openvpn – VPN im Userspace mit shared-secret

- Kurzakte –

<table>
<tr><th colspan="2">Installation</th></tr>
<tr><td colspan="2">
• Installation des Pakets openvpn auf beiden Rechnern/Routern
• Download des Quellpaketes von http://openvpn.sourceforge.net
tar -xvzf openvpn-1.6.0.tar.gz; cd openvpn-x.x.x;
./configure [--disable-lzo]; make; su; make install
• für die Komprimierung weiter installieren: openssl-devel, openssl und lzo, lzo-devel
</td></tr>
<tr><th colspan="2">Besondere Kennzeichen</th></tr>
<tr><td colspan="2">• Verbindung über UDP oder TCP am Port 1194 (Option --proto udp|tcp-client|tcp-server)</td></tr>
<tr><th colspan="2">Konfiguration auf beiden Hosts</th></tr>
<tr><td>make menuconfig</td><td>• ein TUN bzw. TAP Device muss durch den Kernel benutzbar sein
• TUN bietet Programmen über virtuelle Netzwerkschnittstellen die VPN-Funktionalität auf IP-Ebene an und TAP auf Ethernet-Ebene
• Check unter: Device Drivers ->Networking support -> Universal TUN/TAP device driver support
• bei frischer Aktivierung müssen die Treiber neu kompiliert werden
<code>cd /usr/src/linux
make modules
make modules_install</code></td></tr>
<tr><td>/dev/net/tun</td><td>• falls nicht vorhanden, erstellen mit:
<code># mkdir /dev/net
mknod /dev/net/tun c 10 200
chmod 0700 /dev/net/tun</code></td></tr>
<tr><td>openvpn --genkey</td><td>• Generieren eines gemeinsamen Geheimnisses mit
<code># openvpn --genkey --secret /etc/openvpn/peer.key
chmod go-rwx /etc/openvpn/peer.key</code>
• sicher auf den anderen Host in gleiches Verzeichnis kopieren (z. B. via scp)
<code># scp /etc/openvpn/peer.key IP-Gegenseite:/etc/openvpn/</code>
• Der Tunnel wird durch diesen gemeinsamen Schlüssel verschlüsselt.</td></tr>
<tr><td>route</td><td>• für Zugriff auf das Remote-Netz u./o. von dort auf das Internet
• Client: <code>route del default
route add default gw [Virtual_remoteIP]</code>
• Server: <code>echo "1" > /proc/sys/net/ipv4/ip_forward</code></td></tr>
<tr><td>/sbin/iptables</td><td>• notwendig um den UDP-Port 1194 freizugeben:
<code>/sbin/iptables -A INPUT -p udp --sport 1194 -j ACCEPT
/sbin/iptables -A OUTPUT -p udp --dport 1194 -j ACCEPT</code>
• Der Tunnel sollte die einzige Möglichkeit zum Datenaustausch sein</td></tr>
<tr><th colspan="2">Start als Stand-Alone-Dienst</th></tr>
<tr><td>openvpn --config</td><td>• auf beiden Seiten: <code>openvpn --config konfigdatei zB peer-config</code>
• oder ohne Konfigurationsdatei:
<code>openvpn --dev tun0 --remote [Real_remoteIP]
--ifconfig [Virtual_localIP] [Virtual_remoteIP]
--secret /etc/openvpn/peer.key</code></td></tr>
<tr><td>/etc/inittab
/etc/init.d/</td><td>• Einfügen eines Startskriptes [in den Standard-Runlevel]
• alternativ über den Runlevel-Editor im Yast</td></tr>
<tr><th colspan="2">Funktionskontrolle</th></tr>
<tr><td>ping IP_Tunnelende</td><td>• Ping auf das virtuelle Device der Gegenstelle am Zunnelende</td></tr>
<tr><td>tcpdump</td><td>• Test auf verschlüsselte UDP-Pakete</td></tr>
<tr><th colspan="2">Dokumentation</th></tr>
<tr><td colspan="2">man: openvpn, /usr/share/doc/packages/openvpn</td></tr>
</table>

Einrichten einer openvpn-Verbindung mit shared secret

beteiligte Rechner

Hostnamen:	server	client	(eigentlich peers)
echte öff. IPs:	192.12.1.10	192.12.1.20	(öffentlicher Adressraum)
tun-IPs (Tunnel):	10.0.0.1	10.0.0.2	(privater Adressraum!!!)

Installation von openvpn auf beiden Rechnern

vorab benötigte Pakete

```
client:/usr/local/openvpn/openvpn-2.0 # rpm -qa | egrep "(ssl|lzo)"
lzo-devel-1.08-107
openssl-0.9.7e-3
lzo-1.08-107
openssl-devel-0.9.7e-3
```

Kompilieren der neuesten Version ab 2.xx

```
client:/usr/local/openvpn # ls
.  ..  openvpn-2.0.tar.gz
client:/usr/local/openvpn # tar xzf openvpn-2.0.tar.gz
client:/usr/local/openvpn # cd openvpn-2.0/
client:/usr/local/openvpn/openvpn-2.0 # ./configure
config.status: creating Makefile
client:/usr/local/openvpn/openvpn-2.0 # make
client:/usr/local/openvpn/openvpn-2.0 # make install
/usr/bin/install -c 'openvpn' '/usr/local/sbin/openvpn'
/usr/bin/install -c -m 644 './openvpn.8' '/usr/local/man/man8/openvpn.8'
```

Check auf Unterstützung der virtuellen Schnittstellen

```
client:/usr/local/openvpn/openvpn-2.0 # make menuconfig
```

Device Drivers ->Networking support -> Universal TUN/TAP device driver support"

Erstellen des symmetrischen Schlüssels für shared-secret

```
client:/usr/local/openvpn/openvpn-2.0 # mkdir /etc/openvpn
client:/usr/local/openvpn/openvpn-2.0 # cd /etc/openvpn
client:/etc/openvpn # openvpn --genkey --secret peer.key
client:/etc/openvpn # ls -l
-rw-------   1 root root  636 Jul 15 12:55 peer.key
client:/etc/openvpn # scp peer.key 192.12.1.10:/etc/openvpn/
Password:
peer.key                                      100%  636     0.6KB/s   00:00
```

Einrichten des VPN-Tunnels auf beiden Hosts

```
client:/etc/openvpn # openvpn --dev tun --remote 192.12.1.10 --ifconfig
10.0.0.2 10.0.0.1 --secret /etc/openvpn/peer.key

server:/etc/openvpn # openvpn --dev tun --remote 192.12.1.20 --ifconfig
10.0.0.1 10.0.0.2 --secret /etc/openvpn/peer.key
Fri Jul 15 13:24:06 2005 OpenVPN 2.0 i686-suse-linux [SSL] [LZO] [EPOLL]
built on Jul 14 2005
Fri Jul 15 13:24:06 2005 IMPORTANT: OpenVPN's default port number is now
1194, based on an official port number assignment by IANA.  OpenVPN 2.0-
beta16 and earlier used 5000 as the default port.
Fri Jul 15 13:24:06 2005 TUN/TAP device tun0 opened
Fri Jul 15 13:24:06 2005 /sbin/ifconfig tun0 10.0.0.1 pointopoint 10.0.0.2
mtu 1500
Fri Jul 15 13:24:07 2005 UDPv4 link local (bound): [undef]:1194
Fri Jul 15 13:24:07 2005 UDPv4 link remote: 192.12.1.20:1194
Fri Jul 15 13:24:08 2005 Peer Connection Initiated with 192.12.1.20:1194
Fri Jul 15 13:24:09 2005 Initialization Sequence Completed


server:~ # ifconfig
eth0      Link encap:Ethernet  HWaddr 00:02:3F:DB:4C:23
          inet addr:192.12.1.10  Bcast:192.12.1.255  Mask:255.255.255.0
…
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
inet addr:10.0.0.1  P-t-P:10.0.0.2  Mask:255.255.255.255

server:~ # ps axl
F   UID   PID  PPID PRI  NI   VSZ  RSS WCHAN  STAT TTY        TIME COMMAND
...
4     0  7330  5803  16   0  2984 1352 322479 S+   pts/5      0:00 openvpn
...

server:~ # ping 10.0.0.2      (zum Tunnelausgang)
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.57 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.576 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.569 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.569/0.906/1.573/0.471 ms
```

Der richtige Pfad

Das openvpn-Kommando setzt die Route zum anderen Tunnelende:

```
sever:~ # route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.2        *               255.255.255.255 UH    0      0        0 tun0
192.12.1.0      *               255.255.255.0   U     0      0        0 eth0
link-local      *               255.255.0.0     U     0      0        0 eth0
loopback        *               255.0.0.0       U     0      0        0 lo
default         192.12.1.1      0.0.0.0         UG    0      0        0 eth0
```

Alle anderen Adressen werden allerdings wie vorher um den Tunnel herum geleitet!!!
Verschlüsselung erfolgt also nur zur Zieladresse 10.0.0.2!

- einfach in das Startskript für openvpn integrieren -

Um alle Pakete in Fremdnetze vom Client aus durch den Tunnel zu senden, muss die Default-Route neu gesetzt werden auf das Tunnelende:

```
client:~ # route del default
client:~ # route add default gw 10.0.0.2
```

Der Server dagegen muss die Pakete z.B vom WLAN ins Kabel-LAN weiterrouten:

```
server:~ # echo 1 > /proc/sys/net/ipv4/ip_forward
```

Firewall-Regeln für Client und Server

- einfach in das Startskript für openvpn integrieren -

Client und Server

eingehende Pakete nur auf Port 1194 akzeptieren:

```
iptables -A INPUT -i eth0 -p udp --dport 1194 -j ACCEPT
iptables -A INPUT -i eth0 -j DROP
```

Senden nur vom Port 1194, Weiterleiten von Paketen:

```
iptables -A OUTPUT -o eth0 -p udp --dport 1194 -j ACCEPT
iptables -A OUTPUT -o eth0 -j DROP
iptables -A FORWARD -i eth0 -j DROP
```

TUN-Devices freischalten (Kommunikation über den Tunnel erlauben):

```
iptables -A FORWARD -i tun0 -j ACCEPT
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

nur Server

Server muss sicherstellen, dass der Client den Rest der Welt erreicht

```
iptables -A FORWARD -i tun0 -j ACCEPT
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

Erstellen von Konfigurationsdateien

```
server:/etc/openvpn # cat openvpn.server
 # Konfigurationsdatei fuer eine peer-zu-peer-Verbindung
 # Mit
 #     openvpn --config /etc/openvpn/openvpn.server
 # kann der Server gestartet werden.

 # Welcher Port soll verwendet werden? (Standard: UDP/1194)
 ;proto tcp-server  # nur Client kann VPN-Tunnelaufbau anstossen
 ;port 1194

 # Wir verwenden IP-Tunnel - also das tun-Device.
 dev tun

 # die statische IP-Adresse oder dyndns-Account vom Client
 remote 192.12.1.20

 # die getunnelten IP-Adressen von Server und Client
 ifconfig 10.0.0.1 10.0.0.2

 # Pfad zum gleichen symmetrischen Schluessel
 secret /etc/openvpn/peer.key

 # Pruefe, ob noch eine Verbindung mit dem Server besteht.
 # Jede 10 Sekunden einen Ping, falls nach 60 Sekunden
 # keine Antwort ankommt, wird ein Verbindungsabbruch angenommen.
 # haelt Verbindung durch den NAT-Router am Leben
 keepalive 10 60

 # Kompression fuer den Tunnel
 comp-lzo

 # Abgeben der Prozess-Rechte nach dem Start:
 # * user          Nutzer nach Rechteabgabe (mit moeglichst wenig Rechten)
 # * group         Gruppe analog
 # * persist-key   Lese die Schluessel nicht beim Neustart durch
 #                 SIGUSR1 bzw. --ping-restart ein
 # * persist-tun   Analog fuer close und reopen des TUN/TAP-Devices
 user nobody
 group nobody
 persist-key
 persist-tun

 # Traffic-Shaping z.B. zur Entlastung des 256 kBaud-Upstream bei ADSL
 # bei ausgehenden 16 kByte bleibt die Haelfte der Bandbreite für
 # andere Dienste
 shaper 16000

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

 # openvpn als Daemon laufen lassen
 daemon

 # Der fragment und der mssfix-Eintrag sind notwendig, weil es sonst zu Problemen
 # mit Win32-Clienten kommen kann.
 tun-mtu 1500
 fragment 1500
 mssfix

server:/etc/openvpn # openvpn --config /etc/openvpn/openvpn.server
```

```
client:/etc/openvpn # cat openvpn.client
 # Konfigurationsdatei fuer eine peer-zu-peer-Verbindung
 # Mit
 #     openvpn --config /etc/openvpn/openvpn.client
 # kann der Server gestartet werden.

 # Welcher Port soll verwendet werden? (Standard: UDP/1194)
 ;proto tcp-client      # nur Client kann VPN-Tunnelaufbau anstossen
 ;port 1194

 # Wir verwenden IP-Tunnel - also das tun-Device.
 dev tun

 # die statische IP-Adresse oder dyndns-Account vom Server
 remote 192.12.1.10

 # die getunnelten IP-Adressen von Server und Client
 ifconfig 10.0.0.2 10.0.0.1

 # Pfad zum gleichen symmetrischen Schluessel
 secret /etc/openvpn/peer.key

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

 # Kompression fuer den Tunnel
 comp-lzo

 # Abgeben der Prozess-Rechte nach dem Start:
 # * user          Nutzer nach Rechteabgabe (mit moeglichst wenig Rechten)
 # * group         Gruppe analog
 # * persist-key   Lese die Schluessel nicht beim Neustart durch
 #                 SIGUSR1 bzw. --ping-restart ein
 # * persist-tun   Analog fuer close und reopen des TUN/TAP-Devices
 user nobody
 group nobody
 persist-key
 persist-tun

 # Traffic-Shaping z.B. zur Entlastung des 256 kBaud-Upstream bei ADSL
 # bei ausgehenden 16 kByte bleibt die Haelfte der Bandbreite für
 # andere Dienste
 shaper 16000

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

 # auch der Client (peer 2) kann als Daemon laufen
 daemon


client:/etc/openvpn # openvpn --config /etc/openvpn/openvpn.client
```