

DFNRoaming mit end-to-end Sicherheit

Roaming Logistik mit „Multiple Open Source tools“

Mit dem Aufbau eines VPN-Systems, wie es im vorliegenden Artikel beschrieben wird, ist es dem DFNRoaming Anwender möglich, nach erfolgreicher Authentifizierung mit seiner Heimat-IP-Adresse aus einem fremden Netz heraus auf die Netzwerkdienste seiner Einrichtung zuzugreifen, auch „schwierige“ Dienste wie der Zugriff auf eine PBX via (P)-Nat Router ist dann möglich.

Im folgenden werden die u.a. wichtigsten Open Source Tools (OpenVPN, bridge utils, ebtables, OpenVPN GUI, Nullsoft Scriptable Install System (NSIS)) vorgestellt, die zusammengenommen zu einem sehr leistungsstarken VPN System aufgebaut werden können. Die genannten Tools sind sicherlich bekannt, jedoch vielleicht nicht in der vorgestellten Anwendungsumgebung.

- **OpenVPN** ist eine robuste und flexible VPN (Virtual Privat Network) Lösung, welche auf gängige Betriebssysteme (Unix/Linux, Windows 2000/XP und MAC OSX) portiert wurde. Wahlweise via UDP oder TCP als Transportprotokoll kann ein mittels SSL verschlüsselter Tunnel über einen einzigen Port aufgebaut werden. Hierbei können zwei verschiedene Verfahren der Authentifizierung, basierend auf statischen Schlüsseln oder Zertifikaten, angewandt werden.
- **OpenVPN-GUI** ist ein grafisches Front-End für WINDOWS 2000/XP. Es ermöglicht dem Nutzer einen einfachen OpenVPN Zugang.
- **Bridge Utils** ist eine Sammlung von Tools, mit denen man recht einfach eine Layer 2 Bridge aufbauen kann. So ist es möglich u.a. auch mehrere verschiedene Netzwerk Interfaces (auch virtuelle) miteinander zu verbinden. Netzwerk Pakete werden so über ihre Ethernet-Adresse und nicht über ihre IP-Adresse weitergeleitet.
- **Ebtables** ist ein Programm, mit dem sich eine Bridging Firewall installieren lässt. Netzwerk Pakete können u.a. auf ihre Ethernet Adresse hin gefiltert werden.
- **Nullsoft Scriptable Install System (NSIS)** bezeichnet ein Open Source System unter WINDOWS 2000/XP, mit dem sich ein Client Installations-Paket erstellen lässt. Open Source Software lässt sich damit individuell vorkonfigurieren und effizient an autorisierte Anwender verteilen.

Bringt man nun die einzelnen OpenSource Projekte sinnvoll zusammen so erhält man ein leistungsstarkes VPN-System, mit dem sich nahezu alle VPN-Anwendungs-Szenarien sicher, flexibel, robust und anwenderfreundlich realisieren lassen. In unserem skizzierten Beispiel wird die Möglichkeit „routbare Adresse“ über einen VPN-Tunnel zu bridgen realisiert (s. Abbildung 1).

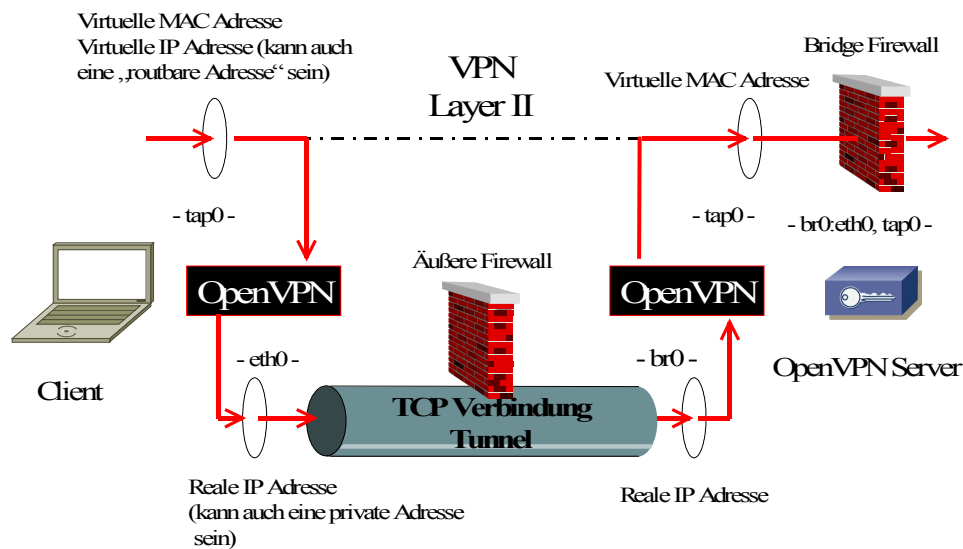


Abb. 1: Schematischer Aufbau eines OpenVPN-Anwendungsszenarios

Installation und Konfiguration eines OpenVPN-Servers

Damit der DFNRoaming Anwender es relativ einfach hat, den Dienst zu nutzen, sind einige wichtige Arbeiten auf der Seite des Rechenzentrums durchzuführen.

Die hier im folgenden beschriebenen Installations – und Konfigurationsarbeiten basieren auf einem Debian System mit einem 2.6. Linux Kernel und einem i686 Prozessor mit einer Ethernet Karte.

Damit der VPN-Server keine Einladung für „unliebsame Gäste“ wird, sollte das System besonders gehärtet werden, denn jeder autorisierte Client öffnet schließlich äußerst sensible Bereiche des lokalen Netzwerkes. Wenn möglich sollten außer SSH und OpenVPN alle anderen Dienste abgeschaltet werden!

Die Standard Prozedur für die Installation von openvpn-2.0.2 ist mit wenigen Schritten erledigt:

1. Herunterladen der Software unter openvpn.net
2. Auspacken
 - `gzip -dc openvpn-2.0.2.tar.gz | tar vfx -`
3. Kompilieren
 - `cd openvpn-2.0.2`
 - `./configure`
 - `make`
 - `make install`

Es könnte sein, dass die eine oder andere Library fehlt, z.B. LZO (compression library). Daher wird an dieser Stelle auf die OpenVPN Dokumentation verwiesen.

Beispiele für die Konfigurationsdatei des VPN-Server findet man unter ~/openvpn-2.0.2/sample-config-files, diese sollte unter /etc/openvpn abgelegt werden, hier ein Beispiel:

```
##### Start OpenVPN Config File #####
# Local ip address the server listen on
local 172.17.0.1

# TCP/UDP port the server listen on
# standard is 1194
port 1194

# What server do we want, TCP or UDP ?
proto tcp-server

# What virtual interface do we want?
# "dev tun" will create a routed IP tunnel,
# "dev tap" will create an ethernet tunnel.
dev tap0

# SSL/TLS root certificate (ca), certificate
# (cert), and private key (key). Each client
# and the server must have their own cert and
# key file. The server and all clients will
# use the same ca file.
ca /etc/openvpn/certs/ca.crt
cert /etc/openvpn/certs/openvpn-server.crt
key /etc/openvpn/certs/openvpn-server.key

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

# Configure server mode for ethernet bridging.
# Here you can use privat address space or a real ip address space!
server-bridge 172.17.0.1 255.255.0.0 172.17.0.3 172.17.255.255

# E.g. you want to give special clients a fixed ip address.
# If comon name of client's certificate is hugo.name.de, you should
# save a file called hugo.name.de in client-config-dir below.
# In hugo.name.de you can type in the requested ip address for the client:
# ifconfig-push 172.17.0.45 255.255.0.0
client-config-dir /etc/openvpn/ccd

# Sorry, only clients what have a --config-config-dir file can pass
ccd-exclusive
```

```
# If client is authenticated change bridge firewall and let the client pass through  
learn-address /etc/openvpn/callef
```

```
# If enabled, this directive will configure  
# all clients to redirect their default  
# network gateway through the VPN, causing  
# all IP traffic such as web browsing and  
# DNS lookups to go through the VPN  
push "redirect-gateway"
```

```
# push the DNS to client  
push "dhcp-option DNS 172.17.0.2"
```

```
# This config item must be copied to  
# the client config file as well.  
cipher AES-128-CBC # AES
```

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

```
# It's a good idea to reduce the OpenVPN  
# daemon's privileges after initialization  
user nobody  
group nogroup
```

```
##### End OpenVPN Config File #####
```

Die hier aufgeführten Konfigurationszeilen beschreiben nur einen Teil der Möglichkeiten des OpenVPN-Systems. Aus Gründen der eigenen Sicherheit sind die im Konfigurationsscript aufgeführten IP-Adresse aus dem privaten IP-Adressbereich. Diese müssten, entsprechend dem eigenen „routbaren“ Adressbereich angepasst werden.

Das Root-Zertifikat, Server Zertifikat sowie die Client Zertifikate lassen sich einfach durch Scripte generieren, die den Quellen der OpenVPN-Software beiliegen. Im Directory `~/openvpn.2.0.2/easy-rsa` befinden sich u.a. folgende Scripte, die nach Anpassung der Umgebungsvariablen in `~/openvpn.2.0.2/easy-rsa/vars`, gemäß den lokalen Gegebenheiten, der Reihe nach aufgerufen werden müssen:

- `~/openvpn.2.0.2/easy-rsa/build-ca <directory name>`
- `~/openvpn.2.0.2/easy-rsa/build-key-server <server name>`
- `~/openvpn.2.0.2/easy-rsa/sign-key <server name>`
- `~/openvpn.2.0.2/easy-rsa/build-key <client name>`
- `~/openvpn.2.0.2/easy-rsa/sign-key <client name>`

Bevor der Server gestartet werden kann, wird explizit das virtuelle Interface „tap0“ generiert. Das erledigt das Kommando `openvpn --mktun --dev tap0`.

Gestartet wird der Server mit `openvpn --config /etc/openvpn/openvpn.conf --daemon`.

Es wird an dieser Stelle empfohlen, ein sogenanntes Start-Stop-Script in `/etc/init.d` abzulegen, so kann beim Hochfahren der Maschine gleich der VPN-Server automatisch mitgestartet werden.

Bridging the gap

Mit dem Kommando `apt-get install bridge-utils` wird die Ethernet Bridging Software installiert. Nun stehen dem Admin die mächtigen Ethernet Bridging Tools zur Verfügung, und die Ethernet Bridge kann wie folgt aufgebaut werden:

- `brctl addbr br0` (generiert das virtuelle Bridge Interface `br0`)
- `brctl addif br0 eth0` (fügt das LAN Interface `eth0` der Bridge `br0` hinzu)
- `brctl addif br0 tap0` (fügt das virtuelle OpenVPN Interface `tap0` der Bridge `br0` hinzu)
- `ifconfig eth0 0.0.0.0 promisc up` (versetzt das Interface `eth0` in den promisc mode)
- `ifconfig tap0 0.0.0.0 promisc up` (versetzt das Interface `tap0` in den promisc mode)
- im Prinzip können beliebig viele Interfaces der Bridge hinzugefügt werden, denkbar ist auch ein UDP-VPN-Server, z.B. auf `tap1`
- `ifconfig br0 172.17.0.1 netmask 255.255.0.0 broadcast 255.255.255.255` (konfiguriert das virtuelle LAN/Bridge Interface `br0`)
- `route add default gw 192.168.0.1 br0` (setzt das Default Gateway von `eth0` auf `br0` um)

Ist es erforderlich, dass das Interface `eth0` mehreren unterschiedlichen Bridges zugeordnet werden muss, z.B. wenn der VPN-Server von unterschiedlichen IP-Adressen erreichbar sein muss, so kann auch ein virtuelles Bridge Interface `br0:1` auf dem virtuellen Bridge Interface `br0` konfiguriert werden. Mit der Eingabe `ifconfig br0:1 172.17.0.155 netmask 255.255.0.0 broadcast 255.255.255.255` erhält man das gewünschte Interface, auf dem all die Ethernet Pakete „gebridged“ werden, die auch auf dem Interface `br0` „gebridged“ werden. Diese Vorgehensweise (Konfiguration eines `br0:1` Interfaces) ist in der einschlägigen Linux-Literatur bisher nicht dokumentiert!

Bridge Firewall

Damit beim „Bridging“ nur die Ethernet Frames gebridged werden, die für den Client auch notwendig sind, ist die Installation einer Ethernet Bridge Firewall erforderlich. Das Programm `eatables` ist ein solches Firewall-Programm. Durch `apt-get install eatables` wird das Programm installiert.

Nachdem ein VPN-Client sich erfolgreich authentifiziert oder sich abgemeldet hat, wird gemäß der Konfiguration des OpenVPN Servers (s.o. Server Konfigurations-Script) umgehend ein Script angestoßen, welches die Bridge Firewall konfiguriert. Durch die OpenVPN Direktive „`learn-address`“ (s.o.) wird der Aufruf eines solchen Scriptes festgelegt. Beim Ausführen des Scriptes bekommt dieses in der Regel drei Parameter mit:

1. Operand: `add`, `update` oder `delete` je nach Status des Clients
2. Adresse: Im Falle eines Bridging Servers bekommt das Script die Ethernetadresse des Clients als Parameter übergeben. Handelt es sich um einen Routing-VPN-Server, so erhält das Script eine IP-Adresse mitgeliefert.
3. Common Name: Bei den Operanden `add` und `update` bekommt das Script zusätzlich den Common Name als Parameter mitgeliefert.

Um z.B. nach erfolgreicher Authentifizierung eines Clients die Bridge Firewall entsprechend zu konfigurieren reichen folgende fünf Regeln:

- `ebtables -t filter -I FORWARD 1 --logical-link br0 -s <clients-mac-address> -p IP --ip-source <clients-ip-address> -j ACCEPT #` lässt alle IPv4 Pakete durch, die die Ethernet Adresse (`clients-mac-address`) und die IP-Adresse (`clients-ip-address`) als Absender Adresse enthalten
- `ebtables -t filter -I FORWARD 1 --logical-link br0 -d <clients-mac-address> -p IP --ip-destination <clients-ip-address> -j ACCEPT #` lässt alle IPv4 durch, die die Ethernet Adresse (`clients-mac-address`) und die IP-Adresse (`clients-ip-address`) als Absender Adresse enthalten
- `ebtables -t filter -I FORWARD 1 -p ARP --arp-ip-src <clients-ip-address> -j ACCEPT #` lässt alle Broadcast Pakete des Clients durch
- `ebtables -t filter -I FORWARD 1 -p ARP --arp-ip-dst <clients-ip-address> -j ACCEPT #` lässt alle Broadcast Pakete durch, die für den Client bestimmt sind
- `ebtables -t filter -A FORWARD -j DROP #` diese Regel muss als letztes stehen, verwirft alle unbekannt Pakete

Ausgestattet mit diesen Information lässt sich nun relativ leicht ein den individuellen Gegebenheiten angepasstes Bridge Firewall Script erstellen.

Individuelles Client OpenVPN Packet

Möchte man individuelle Clients (Client Installer) für seine Nutzer, die unter WINDOWS 2000/XP arbeiten vorkonfigurieren, so eignet sich am besten das Software-Tool NSIS dazu. Dieses Software Tool erhält man unter folgendem Link

<http://www.openvpn.se/files/nsis/nsis205.exe> . Nach dem Download der Software einfach starten und den Installationshinweisen folgen. Nun benötigt man noch das „Source Paket“ zu OpenVPN-GUI, welches man unter, z.B. folgendem Link

http://www.openvpn.se/files/install_packages_source/openvpn_install_source-2.0.2-gui-1.0.3.zip „downloaded“ und anschließend auf der gleichen Windows Maschine, z.B. in C:\OpenVPN-Install auspackt, auf der man zuvor das Software Tool NSIS installiert hat.

Weiterführende Informationen gibt es unter http://openvpn.se/files/howto/openvpn-howto_roll_your_own_installation_package.html .

Die Grundinstallation, um z.B. vorkonfigurierte, individuelle Client Installer zu generieren ist damit gegeben. Um einen Client Installer für seine Nutzer zu erstellen, geht man wie folgt vor:

Zunächst erstellt man ein Konfigurations-Script für den OpenVPN Client, z.B. unter C:\OpenVPN-Install\openvpn\config. Hier ein Beispiel:

```
##### Begin Client Side configuration file Office.conf #####
# vpn server to connect
remote 172.17.0.1

# local tunnel device
dev tap

# port to establish vpn connection on
port 1194
```

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

# Where to find certificates?
ca C:/Programme/OpenVPN/my-certs/ca.crt
cert C:/Programme/OpenVPN/my-certs/openvpn-client.crt
key C:/Programme/OpenVPN/my-certs/openvpn-client.key

# Select a cryptographic cipher.
# If the cipher option is used on the server
# then you must also specify it here.
cipher AES-128-CBC # AES

# enable LZO compression
comp-lzo

##### End Client Side configuration File Office.conf
```

Damit die Zertifikate auch später an der richtigen Stelle gefunden werden, generiert man das Verzeichnis: C:/Programme/OpenVPN/my-certs/ und speichert dort die individuellen Zertifikate (ROOT-CA-Cert, Client-Cert, Client-Key) ab. Client Zertifikate werden auf der Server Maschine generiert (s. Kapitel **Installation und Konfiguration eines OpenVPN-Servers**) und dann in das dafür festgelegte Verzeichnis auf die Client Installer Maschine geschrieben.

Als nächstes editiert man das File, z.B. C:\OpenVPN-Install\openvpn-gui.nsi wie folgt:

- Im Abschnitt:
Section "OpenVPN GUI" SecGUI
trägt man folgende Zeilen ein:
SetOutPath "\$INSTDIR\My-CERTS"
File "\${HOME}\My-CERTS\ca.crt"
File "\${HOME}\My-CERTS\openvpn-client.crt"
File "\${HOME}\My-CERTS\openvpn-client.key"

CreateDirectory "\$INSTDIR\My-CERTS"
- Im Abschnitt:
Section "Uninstall"
trägt man an geeigneter Stelle folgende Zeilen ein:
Delete "\$INSTDIR\My-CERTS\ca.crt"
Delete "\$INSTDIR\My-CERTS\openvpn-client.crt"
Delete "\$INSTDIR\My-CERTS\openvpn-client.key"

RMDir "\$INSTDIR\My-CERTS"

Nach dem Speichern der Ergänzungen in C:\OpenVPN-Install\openvpn-gui.nsi „klickt“ man mit der rechten Maustaste auf das File openvpn-gui.nsi und startet den „NSIS Compiler“ aus dem aufgeklappten Menü. Nun wird ein Client Installer, mit der Bezeichnung openvpn-

2.0.2-gui-1.0.3-install.exe erstellt. Diesen Client Installer kann man an den Nutzer aushändigen. Der Nutzer muss dann nur noch den vorkonfigurierten Client installieren.

Einfach für den Nutzer, aber dennoch sicher

Für den Nutzer gestaltet sich die Installation einfach. Mit einem Doppelklick auf das Client Installer Icon wird der Installations-Prozess gestartet. Der Nutzer folgt dann den Anweisungen des Installer Programms. Nach der Installation befindet sich ein neues Icon auf der rechten Seite der WINDOWS-Start-Leiste, das OpenVPN-GUI Frontend (siehe Abbildung 2). Klickt der Nutzer mit der rechten Maustaste auf das neue Icon, so kann er aus dem sich öffnenden Pop-up-Menü über die Auswahl „Connect“ die Verbindung zum VPN Server aufbauen. Über die Zuweisung der IP-Adresse (das kann eine private oder eine routbare IP-Adresse sein) vom VPN-Server ist dann der VPN Client mit dem G-WiN über einen verschlüsselten Tunnel verbunden.

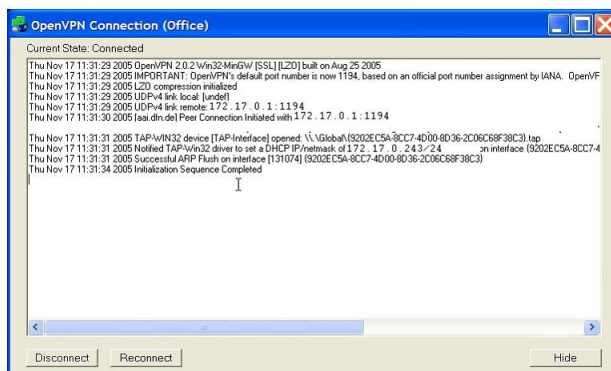


Abb. 2: Mit 2 „Maus-Klicks“ lässt sich ein sicherer Tunnel zum VPN-Server etablieren