Einrichtung einer Multi-Client/Single-Server VPN Umgebung

Von Tim Meusel
Sommer 2011

Inhaltsverzeichnis

Aufgabenstellung:

Auf einem laufenden Debian vServer* soll ein VPN Gateway installiert werden. Die Clients (Linux) sollen untereinander Spiele im LAN-Modus spielen können und über Ipv4 und IPv6 das Internet erreichen.

Hinweis: Alle mit * gekennzeichneten Wörter werden im Glossar erklärt.

Welche Software kann ein VPN realisieren?

IPSec:
+ Die sicherste Möglichkeit, wird unter anderem vom deutschen Auswärtigen Amt genutzt
- Sehr komplizierte Konfiguration
● Layer 3

L2TP
+ Sehr einfache Konfiguration unter Windows
- Aufwendige Konfiguration unter Linux (es ist eventuell nötig den Kernel zu Patchen und somit neu zu kompilieren)
● Layer 2

OpenVPN
+ Einfache Einrichtung unter Linux und Windows Clients sowie Servern
+ Gute Verschlüsselung
● Layer 2 oder 3

PPTP
+ Einfache Konfiguration
- Unterstützt keine Verschlüsselung
● Layer 3

Layer 2 oder Layer 3?

Layer 2
Bei einem VPN auf Layer 2 arbeiten die Netzwerkschnittstellen im Promiscuous Mode*. Dadurch sind alle Clients und der Server in einem großen logischen LAN. Folglich kann jedes Protokoll der Schicht 3 genutzt werden. Wird der TCP/Ip Stack genutzt, sind Broadcasts möglich.

Layer 3
Die zur Verfügung stehende Software arbeitet mit dem TCP/IP Protokoll Stack. Somit ist eine Nutzung von alternativen Protokollen der Schicht 3 nicht möglich (z.B. IPX). Die Konfiguration ist hier aufwendiger und Broadcasts werden nicht unterstützt (jeder Client befindet sich in einem eigenen Subnetz und es gibt ein globales Subnetz in dem sich alle virtuellen VPN Schnittstelen befinden).

Ich habe mich gegen eine Lösung auf Layer 3 entschieden, da das VPN unter anderem für netzwerkfähige Spiele genutzt werden soll. Diese nutzen oft Broadcasts zum finden der Gameserver.

OpenVPN oder L2TP?

Das Betriebssystem des Servers ist Debian 6. Hier ist die Einrichtung von OpenVPN wesentlich einfacher als L2TP. Somit entscheide ich mich für OpenVPN im Layer 2 Modus.

Konzept:

Die Zertifizierung der Clients erfolgt auf Basis von Zertifikaten*. Die Kommunikation wird mit AES* verschlüsselt. Jedem Client wird eine private IPv4-Adresse und ein 64 Bit langer Prefix einer globalen Unicast Ipv6-Adresse zugewiesen (über Router Advertisment*). Ist auf dem Client das VPN aktiviert soll die komplette Kommunikation mit dem Internet über das VPN erfolgen.

Konfiguration des Host Servers:

Der Host Server nutzt ein geroutetes Setup*. Damit alle Pakete bei dem vServer ankommen muss ein IPv6-Prefix zu dem vServer geroutet werden. Dazu muss man der Bridge eine neue Route zuweisen. Dies geschieht durch anhängen des folgenden Befehl in der interface Datei(/etc/network)

```
up ip -6 route add 2a01:4f8:100:5580::/57 via 2a01:4f8:100:5500::2 dev virbr1
```

Der 57Bit Prefix wird zur IPv6-Adresse der Schnittstelle eth0* des vServers geroutet.

Installation und Konfiguration von OpenVPN auf einem vServer:

1. Updaten der Repositorys und Installation von OpenVPN

```
vpn01:~# aptitude update && aptitude install openvpn
```

2. Das Forwarding wird in /etc/sysctl.conf für IPv6 und IPv4 aktiviert.

```
net.ipv6.conf.all.forwarding=1
net.ipv4.ip_forward=1
```

3. Die Konfigurationsdateien für OpenVPN werden aus dem Dokumentationsordner des Paketes in das Laufzeitverzeichnis kopiert.

```
vpn01:~#  cp -r /usr/share/doc/openvpn/examples/easy-rsa/2.0 /etc/openvpn/easy-rsa2
```

4. Folgende Variablen müssen am Ende der /etc/openvpn/easy-rsa2/vars angepasst werden:

```
export KEY_COUNTRY="DE"
export KEY_PROVINCE="NRW"
export KEY_CITY="Hamm"
export KEY_ORG="bastelfreak.org"
export KEY_EMAIL="tim.meusel@bastelfreak.org"
```

5. Der Inhalt der vars Datei muss gesourcet* werden.

```
vpn01:~# source ./vars
```

5. Der Ordner keys unter /etc/openvpn/easy-rsa2/ wird erstellt.

```
vpn01:~#  mkdir /etc/openvpn/easy-rsa2/keys
```

6. Die neue CA* mus initialisiert werden.

```
vpn01:~# ./clean-all
```

```
vpn01:~# ./build-ca
```

7. Das Server-Zertifikat wird erstellt.

```
vpn01:~# ./build-key-server
```

8. Folgendes muss pro Client einmal ausgeführt werden(erstellt Zertifikat und passenden Schlüssel für den Client).

```
vpn01:~# ./build-key Clientname
```

Anmerkung:Möchte man nach dem beenden der Shellsitzung weitere Zertifikate erzeugen, muss Schritt 5 wiederholt werden, danach kann man wie gewohnt mit build-key Zertifikate erzeugen.

9. Nun wird die Diffie-Hellman Datei* erstellt.

```
vpn01:~#  ./build-dh
```

10. Aus Sicherheitsgründen sollte OpenVPN nicht vom root Benutzer ausgeführt werden. Deshalb legen wir einen eigenen Benutzer und eine dazugehörige Gruppe an:

```
vpn01:~# adduser --system --no-create-home --disabled-login openvpn
vpn01:~#  addgroup --system --no-create-home --disabled-login openvpn
```

11. Die Konfiguration(/etc/openvpn/server.conf) muss noch angepasst werden:

```
port 1337
# Port auf dem OpenVPN läuft
proto udp
# Das genutzte Protokoll, UDP ist effizienter als TCP da es weniger
# Overhead produziert.
dev tap
# OSI Schichten wählen, tap entspricht Layer 2, tun Layer 3. Nach dem
# Start von OpenVPN wird eine virtuelle Netzwerkschnittstelle mit dem
# Namen tap0 angelegt.
ca ./easy-rsa2/keys/ca.crt
# Pfad zu der Certificate Authority*.
cert ./easy-rsa2/keys/vpn01.bastelfreak.org.crt
# Pfad zu dem Zertifikat für den Server.
key ./easy-rsa2/keys/vpn01.bastelfreak.org.key
# Pfad zum dem Schlüssel für das Server-Zertifikat.
dh ./easy-rsa2/keys/dh1024.pem
# Pfad zu der Diffie-Hellman-Datei.
server 10.0.0.0 255.255.255.0
# Subnetz für das VPN Netz. 10.0.0.1 ist der Server. 10.0.0.2 ist die
# erste Adresse die an Clients vergeben wird.
ifconfig-pool-persist ipp.txt
# Pfad zur ipp.txt Datei. In dieser werden die Zuordnungen zwischen
# Client-Hostname und seiner VPN IPv4-Adresse gespeichert.
client-to-client
# Clients dürfen untereinander kommunizieren.
keepalive 10 120
# nach 10 Sekunden Inaktivität wird ein Ping gesendet. Nach 120 Sekunden
# ohne Antwort wird die Verbindung neugestartet.
comp-lzo adaptive
# OpenVPN wählt selbst welcher Inhalt komprimiert wird.
user openvpn
# Dieser Benutzer führt OpenVPN aus.
group openvpn
# Zu dieser Gruppe gehört der Benutzer der OpenVPN ausführt.
persist-key
persist-tun
```

```
push "persist-key"
push "persist-tun"
# Diese Optionen verhindern einen Zugriff auf die OpenVPN Ressourcen bei
# einer Trennung der Verbindung (z.B. Zwangstrennung bei PPPoE). Push
# leitet die Einstellungen an die Clients weiter wenn diese verbinden.
log openvpn.log
# Pfad zur Logdatei.
verb 3
# Mit verb(verbose ~ Geschwätzigkeit) wird die Anzahl der Logeinträge
# bestimmt. 0 sind nur schwerwiegende Fehler, 9 sind extrem viele
# Ausgaben die größenteils nur für Programmierer interessant sind. 3 ist
# ein guter Standardwert für das Debugging von Fehlern.
push "redirect-gateway"
# Der Client nutzt seine VPN Schnittstelle als Standardgateway.
cipher AES-256-CBC
# AES Verschlüsselung aktivieren. Standard ist BF-CBC (Blowfish).
Up up.sh
# Das up.sh Shellskript wird beim Start von OpenVPN ausgeführt.
script-security 2
# Der Wert muss auf 2 gesetzt werden. Sonst führt OpenVPN keine up
# Skripte aus
```

11. Das up.sh Skript wird im Verzeichniss /etc/openvpn erstellt. Aus Sicherheitsgründen gehört es dem Benutzer und der Gruppe openvpn. Nur dieser Benutzer darf es ausführen.

```
vpn01:~# touch /etc/openvpn/up.sh
```

```
vpn01:~# chown openvpn:openvpn /etc/openvpn/up.sh
```

```
vpn01:~# chmod 700 /etc/openvpn/up.sh
```

Inhalt der up.sh Datei:

```
#!/bin/bash
ip -6 addr add 2a01:4f8:100:5580::2 dev tap0
# Die Adresse 2a01:4f8:100:5580::2 wird der virtuellen Schnittstelle tap0
# zugewiesen.
iptables -t nat -A POSTROUTING -o tap0 -s 10.0.0.0/24 -j  MASQUERADE
# Durch diesen IPTables Eintrag wird NAT* für die privaten IP-Adressen
# aktiviert.
```

12. Installation und Einrichtung von radvd für Router Advertisment:

```
vpn01:~# aptitude install radvd
```

Folgende Konfiguration wird in die /etc/radvd.conf Datei eingetragen:

```
interface tap0{
      AdvSendAdvert on;
      prefix 2a01:4f8:100:5590::/64 {
            AdvOnLink on;
            AdvAutonomous on;
            AdvRouterAddr on;
      };
};
```

Dadurch werden per Router Advertisment IPv6 Prefixe auf der tap0 Schnittstelle verteilt. Nun muss radvd neugestartet werden:

```
vpn01:~#  /etc/init.d/radvd restart
```

14. OpenVPN Server starten

```
vpn01:~# cd /etc/openvpn
```

```
vpn01:~# openvpn --config /etc/openvpn/server.conf  --daemon
```

Installation und Einrichtung eines VPN Clients unter Debian 6:

1. Installation von OpenVPN und durch den Paketmanager:

```
vpn01:~# aptitude install openvpn
```

3. Die Zertifikate und Schlüssel die auf dem TLS-Server erzeugt wurden müssen auf den Client kopiert werden. Das Verzeichniss ist /etc/openvpn. Hier müssen die ca.crt, CLIENT.crt und CLIENT.key gespeichert werden.

4. Nun wird die Konfigurationsdatei erstellt und angepasst:

```
vpn01:~# touch /etc/openvpn/client.conf
```

```
client
# sagt OpenVPN das es als Client arbeiten soll.
dev tap
proto udp
pull
# Der Client rechnet damit das er Einstellungen vom Server bekommt (push)
remote vpn01.bastelfreak.org 1337
# FQDN* des VPN-Server und der Port auf dem der OpenVPN Server läuft.
ca ca.crt
cert CLIENT.crt
key CLIENT.key
ns-cert-type server
# Server überprüft CLIENT Zertifikate auf Gültigkeit
comp-lzo adaptive
up up.sh
script-security 2
verb 3
resolv-retry infinite
# Der Hostname des Servers wird aufgelöst und gespeichert.
nobind
# Der Client ist lokal nicht an einen bestimmten UDP Port gebunden.
cipher AES-256-CBC
log openvpn.log
```

5. Das up.sh Skript wird im Verzeichniss /etc/openvpn erstellt. Der Benutzer der OpenVPN ausführen soll benötigt auch die Rechte um das Skript auszuführen.

```
#!/bin/sh
ip -6 route add  2a01:4f8:100:5580::2 dev eth0
ip -6 route add default via  2a01:4f8:100:5580::2 dev eth0
```

Das Skript sagt dem Client das jeder IPv6 Datenverkehr von eth0 an die IP-Adresse 2a01:4f8:100:5580::2 geht. Dies ist die Adresse von tap0 des Servers.

6. Installation und Konfiguration von rdnssd:
rdnssd ist ein Client für Router Advertisments. Er hört auf radvd und weist der Schnittstelle auf der das Advertisment ankommt einen IPv6 Prefix zu. rdnssd updated regelmäßig die /etc/resolv.conf, deshalb muss vorher das Paket resolvconf installert werden.

```
vpn01:~# aptitude install resolvconf
```

```
vpn01:~# aptitude install rdnssd
```

Analyse der Logdatei: Serverstart

Zur besseren Lesbarkeit wurde vor jedem Logeintrag das Datum entfernt. Der Server schreibt beim Start folgendes in die openvpn.log:

```
OpenVPN 2.1.3 x86_64-pc-linux-gnu [SSL] [LZO2] [EPOLL] [PKCS11] [MH]
[PF_INET6] [eurephia] built on Oct 22 2010
# Versionsnummer von Openvpn, verwendete Architektur, geladene Module und
#Build Datum
NOTE: OpenVPN 2.1 requires '--script-security 2' or higher to call user-
defined scripts or executables
Diffie-Hellman initialized with 1024 bit key
# die Diffie-Hellman Datei(dh1024.pem) wurde erfolgreich geladen.
/usr/bin/openssl-vulnkey -q -b 1024 -m <modulus omitted>
TLS-Auth MTU parms [ L:1590 D:138 EF:38 EB:0 ET:0 EL:0 ]
# Für den TLS Server wird die MTU eingestellt.
Socket Buffers: R=[124928->131072] S=[124928->131072]
# Die Socket Buffers werden eingestellt.
TUN/TAP device tap0 opened
# Es wird tap0 genutzt
TUN/TAP TX queue length set to 100
# Die Paketwarteschlange fasst 100 Pakete
/sbin/ifconfig tap0 10.0.0.1 netmask 255.255.255.0 mtu 1500 broadcast
10.0.0.255
# Die virtuelle Netzwerkschnittstelle tap0 wurde erstellt
./up.sh tap0 1500 1590 10.0.0.1 255.255.255.0 init
# Das up.sh Skript wurde erfolgreich ausgeführt.
Data Channel MTU parms [ L:1590 D:1450 EF:58 EB:135 ET:32 EL:0 AF:3/1 ]
# Für tap0 wird die MTU eingestellt.
GID set to openvpn
# Die Gruppe openvpn führt OpenVPN aus
UID set to openvpn
# Der User openvpn füht OpenVPN aus
UDPv4 link local (bound): [undef]
UDPv4 link remote: [undef]
# Es besteht keine Verbindung zu einem weiteren OpenVPN-Server.
MULTI: multi_init called, r=256 v=256
#
IFCONFIG POOL: base=10.0.0.2 size=253
# Der interne DHCP Server von OpenVPN vergibt max. 253 Adressen,
angefangen bei 10.0.0.2
Initialization Sequence Completed
# Der Server wurde erfolgreich gestartet
```

Analyse der Logdatei: Verbindungsaufbau (Clientseitig)

Der Client schreibt beim Verbindungsaufbau folgendes in die openvpn.log:

```
OpenVPN 2.1.3 x86_64-pc-linux-gnu [SSL] [LZO2] [EPOLL] [PKCS11] [MH]
[PF_INET6] [eurephia] built on Oct 22 2010
NOTE: the current --script-security setting may allow this configuration
to call user-defined scripts
/usr/bin/openssl-vulnkey -q -b 1024 -m <modulus omitted>
LZO compression initialized
# Die Komprimierung wurde aktiviert.
Control Channel MTU parms [ L:1590 D:138 EF:38 EB:0 ET:0 EL:0 ]
Socket Buffers: R=[124928->131072] S=[124928->131072]
Data Channel MTU parms [ L:1590 D:1450 EF:58 EB:135 ET:32 EL:0 AF:3/1 ]
Local Options hash (VER=V4): 'c6c7c21a'
# Lokaler Hashwert
Expected Remote Options hash (VER=V4): '1a6d5c5d'
# Hashwert des Servers
UDPv4 link local (bound): [undef]
UDPv4 link remote: [AF_INET]188.40.193.202:1337
# IPv4-Adresse und Port des OpenVPN Servers.
TLS: Initial packet from [AF_INET]188.40.193.202:1337, sid=a44b0322
a808b9c3
# Paket zum Zertifikatsaustausch würde vom Server gesendet.
VERIFY OK: depth=1,
/C=DE/ST=NRW/L=Hamm/O=bastelfreak.org/CN=bastelfreak.org_CA/name=bastelfr
eak/emailAddress=tim.meusel@bastelfreak.org
VERIFY OK: nsCertType=SERVER
VERIFY OK:
depth=0,/C=DE/ST=NRW/L=Hamm/O=bastelfreak.org/CN=vpn01.bastelfreak.org/em
ailAddress=tim.meusel@bastelfreak.org
# das Server-Zertifikat wurde erfolgreich validiert.
Data Channel Encrypt: Cipher 'AES-256-CBC' initialized with 256 bit key
# Der Verschlüsselungsalgorythmus wurde auf AES-256-CBC gesetzt.
Data Channel Encrypt: Using 160 bit message hash 'SHA1' for HMAC
authentication
# Die Methode der Authentisierung wurde auf SHA1 gesetzt.
Control Channel: TLSv1, cipher TLSv1/SSLv3 DHE-RSA-AES256-SHA, 1024 bit
RSA
# Der TLS-Server meldet seine Verschlüsselungseinstellungen.
[vpn01.bastelfreak.org] Peer Connection Initiated with
[AF_INET]188.40.193.202:1337
# Der Verbindungsaufbau wurde begonnen.
SENT CONTROL [vpn01.bastelfreak.org]: 'PUSH_REQUEST' (status=1)
# Der Client fragt nach push Anweisungen des Servers.
PUSH: Received control message: 'PUSH_REPLY,redirect-gateway,route-
gateway 10.0.0.1,ping 10,ping-restart 120,ifconfig 10.0.0.2
255.255.255.0'
# Der Client hat seine Einstellungen erhalten und wendet sie an.
OPTIONS IMPORT: timers and/or timeouts modified
OPTIONS IMPORT: --ifconfig/up options modified
OPTIONS IMPORT: route options modified
OPTIONS IMPORT: route-related options modified
# Der Client hat die Einstellungen des Servers (push) übernommen.
ROUTE default_gateway=172.20.60.1
# Ein neues Default-Gateway wurde auf dem Client gesetzt.
TUN/TAP TX queue length set to 100
/sbin/ifconfig tap0 10.0.0.2 netmask 255.255.255.0 mtu 1500 broadcast
10.0.0.255
```

```
up.sh tap0 1500 1590 10.0.0.2 255.255.255.0 init
# Das up.sh Skript wurde ausgeführt.
/sbin/route add -net 188.40.193.202 netmask 255.255.255.255 gw
172.20.60.1
/sbin/route del -net 0.0.0.0 netmask 0.0.0.0
/sbin/route add -net 0.0.0.0 netmask 0.0.0.0 gw 10.0.0.1
# Durch das up.sh Skript wurden einige neue Routen gesetzt.
Initialization Sequence Completed
# Der Verbindungsaufbau war erfolgreich. Die Verbindung steht.
```

Analyse der Logdatei: Verbindungsaufbau (Serverseitig)

Ausser dem Datum schreibt der Server bei einem Verbindungsaufbau auch die IP-Adresse:Port an den Anfang jeders Logeintrags. Aus Datenschutzgründen und zur besseren Lesbarkeit wurden diese Angaben durch [IP:P] ersetzt.

```
MULTI: multi_create_instance called
# Der Server bemerkt eine eingehende Verbindung.
[IP:P] Re-using SSL/TLS context
# Server bereitet sich auf einen Zertifikatsvergleich vor.
[IP:P] LZO compression initialized
[IP:P] Control Channel MTU parms [ L:1590 D:138 EF:38 EB:0 ET:0 EL:0 ]
[IP:P] Data Channel MTU parms [ L:1590 D:1450 EF:58 EB:135 ET:32 EL:0
AF:3/1 ]
[IP:P] Local Options hash (VER=V4): '1a6d5c5d'
[IP:P] Expected Remote Options hash (VER=V4): 'c6c7c21a'
[IP:P] TLS: Initial packet from [AF_INET][IP:P], sid=7de3ceef 141aba66
[IP:P] VERIFY OK: depth=1,
/C=DE/ST=NRW/L=Hamm/O=bastelfreak.org/CN=bastelfreak.org_CA/name=bastelfr
eak/emailAddress=tim.meusel@bastelfreak.org
[IP:P] VERIFY OK: depth=0,
/C=DE/ST=NRW/L=Hamm/O=bastelfreak.org/CN=bastelfreak/emailAddress=tim.meu
sel@bastelfreak.org
[IP:P] Data Channel Encrypt: Cipher 'AES-256-CBC' initialized with 256
bit key
[IP:P] Data Channel Encrypt: Using 160 bit message hash 'SHA1' for HMAC
authentication
[IP:P] Data Channel Decrypt: Cipher 'AES-256-CBC' initialized with 256
bit key
[IP:P] Data Channel Decrypt: Using 160 bit message hash 'SHA1' for HMAC
authentication
[IP:P] Control Channel: TLSv1, cipher TLSv1/SSLv3 DHE-RSA-AES256-SHA,
1024 bit RSA
[IP:P] [bastelfreak] Peer Connection Initiated with [AF_INET][IP:P]
bastelfreak/[IP:P] PUSH: Received control message: 'PUSH_REQUEST'
# Der Server erhält die Anfrage nach push Einstellungen für den Client.
bastelfreak/[IP:P] SENT CONTROL [bastelfreak]: 'PUSH_REPLY,redirect-
gateway,route-gateway 10.0.0.1,ping 10,ping-restart 120,ifconfig 10.0.0.2
255.255.255.0' (status=1)
# Die Einstellungen wurden gesendet.
bastelfreak/[IP:P] MULTI: Learn: 46:5b:94:df:28:e4 -> bastelfreak/[IP:P]
# Der Server merkt sich das die MAC-Adresse 46:5b:94:df:28:e4 zu dem
# Client bastelfreak gehört. Bastelfreak ist in diesem Fall der
# Clientname  der beim erstellen des Zertifikates eingegeben wurde.
bastelfreak/[IP:P] TLS: new session incoming connection from [AF_INET]
[IP:P]
# Es erfolgt nochmal eine Verbindung zum Zertifikatsabgleich
bastelfreak/[IP:P] VERIFY OK: depth=1,
/C=DE/ST=NRW/L=Hamm/O=bastelfreak.org/CN=bastelfreak.org_CA/name=bastelfr
```

```
eak/emailAddress=tim.meusel@bastelfreak.org
bastelfreak/[IP:P] VERIFY OK: depth=0,
/C=DE/ST=NRW/L=Hamm/O=bastelfreak.org/CN=bastelfreak/emailAddress=tim.meu
sel@bastelfreak.org
# Der Abgleich war erfolgreich.
bastelfreak/[IP:P] Data Channel Encrypt: Cipher 'AES-256-CBC' initialized
with 256 bit key
# Die Verschlüsselung wurde erfolgreich eingestellt.
bastelfreak/[IP:P] Data Channel Encrypt: Using 160 bit message hash
'SHA1' for HMAC authentication
bastelfreak/[IP:P] Data Channel Decrypt: Cipher 'AES-256-CBC' initialized
with 256 bit key
bastelfreak/[IP:P] Data Channel Decrypt: Using 160 bit message hash
'SHA1' for HMAC authentication
#bastelfreak/[IP:P] TLS: move_session: dest=TM_ACTIVE src=TM_UNTRUSTED
reinit_src=1
bastelfreak/[IP:P] TLS: tls_multi_process: untrusted session promoted to
semi-trusted
# Da die Verbindung nun verschlüsselt läuft und Client und Serverseitig
alle Zertifikate validiert wurden ist die Verbindung un vertrauenswürdig.
bastelfreak/[IP:P] Control Channel: TLSv1, cipher TLSv1/SSLv3 DHE-RSA-
AES256-SHA, 1024 bit RSA
bastelfreak/[IP:P] PUSH: Received control message: 'PUSH_REQUEST'
bastelfreak/[IP:P] SENT CONTROL [bastelfreak]: 'PUSH_REPLY,redirect-
gateway,route-gateway 10.0.0.1,ping 10,ping-restart 120,ifconfig 10.0.0.2
255.255.255.0' (status=1)
bastelfreak/[IP:P] MULTI: Learn: f6:4f:3e:1b:85:79 → bastelfreak/[IP:P]
# Die Verbindung wurde erfolgreich validiert und aufgebaut. Sie kannn nun
#genutzt werden.
```

Fazit:

Innerhalb zwei Tagen lässt sich eine sehr sichere, flexible und skalierbare Multi-Client/Server VPN Umgebung auf Basis von OpenVPN einrichten. Die nachträgliche Dokumentation war zeitaufwendiger als die VPN Einrichtung. In nachfolgenden Tests wurde versucht durch verschiedene Ansätze in das VPN einzudringen und Daten zu manipulieren. Auf eine bestehende Verbindung wurde eine „Man in the Middle“* Attacke angewandt. Das Bruteforcen der mitgeschnittenen Pakete mittels GPU Computing wurde nach einigen Tagen aufgrund ausbleibenden Erfolgs abgebrochen. Falls man Zugriff auf einen vorhandenen Client hat, kann man diesen mit dem THC-IPv6 Toolkit* versehen. Dadurch lassen sich die IPv6 Einstellungen aller Clients massiv verändern. Die Lücken die THC nutzt kann man mit einigem Aufwand allerdings schließen.

Abschließend kann man sagen das man mit OpenVPN eine brauchbare Infrastruktur bereitstellen kann die auch für brisante Daten geeignet ist. Auf den Einsatz von IPv6 sollte allerdings vorerst verzichtet werden.

Glossar

AES: Im VPN wir die Verschlüsselung AES-256-CBC genutzt. Diese ist sehr sicher. Außerdem unterstützt die CPU des Servers hardwareseitige AES Ver/Entschlüsselung. Somit wird die CPU bei sehr vielen VPN Verbindungen massiv entlastet (http://software.intel.com/en-us/articles/intel-advanced-encryption-standard-aes-instructions-set).

CA: Certificate Authority, mit dieser Zertifizierungsstelle werden alle Clientschlüssel digital signiert (http://de.wikipedia.org/wiki/Certificate_Authority).

dh.pem/Diffie-Hellman Datei: Diffie-Hellman ist ein Verfahren zum Schlüsselaustausch. Dabei erzeugen der Server und der Client einen geheimen Schlüssel mit dem die verschlüsselten Pakete übertragen werden. Solange ein Angreifer nur den Schlüsselaustausch mithören kann(welcher unverschlüsselt abläuft) ist es nicht möglich den Schlüssel zu errechnen. Manipuliert ein Angreifer allerdings die Pakete des Schlüsselaustauschs, kann er den Schlüssel berechnen (http://de.wikipedia.org/wiki/Diffie-Hellman-Schl%C3%BCsselaustausch).

eth0: eth0 ist der Name der ersten Netzwerkverbindung eines Linux-PCs.

Forwarding: Siehe Promiscuous Mode.

FQDN: Fully Quallified Domain Name. Dieser Name ist der absolute Domain Name.

geroutetes Setup: Der Host Server enthält eine virtuelle Bridge. Diese erhält dank dem Promiscuous Mode alle Pakete von eth0. Die Bridge erstellt für jede IP-Adresse/Subnetz der vServer eine Route, welche die Adresse(n) zu der richtigen virtuellen Maschinen leitet.

Gesourcet: Für die Werte wird ein systemweite Aliase gesetzt. Diese sind bis zum verlassen der Shell gültig.

MTU: Maximum Transmission Unit, diese beschreibt die maximale Paketgröße eines Paketes.

Man in the Middle: Bei dieser Art von Attacke steht der Angreifer zwischen der Client/Server Verbindung und hat somit Zugriff auf alle verschickten Pakete.

NAT: Network Adress Translation, ist NAT auf einem Server eingerichtet, greift er Anfragen die aus dem privaten IP-Adressbereich in das Internet gehen ab und manipuliert die Pakete so, das er als Absender in den Paketen steht. Eingehenden Verbindungen leitet er an die passende IP-Adresse aus dem privaten Bereich weiter. Somit kann man viele Computer in einem privaten Netz an einer einzigen öffentlichen IP-Adresse anschliesen. Dieses Verfahren findet häufig bei privaten Internet Anschlüssen statt. Aufgrund nicht vorhandener Adressknappheit bei IPv6 ist es dort hinfällig.
(http://de.wikipedia.org/wiki/Network_Address_Translation#Source_NAT)

Promiscuous Mode: Die Netzwerkkarte leitet alle ankommenden Pakete an das Betriebssystem weiter. Danach greift die virtuelle Schnittstelle auf die Pakete zu und verarbeitet diese.
Router Advertisment: Router Advertisment ist ein ICMPv6-Typ. Der Router verkündet im Netz das er verfügbar ist. Außerdem kann er einige Client Einstellungen übermitteln (http://de.wikipedia.org/wiki/Neighbor_Discovery_Protocol#Router_Advertisement_.E2.80.93_Type_134)

tap0: Die erste virtuelle Netzwerkverbindung die OpenVPN im Layer 2 Modus erstellt heißt tap0.

THC-IPv6 Toolkit: Mit diesem Toolkit kann man IPv6 und ICMPv6 Lücken ausnutzen.

TLS-Server: Der TLS-Server verwaltet alle Zertifikate. Jeder Vpn User kann der TLS-Server sein. Es ist sinnvoll wenn der OpenVPN Server gleichzeitig TLS-Server ist. Die CA signiert die Zertifikate die der TLS-Server ausstellt.

vServer: In diesem Fall ist ein vServer ein virtuelles Linux das durch KVM (http://www.linux-kvm.org/page/Main_Page) bereitgestellt wird.

Zertifikate: Jeder Client und der Server besitzen ein Zertifikat (Schlüssel). Der OpenVPN Server lässt nur Verbindungen mit zertifikaten zu die von einem ihm bekannten TLS-Server stammen.

Quellen & Copyright

1. Buch: OpenVPN von Galileo Computing (ISBN 978-3-8362-1671-5)
2. VPN Grundlagen http://wiki.ubuntuusers.de/OpenVPN
3. IPv6 Prefix Delegation http://www.ripe.net/ripe/docs/ripe-512
4. Ipv6 Router Advertisment http://tools.ietf.org/html/rfc5006
5. Neighbor Discovery for IP version 6 http://tools.ietf.org/html/rfc4861
6. THC-IPv6 Toolkit http://www.thc.org/thc-ipv6/
7. Vortrag über IPv6 und THC http://events.ccc.de/congress/2010/Fahrplan/events/3957.en.html
8. Probleme durch IPv6 http://torrentfreak.com/huge-security-flaw-makes-vpns-useless-for-bittorrent-100617/