
Praktikum Rechnernetze

Protokoll zu Versuch 7 (OpenVPN) von Gruppe 1

Jakob Waibel, Daniel Hiller, Elia Wüstner, Felix Pojtinger

2021-11-30

Inhaltsverzeichnis

1 Einführung	3
1.1 Mitwirken	3
1.2 Lizenz	3
2 CA (=Zertifizierungsstelle) und Schlüssel erzeugen und signieren	4
2.1 Fragen zur Aufgabe	13
3 Konfiguration von Client und Server	14
3.1 Server konfigurieren	14
3.2 Erklären Sie die einzelnen Parameter/Optionen der „server.conf“ und der „client.conf“.	17
3.3 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.	20
4 Analyse	21
4.1 Analyse der Logs	21
4.2 Funktionstest	24
5 Betrachtung via Wireshark	26
6 Erweiterte Konfiguration	28
6.1 Änderung der Konfiguration	29
6.2 Funktionstest	30
7 Zugriffsbeschränkung	31

1 Einführung

1.1 Mitwirken

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/pojntfx/uni-netpractice-notes):



Abbildung 1: QR-Code zum Quelltext auf GitHub

Wenn Ihnen die Materialien gefallen, würden wir uns über einen GitHub-Stern sehr freuen.

1.2 Lizenz

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



Abbildung 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

2 CA (=Zertifizierungsstelle) und Schlüssel erzeugen und signieren

Verzeichnis erstellen und betreten:

```
1 # mkdir openvpn
2 # cd openvpn
```

Git installieren:

```
1 apt install git
```

Repository klonen:

```
1 # git clone https://github.com/OpenVPN/easy-rsa
2 Cloning into 'easy-rsa'...
3 remote: Enumerating objects: 2095, done.
4 remote: Counting objects: 100% (13/13), done.
5 remote: Compressing objects: 100% (11/11), done.
6 remote: Total 2095 (delta 3), reused 4 (delta 0), pack-reused 2082
7 Receiving objects: 100% (2095/2095), 11.72 MiB | 7.01 MiB/s, done.
8 Resolving deltas: 100% (914/914), done.
```

Verschieben und umbenennen einiger Ordner:

```
1 # mv easy-rsa/easyrsa3 easyrsa && rm -r easy-rsa && cd easyrsa
```

PKI-Infrastruktur erstellen:

```
1 root@gl1:~/openvpn/easyrsa# ./easyrsa init-pki
2
3 init-pki complete; you may now create a CA or requests.
4 Your newly created PKI dir is: /root/openvpn/easyrsa/pki
```

Zertifizierungsstelle erstellen:

```
1 # ./easyrsa build-ca
2 Using SSL: openssl OpenSSL 1.1.0l 10 Sep 2019
3
4 Enter New CA Key Passphrase:
5 Re-Enter New CA Key Passphrase:
6 Generating RSA private key, 2048 bit long modulus
7 .....+++++
8 .....+++++
9 e is 65537 (0x010001)
10 You are about to be asked to enter information that will be
    incorporated
11 into your certificate request.
12 What you are about to enter is what is called a Distinguished Name or a
    DN.
13 There are quite a few fields but you can leave some blank
```

```
14 For some fields there will be a default value,
15 If you enter '.', the field will be left blank.
16 -----
17 Common Name (eg: your user, host, or server name) [Easy-RSA CA]:g1.mi.
   hdm-stuttgart.de
18 CA creation complete and you may now import and sign cert requests.
19 Your new CA certificate file for publishing is at:
20 /root/openvpn/easyrsa/pki/ca.crt
```

Entfernen der Passphrase vom RSA Private Key:

```
1 # openssl rsa -in pki/private/ca.key -out pki/private/ca.key
2 Enter pass phrase for pki/private/ca.key:
3 writing RSA key
```

Keypair für Server generieren:

```
1 root@g1:~/openvpn/easyrsa# ./easyrsa gen-req server-g1 nopass
2 Using SSL: openssl OpenSSL 1.1.0l 10 Sep 2019
3 Generating a RSA private key
4 .....
5 .....
6 writing new private key to '/root/openvpn/easyrsa/pki/easy-rsa-4141.
   HiTBNm/tmp.JnuJdp'
7 -----
8 You are about to be asked to enter information that will be
   incorporated
9 into your certificate request.
10 What you are about to enter is what is called a Distinguished Name or a
   DN.
11 There are quite a few fields but you can leave some blank
12 For some fields there will be a default value,
13 If you enter '.', the field will be left blank.
14 -----
15 Common Name (eg: your user, host, or server name) [server-g1]:g1.mi.hdm
   -stuttgart.de
16
17 Keypair and certificate request completed. Your files are:
18 req: /root/openvpn/easyrsa/pki/reqs/server-g1.req
19 key: /root/openvpn/easyrsa/pki/private/server-g1.key
```

Keypair für Client generieren:

```
1 root@gl:~/openvpn/easyrsa# ./easyrsa gen-req client-g1 nopass
2 Using SSL: openssl OpenSSL 1.1.0l 10 Sep 2019
3 Generating a RSA private key
4 .....+++++
5 .....
6 writing new private key to '/root/openvpn/easyrsa/pki/easy-rsa-4162.
   amBaR1/tmp.1aHVQ6'
7 -----
8 You are about to be asked to enter information that will be
   incorporated
9 into your certificate request.
10 What you are about to enter is what is called a Distinguished Name or a
   DN.
11 There are quite a few fields but you can leave some blank
12 For some fields there will be a default value,
13 If you enter '.', the field will be left blank.
14 -----
15 Common Name (eg: your user, host, or server name) [client-g1]:gl.mi.hdm
   -stuttgart.de
16
17 Keypair and certificate request completed. Your files are:
18 req: /root/openvpn/easyrsa/pki/reqs/client-g1.req
19 key: /root/openvpn/easyrsa/pki/private/client-g1.key
```

Analog zur vorherigen Erstellung des Keypairs für den Client haben wir zwei weitere Keypairs `client-g1-2` und `client-g1-3` erstellt.

Einzelne Zertifikate signieren:

Server-Zertifikat:

```
1 # ./easyrsa sign server server-g1
2 Using SSL: openssl OpenSSL 1.1.0l 10 Sep 2019
3
4
5 You are about to sign the following certificate.
6 Please check over the details shown below for accuracy. Note that this
   request
7 has not been cryptographically verified. Please be sure it came from a
   trusted
8 source or that you have verified the request checksum with the sender.
9
10 Request subject, to be signed as a server certificate for 825 days:
11
12 subject=
13     commonName                = gl.mi.hdm-stuttgart.de
14
15
```

```
16 Type the word 'yes' to continue, or any other input to abort.
17 Confirm request details: yes
18 Using configuration from /root/openvpn/easyrsa/pki/easy-rsa-4226.9Qm0XH
   /tmp.jenIQd
19 Check that the request matches the signature
20 Signature ok
21 The Subject's Distinguished Name is as follows
22 commonName               :ASN.1 12:'gl.mi.hdm-stuttgart.de'
23 Certificate is to be certified until Mar  4 13:45:08 2024 GMT (825 days
   )
24
25 Write out database with 1 new entries
26 Data Base Updated
27
28 Certificate created at: /root/openvpn/easyrsa/pki/issued/server-gl.crt
```

Client-Zertifikat:

```
1 # ./easyrsa sign client client-gl
2 Using SSL: openssl OpenSSL 1.1.0l 10 Sep 2019
3
4
5 You are about to sign the following certificate.
6 Please check over the details shown below for accuracy. Note that this
   request
7 has not been cryptographically verified. Please be sure it came from a
   trusted
8 source or that you have verified the request checksum with the sender.
9
10 Request subject, to be signed as a client certificate for 825 days:
11
12 subject=
13 commonName               = gl.mi.hdm-stuttgart.de
14
15
16 Type the word 'yes' to continue, or any other input to abort.
17 Confirm request details: yes
18 Using configuration from /root/openvpn/easyrsa/pki/easy-rsa-4288.S5KRe9
   /tmp.oh8FLq
19 Check that the request matches the signature
20 Signature ok
21 The Subject's Distinguished Name is as follows commonName           :
   ASN.1 12:'gl.mi.hdm-stuttgart.de'
22 Certificate is to be certified until Mar  4 13:45:38 2024 GMT (825 days
   )
23
24 Write out database with 1 new entries
25 Data Base Updated
26
27 Certificate created at: /root/openvpn/easyrsa/pki/issued/client-gl.crt
```

Analog wurden auch die zwei zusätzlich generierten Zertifikate signiert.

Generieren der Diffie-Hellman-Parameter:

```
1 # ./easysrsa gen-dh
2 Using SSL: openssl OpenSSL 1.1.0l 10 Sep 2019
3 Generating DH parameters, 2048 bit long safe prime, generator 2
4 This is going to take a long time
5 .....+.....+.....
6
7
8 DH parameters of size 2048 created at /root/openvpn/easysrsa/pki/dh.pem
```

Gültigkeit der signierten Zertifikate prüfen:

Server-Zertifikat:

```
1 # openssl x509 -in pki/issued/server-g1.crt -text -noout
2 Certificate:
3 Data:
4 Version: 3 (0x2)
5 Serial Number:
6 98:f4:e5:df:00:4d:ce:24:42:e2:26:ae:fe:64:14:bc
7 Signature Algorithm: sha256WithRSAEncryption
8 Issuer: CN = g1.mi.hdm-stuttgart.de
9 Validity
10 Not Before: Nov 30 13:45:08 2021 GMT
11 Not After : Mar  4 13:45:08 2024 GMT
12 Subject: CN = g1.mi.hdm-stuttgart.de
13 Subject Public Key Info:
14 Public Key Algorithm: rsaEncryption
15 Public-Key: (2048 bit)
16 Modulus:
17 00:e3:ee:11:d2:55:a1:cb:fc:2f:78:0c:e7:d5:8b:
18 e2:a0:4b:b4:65:61:8a:75:49:35:1d:69:dd:d2:b9:
19 e9:3e:a8:f1:06:11:3b:d3:84:aa:89:e9:ae:c5:de:
20 ed:37:e4:3f:b3:c0:aa:27:5e:ab:a6:a1:3f:eb:c1:
21 65:89:c1:6a:65:8b:28:10:74:eb:44:50:96:ce:5f:
22 1d:5f:f7:0c:d1:a0:d4:22:2e:46:39:11:fc:89:5e:
23 68:9b:79:8c:28:d0:ea:3c:a2:02:c6:9e:ce:db:d6:
24 3d:5f:e7:2a:ed:02:d9:cb:3e:4d:0a:c1:c6:4e:35:
25 b7:1d:fe:8e:08:c2:ee:a1:b2:a9:7c:66:9f:b3:1b:
26 3b:20:4d:f4:b0:71:b4:5e:b5:4e:62:88:90:bb:f2:
27 87:cd:ba:63:29:68:af:65:96:14:08:2f:78:a3:0d:
28 3c:9b:c8:ac:fa:b3:2a:ed:ff:14:ce:01:af:8c:45:
29 e3:29:4e:3c:19:9b:6a:6e:40:6a:2f:86:ca:6c:9e:
30 1d:dd:ed:2f:72:c6:7a:3e:8a:8d:08:e2:e6:76:b6:
31 33:95:23:54:9a:e6:ea:4e:17:0c:08:c5:86:38:00:
32 2f:d5:70:0e:77:db:47:c4:48:a1:5e:6f:c6:95:1a:
33 4c:9b:b5:5b:41:fb:9d:99:23:8c:f0:55:37:eb:a7:
34 06:cf
```



```
35 Exponent: 65537 (0x10001)
36 X509v3 extensions:
37 X509v3 Basic Constraints:
38 CA:FALSE
39 X509v3 Subject Key Identifier:
40 EF:0D:06:76:FE:C8:B5:0B:3B:1E:D6:E9:98:94:25:29:11:C0:84:72
41 X509v3 Authority Key Identifier:
42 keyid:3F:CC:11:C2:51:85:77:9E:D1:D5:5F:0B:D6:D2:09:7F:79:D8:4D:4C
43 DirName:/CN=g1.mi.hdm-stuttgart.de
44 serial:AE:98:7F:B2:0A:A6:16:A4
45
46 X509v3 Extended Key Usage:
47 TLS Web Server Authentication
48 X509v3 Key Usage:
49 Digital Signature, Key Encipherment
50 X509v3 Subject Alternative Name:
51 DNS:g1.mi.hdm-stuttgart.de
52 Signature Algorithm: sha256WithRSAEncryption
53 97:13:f0:05:3a:91:b5:e7:69:5a:cb:53:70:8e:a4:f5:92:5e:
54 ed:9f:44:a9:11:6c:3f:0f:f8:b2:9e:1b:58:3f:17:a8:e4:9e:
55 79:44:be:c3:63:0c:c7:69:e4:1f:c8:39:8d:3c:3e:cd:ee:ff:
56 a5:88:67:ec:8a:df:6a:c7:45:68:3a:1b:97:b1:21:ba:0f:4f:
57 f2:f2:9d:27:71:cd:a8:f1:14:74:90:96:49:50:de:63:77:ad:
58 b1:87:be:4d:3e:78:59:cd:97:2c:08:b5:0c:f6:48:36:42:97:
59 6c:ab:79:9f:cf:b4:e7:3e:40:ca:65:fe:3f:2d:7a:f3:fc:20:
60 69:84:7b:e2:41:7e:22:db:52:6e:7f:be:9d:21:12:42:92:e3:
61 02:0d:47:3e:42:8a:42:d1:23:5e:c5:6f:16:3b:36:ce:84:c0:
62 71:d8:c6:97:f9:8f:fe:a2:44:92:5b:ee:cd:1b:f2:26:11:84:
63 d6:03:58:eb:44:d7:8e:8f:f7:74:fd:3d:98:17:2e:e1:81:42:
64 d6:77:fc:80:6d:2f:7e:8d:5e:fe:d1:8e:be:ba:9a:01:f8:01:
65 57:cb:5f:09:53:4d:f3:36:e9:cc:ac:25:d0:a2:54:27:28:c6:
66 4b:30:51:e0:13:6a:f2:73:3a:d4:2e:0f:44:72:07:73:56:98:
67 cd:7c:75:35
```

Client-Zertifikat:

```
1 # openssl x509 -in pki/issued/client-g1.crt -text -noout
2 Certificate:
3 Data:
4 Version: 3 (0x2)
5 Serial Number:
6 7d:31:54:75:d7:f4:b3:71:9d:9a:30:52:13:1f:f4:b6
7 Signature Algorithm: sha256WithRSAEncryption
8 Issuer: CN = g1.mi.hdm-stuttgart.de
9 Validity
10 Not Before: Nov 30 13:45:38 2021 GMT
11 Not After : Mar 4 13:45:38 2024 GMT
12 Subject: CN = g1.mi.hdm-stuttgart.de
13 Subject Public Key Info:
14 Public Key Algorithm: rsaEncryption
15 Public-Key: (2048 bit)
```

```
16 Modulus:
17 00:de:a5:7f:88:cb:40:dc:92:76:7b:ac:67:38:ad:
18 b8:e5:86:8d:18:e7:ca:35:ba:5f:92:a3:89:d9:18:
19 58:51:79:c2:5e:02:0c:f3:96:4e:1e:fc:73:9b:0c:
20 d9:3f:05:6d:7d:23:15:38:f5:0f:55:89:86:b3:6c:
21 ac:38:cc:85:8d:3f:97:ec:f6:0e:a7:5e:6e:39:fb:
22 bd:e5:78:ac:0c:04:b8:c9:ac:29:8c:84:90:8b:de:
23 3a:e6:83:b9:c3:82:48:9c:a1:71:d7:0b:15:ef:13:
24 f6:e7:59:84:bb:c9:7e:c3:69:ae:92:1e:f7:b6:39:
25 45:a1:63:72:25:41:a4:30:85:c3:ba:75:23:24:4b:
26 9c:98:58:90:98:38:40:65:1d:09:21:bf:36:9b:3d:
27 f7:2a:65:80:e8:84:67:5b:83:f3:b9:b7:8f:9b:03:
28 d0:db:23:7b:40:4d:f0:9c:c0:a9:81:26:0e:00:7c:
29 24:dd:ee:b0:d8:c5:f2:bf:be:f5:18:86:67:6c:0c:
30 b8:ab:f9:41:a0:c7:60:e2:d2:9c:32:6a:2f:8c:94:
31 c8:bb:1c:2c:80:5c:3b:20:b9:bf:a7:16:80:60:eb:
32 6c:f5:de:cb:34:c1:cc:89:ee:f0:bf:60:7b:56:ef:
33 1b:ca:f0:73:57:ba:b0:0d:a0:52:78:02:a1:d7:7f:
34 78:bf
35 Exponent: 65537 (0x10001)
36 X509v3 extensions:
37 X509v3 Basic Constraints:
38 CA:FALSE
39 X509v3 Subject Key Identifier:
40 A3:F6:BE:EC:50:50:92:A0:A3:1E:12:1A:00:A1:D1:53:B8:33:90:0F
41 X509v3 Authority Key Identifier:
42 keyid:3F:CC:11:C2:51:85:77:9E:D1:D5:5F:0B:D6:D2:09:7F:79:D8:4D:4C
43 DirName:/CN=gl.mi.hdm-stuttgart.de
44 serial:AE:98:7F:B2:0A:A6:16:A4
45
46 X509v3 Extended Key Usage:
47 TLS Web Client Authentication
48 X509v3 Key Usage:
49 Digital Signature
50 Signature Algorithm: sha256WithRSAEncryption
51 6d:18:13:c6:7d:04:58:f8:69:54:c0:74:a1:ec:5c:19:44:74:
52 a5:22:ff:ac:41:96:ca:23:50:4a:14:61:3a:4e:e8:d8:23:03:
53 5e:0c:2b:df:48:db:6c:f0:53:ab:36:57:9f:44:d5:f1:71:ae:
54 24:43:c9:86:52:d1:87:2a:5e:d8:a5:6e:90:c9:86:cc:44:b9:
55 69:2d:47:2a:94:87:46:29:00:8e:32:1b:8c:5e:cf:82:d8:e0:
56 d2:d1:85:87:94:a7:bc:53:c9:8b:eb:74:d2:59:be:44:92:72:
57 7a:85:ce:4c:40:ba:9f:fa:6b:e3:08:da:6a:e6:3b:34:4a:18:
58 25:7a:3d:2a:a1:8c:ad:47:c1:76:cc:a8:5b:46:38:3d:ee:0c:
59 35:68:c4:2f:a1:3b:66:64:e8:88:7a:1e:21:22:99:6e:4d:f2:
60 f1:55:d5:c3:25:ce:ac:27:2b:76:1e:11:6e:5b:78:4f:7b:1e:
61 2b:1e:5f:13:0c:b5:4e:0a:4f:b7:df:e6:85:ef:88:cd:9e:21:
62 e5:70:53:20:16:33:4b:6b:67:28:c7:0c:f5:bd:f6:38:30:47:
63 5a:44:99:c5:28:57:47:88:72:b9:de:a8:ae:ed:d3:c1:78:23:
64 07:9b:d5:2b:92:3f:ad:d8:88:f2:6e:e8:5a:0e:27:d8:7c:b2:
65 94:b5:27:ef
```

Analog wurden auch die weiteren Zertifikate geprüft. Den Daten kann entnommen werden, dass die Zertifikate bis `Mar 4 13:45:38 2024 GMT` gültig sind. Die Zertifikate wurden von `g1.mi.hdm-stuttgart.de` signiert.

Die Ordnerstruktur sieht wie folgt aus:

```
# tree .
.
├── easysrsa
├── openssl-easysrsa.cnf
├── pki
│   ├── ca.crt
│   ├── certs_by_serial
│   │   ├── 1269D2C7D013DEA54189F3EDB5CDD59.pem
│   │   ├── 7D315475D7F4B3719D9A3052131FF4B6.pem
│   │   ├── 8E88712C77364F4C60E4A61AA654D8DE.pem
│   │   └── 98F4E5DF004DCE2442E226AEFE6414BC.pem
│   ├── dh.pem
│   ├── index.txt
│   ├── index.txt.attr
│   ├── index.txt.attr.old
│   ├── index.txt.old
│   ├── issued
│   │   ├── client-g1-2.crt
│   │   ├── client-g1-3.crt
│   │   ├── client-g1.crt
│   │   └── server-g1.crt
│   ├── openssl-easysrsa.cnf
│   ├── private
│   │   ├── ca.key
│   │   ├── client-g1-2.key
│   │   ├── client-g1-3.key
│   │   ├── client-g1.key
│   │   └── server-g1.key
│   ├── renewed
│   │   ├── certs_by_serial
│   │   ├── private_by_serial
│   │   └── reqs_by_serial
│   ├── reqs
│   │   ├── client-g1-2.req
│   │   ├── client-g1-3.req
│   │   ├── client-g1.req
│   │   └── server-g1.req
│   ├── revoked
│   │   ├── certs_by_serial
│   │   ├── private_by_serial
│   │   └── reqs_by_serial
│   ├── safessl-easysrsa.cnf
│   ├── serial
│   └── serial.old
├── vars.example
├── x509-types
│   ├── ca
│   │   └── client
│   │       ├── code-signing
│   │       │   └── COMMON
│   │       │       ├── email
│   │       │       │   └── kdc
│   │       │       │       ├── server
│   │       │       │       └── serverClient
│   └── client
│       ├── code-signing
│       │   └── COMMON
│       │       ├── email
│       │       │   └── kdc
│       │       │       ├── server
│       │       │       └── serverClient
└── 14 directories, 38 files
```

Abbildung 3: Ordnerstruktur

Abschließend verschieben wir den `pki` Ordner ins Home-Verzeichnis:

```
1 # mv pki ~/pki
```

Aus der bisherigen Struktur generieren wir ein `client package`. Dafür muss man sich im `pki` Ordner befinden:

```
1 tar cf gl.tar ca.crt private/client-gl.key issued/client-gl.crt
```

2.1 Fragen zur Aufgabe

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 Zertifikate mit ihrer Seriennummer.

`dh.pem` enthält die Parameter für den Diffie-Hellman-Key-Exchange.

`index.txt` ist die "Master-Datenbank" aller Zertifikate.

In `reqs` sind die Certificate Signing Requests (CSR) enthalten.

`renewed` und `revoked` enthalten Informationen über erneuerte und ungültig gemachte Zertifikate.

Wie ist der Ablauf bei der Erstellung eines eigenen Zertifikates (gemeint sind die Schritte bei der Erstellung)?

Wir benötigen einen separaten Public Key und Private Key für den Server und jeden Client. Außerdem braucht es noch das Zertifikat und Key der CA, um alle Server und Client Zertifikate zu signieren. Bevor sich beide Parteien vertrauen muss der Client die Server Zertifikate authentifizieren und der Server muss die Client Zertifikate authentifizieren. Dieses gegenseitige Authentifizieren erfolgt durch das Sicherstellen, dass ein Zertifikat, welches man bekommt bereits von der CA signiert wurde. Danach kann der Inhalt in dem neu authentifizierten Zertifikat Header, wie z.B. der certificate common name getestet werden.

Schildern Sie den Ablauf der Authentisierung, des Schlüsselaustausches und der Verschlüsselung bei der Verwendung von Zertifikats-basierter Authentisierung in OpenVPN!

Als erstes tauschen Client und Server die Schlüsselpaare aus und verifizieren die Zertifikate. Der Client initiiert mit einer Anfrage an den Server die Verbindung. Der Server verifiziert sich mittels seiner eigenen "certificate chain". Mithilfe der eigenen Kopie des CA Files kann der Client die "certificate chain" überprüfen. Sofern dies klappt, erfolgt der Vorgang erneut umgekehrt, indem der Server die Client "certificate chain" checkt. CCD Dateien werden überprüft. Sie ermöglichen das Vergeben von spezifischen IP-Adressen an einen Client, um z.B. einen DNS Server einem bestimmten Client zuzuordnen oder einen Client zeitweise zu deaktivieren. Falls keine Fehler entstehen, kann die Verbindung aufgebaut werden.

Was bewirkt die Option „nopass“ bei der Keypair-Erzeugung und ist diese sinnvoll?

Schlüsselpaare werden mit dem Argument "nopass" unverschlüsselt gelassen, da Server in der Regel ohne Passworteingabe gestartet werden. Dadurch wird ein unverschlüsselter Schlüssel erzeugt, dessen Zugriff und Dateiberechtigungen daher sorgfältig geschützt werden muss.

Erstellen Sie die CA + Keys + Zertifikate auf dem Server. Das sollte man eigentlich nicht machen, warum?

3 Konfiguration von Client und Server

3.1 Server konfigurieren

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

```
1 # cat server.conf
2 proto udp
3 dev tun
4 ca pki/ca.crt
5 cert pki/issued/server-gl.crt
6 key pki/private/server-gl.key
7 dh pki/dh.pem
8 server 10.8.1.0 255.255.255.0
9 keepalive 10 120
10 comp-lzo
11 persist-key
12 persist-tun
13 verb 3
```

Jetzt kann der OpenVPN-Server gestartet werden:

```
1 # sudo openvpn --config server.conf
2 Tue Nov 30 14:20:50 2021 OpenVPN 2.4.0 x86_64-pc-linux-gnu [SSL (
    OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on
    Oct 14 2018
3 Tue Nov 30 14:20:50 2021 library versions: OpenSSL 1.0.2u 20 Dec 2019,
    LZO 2.08
4 Tue Nov 30 14:20:50 2021 Diffie-Hellman initialized with 2048 bit key
5 Tue Nov 30 14:20:50 2021 ROUTE_GATEWAY 172.31.1.1
6 Tue Nov 30 14:20:50 2021 TUN/TAP device tun0 opened
7 Tue Nov 30 14:20:50 2021 TUN/TAP TX queue length set to 100
8 Tue Nov 30 14:20:50 2021 do_ifconfig, tt->did_ifconfig_ipv6_setup=0
9 Tue Nov 30 14:20:50 2021 /sbin/ip link set dev tun0 up mtu 1500
10 Tue Nov 30 14:20:50 2021 /sbin/ip addr add dev tun0 local 10.8.1.1 peer
    10.8.1.2
11 Tue Nov 30 14:20:50 2021 /sbin/ip route add 10.8.1.0/24 via 10.8.1.2
12 Tue Nov 30 14:20:50 2021 Could not determine IPv4/IPv6 protocol. Using
    AF_INET
13 Tue Nov 30 14:20:50 2021 Socket Buffers: R=[212992->212992] S
    =[212992->212992]
14 Tue Nov 30 14:20:50 2021 UDPv4 link local (bound): [AF_INET][undef
    ]:1194
15 Tue Nov 30 14:20:50 2021 UDPv4 link remote: [AF_UNSPEC]
16 Tue Nov 30 14:20:50 2021 MULTI: multi_init called, r=256 v=256
17 Tue Nov 30 14:20:50 2021 IFCONFIG POOL: base=10.8.1.4 size=62, ipv6=0
18 Tue Nov 30 14:20:50 2021 Initialization Sequence Completed
```

Mit `scp` senden wir das `client package` mit dem Namen `g1.tar` zu unserem Client:

```
1 cp root@135.181.204.42:~/pki/g1.tar ~/
2 root@135.181.204.42's password:
3 g1.tar 100% 20
    KB 389.8KB/s 00:00
```

Nachdem wir die Datei entpackt haben, erstellen wir unser `client.conf` file:

```
1 # cat client.conf
2 client
3 dev tun
4 proto udp
5 remote 135.181.204.42 1194
6 nobind
7 persist-key
8 persist-tun
9 ca ca.crt
10 cert issued/client-g1.crt
11 key private/client-g1.key
12 comp-lzo
13 verb 3
```

Jetzt können wir den client mit unserer Konfiguration starten. Dabei ist zu beachten, dass der OpenVPN-Server auch bereits laufen muss:

```
1 # sudo openvpn --config client.conf
2 [sudo] password for root:
3 2021-11-30 15:58:20 WARNING: Compression for receiving enabled.
   Compression has been used in the past to break encryption. Sent
   packets are not compressed unless "allow-compression yes" is also
   set.
4 2021-11-30 15:58:20 --cipher is not set. Previous OpenVPN version
   defaulted to BF-CBC as fallback when cipher negotiation failed in
   this case. If you need this fallback please add '--data-ciphers-
   fallback BF-CBC' to your configuration and/or add BF-CBC to --data-
   ciphers.
5 2021-11-30 15:58:20 OpenVPN 2.5.3 x86_64-suse-linux-gnu [SSL (OpenSSL)]
   [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on Jun 17
   2021
6 2021-11-30 15:58:20 library versions: OpenSSL 1.1.1l 24 Aug 2021, LZO
   2.10
7 2021-11-30 15:58:20 WARNING: No server certificate verification method
   has been enabled. See http://openvpn.net/howto.html#mitm for more
   info.
8 2021-11-30 15:58:20 TCP/UDP: Preserving recently used remote address: [
   AF_INET]135.181.204.42:1194
9 2021-11-30 15:58:20 Socket Buffers: R=[212992->212992] S
   =[212992->212992]
10 2021-11-30 15:58:20 UDP link local: (not bound)
11 2021-11-30 15:58:20 UDP link remote: [AF_INET]135.181.204.42:1194
12 2021-11-30 15:58:20 TLS: Initial packet from [AF_INET
   ]135.181.204.42:1194, sid=1cf1ee33 f316b385
13 2021-11-30 15:58:20 VERIFY OK: depth=1, CN=g1.mi.hdm-stuttgart.de
14 2021-11-30 15:58:20 VERIFY OK: depth=0, CN=g1.mi.hdm-stuttgart.de
15 2021-11-30 15:58:20 Control Channel: TLSv1.2, cipher TLSv1.2 ECDHE-RSA-
   AES256-GCM-SHA384, peer certificate: 2048 bit RSA, signature: RSA-
   SHA256
16 2021-11-30 15:58:20 [g1.mi.hdm-stuttgart.de] Peer Connection Initiated
   with [AF_INET]135.181.204.42:1194
17 2021-11-30 15:58:21 SENT CONTROL [g1.mi.hdm-stuttgart.de]: '
   PUSH_REQUEST' (status=1)
18 2021-11-30 15:58:21 PUSH: Received control message: 'PUSH_REPLY,route
   10.8.1.1,topology net30,ping 10,ping-restart 120,ifconfig 10.8.1.6
   10.8.1.5,peer-id 0,cipher AES-256-GCM'
19 2021-11-30 15:58:21 OPTIONS IMPORT: timers and/or timeouts modified
20 2021-11-30 15:58:21 OPTIONS IMPORT: --ifconfig/up options modified
21 2021-11-30 15:58:21 OPTIONS IMPORT: route options modified
22 2021-11-30 15:58:21 OPTIONS IMPORT: peer-id set
23 2021-11-30 15:58:21 OPTIONS IMPORT: adjusting link_mtu to 1625
24 2021-11-30 15:58:21 OPTIONS IMPORT: data channel crypto options
   modified
25 2021-11-30 15:58:21 Data Channel: using negotiated cipher 'AES-256-GCM'
26 2021-11-30 15:58:21 Outgoing Data Channel: Cipher 'AES-256-GCM'
```



```
    initialized with 256 bit key
27 2021-11-30 15:58:21 Incoming Data Channel: Cipher 'AES-256-GCM'
    initialized with 256 bit key
28 2021-11-30 15:58:21 ROUTE_GATEWAY 100.64.84.78/255.255.255.240 IFACE=
    wlp3s0 HWADDR=c8:94:02:bd:60:53
29 2021-11-30 15:58:21 TUN/TAP device tun0 opened
30 2021-11-30 15:58:21 /usr/sbin/ip link set dev tun0 up mtu 1500
31 2021-11-30 15:58:21 /usr/sbin/ip link set dev tun0 up
32 2021-11-30 15:58:21 /usr/sbin/ip addr add dev tun0 local 10.8.1.6 peer
    10.8.1.5
33 2021-11-30 15:58:21 /usr/sbin/ip route add 10.8.1.1/32 via 10.8.1.5
34 2021-11-30 15:58:21 WARNING: this configuration may cache passwords in
    memory -- use the auth-nocache option to prevent this
35 2021-11-30 15:58:21 Initialization Sequence Completed
```

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

Client:

```
1 client                # Definiert dass es sich um eine
    Konfigurationsdatei für einen Client handelt.
2 dev tun               # Als virtuelles Netzwerkgerät verwenden
    wir tun, welches nur TCP/IP-Verkehr weiterleitet und keinen
    Broadcast-Verkehr über den VPN-Tunnel bereitstellt.
3 proto udp             # Hier wird festgelegt, welches Protokoll
    auf Ebene 4 verwendet werden soll.
4 remote 135.181.204.42 1194 # Gibt an mit welcher Adresse sich
    verbunden werden soll. Dies wäre auch mit einem Hostnamen möglich.
5 nobind                # Veranlasst OpenVPN dazu einen zufä
    lligen clientseitigen Port zu verwenden.
6 persist-key           # Versucht Zustände über den Neustart der
    Verbindung zu erhalten.
7 persist-tun           # Versucht Zustände über den Neustart der
    Verbindung zu erhalten.
8 ca ca.crt             # Gibt den Pfad zur Zertifikatsdatei der
    Certification Authority an.
9 cert issued/client-gl.crt # Gibt den Pfad zur Zertifikatsdatei des
    Clients an.
10 key private/client-gl.key # Gibt den Pfad zur Key-Datei des Clients
    an.
11 comp-lzo             # Definiert dass keine Kompression
    verwendet werden soll.
12 verb 3               # Definiert die Ausführlichkeit des
    Outputs. 3: Infos über Key-Generierung, Routen, Debugging des TUN/
    TAP-Treibers, Push/Pull/Ifconfig-Pool, Authentifizierung
```

Server:

```
1 proto udp # Hier wird festgelegt, welches Protokoll
   auf Ebene 4 verwendet werden soll.
2 dev tun # Als virtuelles Netzwerkgerät verwenden
   wir tun, welches nur TCP/IP-Verkehr weiterleitet und keinen
   Broadcast-Verkehr über den VPN-Tunnel bereitstellt.
3 ca pki/ca.crt # Gibt den Pfad zur Zertifikatsdatei der
   Certification Authority an.
4 cert pki/issued/server-gl.crt # Gibt den Pfad zur Zertifikatsdatei des
   Servers an.
5 key pki/private/server-gl.key # Gibt den Pfad zur Key-Datei des Servers
   an.
6 dh pki/dh.pem # Gibt den Pfad zur Diffie-Hellman-Key-
   Datei des Servers an.
7 server 10.8.1.0 255.255.255.0 # Damit wird ein VPN-Subnetz unter
   Verwendung des Adressbereichs 10.8.1.XXX eingerichtet.
8 keepalive 10 120 # Hier wird alle 10 Sekunden ein Ping
   abgesetzt und wenn nach 120 Sekunden keine Antwort kommt, gilt die
   Verbindung als down.
9 comp-lzo # Definiert dass keine Kompression
   verwendet werden soll.
10 persist-key # Versucht Zustände über den Neustart der
   Verbindung zu erhalten.
11 persist-tun # Versucht Zustände über den Neustart der
   Verbindung zu erhalten.
12 verb 3 # Definiert die Ausführlichkeit des
   Outputs. 3: Infos über Key-Generierung, Routen, Debugging des TUN/
   TAP-Treibers, Push/Pull/Ifconfig-Pool, Authentifizierung
```


3.3 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.



```
C:\Users\Elia1\OneDrive\Dokumente\MedienInformatik\Lehrstoff\Semester4\PraktRechner\Neuer Ordner>ipconfig
```

Windows-IP-Konfiguration

Unbekannter Adapter LAN-Verbindung:

```
Verbindungsspezifisches DNS-Suffix:
Verbindungslokale IPv6-Adresse . . : fe80::cb5:9d5e:4c41:db92%8
IPv4-Adresse . . . . . : 10.8.1.6
Subnetzmaske . . . . . : 255.255.255.252
Standardgateway . . . . . :
```

Drahtlos-LAN-Adapter LAN-Verbindung* 1:

```
Medienstatus. . . . . : Medium getrennt
Verbindungsspezifisches DNS-Suffix:
```

Drahtlos-LAN-Adapter LAN-Verbindung* 2:

```
Medienstatus. . . . . : Medium getrennt
Verbindungsspezifisches DNS-Suffix:
```

Drahtlos-LAN-Adapter WLAN:

```
Verbindungsspezifisches DNS-Suffix: Speedport_W_724V_Typ_A_05011603_06_003
IPv6-Adresse. . . . . : 2003:cd:271d:f860:f4f2:d559:fca9:9fb2
Verbindungslokale IPv6-Adresse . . : fe80::f4f2:d559:fca9:9fb2%6
IPv4-Adresse . . . . . : 192.168.2.104
Subnetzmaske . . . . . : 255.255.255.0
Standardgateway . . . . . : fe80::1%6
                          192.168.2.1
```

Ethernet-Adapter Bluetooth-Netzwerkverbindung:

```
Medienstatus. . . . . : Medium getrennt
Verbindungsspezifisches DNS-Suffix:
```

Windows verlangt, dass wir die "client.conf" in "client.ovpn" umbenennen. Die "client.ovpn" muss dann neben der "ca.crt", "client-g1.crt", "client-g1.key" abgelegt werden. Anschließend kann über das GUI eine Verbindung etabliert werden.

4 Analyse

4.1 Analyse der Logs

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

Client-Log:

```
1 # sudo openvpn --config client.conf
2 [sudo] password for root:
3 2021-11-30 15:58:20 WARNING: Compression for receiving enabled.
   Compression has been used in the past to break encryption. Sent
   packets are not compressed unless "allow-compression yes" is also
   set.
4 2021-11-30 15:58:20 --cipher is not set. Previous OpenVPN version
   defaulted to BF-CBC as fallback when cipher negotiation failed in
   this case. If you need this fallback please add '--data-ciphers-
   fallback BF-CBC' to your configuration and/or add BF-CBC to --data-
   ciphers.
5 2021-11-30 15:58:20 OpenVPN 2.5.3 x86_64-suse-linux-gnu [SSL (OpenSSL)]
   [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on Jun 17
   2021
6 2021-11-30 15:58:20 library versions: OpenSSL 1.1.1l 24 Aug 2021, LZO
   2.10
7 2021-11-30 15:58:20 WARNING: No server certificate verification method
   has been enabled. See http://openvpn.net/howto.html#mitm for more
   info.
8 2021-11-30 15:58:20 TCP/UDP: Preserving recently used remote address: [
   AF_INET]135.181.204.42:1194
9 2021-11-30 15:58:20 Socket Buffers: R=[212992->212992] S
   =[212992->212992]
10 2021-11-30 15:58:20 UDP link local: (not bound)
11 2021-11-30 15:58:20 UDP link remote: [AF_INET]135.181.204.42:1194
12 2021-11-30 15:58:20 TLS: Initial packet from [AF_INET
   ]135.181.204.42:1194, sid=1cflee33 f316b385
13 2021-11-30 15:58:20 VERIFY OK: depth=1, CN=g1.mi.hdm-stuttgart.de
14 2021-11-30 15:58:20 VERIFY OK: depth=0, CN=g1.mi.hdm-stuttgart.de
15 2021-11-30 15:58:20 Control Channel: TLSv1.2, cipher TLSv1.2 ECDHE-RSA-
   AES256-GCM-SHA384, peer certificate: 2048 bit RSA, signature: RSA-
   SHA256
16 2021-11-30 15:58:20 [g1.mi.hdm-stuttgart.de] Peer Connection Initiated
   with [AF_INET]135.181.204.42:1194
17 2021-11-30 15:58:21 SENT CONTROL [g1.mi.hdm-stuttgart.de]: '
   PUSH_REQUEST' (status=1)
```

```
18 2021-11-30 15:58:21 PUSH: Received control message: 'PUSH_REPLY,route
    10.8.1.1,topology net30,ping 10,ping-restart 120,ifconfig 10.8.1.6
    10.8.1.5,peer-id 0,cipher AES-256-GCM'
19 2021-11-30 15:58:21 OPTIONS IMPORT: timers and/or timeouts modified
20 2021-11-30 15:58:21 OPTIONS IMPORT: --ifconfig/up options modified
21 2021-11-30 15:58:21 OPTIONS IMPORT: route options modified
22 2021-11-30 15:58:21 OPTIONS IMPORT: peer-id set
23 2021-11-30 15:58:21 OPTIONS IMPORT: adjusting link_mtu to 1625
24 2021-11-30 15:58:21 OPTIONS IMPORT: data channel crypto options
    modified
25 2021-11-30 15:58:21 Data Channel: using negotiated cipher 'AES-256-GCM'
26 2021-11-30 15:58:21 Outgoing Data Channel: Cipher 'AES-256-GCM'
    initialized with 256 bit key
27 2021-11-30 15:58:21 Incoming Data Channel: Cipher 'AES-256-GCM'
    initialized with 256 bit key
28 2021-11-30 15:58:21 ROUTE_GATEWAY 100.64.84.78/255.255.255.240 IFACE=
    wlp3s0 HWADDR=c8:94:02:bd:60:53
29 2021-11-30 15:58:21 TUN/TAP device tun0 opened
30 2021-11-30 15:58:21 /usr/sbin/ip link set dev tun0 up mtu 1500
31 2021-11-30 15:58:21 /usr/sbin/ip link set dev tun0 up
32 2021-11-30 15:58:21 /usr/sbin/ip addr add dev tun0 local 10.8.1.6 peer
    10.8.1.5
33 2021-11-30 15:58:21 /usr/sbin/ip route add 10.8.1.1/32 via 10.8.1.5
34 2021-11-30 15:58:21 WARNING: this configuration may cache passwords in
    memory -- use the auth-nocache option to prevent this
35 2021-11-30 15:58:21 Initialization Sequence Completed
```

Server-Log:

```
1 # sudo openvpn --config server.conf
2 Tue Nov 30 14:57:41 2021 OpenVPN 2.4.0 x86_64-pc-linux-gnu [SSL (
    OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on
    Oct 14 2018
3 Tue Nov 30 14:57:41 2021 library versions: OpenSSL 1.0.2u 20 Dec 2019,
    LZO 2.08
4 Tue Nov 30 14:57:41 2021 Diffie-Hellman initialized with 2048 bit key
5 Tue Nov 30 14:57:41 2021 ROUTE_GATEWAY 172.31.1.1
6 Tue Nov 30 14:57:41 2021 TUN/TAP device tun0 opened
7 Tue Nov 30 14:57:41 2021 TUN/TAP TX queue length set to 100
8 Tue Nov 30 14:57:41 2021 do_ifconfig, tt->did_ifconfig_ipv6_setup=0
9 Tue Nov 30 14:57:41 2021 /sbin/ip link set dev tun0 up mtu 1500
10 Tue Nov 30 14:57:41 2021 /sbin/ip addr add dev tun0 local 10.8.1.1 peer
    10.8.1.2
11 Tue Nov 30 14:57:41 2021 /sbin/ip route add 10.8.1.0/24 via 10.8.1.2
12 Tue Nov 30 14:57:41 2021 Could not determine IPv4/IPv6 protocol. Using
    AF_INET
13 Tue Nov 30 14:57:41 2021 Socket Buffers: R=[212992->212992] S
    =[212992->212992]
14 Tue Nov 30 14:57:41 2021 UDPv4 link local (bound): [AF_INET][undef
    ]:1194
15 Tue Nov 30 14:57:41 2021 UDPv4 link remote: [AF_UNSPEC]
```

```
16 Tue Nov 30 14:57:41 2021 MULTI: multi_init called, r=256 v=256
17 Tue Nov 30 14:57:41 2021 IFCONFIG POOL: base=10.8.1.4 size=62, ipv6=0
18 Tue Nov 30 14:57:41 2021 Initialization Sequence Completed
19 Tue Nov 30 14:58:20 2021 141.72.244.138:59463 TLS: Initial packet from
[AF_INET]141.72.244.138:59463, sid=15b6f57f 995bddb5
20 Tue Nov 30 14:58:20 2021 141.72.244.138:59463 VERIFY OK: depth=1, CN=g1
.mi.hdm-stuttgart.de
21 Tue Nov 30 14:58:20 2021 141.72.244.138:59463 VERIFY OK: depth=0, CN=g1
.mi.hdm-stuttgart.de
22 Tue Nov 30 14:58:20 2021 141.72.244.138:59463 peer info: IV_VER=2.5.3
23 Tue Nov 30 14:58:20 2021 141.72.244.138:59463 peer info: IV_PLAT=linux
24 Tue Nov 30 14:58:20 2021 141.72.244.138:59463 peer info: IV_PROTO=6
25 Tue Nov 30 14:58:20 2021 141.72.244.138:59463 peer info: IV_NCP=2
26 Tue Nov 30 14:58:20 2021 141.72.244.138:59463 peer info: IV_CIPHERS=AES
-256-GCM:AES-128-GCM
27 Tue Nov 30 14:58:20 2021 141.72.244.138:59463 peer info: IV_LZ4=1
28 Tue Nov 30 14:58:20 2021 141.72.244.138:59463 peer info: IV_LZ4v2=1
29 Tue Nov 30 14:58:20 2021 141.72.244.138:59463 peer info: IV_LZO=1
30 Tue Nov 30 14:58:20 2021 141.72.244.138:59463 peer info: IV_COMP_STUB=1
31 Tue Nov 30 14:58:20 2021 141.72.244.138:59463 peer info: IV_COMP_STUBv2
=1
32 Tue Nov 30 14:58:20 2021 141.72.244.138:59463 peer info: IV_TCPNL=1
33 Tue Nov 30 14:58:20 2021 141.72.244.138:59463 WARNING: 'cipher' is
present in local config but missing in remote config, local='cipher
BF-CBC'
34 Tue Nov 30 14:58:20 2021 141.72.244.138:59463 Control Channel: TLSv1.2,
cipher TLSv1/SSLv3 ECDHE-RSA-AES256-GCM-SHA384, 2048 bit RSA
35 Tue Nov 30 14:58:20 2021 141.72.244.138:59463 [g1.mi.hdm-stuttgart.de]
Peer Connection Initiated with [AF_INET]141.72.244.138:59463
36 Tue Nov 30 14:58:20 2021 g1.mi.hdm-stuttgart.de/141.72.244.138:59463
MULTI_sva: pool returned IPv4=10.8.1.6, IPv6=(Not enabled)
37 Tue Nov 30 14:58:20 2021 g1.mi.hdm-stuttgart.de/141.72.244.138:59463
MULTI: Learn: 10.8.1.6 -> g1.mi.hdm-stuttgart.de
/141.72.244.138:59463
38 Tue Nov 30 14:58:20 2021 g1.mi.hdm-stuttgart.de/141.72.244.138:59463
MULTI: primary virtual IP for g1.mi.hdm-stuttgart.de
/141.72.244.138:59463: 10.8.1.6
39 Tue Nov 30 14:58:21 2021 g1.mi.hdm-stuttgart.de/141.72.244.138:59463
PUSH: Received control message: 'PUSH_REQUEST'
40 Tue Nov 30 14:58:21 2021 g1.mi.hdm-stuttgart.de/141.72.244.138:59463
SENT CONTROL [g1.mi.hdm-stuttgart.de]: 'PUSH_REPLY,route 10.8.1.1,
topology net30,ping 10,ping-restart 120,ifconfig 10.8.1.6 10.8.1.5,
peer-id 0,cipher AES-256-GCM' (status=1)
41 Tue Nov 30 14:58:21 2021 g1.mi.hdm-stuttgart.de/141.72.244.138:59463
Data Channel Encrypt: Cipher 'AES-256-GCM' initialized with 256 bit
key
42 Tue Nov 30 14:58:21 2021 g1.mi.hdm-stuttgart.de/141.72.244.138:59463
Data Channel Decrypt: Cipher 'AES-256-GCM' initialized with 256 bit
key
```

Aus dem Output lässt sich entnehmen, dass die Verbindung etabliert wurde und damit auch der Tun-

nel initialisiert wurde. Im Folgenden kann man auch sehen, dass nun die `tun` Netzwerkinterfaces angezeigt werden.

4.2 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 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2   inet 127.0.0.1/8 scope host lo
   valid_lft forever preferred_lft forever
3   inet6 ::1/128 scope host
   valid_lft forever preferred_lft forever
4 2: enp2s0f0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc
   pfifo_fast state DOWN group default qlen 1000
5   link/ether 84:a9:38:67:f2:18 brd ff:ff:ff:ff:ff:ff
6 3: wlp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
   state UP group default qlen 1000
7   link/ether c8:94:02:bd:60:53 brd ff:ff:ff:ff:ff:ff
8   inet 100.64.84.65/28 brd 100.64.84.79 scope global dynamic
9     noprefixroute wlp3s0
10    valid_lft 31703sec preferred_lft 31703sec
11   inet6 2001:7c7:2126:4b00:1c82:4bbd:d5bc:2749/64 scope global
12     temporary dynamic
13    valid_lft 597550sec preferred_lft 78897sec
14   inet6 2001:7c7:2126:4b00:7431:96ca:2ac9:c43b/64 scope global
15     dynamic mngtmpaddr noprefixroute
16    valid_lft 2591827sec preferred_lft 604627sec
17   inet6 fe80::3d0a:2eec:1296:52be/64 scope link noprefixroute
18     valid_lft forever preferred_lft forever
19 4: enp6s0f3u1u3c2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc
   pfifo_fast state DOWN group default qlen 1000
20   link/ether 00:50:b6:f5:31:44 brd ff:ff:ff:ff:ff:ff
21 6: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc
   pfifo_fast state UNKNOWN group default qlen 500
22   link/none
23   inet 10.8.1.6 peer 10.8.1.5/32 scope global tun0
24     valid_lft forever preferred_lft forever
25   inet6 fe80::847d:2db8:e5f6:2a09/64 scope link stable-privacy
26     valid_lft forever preferred_lft forever
```

Als Nächstes verwenden wir `ip link`


```
1 # ip link
2 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
   mode DEFAULT group default qlen 1000
3     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
4 2: enp2s0f0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc
   pfifo_fast state DOWN mode DEFAULT group default qlen 1000
5     link/ether 84:a9:38:67:f2:18 brd ff:ff:ff:ff:ff:ff
6 3: wlp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
   state UP mode DORMANT group default qlen 1000
7     link/ether c8:94:02:bd:60:53 brd ff:ff:ff:ff:ff:ff
8 4: enp6s0f3u1u3c2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc
   pfifo_fast state DOWN mode DEFAULT group default qlen 1000
9     link/ether 00:50:b6:f5:31:44 brd ff:ff:ff:ff:ff:ff
10 6: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc
   pfifo_fast state UNKNOWN mode DEFAULT group default qlen 500
11     link/none
```

Jetzt noch ein Test mit `ip route`:

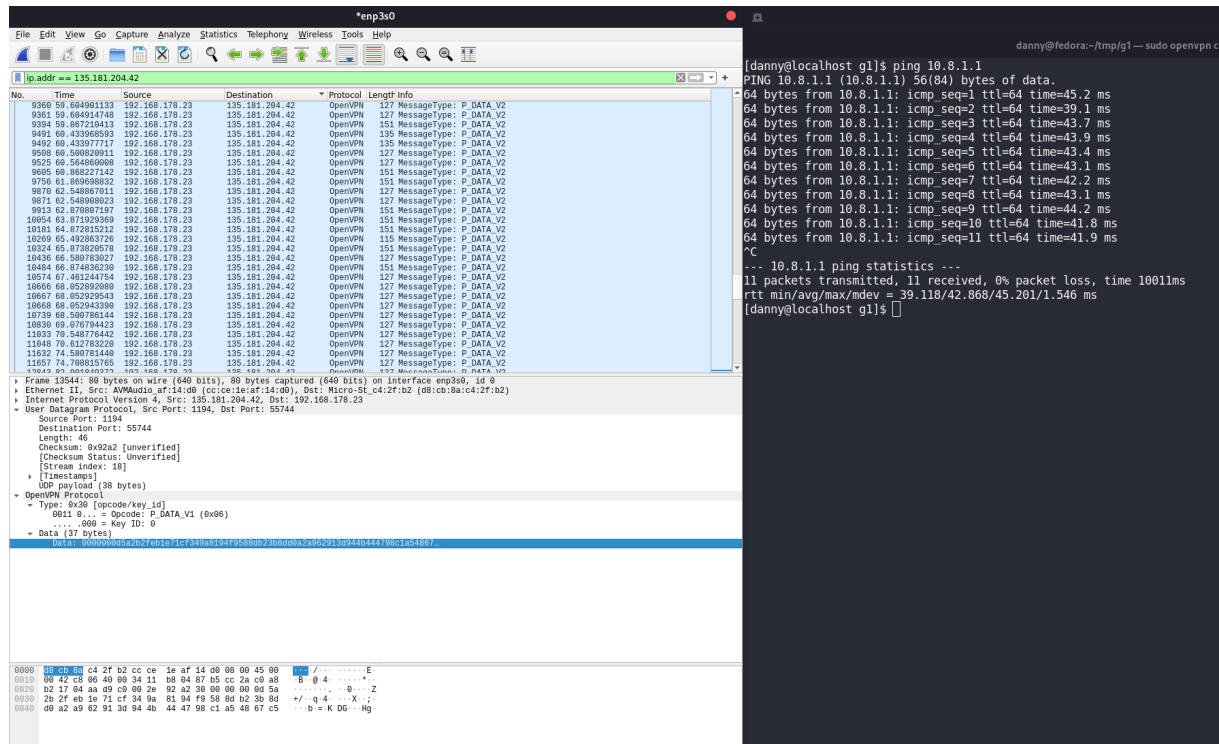
```
1 e0 proto dhcp metric 600
2 10.8.1.1 via 10.8.1.5 dev tun0
3 10.8.1.5 dev tun0 proto kernel scope link src 10.8.1.6
4 100.64.84.64/28 dev wlp3s0 proto kernel scope link src 100.64.84.65
   metric 600
```

Als Letztes noch ein Ping vom Client an das `tun0` Interface des Servers:

```
1 NG 10.8.1.1 (10.8.1.1) 56(84) bytes of data.
2 64 bytes from 10.8.1.1: icmp_seq=1 ttl=64 time=25.5 ms
3 64 bytes from 10.8.1.1: icmp_seq=2 ttl=64 time=25.3 ms
4 64 bytes from 10.8.1.1: icmp_seq=3 ttl=64 time=25.5 ms
5 64 bytes from 10.8.1.1: icmp_seq=4 ttl=64 time=25.0 ms
6 64 bytes from 10.8.1.1: icmp_seq=5 ttl=64 time=25.8 ms
7 64 bytes from 10.8.1.1: icmp_seq=6 ttl=64 time=25.5 ms
8 64 bytes from 10.8.1.1: icmp_seq=7 ttl=64 time=25.2 ms
9 ^C
10 --- 10.8.1.1 ping statistics ---
11 7 packets transmitted, 7 received, 0% packet loss, time 6008ms
12 rtt min/avg/max/mdev = 25.042/25.397/25.771/0.222 ms
```

5 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 image displays a Wireshark capture of network traffic. The top pane shows a list of captured packets, with the first packet being an ICMP Echo (ping) request from 192.168.178.23 to 135.181.204.42. The middle pane shows the details of the selected packet, including the Ethernet II header, Internet Protocol Version 4 header, and User Datagram Protocol header. The bottom pane shows the raw data of the packet in hexadecimal and ASCII format.

Packet List:

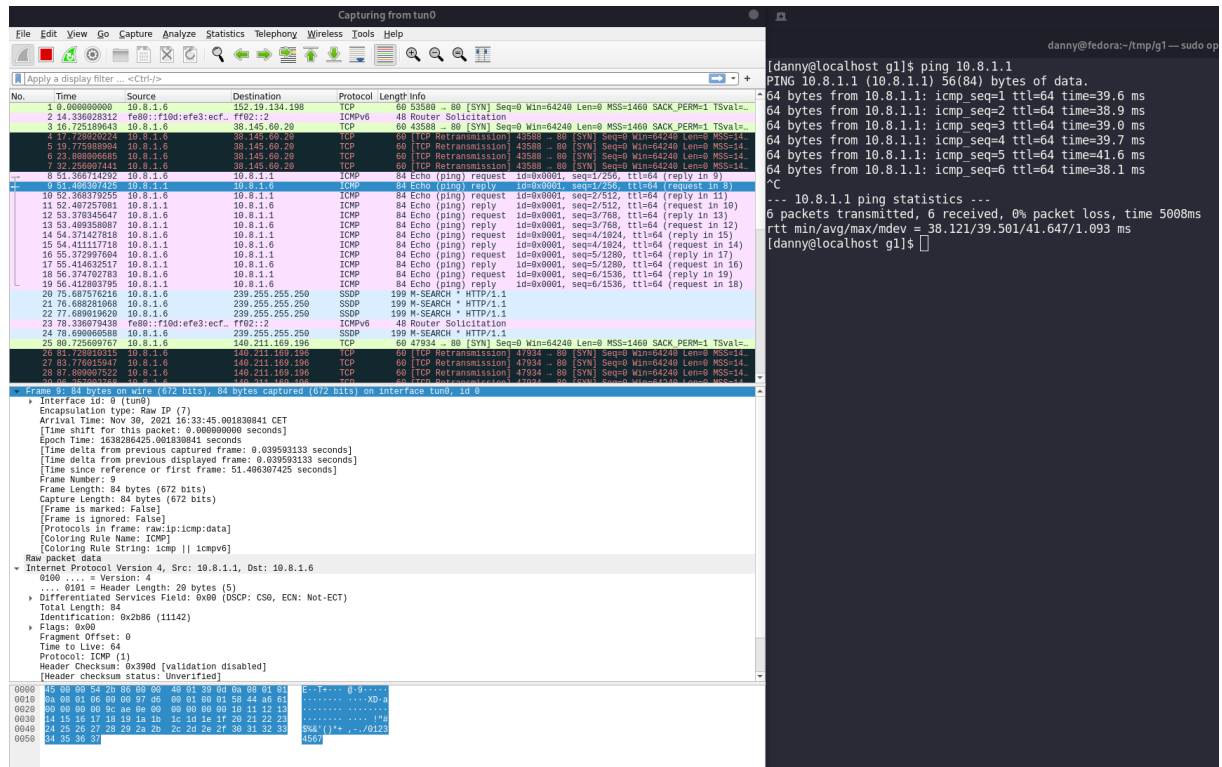
No.	Time	Source	Destination	Protocol	Length	Info
9369	59.684961133	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P DATA_V2
9371	59.684914748	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P DATA_V2
9394	59.867218413	192.168.178.23	135.181.204.42	OpenVPN	151	MessageType: P DATA_V2
9491	60.433985993	192.168.178.23	135.181.204.42	OpenVPN	135	MessageType: P DATA_V2
9492	60.433977717	192.168.178.23	135.181.204.42	OpenVPN	135	MessageType: P DATA_V2
9508	60.580820911	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P DATA_V2
9575	60.584609888	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P DATA_V2
9605	60.868227142	192.168.178.23	135.181.204.42	OpenVPN	151	MessageType: P DATA_V2
9756	61.699698932	192.168.178.23	135.181.204.42	OpenVPN	151	MessageType: P DATA_V2
9870	62.548867011	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P DATA_V2
9871	62.548898023	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P DATA_V2
9913	62.878897137	192.168.178.23	135.181.204.42	OpenVPN	151	MessageType: P DATA_V2
10054	63.871929369	192.168.178.23	135.181.204.42	OpenVPN	151	MessageType: P DATA_V2
10181	64.872815212	192.168.178.23	135.181.204.42	OpenVPN	151	MessageType: P DATA_V2
10209	65.492863726	192.168.178.23	135.181.204.42	OpenVPN	115	MessageType: P DATA_V2
10324	65.873820578	192.168.178.23	135.181.204.42	OpenVPN	151	MessageType: P DATA_V2
10436	66.580753027	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P DATA_V2
10484	66.874838239	192.168.178.23	135.181.204.42	OpenVPN	151	MessageType: P DATA_V2
10574	67.462244754	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P DATA_V2
10666	68.852892080	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P DATA_V2
10667	68.852929543	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P DATA_V2
10668	68.852943380	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P DATA_V2
10739	68.580785144	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P DATA_V2
10830	69.076794423	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P DATA_V2
11033	70.548776442	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P DATA_V2
11048	70.612783220	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P DATA_V2
11632	74.580751440	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P DATA_V2
11657	74.788815765	192.168.178.23	135.181.204.42	OpenVPN	127	MessageType: P DATA_V2

Packet Details:

- Frame 13544: 88 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface em3s0, id 0
- Ethernet II, Src: VMware10, af:14:00 (cc:ce:1a:af:14:00), Dst: Mikro-SL c4:2f:b2 (d8:cb:8a:c4:2f:b2)
- Internet Protocol Version 4, Src: 135.181.204.42, Dst: 192.168.178.23
- User Datagram Protocol, Src Port: 1194, Dst Port: 55744
 - Source Port: 1194
 - Destination Port: 55744
 - Length: 48
 - Checksum: 0x92a2 [unverified]
 - [Checksum Status: Unverified]
 - [Stream index: 10]
 - [Timestamps]
 - UDP payload (28 bytes)
 - OpenVPN Protocol
 - Type: 0x30 [opcode/key_id]
 - 0011 0... = Opcode: P_DATA_V1 (0x06)
 -000 = Key ID: 0
 - Data (27 bytes)

Packet Bytes:

```
0000  00 42 c8 00 00 34 11 08 04 87 05 cc 2a 09 a8 18 04 .....E
0010  02 17 04 aa 09 09 09 2a 92 a2 39 09 00 00 8d 5a 00 00 .....Z
0020  02 2f eb 1e 71 cf 34 9a 81 94 f0 58 80 b2 3b 8d +- / q 4 . X . ;
0030  08 02 a9 02 91 3d 94 4b 44 47 06 c1 a5 4b 07 c5 .....b = K D6 . Hg
```



The screenshot shows a Wireshark packet capture on the `tun0` interface. The packet list displays several ICMP Echo (ping) requests and replies between `10.8.1.1` and `10.8.1.6`. The packet details pane shows the raw packet data for an ICMP Echo request, including the IP header and ICMP header.

Packet List:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.8.1.6	152.19.134.198	TCP	60	53588 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=...
2	14.586026312	fe80::f1d0:efe3::...	fe80::f1d0:efe3::...	ICMPv6	48	Router Solicitation
3	16.725189643	10.8.1.6	38.145.60.20	TCP	60	43588 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=...
4	17.229466024	10.8.1.6	38.145.60.20	TCP	60	[TCP Retransmission] 43588 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=...
5	19.755889044	10.8.1.6	38.145.60.20	TCP	60	[TCP Retransmission] 43588 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=...
6	23.888896585	10.8.1.6	38.145.60.20	TCP	60	[TCP Retransmission] 43588 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=...
7	33.266097444	10.8.1.6	38.145.60.20	TCP	60	[TCP Retransmission] 43588 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=...
8	51.350714292	10.8.1.6	10.8.1.1	ICMP	84	Echo (ping) request id=0x0001, seq=1/256, ttl=64 (reply in 9)
9	52.368379255	10.8.1.1	10.8.1.6	ICMP	84	Echo (ping) reply id=0x0001, seq=2/512, ttl=64 (request in 8)
11	52.407257081	10.8.1.1	10.8.1.6	ICMP	84	Echo (ping) reply id=0x0001, seq=2/512, ttl=64 (request in 10)
12	53.378345647	10.8.1.6	10.8.1.1	ICMP	84	Echo (ping) request id=0x0001, seq=3/768, ttl=64 (reply in 13)
13	53.409358087	10.8.1.1	10.8.1.6	ICMP	84	Echo (ping) reply id=0x0001, seq=3/768, ttl=64 (request in 12)
14	54.371427818	10.8.1.6	10.8.1.1	ICMP	84	Echo (ping) request id=0x0001, seq=4/1024, ttl=64 (reply in 15)
15	54.411177118	10.8.1.1	10.8.1.6	ICMP	84	Echo (ping) reply id=0x0001, seq=4/1024, ttl=64 (request in 14)
16	55.372997604	10.8.1.6	10.8.1.1	ICMP	84	Echo (ping) request id=0x0001, seq=5/1280, ttl=64 (reply in 17)
17	55.414632517	10.8.1.1	10.8.1.6	ICMP	84	Echo (ping) reply id=0x0001, seq=5/1280, ttl=64 (request in 16)
18	56.374702783	10.8.1.6	10.8.1.1	ICMP	84	Echo (ping) request id=0x0001, seq=6/1536, ttl=64 (reply in 19)
19	56.412803795	10.8.1.1	10.8.1.6	ICMP	84	Echo (ping) reply id=0x0001, seq=6/1536, ttl=64 (request in 18)
20	15.687676216	10.8.1.6	239.255.255.250	SSDP	199	M-SEARCH * HTTP/1.1
21	16.688281068	10.8.1.6	239.255.255.250	SSDP	199	M-SEARCH * HTTP/1.1
22	17.689019620	10.8.1.6	239.255.255.250	SSDP	199	M-SEARCH * HTTP/1.1
23	18.38079438	fe80::f1d0:efe3::...	fe80::f1d0:efe3::...	ICMPv6	48	Router Solicitation
24	18.699066588	10.8.1.6	239.255.255.250	SSDP	199	M-SEARCH * HTTP/1.1
25	30.725509767	10.8.1.6	140.211.109.190	TCP	60	47934 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=...
26	31.729103115	10.8.1.6	140.211.109.190	TCP	60	[TCP Retransmission] 47934 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=...
27	33.776915916	10.8.1.6	140.211.109.190	TCP	60	[TCP Retransmission] 47934 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=...
28	37.699092722	10.8.1.6	140.211.109.190	TCP	60	[TCP Retransmission] 47934 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=...
29	39.692565276	10.8.1.6	140.211.109.190	TCP	60	[TCP Retransmission] 47934 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=...

Packet Details (Packet 8):

- Interface ID: 0 (tun0)
- Encapsulation type: Raw IP (7)
- Arrival Time: Nov 30, 2021 16:33:45.001830841 CET
- [Time shift for this packet: 0.000000000 seconds]
- Epoch Time: 1633086425.001830841 seconds
- [Time delta from previous captured frame: 0.039593133 seconds]
- [Time delta from previous displayed frame: 0.039593133 seconds]
- [Time since reference or first frame: 51.406307425 seconds]
- Frame Number: 9
- Frame Length: 84 bytes (672 bits)
- Capture Length: 84 bytes (672 bits)
- [Frame is marked: False]
- [Frame is ignored: False]
- Protocols in frame: raw.ip:icmp:data
- Coloring Rule Name: ICMP
- Coloring Rule String: icmp || icmpv6
- Raw packet data
- Internet Protocol Version 4, Src: 10.8.1.1, Dst: 10.8.1.6
- 0100 = Version: 4
- 0100 = Header Length: 20 bytes (5)
- Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
- Total Length: 84
- Identification: 0x2b86 (11142)
- Flags: 0x00
- Fragment Offset: 0
- Time to Live: 64
- Protocol: ICMP (1)
- Header Checksum: 0x0900 [validation disabled]
- [Header checksum status: Unverified]

Packet 8 (ICMP Echo (ping) request) details:

```

0000 45 00 00 54 20 00 00 00 40 01 99 0d 0a 08 01 01  0...0...0...0...0...0...0...0...
0010 84 00 01 00 00 00 07 00 00 01 00 01 38 24 a0 01  .....XD...
0020 09 09 09 00 8c aa 8a 00 09 00 00 00 11 12 13  090909008c aa 09000000111213
0030 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23  1415161718191a1b1c1d1e1f20212223
0040 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33  2425262728292a2b2c2d2e2f30313233
0050 34 35 36 37  34353637
  
```

Im Interface `enp3s0` werden die Daten durch ein `OpenVPN` Protokoll gehandelt, diese Daten sind, wie im Screenshot zu sehen, verschlüsselt. Im Interface `tun0` werden die Daten mit einem `IPv4` Protokoll gehandelt.

6 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):**



Abbildung 4: VPN Tunnel

Um die Funktion zu testen, nutzen Sie den Dienst „ifconfig.co“ (Versuchen Sie einmal die Adresse im Browser). Für was kann dieser Dienst genutzt werden?

Der Dienst kann genutzt werden um die eigene öffentliche IP-Adresse herauszufinden.

Da bei der Person, welche die Clientverbindung hatte, IPv6 verwendet wurde, lieferte `ifconfig.co` folgendes Ergebnis:

```
1 # curl ifconfig.co
2 2001:7c7:2126:4b00:b016:5c9f:1161:eb14
```

Stattdessen kann `api.ipify.org` verwendet werden:

```
1 # curl api.ipify.org
2 141.72.244.138
```

6.1 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:

```
1 # dig api.ipify.org
2
3 ; <<>> DiG 9.16.23-RH <<>> api.ipify.org
4 ;; global options: +cmd
5 ;; Got answer:
6 ;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 52052
7 ;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1
8
9 ;; OPT PSEUDOSECTION:
10 ; EDNS: version: 0, flags;; udp: 65494
11 ;; QUESTION SECTION:
12 ;api.ipify.org.                IN      A
13
14 ;; ANSWER SECTION:
15 api.ipify.org.                3484    IN      CNAME   api.ipify.org.herokudns
16 .com.
17 api.ipify.org.herokudns.com. 5 IN      A       54.91.59.199
18 api.ipify.org.herokudns.com. 5 IN      A       52.20.78.240
19 api.ipify.org.herokudns.com. 5 IN      A       3.232.242.170
20 api.ipify.org.herokudns.com. 5 IN      A       3.220.57.224
21
22 ;; Query time: 27 msec
23 ;; SERVER: 127.0.0.53#53(127.0.0.53)
24 ;; WHEN: Wed Dec 01 11:55:12 CET 2021
25 ;; MSG SIZE rcvd: 147
```

Wir müssen also diese vier IPs in unsere Konfigurationsdatei einarbeiten:

```
1 # cat server.conf
2 proto udp
3 dev tun
4 ca pki/ca.crt
5 cert pki/issued/server-g1.crt
6 key pki/private/server-g1.key
7 dh pki/dh.pem
8 server 10.8.1.0 255.255.255.0
9 keepalive 10 120
10 comp-lzo
11 persist-key
12 persist-tun
13 verb 3
14 push "route 54.91.59.199 255.255.255.255"
15 push "route 52.20.78.240 255.255.255.255"
16 push "route 3.232.242.170 255.255.255.255"
```

```
17 push "route 3.220.57.224 255.255.255.255"
```

Zusätzlich müssen wir auf neue Firewall-Regeln auf dem Server einfügen:

```
1 # iptables -t nat -A POSTROUTING -s 10.8.1.0/24 -d 54.91.59.199 -j  
  SNAT --to-source 135.181.204.42  
2 # iptables -t nat -A POSTROUTING -s 10.8.1.0/24 -d 52.20.78.240 -j  
  SNAT --to-source 135.181.204.42  
3 # iptables -t nat -A POSTROUTING -s 10.8.1.0/24 -d 3.232.242.170 -j  
  SNAT --to-source 135.181.204.42  
4 # iptables -t nat -A POSTROUTING -s 10.8.1.0/24 -d 3.220.57.224 -j  
  SNAT --to-source 135.181.204.42
```

Außerdem muss das IP-Forwarding eingeschaltet werden:

```
1 # sysctl net.ipv4.conf.all.forwarding=1
```

Was bewirken diese Konfigurationsänderungen? Warum sind sie nötig?

Die Änderung der Konfiguration ist nötig, um den Umfang des VPNs zu erweitern, sodass die Clients mehrere Maschinen (in unserem Fall api.ipify.org) im Servernetz erreichen können und nicht nur die Servermaschine selbst.

6.2 Funktionstest

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

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

```
1 # ip route get 54.91.59.199  
2 54.91.59.199 via 10.8.1.5 dev tun0 src 10.8.1.6 uid 1000  
3 cache
```

```
1 # ip route get 52.20.78.240  
2 52.20.78.240 via 10.8.1.5 dev tun0 src 10.8.1.6 uid 1000  
3 cache
```

```
1 # ip route get 3.232.242.170  
2 3.232.242.170 via 10.8.1.5 dev tun0 src 10.8.1.6 uid 1000  
3 cache
```

```
1 # ip route get 3.220.57.224  
2 3.220.57.224 via 10.8.1.5 dev tun0 src 10.8.1.6 uid 1000  
3 cache
```

Rufen Sie den Dienst „ifconfig.co“ vom Client aus auf. Was ist das Resultat? Warum?

```
1 curl api.ipify.org
2 135.181.204.42
```

Das Resultat zeigt, dass der Traffic nun durch den Server getunnelt wird. Daher bekommen wir bei der Abfrage die IP-Adresse des Servers und nicht mehr unsere eigene IP-Adresse zurück.

7 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:

```
1 # ./revoke-full client-gl
```

Durch das vorangegangene Kommando wurde eine CRL-Datei erstellt (Certificate Revocation List), diese muss nun in ein Verzeichnis kopiert werden, auf das der OpenVPN-Server zugriff hat.

Nun muss noch die CRL-Verifizierung aktiviert werden:

```
1 # curl-verify crl.pem
```

Hierdurch werden die Client-Zertifikate aller sich verbindenden Clients mit der CRL verglichen, und jede positive Übereinstimmung führt zum Abbruch der Verbindung.

Widerruf rückgängig machen

Die “saubere Variante” wäre ein neues Zertifikat zu erstellen, jedoch ist es auch möglich den Widerruf eines Zertifikats rückgängig zu machen. Hierzu muss man im CA-Ordner die `index.txt` bearbeiten, welche die Zertifikats-IDs beinhaltet. Diejenigen, die mit **V** beginnen, sind gültig, und diejenigen mit **R** sind widerrufen. Wir können diese Datei bearbeiten und das erste Zeichen in **V** ändern und die dritte Spalte (das Widerrufsdatum) löschen.

Nun müssen wir die CRL-Datei noch einmal neu generieren und das Zertifikat sollte wieder gültig sein.