Zusammenlegung zweier unabhängiger LANs mithilfe von VPN über ein Trägernetz

©Janßen, Meyer

Betreuer: Prof. Dr. Gerhard Kreutz

8. Juli 2010

Inhaltsverzeichnis

1 Einleitung

In der heutigen Zeit sind Internet und der damit verbundene Datenaustausch nicht mehr wegzudenken. Viele Anwender gehen dazu über, Schnittstellen, wie FTP-, Streaming- und SMB-Server aufzusetzen, um Daten auszutauschen. Für reinen Datenaustausch kann man sich dieser Methoden einfach bedienen. Geht es aber darum, an sensiblen Projekten und von verschiedenen Standorten aus zu arbeiten, kommt man mit dem reinen Portforwarding nicht weiter. Hier soll eine Alternative auf der Basis eines virtuellen privaten Netzwerkes dargestellt werden.

1.1 Motivation

Virtuelle private Netzwerke werden schon seit geraumer Zeit für vereinfachten Datenaustausch verwendet. Es wird jedoch eine stabile, nicht kommerzielle Variante gesucht, die ausnahmslos alle Netzwerkgeräte unterstützen kann, ohne dass jedes Gerät dafür konfiguriert werden muss.
Außerdem ist die Tatsache sehr störend, dass manche Programme oder Anwendungen virtuelle Geräte, welche bei einer VPN-Verbindung erstellt werden, nicht richtig erkennen oder nutzen. Die Lösung dieses Problems wird durch die Verbindung der beiden privaten Netzwerk zu einem gemeinsamen Netz erreicht.

1.2 Problemstellung

Ausgehend von den genannten Tatsachen wurden die folgenden Rahmenbedingungen für das Projekt festgelegt:
Es gilt eine stabile, möglichst wartungsfreie Verbindung zwischen den beide Netzen herzustellen. Bei Ausfall eines Netzes soll das Andere noch problemlos funktionieren.
Es soll möglich gemacht werden, auf Netzwerkgeräte des jeweils anderen Netzes ohne Umwege zuzugreifen.

2 Ansatz

Wir entschieden uns für eine Variante, bei der auf beiden Seiten eine Bridge zwischen der lokalen und der virtuellen Schnittstelle des VPNs erstellt wird. Eine geroutete (vgl. 3.2) Verbindung kommt nicht in Frage, da auf der einen Seite der Router nicht dafür konfiguriert werden kann. Es wurden zwei Rechner ausgewählt, die im jeweiligen Netz dauerhaft online sind. Aus

Sicherheitsgründen wird das Betriebssystem Linux verwendet. Außerdem erweist sich Linux, bei Netzwerkeinrichtungen, um einiges umgänglicher als Microsoft Windows. Als Distribution wird Gentoo verwendet. Konfigurationsdateien und Systembedienungshinweise in dieser Dokumentation werden sich auf Gentoo-Linux beschränken. Sollten einige Beispiele bei anderen Distributionen grundlegend anders sein, wird das Äquivalent von Debian-Linux als Beispiel dargestellt.

2.1 Netzaufbau

Der folgende Netzwerkplan zeigt die beiden vorhandenen Netze vor dem Aufbau des VPNs.

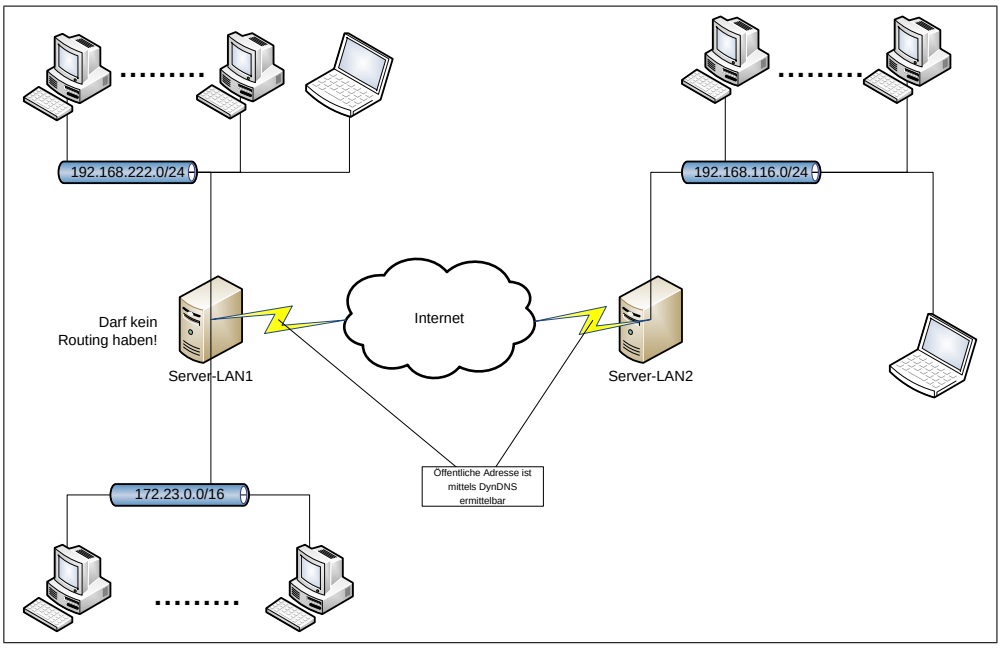

Abbildung 1: Netzplan vor der Umstrukturierung

Um sich in der IP-Adress-Vergabe nicht beschränken zu müssen, haben wir die Subnetmaske von 255.255.255.0 auf 255.255.254.0 abgeändert. Damit die Netzwerke auch im selben Subnet sind, musste das LAN 1 von 192.168.116.0/24 auf 192.168.223.0/23 und das LAN 2 von 192.168.222.0/24 auf 192.168.222.0/23 abgeändert werden.

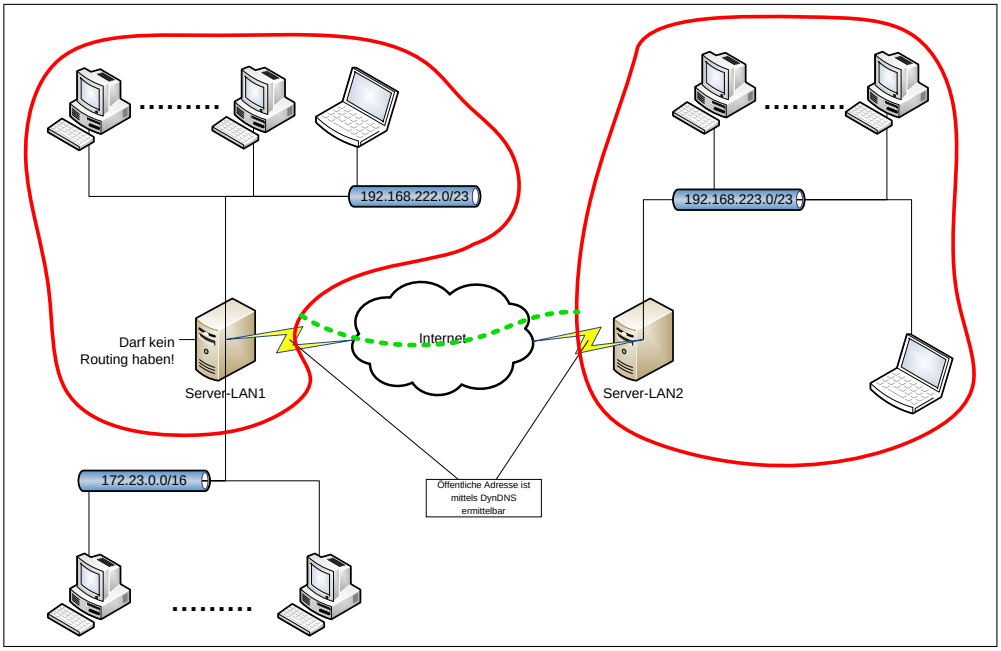

Abbildung 2: Netzplan nach der Umstrukturierung

Die Adressvergabe wird so geregelt, dass, wenn man am Standort LAN 2 eine IP bezieht, eine aus dem Pool 192.168.223.0/23 erhält und äquivalent am Standort LAN 1 eine aus dem Pool 192.168.222.0/23. Als Routing- und DNS-Informationen werden jeweils die des aktuellen Standortes übergeben.

3 VPN Grundlagen

VPN wird dazu benutzt, um in einem Netzwerk ein unabhängiges, eigenes Netzwerk zu erstellen. Dieses kann verschlüsselt sein.

3.1 VPN-Unterschiede

Es gibt mehrere, technische und logische Arten von VPNs.

Logische Unterschiede:

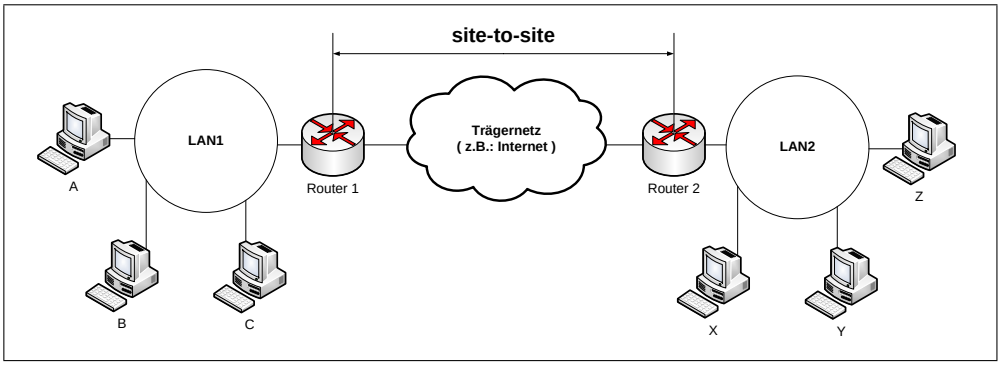

Abbildung 3: site-to-site

Dieser Aufbau verbindet zwei benachbarte Netze über ein Trägernetz miteinander. Diesen nennt man *site-to-site*. Die Teilnehmer der Netze haben somit die Illusion, direkt in einem Netz verbunden zu sein.

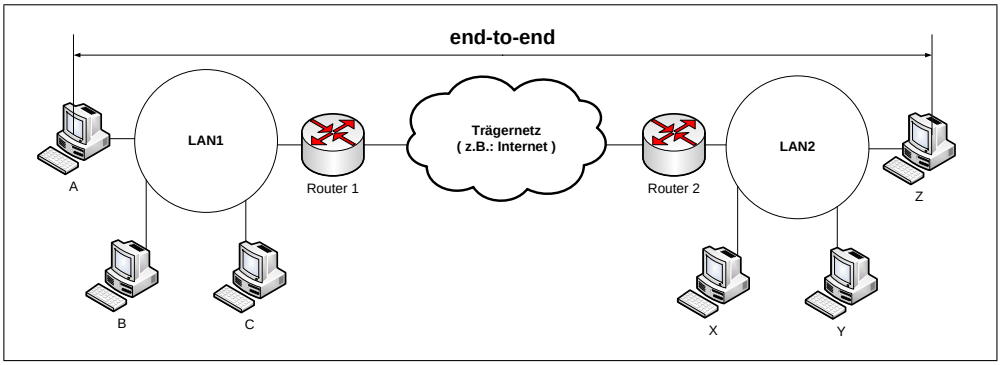

Abbildung 4: end-to-end

Eine andere Möglichkeit ist, die sogenannte *end-to-end* Verbindung. Jeder

Teilnehmer in diesem Aufbau, hat eine eigene Verbindung mit dem VPN-Server. Auch hier können alle Teilnehmer so miteinander kommunizieren, als wären sie physikalisch in einem Netzwerk. Es kann allerdings am Server eingestellt werden, ob die Clients miteinander kommunizieren dürfen. Ist dies nicht der Fall, können Clients nur auf die Ressourcen zugreifen, die der Server freigibt.

Diese beiden logischen Aufbauten können auch gemischt werden.

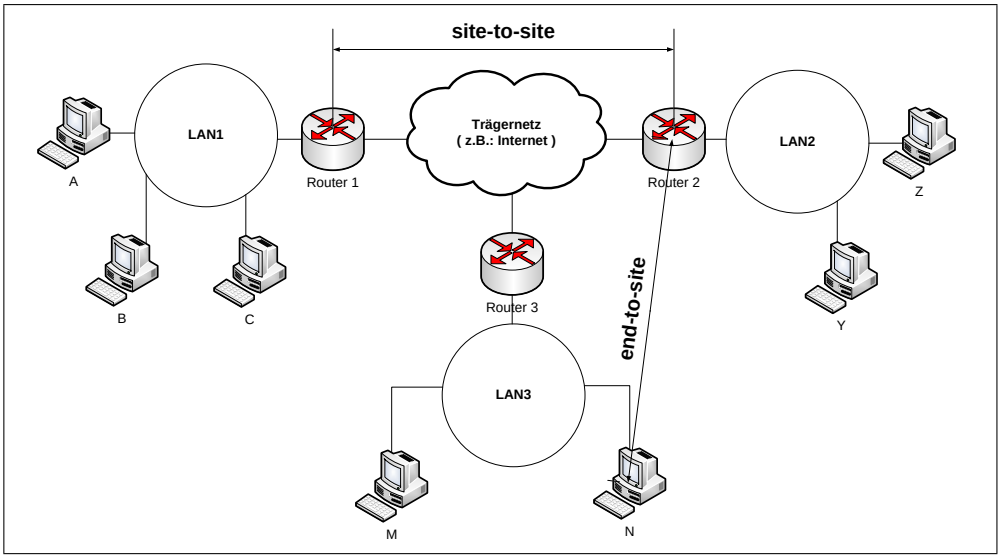

Abbildung 5: Gemischter Aufbau

Technische Unterschiede:
Die zwei wichtigsten und meist benutzten Arten von VPNs, ist der Aufbau mit IPsec und TLS/SSL.
Der Unterschied ist, dass IPsec, im Gegensatz zu TLS/SSL, direkt in der Vermittlungsschicht (Schicht 3 des OSI-Referenzmodels) arbeitet. TLS/SSL arbeiten oberhalb der Transportschicht (>Schicht 4 des OSI-Referenzmodel).

3.2 Verbindung vom Netzwerk zum VPN

Hier entscheidet sich, wo und womit die VPN-Verbindung aufgebaut wird. Ist es der Router an sich, so wird die erste Methode, das Routing verwendet. Hierbei wird in der Konfigurationsdatei als Device-Type *tun* verwendet. Wird die VPN-Verbindung hinter einem Router erstellt, wird im Allgemeinen eine Bridge verwendet. Für eine Bridge muss die virtuelle Netzwerkschnittstelle vom Typ *tap* sein.

Bei dem geroutetem VPN, wird das virtuelle VPN-Netzwerkgerät zu dem Hardwaregerät, oder einem anderen, virtuellen Netzwerkgerät geroutet. Dieses geschieht auf der Vermittlungsschicht (Schicht 3 des OSI-Referenzmodell), mithilfe von sogenannten Routingtabellen. Der große Nachteil ist, dass die angeschlossenen Geräte sich nicht "sehen" können.
Die Bridge hingegen arbeitet auf der Sicherungsschicht (Schicht 2 des OSI-Referenzmodell). Hier wird eine virtuelle Bridge erstellt (brctl addbr <name >).
Mit brctl addif <brname> <ifname> wird die reelle oder virtuelle Netzwerkschnittstelle *ifname* zur Bridge *brname* hinzugefügt (vgl. 7.4).

4 Implementierung

Wir haben uns für die freie VPN-Software, OpenVPN entschieden, da diese auf den meisten Betriebssystemen installierbar ist, beziehungsweise auf vielen zu erwerbenden Netzwerkgeräten entweder vorinstalliert oder nachträglich zu installieren ist. Dieser Umstand macht dieses VPN auch fähig, einzelne site-to-end Verbindungen zu erlauben.
Die Bridges werden über die bridge-utils der verwendeten Distributionen, hier auf Client- und Serverseite Gentoo, eingestellt. Da es sich bei OpenVPN um eine Anwendung handelt, die auf der Serverseite von außen immer erreichbar sein muss, wird empfohlen, immer die aktuellsten Versionen des Programms und die zur Verschlüsselung genutzten SSL-Libraries installiert zu haben.

4.1 TCP/UDP

OpenVPN bietet an, entweder das Protokoll TCP oder UDP zur Übertragung zu nutzen. In unserer Testphase fiel uns auf, dass die Verbindung mit UDP nach einigen Minuten abbrach und das VPN sich neu verbinden musste. Unsere Überprüfung hat ergeben, dass die Datenpakete der VPN-Verbindung in falscher Reihenfolge eintrafen. Aufgrund der von OpenVPN verwendeten Verschlüsselung (MAC), verlangte der OpenVPN-Server den erneuten Austausch der Schlüssel.
Um dieses Phänomen zu testen, haben wir eine VPN-Verbindung über UDP in einem LAN aufgebaut. Diese Verbindung blieb stabil und es wurde keine erneute Verbindung erzwungen. Anfragen bei dem ISP haben uns bei dem Problem nicht weitergeholfen. Aus diesem Grund stellt unser System die Verbindung über das TCP her.
Im Gegensatz zu UDP baut TCP vor Übertragen von Daten eine Verbindung auf. Deshalb wird TCP zu den verbindungorientierten Protokollen gezählt. Durch den sogenannten *HandShake* wird die Verbindung etabliert. Dieses Verfahren sichert den Datenstrom so ab, dass er beim Empfänger fehlerfrei zusammengesetzt wird. Nachteil diese Verfahrens ist, dass durch Zusatzinformationen und dem Aushandeln der Verbindung, der Bandbreitenverbrauch zunimmt, also die Verbindung langsamer wird.

4.2 Konfiguration

Zunächst wird auf beiden Seiten, der Einfachheit halber, eine virtuelle TAP-Schnittstelle, welche beim Booten gestartet wird, erstellt.

```
# cd /etc/init.d/
# ln -s net.lo net.vpn
# rc-update add net.vpn default
```

Diese virtuelle Schnittstelle erfordert, dass sie in die Netzwerkkonfigurationsdatei eingestellt wird.
Ebenso werden auf Client- und Server-Seite die Bridges in die Netzkonfig. eingetragen und entsprechend beim Boot aktiviert. Konfigurationsbeispiele finden sich im Anhang. Die Bridge wird genau wie die VPN-Schnittstelle zum Bootvorgang hinzugefügt

```
# rc-update add net.xbr0 default
```

4.2.1 Serverseite

Auf der Serverseite werden in diesem Fall alle Schlüssel generiert. Das OpenVPN-Paket liefert einige Scripte zur Zertifikatserstellung mit. Im Konfigurationsordner erstellen wir zunächst den Ordner für die Key-Erstellung. Danach wird die Datei mit den benötigten Umgebungsvariablen kopiert und die Scripte symbolisch verlinkt.

Listing 1: Erstellen der Umgebung zur Generierung der Keys

```
cd /etc/openvpn
mkdir easy-rsa
cd easy-rsa
cp -v /usr/share/openvpn/easy-rsa/vars ./
# '/usr/share/openvpn/easy-rsa/vars' -> './vars'
ln -sv /usr/share/openvpn/easy-rsa/{build-ca|build-key|build
    -dh|build-key-server|clean-all|pkitool|openssl.cnf} .
```

Die Datei *vars* wird nun, den Ansprüchen entsprechend, editiert.

Listing 2: /etc/openvpn/easy-rsa/vars

```
export EASY_RSA="`pwd`"
export KEY_CONFIG="$EASY_RSA/openssl.cnf"
export KEY_DIR="$EASY_RSA/keys"
export KEY_SIZE=2048   # 1024 waere als Key-Groesse etwas zu
    unsicher, 4096 waere paranoid.
export CA_EXPIRE=9999
export KEY_EXPIRE=9999
export KEY_COUNTRY="DE"
export KEY_PROVINCE="Niedersachsen"
```

```
9  export KEY_CITY="Emden"
10 export KEY_ORG="TestSystem"
11 export KEY_EMAIL="root@testnetz"
```

Als nächster Schritt wird die Certificate-Authority erstellt. Dieser Schlüssel ist der wichtigste im gesamten Verschlüsselungsverfahren. Je nachdem wie sicher das VPN-System sein muss, wird dieser Schlüssel auf einem externen Server ausgelagert. Die Schlüsseldatei kann auch mit einem Passwort gesichert werden (Option: --pass). Da sich diese VPN-Installation aber auf einem komplett verschlüsseltem System befindet, wird der Private-CA-Key ohne Passwort auf diesem Rechner gespeichert.

Listing 3: CA-Key-Erstellung

```
1 cd /etc/openvpn/easy-rsa
2 source vars # nach jeder |"Anderung in "vars" muss die Datei
      "gesourced" werden, dass hei|ss t die
     Umgebungsvariablen werden geupdatet.
3 ./clean-all # evtl. vorhandene Zertifikatsdateien in KEY_DIR
      werden gel|"oscht!!
4 ./build-ca
5 # "Organizational Unit Name" -> leer lassen
6 ls /etc/openvpn/easy-rsa/keys/ca* # ca.crt (Zertifikat) ca.
     key (Private-Key)
7 openssl x509 -in /etc/openvpn/easy-rsa/keys/ca.crt -text -
     noout | less # Zertifikat prÃ¼fen
8 openssl rsa  -in /etc/openvpn/easy-rsa/keys/ca.key -text -
     noout | less # Key anschauen
```

Bei der Erstellung des Server-Zertifikates ist es wichtig, einen *individuellen* Common Name zu vergeben, der die VPN-Verbindung eindeutig kennzeichnet. Hier wird *xyz-server* verwendet. Der Common Name taucht in der Client-Konfigurationsdatei wieder auf und dient als eindeutige Identifizierung um *man-in-the-middle-Angriffe* zu verhindern.

Listing 4: Server-Key-Erstellung

```
1 ./build-key-server xyz-Server # xyz-Server: Common Name des
     Zertifikats
2 # "challenge password" -> leer lassen
3 # "An optional company name" -> leer lassen
4 # "Sign the certificate? [y/n]" -> y
5 # "1 out of 1 certificate requests certified, commit?" -> y
6 rm /etc/openvpn/easy-rsa/keys/server.csr # wird nicht
     benoetigt, da Key/Zertifikat zugleich erstellt werden
7 ls /etc/openvpn/easy-rsa/keys/server.*   # server.crt server
     .key
8 # Ersetze "server" durch den gewaehlten Common Name (z.B. "
     xyz-Server").
```

Die Client-Schlüssel werden folgendermaßen erstellt:

Listing 5: Client-Key-Erstellung

```
1 ./build-key client1 # Angaben analog zum Server-Zertifikat
2 rm /etc/openvpn/easy-rsa/keys/client1.csr # wird nicht
    benoetigt, da Key/Zertifikat zugleich erstellt werden
3 ls /etc/openvpn/easy-rsa/keys/client1.* # client1.crt
    client1.key
```

Als Nächstes wird der Diffie-Hellman-Parameter erstellt. Diesen Key benötigt OpenVPN für die sichere Aushandelung der Verschlüsselung.

Listing 6: Diffie Hellman Parameter - Erstellung

```
1 ./build-dh # dauert ca. 5-1000 min., je nach KEY_SIZE in "
    vars"
2 ls /etc/openvpn/easy-rsa/keys/dh* # dh2048.pem (bzw. dh1024.
    pem, je nach KEY_SIZ)
```

Ebenso arbeitet dieses System mit einem statischen Key, der vor dem eigentlichen Schlüesselaustausch übertragen wird. Dieser wird am Ende zusammen mit den Client-Keys auf den Client übertragen.

Listing 7: PreSharedKey-Erstellung

```
1 openvpn --genkey --secret /etc/openvpn/easy-rsa/keys/ta.key
2 ls /etc/openvpn/easy-rsa/keys/ta.key # ta.key
```

Nun müssen die Schlüsseldateien nur noch auf den Client übertragen werden. Dies sollte auf einem möglichst sicheren Weg passieren, wie in dem folgenden Beispiel über scp.

Listing 8: Verteilen der Schlüssel

```
1 cd /etc/openvpn/easy-rsa/keys
2 scp ca.crt ta.key client1.* 192.168.223.10:/etc/openvpn/
    testnetz/
```

Als Letztes muss für den Server die Konfigurationsdatei geschrieben werden. Im Anhang (7.1) findet sich die komplette Datei.
Die Variablen *local* und *port* bezeichnet die IP-Adresse und den dazugehörigen Port, an der OpenVPN auf Verbindungen wartet.
proto bezeichnet das verwendete Protokoll zur Übertragung, es sind *tcp* und *udp* möglich. UDP ist, da es auf einen *HandShake* verzichtet, schneller als TCP, jedoch auch fehleranfälliger. Um die Verbindung zu garantieren wird in diesem Fall TCP genutzt.
Der *server-bridge*-Parameter ist für die IP-Vergabe zuständig. Hier muss der DHCP-Server entsprechend konfiguriert sein, damit keine Komplikationen entstehen.

cipher stellt den Daten-Verschlüsselung-Algorithmus ein. OpenVPN bietet drei verschiedenen Algorithmen an. AES-256-CBC ist, aufgrund der verwendeten Algorithmen und Länge, der sicherste.
comp-lzo aktiviert die Kompression, diese ist nur bei langsameren Verbindungen wie DSL zu empfehlen Bei einer LAN-VPN-Verbindung verbraucht es nur unnötig Ressourcen.

Nun darf der OpenVPN-Dienst gestartet werden. Damit Verbindungen von außen an den VPN-Server gelangen können, muss eine Port-Weiterleitung im Router eingerichtet werden. Diese ist in diesem Fall:

```
1 iptables -t nat -A PREROUTING -p \mbox{TCP} --dport 1234 -i
    ppp0 -j DNAT --to 192.168.222.22:1234
```

Listing 9: Starten des VPN-Daemons

```
1 /etc/init.d/openvpn start
```

4.2.2 Clientseite

Die Client-Konfiguration unterscheidet sich nur wenig von der Serverkonfiguration. Wir gehen davon aus, dass die benötigten Keys und Zertifikate wie in den vorhergehenden Schritten erstellt und übertragen wurden.
In der openvpn.conf wird *local* mit *remote* ausgetauscht und anstatt der lokalen IP-Adresse wird die entfernte, oder der Hostname des OpenVPN-Servers eingetragen. Durch den Parameter *client* wird OpenVPN im Clientmodus gestartet.
Besonders bei Verbindungen zu dynamischen IP-Adressen ist die Option *resolv-retry infinite* wichtig. Sollte die Verbindung nach dem, durch *keepalive* angegeben Zeitraum nicht wieder hergestellt sein, wird versucht die IP zu dem Hostnamen neu aufzulösen.
ns-cert-type server und *tls-remote server* sind die Einstellungen für das Serverzertifikat und für die Art der Authentifizierung.
Die komplette Konfiguration ist im Anhang (vgl. 7.2) zu finden.

Nun wird der Daemon auf der Clientseite gestartet und die Verbindung wird hergestellt. Bei Erfolg sollte jeder Rechner in der Lage sein, einen oder mehrere Rechner aus dem anderen Netz zu pingen.

Listing 10: Starten des VPN-Daemons

```
1 /etc/init.d/openvpn start
```

5 Ausblick - Dienste in zusammengelegten Netzen DHCP

Das "Dynamic Host Configuration Protocol" wird in beiden Netzwerken verwendet. Ein DHCP-Server kann auf Anfrage für beliebige Netzwerkgeräte IP-Adressen, Routing-, DNS- und weitere Informationen bereitstellen.
In unserem Fall haben beide Netze eine Internetverbindung über einen Router an jedem Standort. Logisch betrachtet haben alle Geräte in beiden Netzen die Möglichkeit, einen oder beide Router als Verbindung mit dem Internet zu nutzen. Um zu verhindern, dass sämtliche Internetverbindungen über das VPN geschickt werden, verteilen die DHCP-Server in dem System die entsprechnden Informationen an die Clients, damit diese nur die lokale Internetverbindung nutzen.
Das Problem dabei ist, dass immer die Informationen vom schnelleren DHCP-Server verwendet werden und nicht nur die vom lokalen Server.
Wir haben das Problem mittels *ebtables* gelöst. Die Ports 67 (DHCP-Server-Port) und 68 (DHCP-Client-Port) wurden jeweils auf der Bridge beider Standorte gesperrt. Dieses funktioniert aber nur, wenn der DHCP-Server-Dienst jeweils auf dem Server mit der VPN-Bridge läuft. Versuche, auf anderen Geräten einen DHCP-Server zu nutzen, ergaben, dass Anfragen und Antworten wieder ungeblockt über das VPN übertragen wurden.
Dieses Verhalten kann folgendermaßen erklärt werden: ebtables ist nur in der Lage, Pakete vor Betreten oder nach Verlassen der Bridge zu filtern. Werden die DHCP-Antwortpakete auf dem System generiert, welches die VPN-Verbindung aufbaut, so kann ebtables das Paket analysieren und verwalten. Pakete, die an die reelle Netzwerkschnittstelle der Bridge kommen, können nur *vor* der Bridge abgefangen werden, da die DHCP-Information vor dem Austreten durch OpenVPN verschlüsselt wird. Dadurch hat ebtables keine Möglichkeit, das Paket komplett auszulesen.
In unserem Fall ist die Filterung der DHCP-Pakete vor der Bridge nicht möglich, da noch eine zweite, virtuelle Schnittstelle mit der Bridge verbunden ist. Über diese Schnittstelle müssen DHCP-Informationen übertragbar sein.

6 Fazit

Zusammenfassend ist zu sagen, dass OpenVPN eine stabile und kostengünstige Methode ist, um zwei oder mehr Netzwerke über ein Trägernetz – wie das Internet – zusammen zu legen. Im Laufe unseres Projekts haben sich einige, aber lösbare Schwierigkeiten bei dem Bridging ergeben.
Durch die Zusammenlegung mehrerer Netzwerke, werden viele Sicherheitslücken verhindert, die durch Portforwarding vieler Ports entstehen. Innerhalb des VPNs können die Protokolle und Dienste, wie FTP, SMB, und SSH weiter genutzt werden, ohne dass von außerhalb jemand darauf Einfluss nehmen kann.
Portforwarding kann natürlich nicht vollkommen ausgeschlossen werden. Jedoch benötigt man nicht für jeden Dienst eine Portweiterleitung, die potenziell gefährlich sein kann. Wartung und Fehlerbehebung sind durch eine übersichtliche Logdatei einfach und zeitsparend.
Je nach Anwendungsgebiet, wägt der Benutzer ab, ob oder wie sicher die Verschlüsselung sein soll. Die Stärke der Verschlüsselung steht immer im Zusammenhang mit der zur Verfügung stehenden Bandbreite und Rechenleistung.
Durch unser Projekt wurde uns nochmals deutlich, dass Open Source Produkte, wie OpenVPN, immer als Alternative zu kommerziellen Produkten bedacht werden sollten.

7 Anhang

7.1 openvpn.conf - Serverseite

Listing 11: /etc/openvpn/openvpn.conf

```
local 192.168.222.22
port 1234
proto tcp
dev vpn
dev-type tap
ca /etc/openvpn/easy-rsa/keys/ca.crt
cert /etc/openvpn/easy-rsa/keys/server.crt
key /etc/openvpn/easy-rsa/keys/server.key  # This file
    should be kept secret
dh /etc/openvpn/easy-rsa/keys/dh2048.pem
tls-auth /etc/openvpn/easy-rsa/keys/ta.key 0
#this will assign connecting clients address between the
    range of 100 and 150
server-bridge 192.168.222.22 255.255.254.0 192.168.222.100
    192.168.222.111

#this will allow for people to get the same IP address after
     a reconnect
ifconfig-pool-persist /etc/openvpn/ipp.txt

keepalive 10 120        #defines time-out for client-
    connections
cipher AES-256-CBC
comp-lzo
max-clients 15
user nobody
group nobody
persist-key # do not re-read keyfiles while ping-restart
persist-tun # do not close and reopen virtual if while ping-
    restart
status /tmp/openvpn-status.log
log-append  /var/log/openvpn.log
client-to-client # enables communication between clients
verb 3
mode server # this is the server
tls-server # we need a preshared key
```

7.2 openvpn.conf - Clientseite

Listing 12: /etc/openvpn/openvpn.conf

```
remote testnetz       # IP-Adresse des VPN-Servers bzw. des
    NAT-Routers
```

```
2  port 1234
3  client                  # Client-Modus
4  proto tcp               # möglich wäre auch "udp"
5  dev tap1
6  dev-type tap
7  ca /etc/openvpn/testnetz/ca.crt
8  cert /etc/openvpn/testnetz/client1.crt
9  key /etc/openvpn/testnetz/client1.key
10 tls-auth /etc/openvpn/testnetz/ta.key 1  # "0" beim VPN-
       Server, "1" bei VPN-Clients
11 keepalive 10 120
12 #cipher BF-CBC  (=Default; sehr schnell)
13 #cipher AES-128-CBC (sicherer)
14 cipher AES-256-CBC (am sichersten)
15 comp-lzo
16 user nobody
17 group nogroup
18 persist-key
19 persist-tun
20 resolv-retry infinite
21 status /var/log/openvpn.testnetz.log
22 #chroot /etc/openvpn/chroot
23 ns-cert-type server     # nur ein Server-Zertifikat kann eine
       VPN-Verbindung zum Client aufbauen
24 tls-remote server  # = Common Name des Servers bei der
       Server-Zertifikatsgenerierung.
25 verb 3
```

7.3 net-config - Serverseite / Clientseite

Listing 13: /etc/conf.d/net

```
1  # This blank configuration will automatically use DHCP for
       any net.*
2  # scripts in /etc/init.d.  To create a more complete
       configuration,
3  # please review /etc/conf.d/net.example and save your
       configuration
4  # in /etc/conf.d/net (this file :]!).
5  dns_domain_lo="stuwo-steinweg.de"
6  dns_domain_eth0="stuwo-steinweg.de"
7
8  depend_xbr0() {
9     need net.vpn net.vpn2 net.eth1
10       }
11
12 config_eth1=( "null" )
13
14 tuntap_vpn="tap"
15 config_vpn=( "0.0.0.0" )
```

```
tuntap_vpn2="tap"
config_vpn2=( "0.0.0.0" )
bridge_xbr0="eth1 vpn vpn2"

routes_xbr0=( "default via 192.168.222.250" )
dns_servers_xbr0=( "192.168.222.250" )
config_xbr0=( "192.168.222.22 netmask 255.255.254.0 brd
    192.168.222.255" )
```

7.4 net-config - Debian (Bridge-Beispiel)

Listing 14: /etc/network/interfaces

```
# This file describes the network interfaces available on
    your system
# and how to activate them. For more information, see
    interfaces(5).

# The loopback network interface
auto lo eth0 eth1
iface lo inet loopback


auto br0
 iface br0 inet static
        address 192.168.222.110
        network 192.168.222.0
        netmask 255.255.254.0
        broadcast 192.168.222.255
        gateway 192.168.222.250
        dns-nameservers 192.168.222.250
        bridge_ports eth1
        bridge_fd 9
        bridge_hello 2
        bridge_maxage 12
        bridge_stp off
```

Listing 15: /etc/rc.local

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser
    runlevel.
# Make sure that the script will "exit 0" on success or any
    other
# value on error.
#
# In order to enable or disable this script just change the
    execution
```

```
10 # bits.
11 #
12 # By default this script does nothing.
13
14 while [ "`ifconfig|grep -o tap0`" = "" ] ; do
15         sleep2
16 done
17         brctl addif br0 tap0
18 exit 0
```

7.5 ebtables

Listing 16: ebtable.sh

```
1 #!/bin/bash
2
3 VPN=tap0
4 IPT=/sbin/ebtables
5
6 # Flush rules
7 $IPT -F
8
9 ######## INPUT CHAIN ########
10 $IPT -I INPUT 0 -i $VPN -p IPv4 --ip-protocol udp --ip-
    destination-port 67 -j DROP
11 $IPT -I INPUT 0 -i $VPN -p IPv4 --ip-protocol udp --ip-
    source-port 68 -j DROP
```

Glossar

ISP Internet Service Provider. Auf beiden Seiten der Testsysteme der Anbieter *1 und 1*.

MAC Message Authentication Code
In der Kryptographie bezeichnet der MAC einen erzeugten Prüfteil zur Authentifizierung einer gesendeten Nachricht und dient damit der Sicherung vor unbemerkter Manipulation der gesendeten Nachricht. .

scp Secure Copy, Secure CoPy (Abk. SCP), ist ein Protokoll, sowie ein Programm, zur verschlüsselten Übertragung von Daten zwischen zwei Computern über ein Rechnernetz..

VPN Virtual Private Network oder Virtuelles Privates Netzwerk.

Literatur

[1] [Gentoo-Wiki] OpenVPN:
http://en.gentoo-wiki.com/wiki/OpenVPN, 12.02.2010

[2] [OpenVPN]:
http://openvpn.net, 11.02.2010.

[3] [OpenVPN e.V.]:
http://www.openvpn.eu, 11.02.2010.

[4] [heise Security] VPN-Knigge:
http://www.heise.de/security/artikel/VPN-Knigge-270796.html,
11.02.2010.