



JOIN Homepage -- Howto: OpenVPN IPv6 Tunnel Broker Guide

JOIN IPv6 Projekt Westfälische Wilhelms-Universität Münster

Table of Contents

OpenVPN IPv6 Tunnel Broker Guide.....	1
<u>Table of Contents:</u>	1
<u>1. Introduction</u>	1
<u>2. Definition of the term "tunnel broker"</u>	1
<u>3. Tunnel broker clients</u>	3
<u>4. Installation of tunnel broker components</u>	4
<u>4.1. Installation of OpenSSL CA</u>	4
<u>4.2. Installation of OpenVPN server</u>	7
<u>4.3. Installation of user database</u>	7
<u>5. Functionality of tunnel broker and its components</u>	7
<u>5.1. OpenSSL CA</u>	7
<u>5.2. OpenVPN server</u>	8
<u>6. Routing configuration</u>	9
<u>7. Sample server configuration</u>	11
<u>8. Sample subnet client configuration</u>	12
<u>9. Management</u>	13
<u>10. Client user guide</u>	15
<u>11. Appendix</u>	15
OpenVPN-IPv6-Tunnelbroker-Guide.....	16
<u>Inhaltsverzeichnis:</u>	16
<u>1. Einleitung</u>	16
<u>2. Definition des Begriffs "Tunnelbroker"</u>	16
<u>3. Tunnelbroker-Klienten</u>	18
<u>4. Installation der Tunnelbroker-Komponenten</u>	19
<u>4.1. Installation der OpenSSL-CA</u>	19
<u>4.2. Installation des OpenVPN-Servers</u>	22
<u>4.3. Installation der Nutzerdatenbank</u>	22
<u>5. Funktionalität des Tunnelbrokers und seiner Komponenten</u>	23
<u>5.1. OpenSSL-CA</u>	23
<u>5.2. OpenVPN-Server</u>	24
<u>6. Routing-Konfiguration</u>	24
<u>7. Beispielkonfiguration für einen Server</u>	26
<u>8. Beispielkonfiguration für einen Klienten</u>	28
<u>9. Management</u>	29
<u>10. Klienten-User-Guide</u>	31
<u>11. Anhang</u>	31

OpenVPN IPv6 Tunnel Broker Guide

Copyright © 2004 by [Christian Strauf](#)

Acknowledgements go to people from the University of Erlangen for inspiring us with the idea to use OpenVPN for a tunnel broker service. Thank you, guys, it works like a charm!

Table of Contents:

1. [Introduction](#)
2. [Definition of the term "tunnel broker"](#)
3. [Tunnel broker clients](#)
4. [Installation of tunnel broker components](#)
 1. [Installation of OpenSSL CA](#)
 - ◇ [Creating an openssl.cnf](#)
 - ◇ [Creating keys and certificates for your CA](#)
 2. [Installation of OpenVPN server](#)
 3. [Installation of user database](#)
5. [Functionality of tunnel broker and its components](#)
 1. [OpenSSL CA](#)
 2. [OpenVPN server](#)
6. [Routing configuration](#)
7. [Sample server configuration](#)
8. [Sample subnet client configuration](#)
9. [Management \(and download of JOIN tunnel broker scripts\)](#)
10. [Client user guide](#)
11. [Appendix](#)

1. Introduction

This document describes the process of setting up an OpenVPN based IPv6 tunnel broker to enable ISP-independent IPv6 connectivity that is authenticated, secure, stable, and IPv4 source-address independent. It provides an insight into necessary configurations, gives an overview over administrative tasks and possible caveats.

Important notice for Client users: This document deals with the construction of a whole tunnel broker. For instructions on how to install the client software, please read the paragraph [Client user guide](#).

[Back to top](#)

2. Definition of the term "tunnel broker"

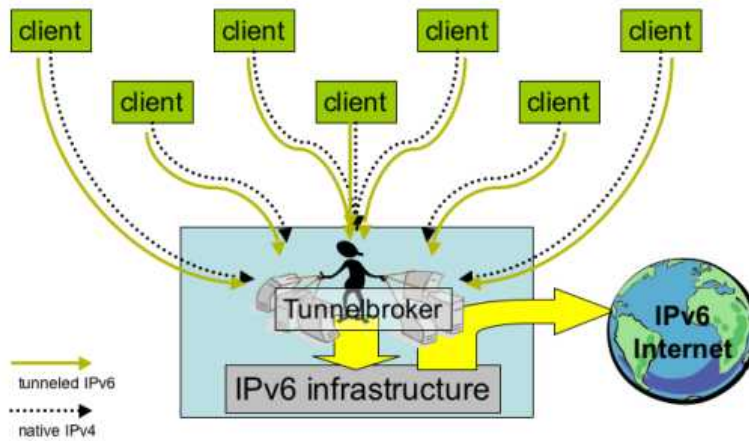


Fig.: Schema of a typical tunnel broker

There are many different definitions of the term "tunnel broker". To clarify what this term means in the context of this document, it is necessary to summarise the tasks that an OpenVPN-based tunnel broker should fulfil:

- provide IPv6 connectivity to a subscribed client
- manage a set of X.509 certificates and keys and a certification authority (CA)
- check authorisation of a client
- assign a fixed IPv6 prefix to each client (either /64 or /128 for a single address)
- adjust routing according to prefix-/address-assignment
- on subscription of a new client, create client configuration for server and as archive file for client
- handle subscription information

To handle all of the above tasks, the tunnel broker needs to consist at least of the following components:

- OpenVPN server(s)
- OpenSSL certification authority (CA)
- client database
- dedicated router for clients (is identical to OpenVPN server)
- IPv6 infrastructure to route IPv6 traffic to and from clients

To visualise the interaction of these components, take a look at the following figure.

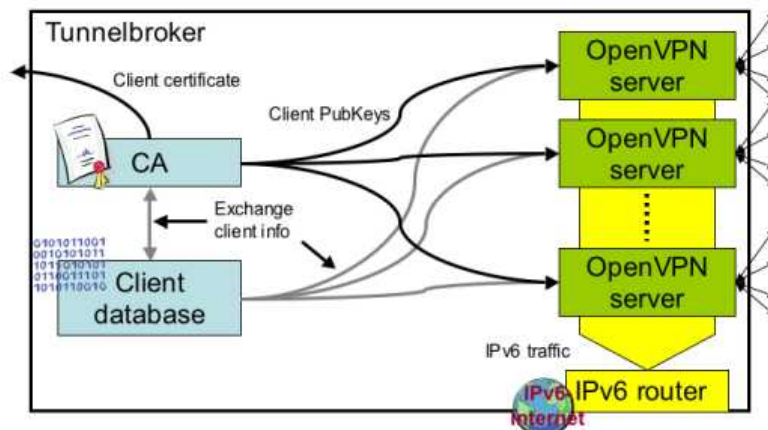


Fig.: Interaction of tunnel broker components

The components in detail may look like this:

- **OpenVPN server:** powerful Linux or *BSD PC with latest OpenVPN software (at the time of writing of this document, this is a version that is more recent than 1.6_rc2); JOIN utilise a Linux server for their installation
- **OpenSSL CA:** may be any kind of machine with an OpenSSL installation which provides the *openssl*-binary to create X.509 keys and certificates
- **Client database:** almost any form of database for holding information about clients ranging from simple text file to dedicated database systems
- **Dedicated IPv6 router:** normally the same Linux or *BSD machine that runs the OpenVPN server; routes need to be adjusted on that particular machine
- **IPv6 infrastructure:** your institution's IPv6 backbone

The above components form what we call a "tunnel broker" for the remainder of this document. It is clear that for the sake of scalability, many of the services (e.g. the OpenVPN server) may be spread across numerous different servers. This is not difficult to achieve and can easily be implemented.

"2. Definition of the term "tunnel broker"

[Back to top](#)

3. Tunnel broker clients

A very important part of a tunnel broker are — obviously — the tunnel broker clients. To understand what functionality a tunnel broker needs to implement, it is necessary to have a look at the different types of clients that need to be connected to the tunnel broker.

First of all, one needs to identify the network topology that a potential client will most likely reside in. It is assumed that any tunnel broker client only has native global IPv4 connectivity and no global IPv6 connectivity. From a practical viewpoint, having global IPv6 connectivity additionally to the tunnel broker IPv6 connectivity is possible. However, this scenario is not a standard scenario where an IPv6 tunnel broker would be employed. Additionally, effects that are imposed by having two types of global connectivity still need to be investigated. No serious problems are to be expected but tests have not yet been conducted to verify this.

One differentiates between two types of clients that reside in two different network topologies for this particular OpenVPN IPv6 tunnel broker:

- **Hermit client:** a lone client that will be assigned a /128 address
- **Subnet client:** a client that will be assigned a /64 prefix and that may act as a router for a subnet where it may announce this /64 prefix to other hosts that use the client as their default router

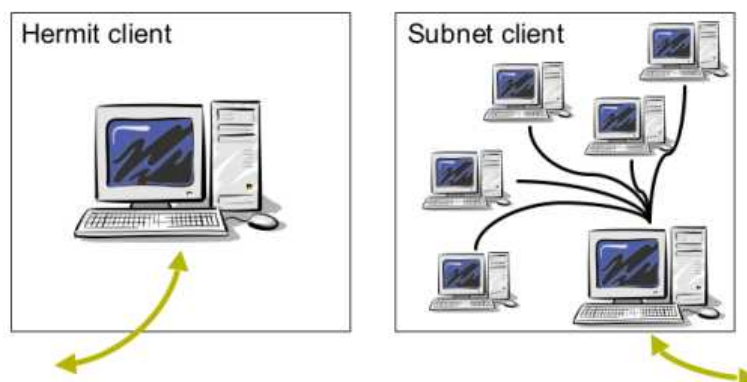


Fig.: Types of tunnel broker clients

Though the OpenVPN client runs on many different platforms and hence a tunnel broker that is based on OpenVPN should be able to accommodate quite a large number of different client OS', this document will only deal with Linux and Windows clients for now, since those are the client OS' that haven been tested as tunnel broker clients so far. However, extending the support to other client OS' should be trivial.

["3. Tunnel broker clients"](#)

[Back to top](#)

4. Installation of tunnel broker components

The upcoming paragraphs deal with the software and hardware installation of all the necessary tunnel broker components. Please note that in most cases this document will not list the exact and detailed installation information of a software if the documentation that comes with the respective software packages is sufficient to explain the process. Since most of the software installation is trivial, this will not pose any problems.

To further facilitate the understanding of the concepts and the installation (especially if the scripts provided by JOIN are to be used for managing the tunnel broker), it is assumed that the CA, the OpenVPN server and the database are located on the same Linux server. The abstraction of the concept to spread these services across several servers and perhaps different operating systems is a trivial step.

4.1. Installation of OpenSSL CA

To set up an OpenSSL certification authority it is sufficient to install the OpenSSL package on your Linux distribution (check that the `openssl-binary` is present and executable as root). The more difficult step is to create an appropriate configuration file and to create the necessary CA certificates and keys. This document will present a sample `openssl.cnf` configuration file that is self-explanatory and that may be edited to fit individual needs. It will also list the necessary steps to create all CA keys and certificates.

Creating an `openssl.cnf`

The following listing shows the `openssl.cnf` that JOIN uses for its OpenSSL CA:

```
#
# OpenSSL example configuration file.
#
HOME                = .
RANDFILE            = $ENV::HOME/.rnd
oid_section         = new_oids

[ new_oids ]

[ ca ]
default_ca         = CA_default

[ CA_default ]
```

JOIN Homepage -- Howto: OpenVPN IPv6 Tunnel Broker Guide

```
dir                = /usr/local/etc/openvpn/ssl           # Adjust this!
certs              = $dir
crl_dir           = $dir
database          = $dir/index.txt
new_certs_dir     = $dir

certificate        = $dir/tmp-ca.crt                   # Adjust this!
serial            = $dir/serial
crl               = $dir/crl.pem
private_key       = $dir/tmp-ca.key                   # Adjust this!
RANDFILE          = $dir/.rand

x509_extensions   = usr_cert

name_opt          = ca_default
cert_opt          = ca_default
default_days      = 365                               # Adjust this!
default_crl_days = 30
default_md        = md5
preserve          = no
policy            = policy_match

[ policy_match ]
countryName       = match
stateOrProvinceName = match
organizationName  = match
organizationalUnitName = optional
commonName        = supplied
emailAddress      = optional

[ policy_anything ]
countryName       = optional
stateOrProvinceName = optional
localityName      = optional
organizationName  = optional
organizationalUnitName = optional
commonName        = supplied
emailAddress      = optional

[ req ]
default_bits      = 1024
default_keyfile   = privkey.pem
distinguished_name = req_distinguished_name
attributes        = req_attributes
x509_extensions  = v3_ca
string_mask       = nombstr

[ req_distinguished_name ]
countryName       = Country Name (2 letter code)
countryName_default = DE                               # Adjust this!
countryName_min   = 2
countryName_max   = 2

stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = Nordrhein-Westfalen    # Adjust this!

localityName       = Locality Name (eg, city)
localityName_default = Muenster                       # Adjust this!

0.organizationName = Organization Name (eg, company)
0.organizationName_default = ZIV, WWU Muenster        # Adjust this!

organizationalUnitName = Organizational Unit Name (eg, section)
organizationalUnitName_default = JOIN-Projekt         # Adjust this!

commonName         = Common Name (eg, YOUR name)
```

JOIN Homepage -- Howto: OpenVPN IPv6 Tunnel Broker Guide

```
commonName_max                = 64

emailAddress                   = Email Address
emailAddress_max               = 64

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

unstructuredName               = An optional company name

[ usr_cert ]
basicConstraints=CA:FALSE
nsComment                      = "OpenSSL Generated Certificate"
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always

[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment

[ v3_ca ]
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer:always
basicConstraints = CA:true

[ crl_ext ]
authorityKeyIdentifier=keyid:always,issuer:always
```

Please note that you need to include "`-config openssl.cnf`" (where `openssl.cnf` is to be given with full path if not located in current directory) in every call of the `openssl`-binary. If you use the management scripts provided by JOIN, you need to adjust the variable pointing to your OpenSSL-installation.

Attention: You must initialise the files `index.txt` and `serial` when setting up the CA. You can create `index.txt` as an empty file and `serial` with the content "01". Additionally, you need a DH-hash that you can create with the command `openssl dhparam -out /usr/local/etc/openvpn/ssl/dh1024.pem 1024`. You can use the script `setup-ca.sh` from the JOIN configuration script package to do this task.

Creating keys and certificates for your CA

Important: This document shows a rather lax use of a CA. Ideally, you should take the process of running a CA rather seriously, especially if you want to hold people liable for the abuse of your tunnel broker. If you already have an established CA, please try to use it for creating client and server certificates. This will add to the security of your tunnel broker. Please use "your own private little CA" just for experimental purposes or if you think that you can afford to keep the security rather lax.

To create your own CA keys and certificates, run the following command:

```
openssl req -days 365 -new -x509 -keyout tmp-ca.key \
-out tmp-ca.crt -config openssl.cnf
```

This command will create the secret key called "`tmp-ca.key`" and the certificate "`tmp-ca.crt`". Please note that the notation "`tmp-ca.*`" is intentional to make administrators aware that they are dealing with an experimental or temporary CA. Please make sure that the permissions for the secret key do not allow it to be read by any other user than the one who

created it ("chmod 600 tmp-ca.key").

With these basic steps the setup of the OpenSSL CA is completed.

4.2. Installation of OpenVPN server

At the time this document was written, a very recent CVS-version of the OpenVPN software was needed to have all functionality that was necessary for running the OpenVPN based IPv6 tunnel broker. The OpenVPN software can be found at

<http://openvpn.sourceforge.net/>

Please follow the installation instructions for the CVS version on the OpenVPN site. The installation is not expected to pose any problems.

It is assumed that the OpenVPN sources are compiled with the given installation prefix `/usr/local` which means that the configuration files can be found in `/usr/local/etc/openvpn`. Naturally, any other prefix works just as well but for the sake of readability, the `/usr/local` prefix will be used for the remainder of the documentation.

4.3. Installation of user database

The installation of a user database is completely up to you. The more users need to be accommodated by the tunnel broker, the more sophisticated the user database should be. For testing purposes, a simple text file may suffice.

If you plan to use the administration scripts provided by JOIN, please be aware that the user management is not yet implemented and you might want to integrate the user management part yourself.

Since there are significant differences between each local user database requirement, this document will not deal with the setup of the database in detail.

["4. Installation of tunnel broker components"](#)

[Back to top](#)

5. Functionality of tunnel broker and its components

All components of the tunnel broker serve a particular purpose. To explain the concept behind the tunnel broker, it is necessary to understand why a certain component is needed and what it actually does. This paragraph deals with each component separately and it will later explain how all the components work together and it will explain, what a typical connection by a client looks like from the client's and the server's point of view.

5.1. OpenSSL CA

For many tunnel brokers, having some form of access control and authorisation is mandatory. It was one of the prerequisites for JOIN when working on the OpenVPN based

tunnel broker. OpenVPN offers a very flexible and secure way of authorising access using X.509 certificates and keys. OpenVPN uses functionality provided by the OpenSSL library (e.g. TLS key exchange). The OpenSSL CA is needed to sign certificates for clients that would like to connect to the tunnel broker.

When a new client subscribes to the tunnel broker service, the following things have to be done:

1. Create an X.509 key and certificate for the client.
2. CA verifies the identity and authorisation of the client to actually use the tunnel broker and then signs the certificate of the client.
3. The CA's certificate is given to both the client and the server. It is used to verify the signature of the X.509 certificates of both client and server when they execute a TLS key exchange.

The CA is the trusted intermediary instance that both the server and the client trust.

When a client starts an OpenVPN-connection to the server, the following steps are exercised by client and server:

1. TLS handshake and key exchange is started.
2. Both server and client verify the Common Names of each other's public certificates. Only on a positive verification of the CN the server or client continue the negotiations. (The CN verification can be seen as an initial sanity check.)
3. After the CN verification, the key exchange is started. Server and client verify each other's keys using the CA's certificate to find out if the signatures of the certificates are valid. If yes, both client and server proceed.
4. The server uses the client's certificate to cipher the stream that is sent to the client and it uses OpenSSL functionality to multiplex the tunnel into a UDP/IPv4 stream. The client uses his private key to decrypt the ciphered stream after de-multiplexing it. The client's use of the server's certificate is analogue.

This summarises where the OpenSSL CA and the X.509 certificates and keys created by the CA play a key role.

Note: JOIN's OpenVPN based tunnel broker used encrypted tunnel streams in an initial version. However, now it uses the "null" cipher instead of the blowfish block stream cipher. This solves some performance and overhead issues.

5.2. OpenVPN server

The OpenVPN server is the actual heart of the tunnel broker. OpenSSL CA and user database are merely the framework or better say the helper applications that enable a controlled use of OpenVPN for a tunnel broker. This document will not go into deep technical detail. Rather, the most important functions of OpenVPN shall be described here.

The most important function of OpenVPN is obviously to provide some form of tunnel for IPv6 over UDP/IPv4. To understand what OpenVPN does to tunnel, one can have a look at what happens when client connects to an OpenVPN server:

1. Server verifies the identity of the client.
2. OpenVPN creates either a tun interface (P-t-P) or a tap interface (ethernet bridge) on both ends.

3. The tunnel interfaces are configured with IPv6 addresses.
4. The routing on the OpenVPN server is adjusted to route a pre-defined IPv6 prefix to the client via the tunnel interface. The client in turn adjusts its IPv6 default route to use the tunnel interface.

Please note that for subnet clients, a special routing prefix is used to route IPv6 traffic. The reason for this is to not use up any IPv6 addresses from the assigned /64 prefix. Hence, the full assigned /64 prefix is at the client's disposal for local address assignment.

The OpenVPN server and client call external scripts upon establishing a new connection. These scripts are:

- a TLS verification script (used for checking CNs of X.509 certificates)
- an "up" script that calls external programs to configure interfaces and to set up routes or starting certain programs after setting up the tunnel
- a "down" script that may be used to cleanly dispose of the tunnel (end programs etc.)

"5. Functionality of tunnel broker and its components"

[Back to top](#)

6. Routing configuration

An important task when setting up and maintaining a tunnel broker is the organisation of the routing. Not only does one have to set up routes to the OpenVPN server which itself has to act as a router, one also needs to dynamically add routes upon a client connection and remove them afterwards. The routing itself is not very complicated but there are a few things that need to be taken into account when planning the routing. Here are a few prerequisites that need to be met for JOIN's tunnel broker:

- A /56 prefix is at the tunnel broker's disposal to assign addresses for clients (either single IPv6 addresses or complete prefixes).
- Subnet clients must be able to freely assign addresses from the /64 prefix they get assigned.
- All clients must be identifiable by their respective addresses (hermit clients) or prefixes (subnet clients).

JOIN came up with a concept that would meet all of the above prerequisites. The concept has the following form:

- The prefix used for JOIN's tunnel broker is `2001:638:500:f100::/56`.
- All hermit hosts receive a /128 address coming from the prefix `2001:638:500:f1ff::/64`.
- All subnet hosts receive /64 prefixes ranging from `2001:638:500:f101::/64` to `2001:638:500:f1fe::/64`.
- The prefix `2001:638:500:f100::/64` is solely used for routing purposes for subnet hosts. Example: for a subnet client that is provided with the prefix `2001:638:500:f1ab::/64`, the routing address `2001:638:500:f100::f1ab:1/112` is used for the P-t-P interface on the OpenVPN server's side, the address `2001:638:500:f100::f1ab:2/112` is used on the OpenVPN client's side. This way routes can be assigned uses those routing addresses as Next-Hop. No addresses from `2001:638:500:f1ab::/64` are used and hence the client can freely assign addresses from this prefix locally.

- All clients are assigned the same addresses or prefixes every time they connect. There is a 1:1 mapping between address/prefix and X.509 certificate. This helps to identify tunnel broker clients by they IPv6 addresses.

The following figure depicts a sample address assignment and routing configuration for a subnet client.

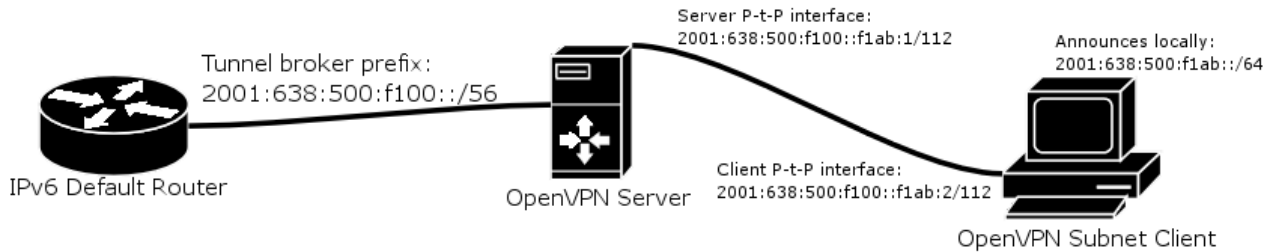


Fig.: Address assignment and routing configuration for a subnet client

To show how the routing is arranged for several subnet clients, please see the following figure.

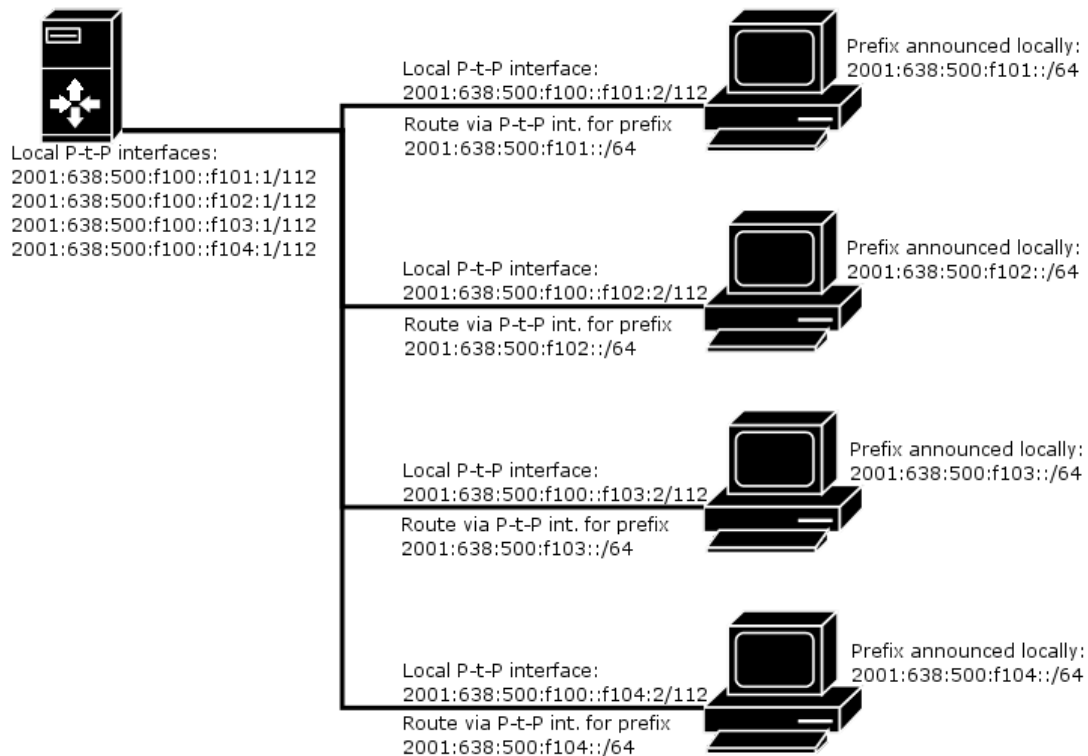


Fig.: Routing for several subnet clients

The hermit client case is trivial: the server's local interface gets a fe80::1/64 IP address to have a next-hop address for the client on the other side. A route to the /128 IPv6 address of the client is added via the corresponding tunnel interface.

"6. Routing configuration"

[Back to top](#)

7. Sample server configuration

To give the reader an idea what a typical server configuration may look like, this chapter will present a number of sample configuration files.

The server needs three mandatory configuration files per client (note: each client has his own server instance with its own configuration; also, each server occupies exactly one UDP port):

- basic configuration file
- TLS verification script
- "up" script to initialise tunnel interface and routing

A sample configuration file looks like this:

```
daemon server-christian.strauf
dev tun
tun-ipv6

up /usr/local/etc/openvpn/server-conf/christian.strauf.up
tls-server

log-append /usr/local/etc/openvpn/server-logs/christian.strauf.log

dh /usr/local/etc/openvpn/ssl/dh1024.pem
ca /usr/local/etc/openvpn/ssl/tmp-ca.crt
cert /usr/local/etc/openvpn/ssl/servers/christian.strauf.crt
key /usr/local/etc/openvpn/ssl/servers/christian.strauf.key
tls-verify /usr/local/etc/openvpn/server-conf/christian.strauf.tls-verify

port 5000

persist-tun
ping 15
ping-restart 45
ping-timer-rem
persist-key

verb 3
```

The first part of the configuration file configures the OpenVPN server itself (act as a daemon, use a tun P-t-P interface and optimise multiplexing for IPv6 tunnelling). The second part tells the server where to find the "up" script and the server should authorise connection via TLS. The third part defines where to write log messages. The fourth part tells the server where to find the Diffie Hellmann hash, the CA certificates, the client's public key, the server's private key and the TLS verification script that check's the client certificate's Common Name. The fifth part tells the server which UDP port to use. The sixth part is very important to guarantee the OpenVPN tunnel's stability when used on dial-up links. It tells the tunnel to try to be as persistent as possible by checking the status of the tunnel with regular pings and other techniques. These options also suffice in many cases to help a client to traverse a NAT gateway because the gateway might be able to recognise that there is constant traffic coming in over the same port. The seventh and last part sets the verbosity level of the log messages.

Another important configuration file is the "up" script that is run by the OpenVPN software after the tunnel has been established. The "up" script configures the local tunnel interface and handles the route setup.

```
#!/bin/bash

INTERFACE=$1; shift;
TUN_MTU=$1; shift;
UDP_MTU=$1; shift;
LOCAL_IP=$1; shift;
REMOTE_IP=$1; shift;
MODUS=$1; shift;

ip link set ${INTERFACE} up
ip link set mtu ${TUN_MTU} dev ${INTERFACE}

ip -6 addr add 2001:638:500:f100::f101:1/112 dev ${INTERFACE}
ip -6 addr add fe80::f101:1/64 dev ${INTERFACE}
ip -6 route add 2001:638:500:f101::/64 dev ${INTERFACE}
exit 0
```

The content of the "up" script should be self-explanatory.

["7. Sample server configuration"](#)

[Back to top](#)

8. Sample subnet client configuration

This paragraph will present a sample subnet client configuration. Since a hermit client's configuration is almost the same, it is easy to deduct the configuration for it from this subnet client's sample configuration.

The basic configuration file of the client that corresponds to the server in the above paragraph looks like the following:

```
daemon client-christian.strauf
dev tun
tun-ipv6

remote corello.uni-muenster.de

up /etc/openvpn/christian.strauf.up

tls-client
ca /etc/openvpn/tmp-ca.crt
cert /etc/openvpn/christian.strauf.crt
key /etc/openvpn/christian.strauf.key

tls-verify /etc/openvpn/tls-verify

port 5000

persist-tun
ping 15
ping-restart 45
ping-timer-rem
persist-key

verb 3
```

The configuration is analog to the server configuration. The only differences are that a remote server is defined that the OpenVPN client tries to connect to and that no DH hash is used. If you use JOIN's management scripts, all configuration files have to be placed in

/etc/openvpn.

The "up" script of a subnet client also sets up the tunnel interface and it creates a default route via the tunnel interface. Additionally, it enables IPv6 packet forwarding and it adds a route for the /64 prefix that is assigned to the client via an internal interface (in most cases eth0):

```
#!/bin/bash

INTERFACE=$1; shift;
TUN_MTU=$1;  shift;
UDP_MTU=$1;  shift;
LOCAL_IP=$1; shift;
REMOTE_IP=$1; shift;
MODUS=$1;    shift;

ip link set ${INTERFACE} up
ip link set mtu ${TUN_MTU} dev ${INTERFACE}

ip -6 addr add 2001:638:500:f100::f101:2/112 dev ${INTERFACE}
ip -6 addr add fe80::f101:2/64 dev ${INTERFACE}
ip -6 route add default dev ${INTERFACE} metric 1

sysctl -w net.ipv6.conf.all.forwarding=1
ip -6 addr show dev eth0 | grep 2001:638:500:f101::1/64 \
    >/dev/null 2>1|| ip -6 addr \
    add 2001:638:500:f101::1/64 dev eth0
```

["8. Sample subnet client configuration"](#)

[Back to top](#)

9. Management

The management of the tunnel broker is rather complicated to handle manually. Therefore, JOIN team members have written a very basic bash script that handles certificate creation, certificate signatures, server- & client-configurations and the creation of an archive file that may be given to users to set up the OpenVPN client on their machines. However, the script as of now can only be used to create client accounts, it cannot remove accounts and it does not handle the storage of client information in a user database. Therefore, the script should only be used as a reference for tunnel broker administrator how to set up local management. The script is released under the GNU GPL and may be modified to fit individual needs.

`create-client-conf.sh` is a script that is used to...

- ... create a client ID (e.g. christian.strauf),
- ... create an X.509 key and certificate for the client,
- ... sign the certificate using the CA's key,
- ... read routing relevant information from commandline (hermite or subnet client, Linux or Windows host, prefix to use),
- ... create the server's configuration files and scripts,
- ... put all client relevant configuration files into an archive that is given to the user.

To facilitate debugging of the tunnel on the client's side, `join-openvpn-sanity-check.sh` can be run on a Linux subnet client. The script collects a number of different information and tries to analyse if the information makes

sense. It tests the setup for potential errors and reports these errors back to the user. The script does not modify the system, it is "read-only". It also does not send information somewhere, it merely displays information on the console that can be used by the user to identify the source of a problem.

```

erog:~/Texte/JOIN-Dokumente/Tunnelbroker/Skripte-und-Con [gargoyl
[gargoyl@etc/openvpn] # ./join-openvpn-sanity-check.sh

JOIN OpenVPN Tunnelbroker Client Sanity Check      Version 0,3
-----
Copyright (C) 2004 by Christian Strauf <strauf@uni-muenster.de>
                        <join@uni-muenster.de>
                        http://www.join.uni-muenster.de

Please note: this script only works for a Linux OpenVPN Client
            that announces a /64 prefix to a local subnet.

Checking existence of a configuration file... OK
Checking client ID... (christian.strauf) OK
Checking existence and executability of up-script... OK
Checking existence of SSL certificate... OK
Checking existence of SSL key... OK
Checking permissions of SSL key... OK
Checking existence and executability of tls-verify script... OK
Checking existence of CA certificate... OK
Checking reachability of tunnelbroker over IPv4... OK
Checking IPv6 connectivity to ftp.ipv6.join.uni-muenster.de... OK
Checking if OpenVPN is running... OK
Checking that OpenVPN port is not occupied by other app... OK
Checking presence of "sysctl"... OK
Checking presence of "ip"... OK
Checking if IPv6 packet forwarding is enabled... OK
Checking if correct address+prefix has been configured for eth0... OK
Checking if tunnel device has correct IPv6 address... OK
Checking that there is only one IPv6 default route... OK
Checking default route is via tunnel device... OK
Checking for tun driver support in kernel... OK
Checking for IPv6 support in kernel ;-)... OK

System infos:
-----
Kernel:      Linux 2.6.3-gentoo-r1 #1 SMP Fri Feb 20 15:08:52 CET 2004
Hardware:    i686 AMD Athlon(tm) Processor AuthenticAMD
OS:          GNU/Linux
OpenVPN version:
OpenVPN 1.6_beta1 i686-pc-linux-gnu [SSL] [LZO] built on Jan 29 2004
Copyright (C) 2002-2004 James Yonan <jim@yanon.net>
ip tool version: ip utility, iproute2-ss010824

Test summary:
-----
No. of tests: 21
PASSED tests: 21
FAILED tests: 0
SKIPPED tests: 0

[gargoyl@etc/openvpn] # █

```

Fig.: Output of `join-openvpn-sanity-check.sh`

Both scripts may be freely downloaded and distributed under the terms of the GNU GPL, however it is important to note that the scripts should only be used as a reference for a local installation rather than as a full featured "all-in-one" package.

Note: Please make sure to read the README and INSTALL files that are included in the tarball because they contain important setup information for the scripts.

- [join-tunnel-broker-scripts-current.tar.gz](#) (MD5-sum: 8d57bd67221d182e29b48a552c427c50)
- [join-tunnel-broker-scripts-0.7.tar.gz](#) (MD5-sum: 8d57bd67221d182e29b48a552c427c50)
- [join-tunnel-broker-scripts-0.6.tar.gz](#) (MD5-sum: f38bbda571b988af2dee231b71a238f8)
- [join-tunnel-broker-scripts-0.5.tar.gz](#) (MD5-sum: 4c81b3f49e1e2f49f2778ace3e86b170)
- [join-tunnel-broker-scripts-0.4.tar.gz](#) (MD5 sum: 6141a53de362d1ca230326b170678172)

- [join-tunnel-broker-scripts-0.3.tar.gz](#) (MD5 sum: dcbc4857d717407508e6f03693a2d156)

["9. Management"](#)

[Back to top](#)

10. Client user guide

The installation of the OpenVPN based IPv6 tunnel broker client consists of three different steps:

1. Subscription to the service and receiving of configuration files from tunnel broker.
2. Installation of the OpenVPN client.
3. Installation of the OpenVPN configuration files provided by the tunnel broker.

The subscription part is the non-technical part. The technical part starts with the installation of the OpenVPN client. The client should be newer than version 1.6_rc3. It can be downloaded at <http://openvpn.sourceforge.net/>. Please install the package according to the documentation that can be found on the OpenVPN homepage (quick guide: under Windows, simply double-click the EXE file; under Linux, make sure that you have tun-driver- and IPv6-support in your kernel and that you have OpenSSL-devel and LZO-devel packages installed and do a `./configure ; make ; make install` — for details, please refer to OpenVPN's documentation).

The configuration files that are provided by the tunnel broker must be copied to either `/etc/openvpn` (Linux) or to the `config` subdirectory of your OpenVPN installation (Windows). Starting the client is trivial:

- `openvpn --config /etc/openvpn/<your.ID>.conf` (Linux)
- run OpenVPN either as a service or from `cmd.exe` (Windows)

["10. Client user guide"](#)

[Back to top](#)

11. Appendix

Important links:

- [OpenVPN homepage](#)

This document is also available as [bi-lingual PDF](#) (264KB).

["11. Appendix"](#)

[Back to top](#)

OpenVPN-IPv6-Tunnelbroker-Guide

Copyright © 2004 by Christian Strauf

Ein besonderes Dankeschön geht an die Leute von der Universität Erlangen, die uns auf die Idee gebracht haben, OpenVPN als Grundlage für einen Tunnelbroker zu verwenden. Vielen herzlichen Dank, es funktioniert wirklich ganz vorzüglich!

Inhaltsverzeichnis:

1. Einleitung
2. Definition des Begriffs "Tunnelbroker"
3. Tunnelbroker-Klienten
4. Installation der Tunnelbroker-Komponenten
 1. Installation der OpenSSL-CA
 - ◇ Erstellung einer openssl.cnf
 - ◇ Erstellung von Schlüsseln und Zertifikaten für Ihre CA
 2. Installation des OpenVPN-Servers
 3. Installation der Nutzerdatenbank
5. Funktionalität des Tunnelbrokers und seiner Komponenten
 1. OpenSSL-CA
 2. OpenVPN-Server
6. Routing-Konfiguration
7. Beispielkonfiguration für einen Server
8. Sample subnet client configuration
9. Management (and download of JOIN tunnel broker scripts)
10. Klienten-User-Guide
11. Anhang

1. Einleitung

Dieses Dokument beschreibt die Vorgehensweise zur Installation eines OpenVPN-basierten Tunnelbrokers, der eine ISP-unabhängige IPv6-Konnektivität erlaubt, welche authentifiziert, sicher, stabil und IPv4-Quelladressen-unabhängig ist. Das Dokument liefert eine Einsicht in die nötigen Konfigurations- und Management-Arbeiten und in mögliche Probleme.

Wichtiger Hinweis für Nutzer des Klienten: Dieses Dokument beschäftigt sich mit der Konstruktion eines kompletten Tunnelbrokers. Für Hinweise, wie Sie nur den Klienten bei sich installieren, lesen Sie bitte den Abschnitt Klienten-User-Guide.

Seitenanfang

2. Definition des Begriffs "Tunnelbroker"

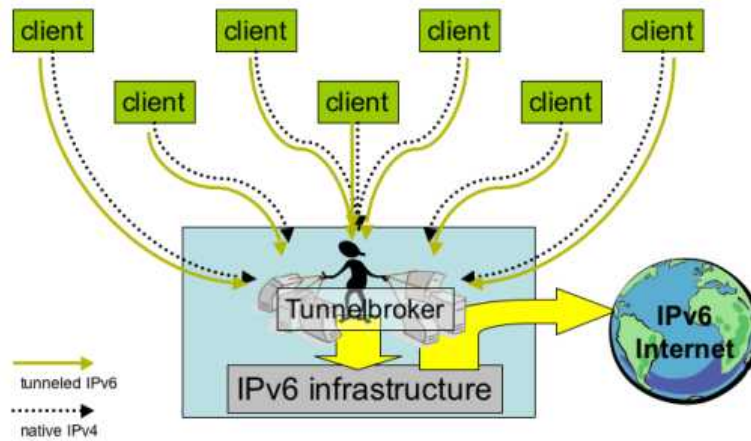


Abb.: Schema eines typischen Tunnelbrokers

Es gibt viele verschiedene Definitionen des Begriffs "Tunnelbroker". Um zu verdeutlichen, was im Verlauf dieses Dokument unter diesem Begriff verstanden wird, sollen zunächst einmal die Aufgaben eines Tunnelbrokers aufgelistet werden:

- Zurverfügungstellung von IPv6-Konnektivität für angemeldete Klienten
- Verwaltung einer Sammlung von X.509-Zertifikaten und -Schlüsseln und einer Certification Authority (CA)
- Überprüfung der Authorisierung eines Klienten
- Zuordnung eines festen IPv6-Präfixes für jeden Klienten (entweder ein /64-Präfix oder eine einzelne /128-Adresse)
- Anpassung des Routings entsprechend der Präfix- bzw. Adresszuordnung
- Erstellung der Klienten- und Server-Konfiguration bei der Anmeldung eines neuen Klienten
- Verwaltung der Anmeldeinformationen

Um alle obigen Aufgaben erledigen zu können, muss ein Tunnelbroker mindestens aus den folgenden Komponenten bestehen:

- OpenVPN-Server
- OpenSSL Certification Authority (CA)
- Klientendatenbank
- Dedizierter Router für die Klienten (ist i.d.R. identisch mit dem OpenVPN-Server)
- IPv6-Infrastruktur, um den IPv6-Verkehr vom und zum Klienten zu routen

Die Interaktion dieser Komponenten wird in der folgenden Abbildung verdeutlicht.

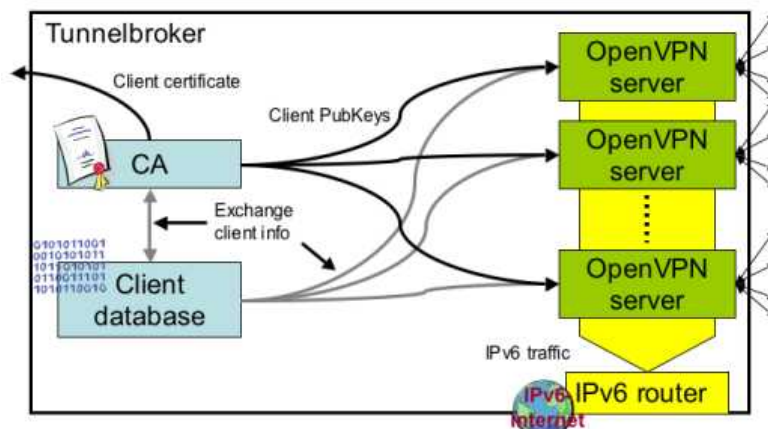


Abb.: Interaktion der Tunnelbroker-Komponenten

Im Detail können die einzelnen Komponenten wie folgt aussehen:

- **OpenVPN-Server:** leistungsfähiger Linux- oder *BSD-PC mit der neuesten OpenVPN-Software (bzw. einer Version, die neuer ist als 1.6_rc2); JOIN verwendet einen Linux-Server für die Installation.
- **OpenSSL-CA:** eine beliebige Maschine mit einer OpenSSL-Installation, die das *openssl*-Binary zur Verfügung stellt, um X.509-Schlüssel und -Zertifikate zu erstellen.
- **Klientendatenbank:** eine beliebige Art von Datenbank, um Information über die Klienten zu verwalten; kann eine Textdatei sein, kann aber auch ein dediziertes Datenbanksystem sein.
- **Dedizierter IPv6-Router:** normalerweise dieselbe Linux- bzw. *BSD-Maschine, auf der der OpenVPN-Server läuft; alle Routen, die den Tunnelbroker betreffen, müssen auf dieser Maschine verwaltet werden.
- **IPv6-Infrastruktur:** der IPv6-Backbone Ihrer Institution.

Die obigen Komponenten bilden das, was im weiteren Verlauf dieses Dokuments unter "Tunnelbroker" verstanden wird. Es ist offensichtlich, dass aus Gründen der Skalierbarkeit viele der Services auf verschiedene Server verteilt werden können. Dies ist nicht schwierig und sollte sich leicht implementieren lassen.

"2. Definition des Begriffs "Tunnelbroker"

Seitenanfang

3. Tunnelbroker-Klienten

Ein sehr wichtiger Bestandteil eines Tunnelbrokers sind — offensichtlich — die Tunnelbroker-Klienten. Um die Funktionalitäten zu verstehen, die ein Tunnelbroker bieten muss, werden nun die verschiedenen Typen von Klienten betrachtet, die sich mit dem Tunnelbroker verbinden können sollen.

Zuerst müssen die möglichen Netzwerktopologien betrachtet werden, in denen sich ein potentieller Klient befinden kann. Es wird angenommen, dass ein Tunnelbroker-Klient nur native globale IPv4-Konnektivität besitzt, jedoch keine globale IPv6-Konnektivität. Aus praktischer Sicht sollte es möglich sein, simultan native globale IPv6-Konnektivität und IPv6-Konnektivität über den Tunnelbroker zu haben. Dies ist jedoch kein Standardszenario, in dem ein Tunnelbroker eingesetzt werden würde. Die möglicherweise auftretenden Effekte bei einer solchen Konfiguration müssen noch erforscht werden. Es sind keine ernstesten Probleme zu erwarten, aber trotzdem müssen Tests in diese Richtung erst noch durchgeführt werden.

Für den hier besprochenen, speziellen Fall eines Tunnelbrokers werden zwei verschiedene Netzwerktopologien auf Klientenseite unterschieden:

- **Hermit client (Einzel-Host-Klient):** ein einzelner Klient, dem eine /128-Adresse zugewiesen wird
- **Subnet client (Subnetzclient):** ein Klient, dem ein /64-Präfix zugewiesen wird, den er wiederum lokal announce kann und den er für lokale Klienten routet

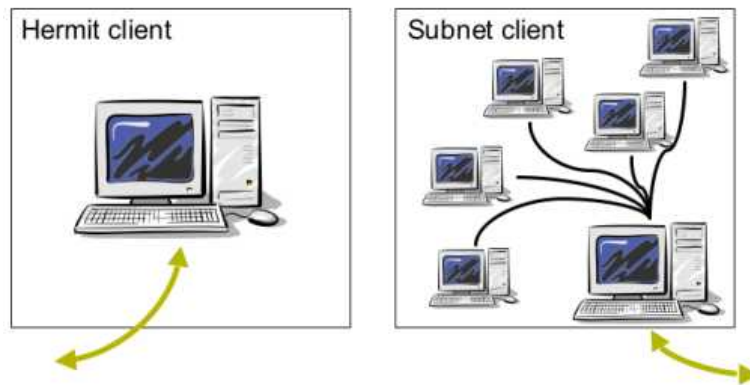


Abb.: Typen von Tunnelbroker-Klienten

Obwohl der OpenVPN-Klient auf vielen verschiedenen Plattformen läuft und eine Nutzung des Tunnelbroker-Services auf allen diesen Plattformen theoretisch denkbar ist, konzentriert sich dieses Dokument nur auf Linux- und Windows-Klienten, da dies die Betriebssysteme sind, mit denen der Tunnelbroker bisher getestet wurde. Die Unterstützung auch auf andere Betriebssysteme zu erweitern, sollte trivial sein.

["3. Tunnelbroker-Klienten"](#)

[Seitenanfang](#)

4. Installation der Tunnelbroker-Komponenten

Die folgenden Abschnitte beschäftigen sich mit der Software- und Hardware-Installation aller für den Tunnelbroker nötigen Komponenten. In den meisten Fällen wird dieses Dokument keine exakten und detaillierten Installationsanweisungen geben, wenn die Dokumentation der entsprechenden Software hinreichend gut ist. Da die meisten nötigen Software-Installationen trivial sind, sollte dies keine größeren Probleme bereiten.

Um das Verständnis des Konzepts und der Installation (insbesondere bei der Benutzung der von JOIN zur Verfügung gestellten Skripte) zu vereinfachen, wird angenommen, dass die CA, der OpenVPN-Server und die Datenbank sich auf demselben Linux-Rechner befinden. Die Verteilung dieser Services auf verschiedene Systeme ist trivial.

4.1. Installation der OpenSSL-CA

Um eine OpenSSL Certification Authority aufzubauen, reicht es, das OpenSSL-Paket der verwendeten Linux-Distribution zu installieren (es reicht zu überprüfen, ob das `openssl`-Binary installiert und als `root` ausführbar ist). Der schwierigere Schritt ist die Erstellung des für die CA nötigen Konfigurationsfiles und der Zertifikate bzw. Schlüssel. Dieses Dokument zeigt ein exemplarisches `openssl.cnf`, welches selbsterklärend ist und welches an die lokalen Bedürfnisse angepasst werden kann. Es werden weiterhin die nötigen Arbeiten erklärt, die zum Erstellen von CA-Schlüsseln und -Zertifikaten nötig sind.

Erstellung einer `openssl.cnf`

Das folgende Listing zeigt die `openssl.cnf`, die JOIN für die OpenSSL-CA verwendet:

```
#
```

JOIN Homepage -- Howto: OpenVPN IPv6 Tunnel Broker Guide

```
# OpenSSL example configuration file.
#
HOME = .
RANDFILE = $ENV::HOME/.rnd
oid_section = new_oids

[ new_oids ]

[ ca ]
default_ca = CA_default

[ CA_default ]

dir = /usr/local/etc/openvpn/ssl # Adjust this!
certs = $dir
crl_dir = $dir
database = $dir/index.txt
new_certs_dir = $dir

certificate = $dir/tmp-ca.crt # Adjust this!
serial = $dir/serial
crl = $dir/crl.pem
private_key = $dir/tmp-ca.key # Adjust this!
RANDFILE = $dir/.rand

x509_extensions = usr_cert

name_opt = ca_default
cert_opt = ca_default
default_days = 365 # Adjust this!
default_crl_days = 30
default_md = md5
preserve = no
policy = policy_match

[ policy_match ]
countryName = match
stateOrProvinceName = match
organizationName = match
organizationalUnitName = optional
commonName = supplied
emailAddress = optional

[ policy_anything ]
countryName = optional
stateOrProvinceName = optional
localityName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
emailAddress = optional

[ req ]
default_bits = 1024
default_keyfile = privkey.pem
distinguished_name = req_distinguished_name
attributes = req_attributes
x509_extensions = v3_ca
string_mask = nombstr

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

JOIN Homepage -- Howto: OpenVPN IPv6 Tunnel Broker Guide

```
stateOrProvinceName          = State or Province Name (full name)
stateOrProvinceName_default  = Nordrhein-Westfalen # Adjust this!

localityName                 = Locality Name (eg, city)
localityName_default         = Muenster # Adjust this!

0.organizationName           = Organization Name (eg, company)
0.organizationName_default   = ZIV, WWU Muenster # Adjust this!

organizationalUnitName       = Organizational Unit Name (eg, section)
organizationalUnitName_default = JOIN-Projekt # Adjust this!

commonName                   = Common Name (eg, YOUR name)
commonName_max               = 64

emailAddress                  = Email Address
emailAddress_max             = 64

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

unstructuredName             = An optional company name

[ usr_cert ]
basicConstraints=CA:FALSE
nsComment                  = "OpenSSL Generated Certificate"
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always

[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment

[ v3_ca ]
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer:always
basicConstraints = CA:true

[ crl_ext ]
authorityKeyIdentifier=keyid:always,issuer:always
```

Wichtig ist, dass allen Aufrufen des `openssl`-Binärs die Option `"-config openssl.cnf"` hinzuzufügen ist (`openssl.cnf` muss ggf. mit vollem Pfad angegeben werden). Bei der Verwendung der Management-Skripte von JOIN muss entsprechend der Pfad der OpenSSL-Installation im Skript angepasst werden.

Achtung: Sie müssen die Dateien `index.txt` und `serial` bei der Einrichtung der CA initialisieren. Dazu legen Sie `index.txt` als leere Datei an und schreiben den Inhalt "01" in `serial`. Zusätzlich benötigen Sie noch ein DH-hash, das Sie mit `openssl dhparam -out /usr/local/etc/openvpn/ssl/dh1024.pem 1024` erstellen können. Sie können für diese Arbeiten das Skript `setup-ca.sh` aus den JOIN-Konfigurationskripten verwenden.

Erstellung von Schlüsseln und Zertifikaten für Ihre CA

Wichtig: Dieses Dokument zeigt einen recht laschen Umgang mit einer CA. Im Idealfall sollten Sie den Betrieb einer CA sehr ernst nehmen, insbesondere dann, wenn Sie Klienten für den Missbrauch Ihrer CA zur Verantwortung ziehen können möchten. Sollten Sie bereits über eine

etablierte CA verfügen, versuchen Sie diese für die Erstellung von Klient- und Server-Zertifikaten zu verwenden. Dies erhöht die Sicherheit Ihres Tunnelbrokers. Bitte nutzen Sie Ihre "eigene kleine CA" nur zu experimentellen Zwecken oder wenn Sie denken, dass Sie mit einem niedrigeren Sicherheitsstandard auskommen.

Um CA-Schlüssel und -Zertifikate zu generieren, ist das folgende Kommando zu verwenden:

```
openssl req -days 365 -new -x509 -keyout tmp-ca.key \  
-out tmp-ca.crt -config openssl.cnf
```

Dieses Kommando erstellt einen geheimen Schlüssel namens "tmp-ca.key" und ein Zertifikat namens "tmp-ca.crt". Es ist zu beachten, dass die Namen "tmp-ca.*" bewusst so gewählt wurden, um hervor zu heben, dass es sich um eine temporäre CA handelt. Es ist sicher zu stellen, dass die Leserechte des geheimen Schlüssels so gesetzt sind, dass nur der erstellende Benutzer (z.B. root) ihn lesen kann.

Mit diesen Grundsritten ist das Setup der OpenSSL CA beendet.

4.2. Installation des OpenVPN-Servers

Zur Zeit, als dieses Dokument erstellt wurde, wurde eine sehr neue OpenVPN-Version benötigt, um alle Funktionalitäten zum Tunneln von IPv6 zur Verfügung zu haben. Die OpenVPN-Software kann hier herunter geladen werden:

<http://openvpn.sourceforge.net/>

Bitte folgend Sie den Installationsanweisungen, die Sie auf der OpenVPN-Seite finden. Die Installation ist erfahrungsgemäß unproblematisch.

Es wird für das verbleibende Dokument angenommen, dass der Installationspräfix `/usr/local` ist, was bedeutet, dass die Konfigurationsdateien unter `/usr/local/etc/openvpn` zu finden sind. Selbstverständlich funktioniert ein beliebiger anderer Präfix ebenfalls, aber zur Erhöhung der Lesbarkeit wird im weiteren Verlauf nur noch `/usr/local` verwendet.

4.3. Installation der Nutzerdatenbank

Die Installation der Nutzerdatenbank ist vollkommen Ihnen überlassen. Je mehr Nutzer den Tunnelbroker verwenden wollen, desto effizienter sollte die verwendete Datenbank sein. Zu Testzwecken kann durchaus eine einfache Textdatei reichen.

Wenn Sie das von JOIN zur Verfügung gestellte Administrationsskript verwenden sollten Sie wissen, dass das Nutzer-Management dort noch nicht integriert ist. Den Management-Teil sollten Sie also ggf. selbst dort einbauen.

Da es signifikante Unterschiede in den lokalen Anforderungen an eine Nutzerdatenbank gibt, wird dieses Dokument nicht näher auf die Installation einer solchen Datenbank eingehen.

"4. Installation der Tunnelbroker-Komponenten"

[Seitenanfang](#)

5. Funktionalität des Tunnelbrokers und seiner Komponenten

Alle Komponenten des Tunnelbrokers dienen einem bestimmten Zweck. Um das Konzept hinter dem Tunnelbroker besser erklären zu können, ist es wichtig zu wissen, warum eine bestimmte Komponente nötig ist und welche Aufgabe sie genau erfüllt. Dieser Abschnitt behandelt jede Komponente separat und er wird später erläutern, wie alle besprochenen Komponenten zusammen arbeiten und wie eine typische Verbindung eines Klienten von Seite des Klienten und von Seite des Servers aussieht.

5.1. OpenSSL-CA

Für viele Tunnelbroker ist eine Form von Authentifizierung obligatorisch. Dies galt auch für den Tunnelbroker, den JOIN betreiben wollte. OpenVPN bietet einen sehr flexiblen Mechanismus zur Authentifizierung mit X.509-Zertifikaten und -Schlüsseln. Die von OpenVPN verwendeten Funktionalitäten werden von der OpenSSL-Bibliothek zur Verfügung gestellt (z.B. TLS-Key-Exchange). Die OpenSSL-CA wird zum Unterzeichnen von Zertifikaten von Klienten gebraucht, die den Tunnelbroker verwenden wollen.

Wenn ein neuer Klient Zugang zum Tunnelbroker bekommen soll, müssen die folgenden Dinge geschehen:

1. Erstellung eines X.509-Schlüssels und -Zertifikats für den Klienten.
2. CA verifiziert die Identität und die Autorisierung des Klienten, den Tunnelbroker zu benutzen und signiert dann das Zertifikat des Klienten.
3. Das Zertifikat der CA wird dem Server und dem Klienten gegeben. Es wird dazu verwendet, die Signatur der X.509-Zertifikate von Server und Klient zu überprüfen, wenn der TLS-Key-Exchange durchgeführt wird.

Die CA ist die vermittelnde Instanz, der beide Seiten (Server und Klient) vertrauen.

Sobald der Klient eine OpenVPN-Verbindung zum Server aufbaut, werden die folgenden Schritte sowohl vom Klienten, als auch vom Server durchgeführt:

1. Start des TLS-Handshakes und -Key-Exchanges.
2. Server und Klient überprüfen die Common Names der jeweils anderen öffentlichen Zertifikate. Nur bei einem positiven Ausgang dieser Verifikation wird der Verbindungsaufbau fortgesetzt. (Die CN-Verifikation kann als initialer Logik-Check betrachtet werden.)
3. Nach der CN-Verifikation wird der Key-Exchange gestartet. Server und Klient überprüfen nun gegenseitig ihre Schlüssel mit Hilfe des Zertifikats der CA, um heraus zu finden, ob die Signaturen gültig sind. Falls ja, so fahren beide fort.
4. Der Server verwendet das Zertifikat des Klienten, um den Stream zum Klienten mit Hilfe von OpenSSL zu einem verschlüsselten UDP/IPv4-Stream zu multiplexen. Der Klient verwendet seinen privaten Schlüssel, um den Stream wieder zu entschlüsseln. Die umgekehrte Richtung funktioniert analog.

Dies fasst die Rolle der OpenSSL-CA und die Schlüsselrolle der X.509-Zertifikate und -Keys zusammen.

Beachten Sie: JOINs OpenVPN-basierter Tunnelbroker verwendete in einer ersten Version verschlüsselte Tunnel-Streams. Mittlerweile verwendet er den "null"-Cipher anstatt des Blowfish-Ciphers. Dies verbessert die Performanz und beseitigt Overhead-Probleme.

5.2. OpenVPN-Server

Der OpenVPN-Server ist das eigentliche Herz des Tunnelbrokers. OpenSSL-CA und Nutzerdatenbank sind nur "Beigaben" oder besser noch "Hilfsapplikationen", die eine bessere Kontrolle über den Tunnelbroker an sich erlauben. Dieses Dokument wird nicht tief in technische Details gehen. Vielmehr sollen hier die wesentlichen Funktionen von OpenVPN beschrieben werden.

Die wichtigste Funktion von OpenVPN ist offensichtlich die Zurverfügungstellung eines IPv6-Tunnels per UDP/IPv4. Um zu verstehen, was OpenVPN tut, um den Tunnel aufzubauen, kann man sich anschauen, was bei der Verbindung eines Klienten zum Server geschieht:

1. Server verifiziert die Identität des Klienten.
2. OpenVPN erstellt entweder ein tun-Interface (P-t-P) oder ein tap-Interface (Ethernet-Bridge) auf beiden Seiten.
3. Die Tunnel-Interfaces werden mit IPv6-Adressen konfiguriert.
4. Das Routing auf dem OpenVPN-Server wird so angepasst, dass ein vordefinierter IPv6-Präfix über den Tunnel zum Klienten geroutet wird. Der Klient wiederum erstellt eine IPv6-Default-Route über das Tunnel-Interface auf seiner Seite.

Bitte beachten Sie, dass für Subnetzklienten ein spezieller Routing-Präfix für das Routing von IPv6-Verkehr verwendet wird. Der Grund dafür ist, dass keine Adressen des zugeordneten /64-Präfixes aufgebraucht werden sollen. So stehen dem Klienten wirklich alle Adressen des /64-Präfixes zur Verfügung.

Der OpenVPN-Server und der -Klient rufen beim Aufbau einer Verbindung externe Skripts auf. Diese sind:

- ein TLS-Überprüfungsskript (wird benutzt, um die CNs der X.509-Zertifikate zu überprüfen)
- ein "up"-Skript, welches externe Programme zur Konfigurierung des Interfaces und zur Erstellung von Routen und auch andere Programme aufruft, die nach dem Start des Tunnels laufen sollen
- ein "down"-Skript, das zum sauberen Beenden des Tunnels verwendet wird (Beendigung von Programme etc.)

"5. Funktionalität des Tunnelbrokers und seiner Komponenten"

[Seitenanfang](#)

6. Routing-Konfiguration

Eine wichtige Aufgabe bei der Verwaltung eines Tunnelbrokers ist das Management des Routings. Es müssen nicht nur Routen zum OpenVPN-Server an sich erstellt werden, es müssen zusätzlich dynamisch Routen auf dem OpenVPN-Server angelegt und entfernt werden, wenn Klienten sich verbinden und anschließend die Verbindung wieder trennen. Das Routing an sich ist nicht sonderlich kompliziert, aber einige Dinge müssen dabei beachtet werden, wenn das Routing geplant wird. Es sollen nun einige Voraussetzungen geschildert werden, die beim JOIN-Tunnelbroker erfüllt sein mussten:

- Dem Tunnelbroker steht ein /56-Präfix zur Verfügung, um Klienten Adressen zuzuordnen (entweder einzelne Adressen oder ganze Präfixe).

- Subnetzklienten müssen in der Lage sein, frei Adressen aus dem ihnen zugeordneten /64-Präfix vergeben zu können.
- Alle Klienten müssen über ihre jeweilige Adresse (Einzel-Hosts) oder ihren jeweiligen Präfix (Subnetzklient) identifizierbar sein.

JOIN entwickelte folgendes Konzept, um alle obigen Bedingungen zu erfüllen:

- Der für den JOIN-Tunnelbroker verwendete Präfix lautet `2001:638:500:f100::/56`.
- Alle Einzel-Hosts erhalten eine /128-Adresse aus dem Präfix `2001:638:500:f1ff::/64`.
- Alle Subnetzklienten erhalten einen /64-Präfix aus dem Bereich `2001:638:500:f101::/64` bis `2001:638:500:f1fe::/64`.
- Der Präfix `2001:638:500:f100::/64` wird ausschließlich für das Routing bei Subnetzklienten verwendet. Beispiel: für einen Subnetzklienten, dem der Präfix `2001:638:500:f1ab::/64` zugeordnet wurde, wird die Adresse `2001:638:500:f100::f1ab:1/112` für das P-t-P-Interface auf der Server-Seite und die Adresse `2001:638:500:f100::f1ab:2/112` auf der Klientenseite verwendet. Auf diese Weise können diese Adressen bei der Zuweisung von Routen als Next-Hop verwendet werden. Dabei werden keinerlei Adressen aus dem Präfix `2001:638:500:f1ab::/64` verbraucht, und alle Adressen stehen dem Klienten zur Vergabe zur Verfügung.
- Allen Klienten wird bei der Verbindung jedes Mal dieselbe Adresse bzw. derselbe Präfix zugeordnet. Dadurch hat man eine 1:1-Abbildung zwischen Adresse/Präfix und dem X.509-Zertifikat. Dies hilft bei der Identifizierung von Tunnelbroker-Klienten über ihre IPv6-Adresse.

Die folgende Abbildung zeigt das Beispiel einer Adresszuweisung und Routing-Konfiguration für einen Subnetzklienten.

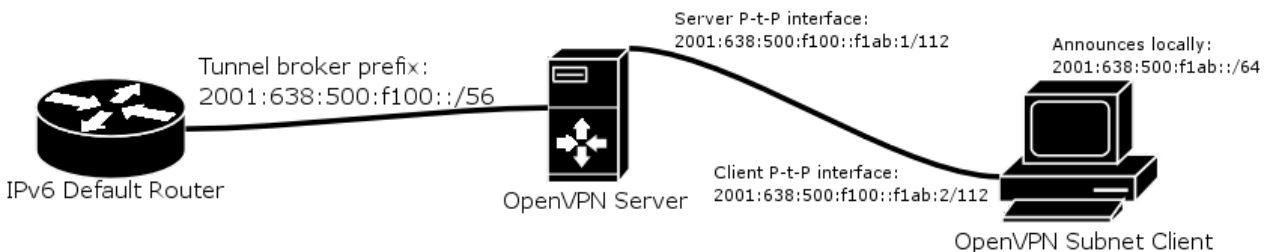


Abb.: Adresszuweisung und Routing-Konfiguration für einen Subnetzklienten

Um zu illustrieren, wie das Routing für mehrere Subnetzklienten organisiert wird, betrachten Sie bitte die folgende Abbildung.

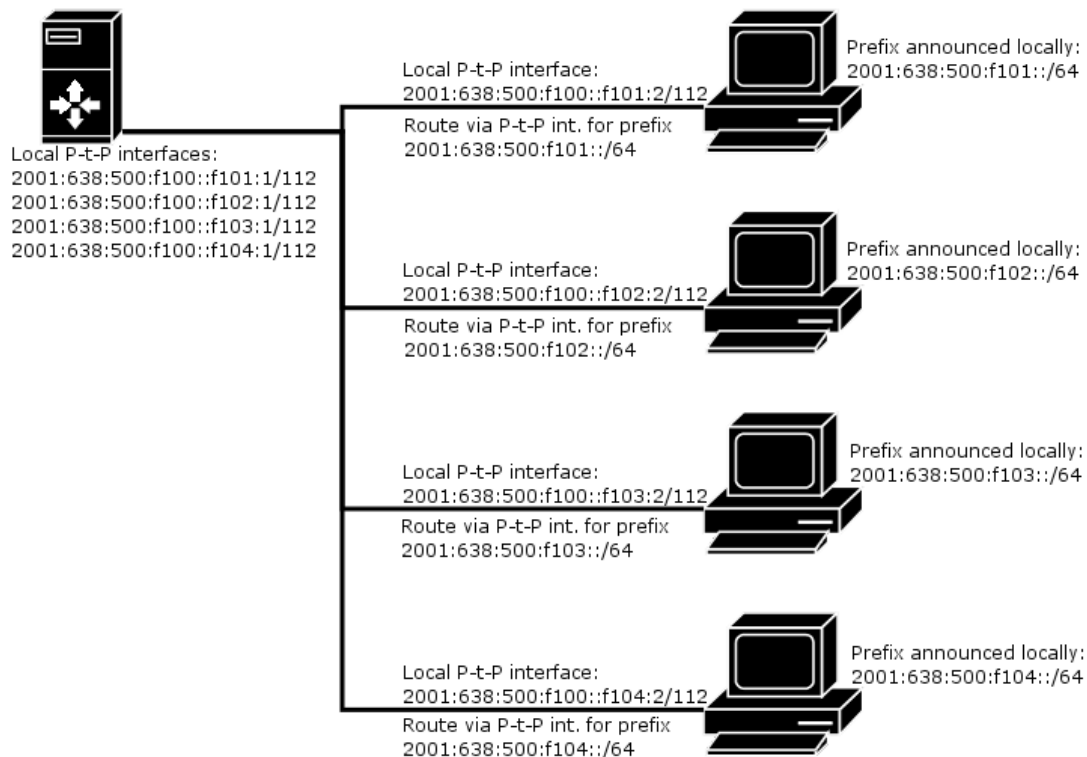


Abb.: Routing für mehrere Subnetzklanten

Der Einzel-Host-Fall ist trivial: das lokale Interface des Servers bekommt die Adresse fe80::1/64, damit dem Klient eine Next-Hop-Adresse zur Verfügung steht. Eine Route für die vergebene /128-Adresse wird dem korrespondierenden Tunnel-Interface zugeordnet.

["6. Routing-Konfiguration"](#)

[Seitenanfang](#)

7. Beispielkonfiguration für einen Server

Um dem Leser eine Vorstellung von einer typischen Server-Konfiguration zu vermitteln, wird dieser Abschnitt ein Reihe von Beispielkonfigurationsdateien vorstellen.

Der Server benötigt drei obligatorische Konfigurationsdateien pro Klient (beachten Sie: jeder Klient bekommt eine eigene Server-Instanz mit einer eigenen Konfiguration; zusätzlich belegt jeder Server genau einen UDP port):

- Basiskonfigurationsdatei
- TLS-Verifikationsskript
- "up"-Skript zum Initialisieren des Tunnel-Interfaces und des Routings

Ein Beispielkonfigurationsfile sieht wie folgt aus:

```
daemon server-christian.strauf
dev tun
tun-ipv6

up /usr/local/etc/openvpn/server-conf/christian.strauf.up
tls-server

log-append /usr/local/etc/openvpn/server-logs/christian.strauf.log
```

JOIN Homepage -- Howto: OpenVPN IPv6 Tunnel Broker Guide

```
dh /usr/local/etc/openvpn/ssl/dh1024.pem
ca /usr/local/etc/openvpn/ssl/tmp-ca.crt
cert /usr/local/etc/openvpn/ssl/servers/christian.strauf.crt
key /usr/local/etc/openvpn/ssl/servers/christian.strauf.key
tls-verify /usr/local/etc/openvpn/server-conf/christian.strauf.tls-verify

port 5000

persist-tun
ping 15
ping-restart 45
ping-timer-rem
persist-key

verb 3
```

Der erste Teil der Konfigurationsdatei konfiguriert den OpenVPN-Server selbst (laufe als Daemon, verwende ein tun-P-t-P-Interface und optimiere das Multiplexing für das Tunneln von IPv6). Der zweite Teil zeigt dem Server, wo das "up"-Skript zu finden ist und dass der Server Verbindungen mit Hilfe von TLS authentifizieren soll. Der dritte Teil beschreibt, wo Log-Messages hin geschrieben werden sollen. Der vierte Teil sagt dem Server, wo der Diffie-Hellmann-Hash, die CA-Zertifikate, der Public-Key des Klienten, der private Schlüssel des Servers und das TLS-Verifikationskript zu finden sind, um den Common Name des Klienten zu überprüfen. Der fünfte Teil sagt dem Server, welchen UDP-Port er verwenden soll. Der sechste Teil ist sehr wichtig, um die Stabilität des Tunnels bei der Verwendung von instabilen Einwahlverbindungen zu gewährleisten. Die dort gemachten Einstellungen sorgen dafür, dass der Tunnel so persistent wie möglich ist, in dem er seinen Status mit Hilfe von regelmäßigen Pings und anderen Techniken überprüft. Diese Einstellungen sind meist auch ausreichend, um dem Tunnel das Traversieren von NAT-Gateways zu ermöglichen. Der siebte Teil setzt die Verbotitäts-Level der Log-Messages.

Eine weitere wichtige Konfigurationsdatei ist das "up"-Skript, das von der OpenVPN-Software nach der Herstellung des Tunnels aufgerufen wird. Es konfiguriert das lokale Tunnel-Interface und verwaltet das Setup der Routen.

```
#!/bin/bash

INTERFACE=$1; shift;
TUN_MTU=$1; shift;
UDP_MTU=$1; shift;
LOCAL_IP=$1; shift;
REMOTE_IP=$1; shift;
MODUS=$1; shift;

ip link set ${INTERFACE} up
ip link set mtu ${TUN_MTU} dev ${INTERFACE}

ip -6 addr add 2001:638:500:f100::f101:1/112 dev ${INTERFACE}
ip -6 addr add fe80::f101:1/64 dev ${INTERFACE}
ip -6 route add 2001:638:500:f101::/64 dev ${INTERFACE}
exit 0
```

Der Inhalt des "up"-Skripts sollte selbsterklärend sein.

["7. Beispielkonfiguration für einen Server"](#)

[Seitenanfang](#)

8. Beispielkonfiguration für einen Klienten

Dieser Abschnitt stellt eine Beispielkonfiguration für einen Klienten vor. Da die Konfiguration für einen Einzel-Hosts fast identisch ist, wird hier nur die Konfiguration eines Subnetzclients erläutert.

Die Grundkonfiguration des zum obigen Server korrespondierenden Klienten sieht wie folgt aus:

```
daemon client-christian.strauf
dev tun
tun-ipv6

remote corello.uni-muenster.de

up /etc/openvpn/christian.strauf.up

tls-client
ca /etc/openvpn/tmp-ca.crt
cert /etc/openvpn/christian.strauf.crt
key /etc/openvpn/christian.strauf.key

tls-verify /etc/openvpn/tls-verify

port 5000

persist-tun
ping 15
ping-restart 45
ping-timer-rem
persist-key

verb 3
```

Die Konfiguration ist analog zur Server-Konfiguration. Der einzige Unterschied ist die Festlegung des Remote-Servers, der vom Klienten kontaktiert werden soll und dass kein DH-Hash verwendet wird. Wenn Sie JOINS Management-Skripte verwenden, so müssen die Konfigurationsdateien in `/etc/openvpn` liegen.

Das "up"-Skript eines Subnetzclients initialisiert das Tunnel-Interface und erstellt eine Default-Route darüber. Es stellt zusätzlich das Packet-Forwarding an und erstellt eine Route für den dem Klienten zugeordneten /64-Präfix über das interne Interface (in den meisten Fällen eth0):

```
#!/bin/bash

INTERFACE=$1; shift;
TUN_MTU=$1; shift;
UDP_MTU=$1; shift;
LOCAL_IP=$1; shift;
REMOTE_IP=$1; shift;
MODUS=$1; shift;

ip link set ${INTERFACE} up
ip link set mtu ${TUN_MTU} dev ${INTERFACE}

ip -6 addr add 2001:638:500:f100::f101:2/112 dev ${INTERFACE}
ip -6 addr add fe80::f101:2/64 dev ${INTERFACE}
ip -6 route add default dev ${INTERFACE} metric 1

sysctl -w net.ipv6.conf.all.forwarding=1
```

```
ip -6 addr show dev eth0 | grep 2001:638:500:f101::1/64 \  
>/dev/null 2>1|| ip -6 addr \  
add 2001:638:500:f101::1/64 dev eth0
```

["8. Beispielkonfiguration für einen Klienten"](#)

[Seitenanfang](#)

9. Management

Das Management des Tunnelbrokers gestaltet sich für eine manuelle Verwaltung zu kompliziert. Aus diesem Grund haben JOIN-Mitglieder ein sehr einfaches Bash-Skript geschrieben, das die Zertifikatserstellung, die Zertifikatssignatur, die Server- & Klientenkonfiguration und die Erstellung von Archivdateien mit Konfigurationsdateien zur Herausgabe an neue Klienten regelt. Allerdings kann das Skript bisher nur Klienten-Accounts erstellen, jedoch keine löschen und es kann auch keine Daten in Datenbanken verwalten. Deswegen sollte das Skript nur als Referenz für einen potentiellen Tunnelbroker-Administrator verwendet werden, um ihm die Verwaltung des Tunnelbrokers zu erleichtern bzw. ihm als Grundlage für eigene Verwaltungsprogramme zu dienen. Das Skript steht unter der GNU GPL und darf frei modifiziert werden, um individuellen Ansprüchen zu genügen.

`create-client-conf.sh` ist ein Skript, das verwendet wird, um...

- ... Klient-IDs zu erstellen (z.B. christian.strauf),
- ... X.509-Zertifikate und -Schlüssel für Klienten zu erstellen,
- ... das Zertifikat mit dem CA-Schlüssel zu unterzeichnen,
- ... routing-relevante Informationen auf der Kommandozeile abzufragen (Einzel-Host oder Subnetzclient, Linux oder Windows, zu verwendenden Präfix),
- ... die Konfigurationsdateien für den Server zu erstellen,
- ... alle Klientenkonfigurationsdateien in eine Archivdatei zu packen, die dem Klienten zur Verfügung gestellt wird.

Um das Debugging auf Seiten des Klienten zu erleichtern, kann das Skript `join-openvpn-sanity-check.sh` auf einem Linux-Subnetzclienten verwendet werden. Das Skript sammelt eine Reihe von Informationen über das System und versucht zu analysieren, ob diese Informationen Sinn machen. Es testet das Setup auf potentielle Fehler und gibt dem Nutzer einen Bericht aus. Das Skript nimmt keine Modifikationen vor, es ist "read-only". Es versendet des Weiteren keinerlei Informationen über das Netz, es zeigt diese lediglich auf der Konsole an, damit sie zum Debugging verwendet werden können.

```

erog:~/Texte/JOIN-Dokumente/Tunnelbroker/Skripte-und-Con [gargoyle:/etc/openvpn] # ./join-openvpn-sanity-check.sh
JOIN OpenVPN Tunnelbroker Client Sanity Check      Version 0.3
-----
Copyright (C) 2004 by Christian Strauf <strauf@uni-muenster.de>
                        <join@uni-muenster.de>
                        http://www.join.uni-muenster.de

Please note: this script only works for a Linux OpenVPN Client
that announces a /64 prefix to a local subnet.

Checking existence of a configuration file... OK
Checking client ID... (christian.strauf) OK
Checking existence and executability of up-script... OK
Checking existence of SSL certificate... OK
Checking existence of SSL key... OK
Checking permissions of SSL key... OK
Checking existence and executability of tls-verify script... OK
Checking existence of CA certificate... OK
Checking reachability of tunnelbroker over IPv4... OK
Checking IPv6 connectivity to ftp.ipv6.join.uni-muenster.de... OK
Checking if OpenVPN is running... OK
Checking that OpenVPN port is not occupied by other app... OK
Checking presence of "sysctl"... OK
Checking presence of "ip"... OK
Checking if IPv6 packet forwarding is enabled... OK
Checking if correct address+prefix has been configured for eth0... OK
Checking if tunnel device has correct IPv6 address... OK
Checking that there is only one IPv6 default route... OK
Checking default route is via tunnel device... OK
Checking for tun driver support in kernel... OK
Checking for IPv6 support in kernel :-)... OK

System infos:
-----
Kernel:      Linux 2.6.3-gentoo-r1 #1 SMP Fri Feb 20 15:08:52 CET 2004
Hardware:    i686 AMD Athlon(tm) Processor AuthenticAMD
OS:          GNU/Linux
OpenVPN version:
  OpenVPN 1.6_beta1 i686-pc-linux-gnu [SSL] [LZO] built on Jan 29 2004
  Copyright (C) 2002-2004 James Yonan <jim@yonan.net>
ip tool version: ip utility, iproute2-ss010824

Test summary:
-----
No. of tests: 21
PASSED tests: 21
FAILED tests: 0
SKIPPED tests: 0

[gargoyle:/etc/openvpn] # █

```

Abb.: Ausgabe von join-openvpn-sanity-check.sh

Beide Skripte können im Sinne der GNU GPL frei heruntergeladen und verbreitet werden. Es ist jedoch zu beachten, dass diese Skripte nur als Referenz für eine lokale Installation verstanden werden sollten und nicht als "All-in-one"-Paket.

Beachten Sie: Bitte lesen Sie zuerst die Dateien README und INSTALL, die Sie im Tarball der Skripte finden. Sie enthalten wichtige Informationen zur Installation.

- [join-tunnel-broker-scripts-current.tar.gz](#) (MD5-Checksumme: 8d57bd67221d182e29b48a552c427c50)
- [join-tunnel-broker-scripts-0.7.tar.gz](#) (MD5-Checksumme: 8d57bd67221d182e29b48a552c427c50)
- [join-tunnel-broker-scripts-0.6.tar.gz](#) (MD5-Checksumme: f38bbda571b988af2dee231b71a238f8)
- [join-tunnel-broker-scripts-0.5.tar.gz](#) (MD5-Checksumme: 4c81b3f49e1e2f49f2778ace3e86b170)
- [join-tunnel-broker-scripts-0.4.tar.gz](#) (MD5-Checksumme: 6141a53de362d1ca230326b170678172)
- [join-tunnel-broker-scripts-0.3.tar.gz](#) (MD5-Checksumme: dcbc4857d717407508e6f03693a2d156)

"9. Management"

Seitenanfang

10. Klienten–User–Guide

Die Installation eines OpenVPN–basierten IPv6–Tunnelbroker–Klienten besteht aus drei verschiedenen Schritten:

1. Anmeldung für den Service und Empfang von Konfigurationsfiles vom Tunnelbroker.
2. Installation des OpenVPN–Klienten.
3. Installation der OpenVPN–Konfigurationsdateien, die vom Tunnelbroker zur Verfügung gestellt wurden.

Die Anmeldung ist offensichtlich der nicht–technische Teil. Der technische Teil beginnt mit der Installation des OpenVPN–Klienten. Der Klient sollte in einer Version verwendet werden, die neuer als 1.6_rc3 ist. Er kann auf der Seite <http://openvpn.sourceforge.net/> herunter geladen werden. Bitte installieren Sie das Paket gemäß der Dokumentation, die auf der OpenVPN–Homepage heruntergeladen werden kann (Schnellanleitung: Unter Windows einfach Doppelklick auf die EXE–Datei; unter Linux sicherstellen, dass tun– und IPv6–Support im Kernel aktiviert ist und dass die OpenSSL–Devel– und LZO–Devel–Pakete installiert sind, so dann `./configure ; make ; make install` ausführen -- Details entnehmen Sie bitte der OpenVPN–Dokumentation).

Die Konfigurationsdateien, die durch den Tunnelbroker zur Verfügung gestellt wurden, müssen entweder nach `/etc/openvpn` (Linux) oder in das `config`–Unterverzeichnis Ihrer OpenVON–Installation (Windows) kopiert werden. Das Starten des Klienten ist trivial:

- `openvpn --config /etc/openvpn/<Ihre.ID>.conf` (Linux)
- lassen Sie OpenVPN entweder als Service laufen oder starten Sie den Klienten as `cmd.exe` (Windows)

["10. Klienten–User–Guide"](#)

[Seitenanfang](#)

11. Anhang

Wichtige Links:

- [OpenVPN–Homepage](#)

Dieses Dokument steht auch als [zweisprachiges PDF](#) (264KB) zur Verfügung.

["11. Anhang"](#)

[Back to top](#)