

# Praktikum Rechnernetze

Protokoll zu Versuch 7 (OpenVPN) von Gruppe 1

---

Jakob Waibel   Daniel Hiller   Elia Wüstner   Felix Pojtinger

2021-11-30

# Einführung

---

Diese Materialien basieren auf Professor Kiefers “Praktikum Rechnernetze”-Vorlesung der HdM Stuttgart.

**Sie haben einen Fehler gefunden oder haben einen Verbesserungsvorschlag?** Bitte eröffnen Sie ein Issue auf GitHub ([github.com/poijntfx/uni-netpractice-notes](https://github.com/poijntfx/uni-netpractice-notes)):



**Figure 1:** QR-Code zum Quelltext auf GitHub

Dieses Dokument und der enthaltene Quelltext ist freie Kultur bzw. freie Software.



**Figure 2:** Badge der AGPL-3.0-Lizenz

Uni Network Practice Notes (c) 2021 Jakob Waibel, Daniel Hiller,  
Elia Wüstner, Felix Pojtinger

SPDX-License-Identifier: AGPL-3.0

**CA (=Zertifizierungsstelle) und  
Schlüssel erzeugen und signieren**

---

# CA (=Zertifizierungsstelle) und Schlüssel erzeugen und signieren

Verzeichnis erstellen und betreten:

```
# mkdir openvpn  
# cd openvpn
```

Git installieren:

```
apt install git
```

Repository klonen:

```
# git clone https://github.com/OpenVPN/easy-rsa  
Cloning into 'easy-rsa' ...  
remote: Enumerating objects: 2095, done.  
remote: Counting objects: 100% (13/13), done.  
remote: Compressing objects: 100% (11/11), done.  
remote: Total 2095 (delta 3), reused 4 (delta 0), p4
```

### **Beschreiben Sie kurz den Sinn der Dateien in diesen Ordnern**

Die `ca.crt` Datei ist öffentlich. User, Server und Client können damit beweisen, dass sie sich im selben vertrauten Netz befinden. Jeder daran beteiligte User und Server muss eine Kopie dieser Datei besitzen.

`ca.key` ist der private Schlüssel, mit dem die CA Zertifikate für Server und Clients signiert werden. Die `ca.key` Datei sollte nur auf der CA Maschine liegen, denn der Schlüssel darf nicht in die Hände eines Angreifers gelangen.

Die Private Keys liegen im Ordner `private` und im Ordner `issued` sind die signierten Zertifikate (Public Keys) für eine gegenseitige Bestätigung zwischen Server und Client.

Der Ordner `certs_by_serial` enthält alle von der CA signierten

# Konfiguration von Client und Server

---



## Server konfigurieren

Analog zu der in der Versuchsanleitung geschilderten Konfigurationsdatei wird im Folgenden eine angepasste `server.conf` dargestellt:

```
# cat server.conf
proto udp
dev tun
ca pki/ca.crt
cert pki/issued/server-g1.crt
key pki/private/server-g1.key
dh pki/dh.pem
server 10.8.1.0 255.255.255.0
keepalive 10 120
comp-lzo
persist-key
persist-tun
```

**Erklären Sie die einzelnen Parameter/Optionen der „server.conf“ und der „client.conf“.**

Client:

<code>client</code>	<code># Definiert dass es s</code>
<code>dev tun</code>	<code># Als virtuelles Netz</code>
<code>proto udp</code>	<code># Hier wird festgelegt</code>
<code>remote 135.181.204.42 1194</code>	<code># Gibt an mit welcher</code>
<code>nobind</code>	<code># Veranlasst OpenVPN</code>
<code>persist-key</code>	<code># Versucht Zustände ü</code>
<code>persist-tun</code>	<code># Versucht Zustände ü</code>
<code>ca ca.crt</code>	<code># Gibt den Pfad zur Z</code>
<code>cert issued/client-g1.crt</code>	<code># Gibt den Pfad zur Z</code>
<code>key private/client-g1.key</code>	<code># Gibt den Pfad zur K</code>
<code>comp-lzo</code>	<code># Definiert dass kein</code>
<code>verb 3</code>	<code># Definiert die Ausfü</code>

Versuchen Sie ebenfalls mit einem Windows-Client eine Verbindung zu Ihrem Server aufzubauen. Die Client-Software können Sie von: <https://openvpn.net/index.php/open-source/downloads.html> herunterladen.



# Analyse

---

## Analyse der Logs

**Inspizieren Sie die Log-Statements des Servers und des Clients. Ist ein Tunnel etabliert?**

Client-Log:

```
# sudo openvpn --config client.conf
[sudo] password for root:
2021-11-30 15:58:20 WARNING: Compression for receive
2021-11-30 15:58:20 --cipher is not set. Previous O
2021-11-30 15:58:20 OpenVPN 2.5.3 x86_64-suse-linux-
2021-11-30 15:58:20 library versions: OpenSSL 1.1.1
24 Aug 2021, LZO 2.10
2021-11-30 15:58:20 WARNING: No server certificate
See http://openvpn.net/howto.html#mitm for more info
2021-11-30 15:58:20 TCP/UDP: Preserving recently us
2021-11-30 15:58:20 Socket Buffers: R=[212992->2129
2021-11-30 15:58:20 UDP link local (not bound)
```

## Funktionstest

Überprüfen Sie mit den Tools `ip link`, `ip address` und `ip route` die erzeugten Netzwerkkonfigurationen. Im Anschluss überprüfen Sie die Funktion des Tunnels mit einem Ping vom Client auf das `tun0` Device des Servers.

Zuerst verwenden wir `ip a`:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: enp2s0f0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500  
    link/ether 84:a9:38:67:f2:18 brd ff:ff:ff:ff:ff:ff  
3: wlp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500  
    link/ether 00:04:23:04:62:52 brd ff:ff:ff:ff:ff:ff
```

# Betrachtung via Wireshark

---

# Betrachtung via Wireshark

Stellen Sie den Unterschied der Datenpakete (verschlüsselt, unverschlüsselt) mit Wireshark dar. Nutzen Sie dazu einen einfachen ping-Befehl. Beachten Sie, dass der Verkehr für Wireshark auf unterschiedlichen Interfaces stattfindet.

The screenshot displays two windows. The top window is a terminal running a ping command on a Fedora system. The bottom window is Wireshark, capturing traffic on the 'enp3s0' interface.

**Terminal Output:**

```
danny@fedora:~/tmp/g1$ sudo openvpn
[danny@localhost g1]$ ping 10.8.1.1
PING 10.8.1.1 (10.8.1.1) 56(84) bytes of data.
 64 bytes from 10.8.1.1: icmp seq=1 ttl=64 time=45.2 ms
 64 bytes from 10.8.1.1: icmp seq=2 ttl=64 time=39.1 ms
 64 bytes from 10.8.1.1: icmp seq=3 ttl=64 time=43.7 ms
 64 bytes from 10.8.1.1: icmp seq=4 ttl=64 time=43.9 ms
 64 bytes from 10.8.1.1: icmp seq=5 ttl=64 time=43.4 ms
 64 bytes from 10.8.1.1: icmp seq=6 ttl=64 time=43.1 ms
 64 bytes from 10.8.1.1: icmp seq=7 ttl=64 time=42.2 ms
 64 bytes from 10.8.1.1: icmp seq=8 ttl=64 time=43.1 ms
 64 bytes from 10.8.1.1: icmp seq=9 ttl=64 time=44.2 ms
 64 bytes from 10.8.1.1: icmp seq=10 ttl=64 time=41.8 ms
 64 bytes from 10.8.1.1: icmp seq=11 ttl=64 time=41.9 ms
^C
... 10.8.1.1 ping statistics ...
11 packets transmitted, 11 received, 0% packet loss, time 10011ms
rtt min/avg/max/mdev = 39.118/42.868/45.201/1.546 ms
[danny@localhost g1]$
```

**Wireshark Packet List:**

No.	Time	Source	Destination	Protocol	Length	Info
9360	0.00000133	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P_DATA_V2
9361	0.000014748	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P_DATA_V2
9364	0.000021843	192.168.178.23	135.181.204.42	OpenVPN	151	MessageType: P_DATA_V2
9401	0.000040593	192.168.178.23	135.181.204.42	OpenVPN	135	MessageType: P_DATA_V2
9402	0.000077717	192.168.178.23	135.181.204.42	OpenVPN	135	MessageType: P_DATA_V2
9408	0.000083911	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P_DATA_V2
9425	0.00008688	192.168.178.23	135.181.204.42	OpenVPN	151	MessageType: P_DATA_V2
9485	0.000221142	192.168.178.23	135.181.204.42	OpenVPN	151	MessageType: P_DATA_V2
9701	0.000000832	192.168.178.23	135.181.204.42	OpenVPN	135	MessageType: P_DATA_V2
9878	0.000001811	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P_DATA_V2
9879	0.000000923	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P_DATA_V2
9913	0.000001197	192.168.178.23	135.181.204.42	OpenVPN	151	MessageType: P_DATA_V2
10004	0.000001039	192.168.178.23	135.181.204.42	OpenVPN	151	MessageType: P_DATA_V2
10181	0.000001522	192.168.178.23	135.181.204.42	OpenVPN	151	MessageType: P_DATA_V2
10209	0.000003726	192.168.178.23	135.181.204.42	OpenVPN	119	MessageType: P_DATA_V2
10324	0.000002976	192.168.178.23	135.181.204.42	OpenVPN	151	MessageType: P_DATA_V2
10438	0.000003827	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P_DATA_V2
10481	0.000003039	192.168.178.23	135.181.204.42	OpenVPN	151	MessageType: P_DATA_V2
10674	0.000044754	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P_DATA_V2
10685	0.000003089	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P_DATA_V2
10687	0.000029543	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P_DATA_V2
10688	0.000042199	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P_DATA_V2
10739	0.000007844	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P_DATA_V2
10838	0.000076443	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P_DATA_V2
11033	0.000076442	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P_DATA_V2
11048	0.000076320	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P_DATA_V2
11052	0.000078448	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P_DATA_V2
11057	0.000015785	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P_DATA_V2

**Wireshark Packet Details:**

- Frame 13544: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface enp3s0, 16 B
- Ethernet II, Src: VMXnet3 adapter (00:0c:29:14:78:00), Dst: Micro-SG (08:00:27:00:00:00:00:00:00:00:00:00)
- Internet Protocol Version 4, Src: 192.168.178.23, Dst: 135.181.204.42
- User Datagram Protocol, Src Port: 1194, Dst Port: 55744
- Source Port: 1194
- Destination Port: 55744
- Length: 60
- Checksum: 8x92a2 [unverified]
- Checksum Status: Unverified
- Stream index: 31
- [TimeLaps]
- UDP payload (36 bytes)
- OpenVPN Protocol
- length: 60
- 0018 0... = Opcode: P\_DATA\_V1 (0x00)
- .... 0000 = Key ID: 0
- = Data (37 bytes)



# Erweiterte Konfiguration

---

## Erweiterte Konfiguration

\*\* Bis hierher haben wir nur Datenverbindung vom Client bis zum Server realisiert (In der Grafik grün dargestellt). Der Sinn einer VPN-Verbindung ist häufig die Network-to-Network-Anbindung. Eine ähnliche Verbindung ist eine Client-Verbindung über den VPN-Server nach draußen ins Internet. Folgende Grafik veranschaulicht die gewünschte Verbindung (rot dargestellt):\*\*



## Änderung der Konfiguration

Die Datei `server.conf` muss um die IP des servers von `api.ipify.org` erweitert werden. Mit Dig können die IPs der Server verwendet werden. Wir erhalten hier mehrere IPs, da anscheinend Loadbalancing verwendet wird:

```
# dig api.ipify.org
```

```
; <<>> DiG 9.16.23-RH <<>> api.ipify.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id:
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY:

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
```

**Starten Sie den Open-VPN Client neu. Überprüfen Sie die Routen.**

Nach dem Neustarten des Clients sehen die Routen wie folgt aus:

```
# ip route get 54.91.59.199
54.91.59.199 via 10.8.1.5 dev tun0 src 10.8.1.6 uid
    cache
```

```
# ip route get 52.20.78.240
52.20.78.240 via 10.8.1.5 dev tun0 src 10.8.1.6 uid
    cache
```

```
# ip route get 3.232.242.170
3.232.242.170 via 10.8.1.5 dev tun0 src 10.8.1.6 uid
    cache
```

```
# ip route get 3.220.57.224
```

# Zugriffsbeschränkung

---

## Zugriffsbeschränkung

**\*\*** Angenommen ein Client soll keinen Zugriff mehr über Ihren OpenVPN-Server erhalten. Wie verhindern Sie das, ohne dass Sie Zugang zum Client bekommen? Am Ende des Versuchs können sie die Methode für alle vergebenen Client-Zertifikate durchführen und testen. Können Sie diesen Vorgang wieder rückgängig machen, so das der Client wieder am VPN „teilnehmen“ kann?**\*\***

### Widerruf

Wenn wir das Zertifikat widerrufen, führt dies dazu, dass das Zertifikat ungültig wird und nicht mehr für Authentifizierungszwecke genutzt werden kann.

Dies kann mit folgendem Kommando geschehen:

```
# ./revoke--full client --g1
```

Durch das vorangegangene Kommando wurde eine CRL-Datei