Application Note 76

How to configure an Ubuntu OpenVPN server and a Digi TransPort WR as an OpenVPN client
1 INTRODUCTION

1.1 Outline

This document describes how to configure an Ubuntu OpenVPN server and a TransPort WR router as an OpenVPN client.

OpenVPN can be used for securely connecting the WR router to a central office network for access to services on the LAN side of the OpenVPN server, such as corporate messaging services, file servers and print servers for example.

From the OpenVPN website:

OpenVPN is a full-featured SSL VPN which implements OSI layer 2 or 3 secure network extension using the industry standard SSL/TLS protocol, supports flexible client authentication methods based on certificates, smart cards, and/or username/password credentials, and allows user or group-specific access control policies using firewall rules applied to the VPN virtual interface. OpenVPN is not a web application proxy and does not operate through a web browser.

OpenVPN 2.0 expands on the capabilities of OpenVPN 1.x by offering a scalable client/server mode, allowing multiple clients to connect to a single OpenVPN server process over a single TCP or UDP port.

For the purposes of this application note, the following scenario will be used:
OpenVPN is certificate based, so there will be certificates on the two peers.

A PC will be needed that can be used to install the OpenVPN Easy-RSA certificate authority and create & sign the certificates. The Ubuntu system machine will be used for this example purpose as well as OpenVPN server.

1.2 Assumptions

This guide has been written for use by technically competent personnel with a good understanding of the communications technologies used in the product and of the requirements for their specific application. It also assumes a basic ability to access and navigate a Digi Transport router and configure it with basic routing functions.

This application note applies to:

**Model:** Digi Transport WR21

**Other Compatible Models:** All Digi WR Transpost models

**Firmware versions:** 5.077 and later

**Configuration:** This Application Note assumes the devices are set to their factory default configurations. Most configuration commands are only shown if they differ from the factory default.

**Software used:** OpenVPN 2.4.0 on Ubuntu 17.04

**Acknowledgement:** Much of the OpenVPN documentation has been taken directly from the HOWTO pages at the OpenVPN website. Please see [http://openvpn.net/index.php/open-source/documentation/howto.html](http://openvpn.net/index.php/open-source/documentation/howto.html) for more details

1.3 Corrections

Requests for corrections or amendments to this Application Note are welcome and should be addressed to: tech.support@digi.com

Requests for new Application Notes can be sent to the same address.

1.4 Version

<table>
<thead>
<tr>
<th>Version Number</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Published</td>
</tr>
</tbody>
</table>
2 OPENVPN & EASY-RSA SETUP

2.1 Download the OpenVPN installation package and install the software

Open the terminal on the Ubuntu machine and download the latest OpenVPN with the following commands:

```bash
sudo apt-get update
sudo apt-get install openvpn easy-rsa
```

The output will be like the following:
2.2 Setting up Certificate Authority (CA) and generating certificates and keys

The first step in building an OpenVPN 2.x configuration is to establish a PKI (public key infrastructure). The PKI consists of:

- a separate certificate (also known as a public key) and private key for the server and each client
- a master Certificate Authority (CA) certificate and key which is used to sign each of the server and client certificates.

OpenVPN supports bidirectional authentication based on certificates, meaning that the client must authenticate the server certificate and the server must authenticate the client certificate before mutual trust is established.

Both server and client will authenticate the other by first verifying that the presented certificate was signed by the master certificate authority (CA), and then by testing information in the now-authenticated certificate header, such as the certificate common name or certificate type (client or server).

This security model has a number of desirable features from the VPN perspective:

- The server only needs its own certificate/key -- it doesn't need to know the individual certificates of every client which might possibly connect to it.
- The server will only accept clients whose certificates were signed by the master CA certificate (which we will generate below). And because the server can perform this signature verification without needing access to the CA private key itself, it is possible for the CA key (the most sensitive key in the entire PKI) to reside on a completely different machine, even one without a network connection.
- If a private key is compromised, it can be disabled by adding its certificate to a CRL (certificate revocation list). The CRL allows compromised certificates to be selectively rejected without requiring that the entire PKI be rebuilt.
- The server can enforce client-specific access rights based on embedded certificate fields, such as the Common Name.

Note that the server and client clocks need to be roughly in sync or certificates might not work properly.
2.2.1 Generate the master Certificate Authority (CA) certificate & key

Note: If certificates and key files have already been created, skip to section 3.

First of all, we need to create the CA directory and move into it:

```
make-cadir ~/openvpn-ca2
cd ~/openvpn-ca
```

Then, the “vars” file must be edited, using a text editor like “nano”:

```
nano vars
```

The following values needs to be edited in the vars file, with the informations that will be placed in the certificates:

```
# These are the default values for fields
# which will be placed in the certificate.
# Don't leave any of these fields blank.
Terminal DUNTRY="DE"
export KEY_PROVINCE="BY"
export KEY_CITY="Munich"
export KEY_ORG="Digi"
export KEY_EMAIL="support@digi.com"
export KEY_OU="Support"
# X509 Subject Field
export KEY_NAME="server"
```

Please note that in the vars file there is also the possibility to change the Key size to use, in this example a key size of 1024 will be used:

```
# Increase this to 2048 if you # are paranoid. This will slow # down TLS negotiation performance # as well as the one-time DH parms # generation process.
export KEY_SIZE=1024
```

Once the vars file is edited, save it (Ctrl + O) and exit from the editor (Ctrl + X).

Then, issue the source command for that file:

```
source vars
```
You should see the following if it was sourced correctly:

```
Note: If you run ./clean-all, I will be doing a rm -rf on /home/digi/openvpn-ca2/keys
```

Make sure we’re operating in a clean environment by typing:

```
./clean-all
```

Now, the root CA can be created with the following command:

```
./build-ca
```

The output will be like as following:

```bash
Generating a 1024 bit RSA private key
```

Please note that most parameters can be defaulted.
2.2.2 Generate certificate & key for server

Issue the following command on the Ubuntu terminal in order to build the certificates/key for the server side:

```bash
./build-key-server server
```

Please note that most parameters can be defaulted. Be sure that when the Common Name is queried, enter "server". Two other queries require positive responses, "Sign the certificate? [y/n]" and "1 out of 1 certificate requests certified, commit? [y/n]".

The output will be like the following:

```
Generating a 1024 bit RSA private key
.................+++++
writing new private key to 'server.key'
****
You are about to be asked to enter information that will be incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.' the field will be left blank.
*****
Country Name (2 letter code) [DE]:
State or Province Name (full name) [BY]:
Locality Name (eg, city) [Munich]:
Organization Name (eg, company) [DigiL]:
Organizational Unit Name (eg, section) [support]:
Common Name (eg, your name or your server's hostname) [server]:
Name [EasyRSA2]:
Email Address [support@digi.com]:
```

Please enter the following 'extra' attributes to be sent with your certificate request
A challenge password []:
An optional company name []:
Using configuration from /home/digi/openvpn-ca2/openssl-1.0.0.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName :PRINTABLE:'DE'
stateOrProvinceName :PRINTABLE:'BY'
localityName :PRINTABLE:'Munich'
organizationName :PRINTABLE:'DigiL'
organizationalUnitName :PRINTABLE:'support'\commonName :PRINTABLE:'server'\nname :PRINTABLE:'EasyRSA2'
emailAddress :IA5STRING:'support@digi.com'
Certificate is to be certified until Oct 1 13:43:33 2027 GMT (3650 days)
Sign the certificate? [y/n] :y

1 out of 1 certificate requests certified, commit? [y/n] :y
Write out database with 1 new entries
Data Base Updated
### 2.2.3 Generate certificates & keys for the client

Generating client certificates is very similar to the previous step:

```
./build-key client1
```
2.2.4 Generate Diffie Hellman parameters

Diffie Hellman parameters must be generated for the OpenVPN server with the following command:

```bash
./build-dh
```

![Command output]

2.2.5 Generate TLS-AUTH key file

The tls-auth directive adds an additional HMAC signature to all SSL/TLS handshake packets for integrity verification. Any UDP packet not bearing the correct HMAC signature can be dropped without further processing. The tls-auth HMAC signature provides an additional level of security above and beyond that provided by SSL/TLS. It can protect against:

- DoS attacks or port flooding on the OpenVPN UDP port.
- Port scanning to determine which server UDP ports are in a listening state.
- Buffer overflow vulnerabilities in the SSL/TLS implementation.
- SSL/TLS handshake initiations from unauthorized machines (while such handshakes would ultimately fail to authenticate, tls-auth can cut them off at a much earlier point).

Using tls-auth requires that you generate a shared-secret key that is used in addition to the standard RSA certificate/key:

```bash
sudo openvpn --genkey --secret keys/ta.key
```

This command will generate an OpenVPN static key and write it to the file ta.key. This key should be copied over a pre-existing secure channel to the server and all client machines. It can be placed in the same directory as the RSA .key and .crt files.
### 2.2.6 Key Files

Now we will find our newly-generated keys and certificates in the keys subdirectory:

```bash
cd ~/openvpn-ca2/keys
```

Here is an explanation of the relevant files:

<table>
<thead>
<tr>
<th>Filename</th>
<th>Needed By</th>
<th>Purpose</th>
<th>Secret</th>
</tr>
</thead>
<tbody>
<tr>
<td>ca.crt</td>
<td>server + all clients</td>
<td>Root CA certificate</td>
<td>NO</td>
</tr>
<tr>
<td>ca.key</td>
<td>key signing machine only</td>
<td>Root CA key</td>
<td>YES</td>
</tr>
<tr>
<td>dh{n}.pem</td>
<td>server only</td>
<td>Diffie Hellman parameters</td>
<td>NO</td>
</tr>
<tr>
<td>server.crt</td>
<td>server only</td>
<td>Server Certificate</td>
<td>NO</td>
</tr>
<tr>
<td>server.key</td>
<td>server only</td>
<td>Server Key</td>
<td>YES</td>
</tr>
<tr>
<td>client1.crt</td>
<td>client1 only</td>
<td>Client1 Certificate</td>
<td>NO</td>
</tr>
<tr>
<td>client1.key</td>
<td>client1 only</td>
<td>Client1 Key</td>
<td>YES</td>
</tr>
<tr>
<td>ta.key</td>
<td>server+ all clients</td>
<td>Shared secret for TLS</td>
<td>YES</td>
</tr>
</tbody>
</table>

The final step in the key generation process is to copy all files to the machines which need them, taking care to copy secret files over a secure channel.
3 UBUNTU OPENVPN SERVER CONFIGURATION

The following steps explain the configuration that needs to be done on the Ubuntu OpenVPN server.

3.1 Install the OpenVPN software

This step is only required if the OpenVPN server is a different PC to the one used to create RSA certificates earlier. In this example the same Ubuntu system is used as per section above, so the software is already installed and ready to use.

Please follow same steps as section 2.1 if a new installation is needed on the server.

3.2 Check Network Interfaces

Check Network Interfaces on the Ubuntu Server using the following command:

```
ifconfig
```

The IP addresses on the WAN interface (enp0s3) and LAN interface (enp0s8) will be used later for the configuration and test of the OpenVPN connection:

<table>
<thead>
<tr>
<th>Interface</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAN</td>
<td>10.104.1.125/24</td>
</tr>
<tr>
<td>LAN</td>
<td>172.16.0.1/24</td>
</tr>
</tbody>
</table>
3.3 Install the SSL certificates

The SSL certificates that were created earlier should now be securely transferred from the Certificate Authority PC to the config folder of the Server machine (the Ubuntu system in this case).

The files that should be moved are:

- `ca.crt`
- `dh1024.pem`
- `server.crt`
- `server.key`
- `ta.key`

In order to copy them to the correct OpenVPN folder, issue the following command:

```
sudo cp ca.crt server.crt server.key dh1024.pem ta.key /etc/openvpn
```
### 3.4 Configure the OpenVPN Server (server.conf)

Next, we need to copy and unzip a sample OpenVPN configuration file into configuration directory so that can be used as a basis for the configuration:

```
gunzip -c /usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz | sudo tee /etc/openvpn/server.conf
```

This command will shows you the file content:

```
[...]
```

The configuration file must be edited, using a text editor like “nano”:

```
sudo nano /etc/openvpn/server.conf
```
The text file will be opened in the terminal window and the following sections should be edited/checked:

**local 10.104.1.125**

This is the local IP on which the OpenVPN server should listen on, so it is set in this case as the IP address that the server has on its WAN Interface.

---

**server 10.8.0.0. 255.255.255.0**

This is the subnet that will be associated to the OpenVPN connections. The server will take the first address (so 10.8.0.1) and the others will be available for the clients.
push “route 172.16.0.0 255.255.255.0”

This is the route to be pushed to the clients in order for them to reach the private LAN behind the server.

push “dhcp-option DNS 208.67.220.220”

This is the command to push the DNS server address to clients. In this example the public OpenDNS address is set:

push “route 192.168.10.0 255.255.255.0”
push “route 172.16.0.0 255.255.255.0”

push “dhcp-option DNS 208.67.220.220”

This is the command to push the DNS server address to clients. In this example the public OpenDNS address is set:
dh  dh1024.pem

The DH parameters generated before must be set here.

```
# Diffie hellman parameters.
# Generate your own with:
# openssl dhparam -out dh2048.pem 2048
dh  dh1024.pem
```

tls-auth  ta.key  0

The TLS_AUTH key generated before must be set here.

**NOTE:** after the name of the key file, a direction parameter must be set. On server side this should be 0 as in this example. On the client side should be “1” in general, but please refer to section 4.5 for the correct setting on TransPort WR clients.

```
# For extra security beyond that provided
# by SSL/TLS, create an "HMAC firewall"
# to help block DoS attacks and UDP port flooding.
#
# Generate with:
# openvpn --genkey --secret ta.key
#
# The server and each client must have
# a copy of this key.
# The second parameter should be '0'
# on the server and '1' on the clients.
tls-auth  ta.key  0  # This file is secret
```
cipher AES-256-CBC

A cryptographic cipher needs to be chosen. For this application AES-256-CBC is set.

;comp-lzo

Be sure that the compression “comp.lzo” is disabled (so commented with the “;”), as this is not supported on TransPort WR routers.

When finished, save the file (Ctrl + O) and exit from the editor (Ctrl + X).
3.5 Start the OpenVPN Server

Use the following command to start the OpenVPN service:

```bash
sudo systemctl start openvpn@server
```

To double ready to accept client connections, issue the following: check that the OpenVPN server has started successfully and it

```bash
sudo systemctl status openvpn@server
```

An output like the following should be displayed, showing that the status is “active (running)”:  

![OpenVPN status output](https://example.com/openvpn_status.png)

You can also check that the OpenVPN tun0 interface is available by typing:

```bash
ip addr show tun0
```

This command should show as output a configured interface as following:

![OpenVPN interface output](https://example.com/openvpn_interface.png)

If everything is ok, the service can be also configured to start automatically at boot:

```bash
sudo systemctl enable openvpn@server
```
4 TRANSPORT WR CONFIGURATION

4.1 WAN Interface configuration

In this example the Client has the Mobile interface as the WAN interface and it is configured as follows:

 CONFIGURATION - NETWORK > INTERFACES > MOBILE

Where:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Plan/APN</td>
<td>internet.t-d1.de</td>
<td>Enter the APN of your mobile provider</td>
</tr>
</tbody>
</table>

**Please note:** Depending on provider, a SIM PIN or Username/Password may be required. If needed, enter them in the appropriate fields.
4.2 LAN Interface configuration

In this example, the LAN interface is configured with a static address as follows:

**CONFIGURATION - NETWORK > INTERFACES > ETHERNET > ETH 0**

![Configuration interface screenshot]

Where:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Address</td>
<td>172.16.1.1</td>
<td>Enter the IP address of the LAN interface for the router</td>
</tr>
<tr>
<td>Mask</td>
<td>255.255.255.0</td>
<td>Enter the subnet mask</td>
</tr>
</tbody>
</table>
4.3 Transfer Certificates and Key files

Before to transfer the Certificates and Key files on the client, they must be renamed as follows as .pem files:

<table>
<thead>
<tr>
<th>Filename</th>
<th>Purpose</th>
<th>New FileName</th>
</tr>
</thead>
<tbody>
<tr>
<td>ca.crt</td>
<td>Root CA certificate</td>
<td>caovpn.pem</td>
</tr>
<tr>
<td>client1.crt</td>
<td>Client1 Certificate</td>
<td>certcli1.pem</td>
</tr>
<tr>
<td>client1.key</td>
<td>Client1 Key</td>
<td>privcli1.pem</td>
</tr>
<tr>
<td>ta.key</td>
<td>Shared secret for TLS Authentication</td>
<td>ta.key</td>
</tr>
</tbody>
</table>

Note that the ta.key file for TLS authentication don’t need to be renamed.

Once done that, the files can be transferred to the Client using for example an FTP client, connected with the TransPort router with usual username and password.

Please note that you may need to change your IP on the laptop accordingly with the new IP address configured on the ETH0 of the router.
4.4 SSL Certificates configuration

When the certificates have been transferred to the Client, the router needs to be configured so it knows which client certificate files to use:

**CONFIGURATION – NETWORK > SSL**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Certificate Filename</td>
<td>certcli1.pem</td>
<td>The name of the required certificate file is selected from those available on the router’s filing system from this drop-down list. In this example this the one just transferred to the router.</td>
</tr>
<tr>
<td>Client Private Key Filename</td>
<td>privcli1.pem</td>
<td>The name of the file that contains the private key that matches the public key stored in the above parameter, is selected from this drop-down list. In this example this the one just transferred to the router.</td>
</tr>
</tbody>
</table>
4.5 OpenVPN Client mode configuration

An OpenVPN interface will be configured on the TransPort router that acts as OpenVPN client:

CONFIGURATION - NETWORK > VIRTUAL PRIVATE NETWORKING (VPN) > OPENVPN > OPENVPN 0
Where:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>toUbuntuServer</td>
<td>Friendly name for this interface</td>
</tr>
<tr>
<td>Port</td>
<td>1194 (default)</td>
<td>This is the TCP or UDP port number that the server will listen on for incoming VPN connections</td>
</tr>
<tr>
<td>Protocol</td>
<td>UDP (default)</td>
<td>This will either be TCP or UDP. It is up to the reader to decide which protocol to use, both the server and all clients must use the same protocol. See note with regards to protocol choice in the previous section</td>
</tr>
<tr>
<td>Keepalive TX Interval</td>
<td>10</td>
<td>Keepalive interval: Interval between OpenVPN ping transmissions. These are required to detect the operational state of the VPN connection.</td>
</tr>
<tr>
<td>Keepalive RX Timeout</td>
<td>120</td>
<td>Keepalive timeout before VPN is marked as down: If the server hasn’t received a ping from the client in the time limit specified, the tunnel will be marked as down</td>
</tr>
<tr>
<td>Cipher</td>
<td>AES-256-CBC</td>
<td>Encryption algorithm to use. The cipher is not negotiated during tunnel establishment. The server and all clients must be configured to use the same cipher. If the ciphers do not match, decryption errors will occur.</td>
</tr>
<tr>
<td>Digest</td>
<td>SHA1 (default)</td>
<td>Authentication algorithm to use. The digest is not negotiated during tunnel establishment. The server and all clients must be configured to use the same digest. If the ciphers do not match, authentication errors will occur.</td>
</tr>
<tr>
<td>Source IP address</td>
<td>From outgoing interface (default)</td>
<td>The IP address of the outgoing interface will be used as the source IP address</td>
</tr>
<tr>
<td>Client Mode</td>
<td>Selected</td>
<td>Use Client mode</td>
</tr>
<tr>
<td>Connect to OpenVPN server</td>
<td>10.104.1.125</td>
<td>Public IP address of OpenVPN server</td>
</tr>
<tr>
<td>Automatically connect interface</td>
<td>✓</td>
<td>Connects to the OpenVPN server automatically, always on mode.</td>
</tr>
<tr>
<td>Obtain IP address from the OpenVPN server</td>
<td>✓</td>
<td>This interface will obtain an IP address from the OpenVPN server</td>
</tr>
<tr>
<td>Obtain routes from the OpenVPN server</td>
<td>✓</td>
<td>Routing information will be obtained from the OpenVPN server</td>
</tr>
<tr>
<td>Obtain DNS Server IP address from the OpenVPN server</td>
<td>✓</td>
<td>DNS Server information will be obtained from the OpenVPN server</td>
</tr>
<tr>
<td>Use file for TLS Auth Key</td>
<td>Selected</td>
<td>Enables the use of a key file for the TLS Authentication</td>
</tr>
<tr>
<td>TLS password filename</td>
<td>ta.key</td>
<td>Select the TLS password file just uploaded</td>
</tr>
<tr>
<td>TLS Auth Key direction</td>
<td>Inverse</td>
<td>Select the direction for the TLS Authentication Key. On the OpenVPN documentation is indicated to choose “1” on the client side. For TransPort WR router, there are 3 options Bidirectional, Normal and Inverse. On client side, “Inverse” should be used. So please note that this will be shown as “ovpn 0 tlskeydir 2” in the configuration file for the WR, not “1”. (1 will be Normal and 0 will be Bidirectional)</td>
</tr>
</tbody>
</table>

Page | 27
5 TEST OPENVPN CONNECTION

5.1 OpenVPN Connection Status

To check the OpenVPN connection status on the client, browse to:

MANAGEMENT - CONNECTIONS > VIRTUAL PRIVATE NETWORKING (VPN) > OPENVPN > OVPN 0

This will show if the connection is active and all the network settings pushed by the server, as well as traffic statistics details.
On Server side check the status of the OpenVPN Server and latest eventlogs will be displayed, showing the Client connection:

```
digi@Digi:~/openvpn-ca2/keys$ sudo systemctl status openvpn@server
● openvpn@server.service - OpenVPN connection to server
    Loaded: loaded (/lib/systemd/system/openvpn@.service; enabled; vendor preset: enabled)
    Active: active (running) since Thu 2017-10-05 15:57:19 CEST; 23h ago
      Docs: man:openvpn(8)
              https://community.openvpn.net/openvpn/wiki/Openvpn23ManPage
              https://community.openvpn.net/openvpn/wiki/HOWTO
    Main PID: 12394 (openvpn)
    CGroup: /system.slice/system-openvpn.slice/openvpn@server.service
         └─12394 /usr/sbin/openvpn --daemon ovpn-server --status /run/openvpn/server.status 10 --cd /etc/openvpn --script-security 2 --config /etc/openvpn/server.conf --writepid /run/openvpn/server.pid

Okt 06 15:25:12 Digi ovpn-server[12394]: 10.104.1.115:13039 Data Channel Decrypt: Cipher 'AES-256-CBC' initialized with 256 bit key
Okt 06 15:25:12 Digi ovpn-server[12394]: 10.104.1.115:13039 Data Channel Decrypt: Using 160 bit message hash 'SHA1' for HMAC authentication
Okt 06 15:25:12 Digi ovpn-server[12394]: MULTI: new connection by client 'client1' will cause previous active sessions by this client to be dropped. Remember to use the --duplicate-cn option if you want
Okt 06 15:25:12 Digi ovpn-server[12394]: MULTI_sva: pool returned IPv4=10.8.0.10, IPv6=(Not enabled)
Okt 06 15:25:12 Digi ovpn-server[12394]: MULTI: Learn: 10.8.0.10 -> client1/10.104.1.115:13039
Okt 06 15:25:12 Digi ovpn-server[12394]: MULTI: primary virtual IP for client1/10.104.1.115:13039: 10.8.0.10
Okt 06 15:25:12 Digi ovpn-server[12394]: client1/10.104.1.115:13039 PUSH: Received control message: 'PUSH_REQUEST'
Okt 06 15:25:12 Digi ovpn-server[12394]: client1/10.104.1.115:13039 SENT CONTROL [client1]: 'PUSH_REPLY,route 172.16.0.0 255.255.255.0,dhcp-option DNS 208.67.220.220,route 10.8.0.1,topology net30,ping 10,
lines 1-20/20 (END)```
5.2 Routing Table

To better check that all routing information are correct in order to have the connection working as expected, check the routing table:

**MANAGEMENT - NETWORK STATUS > IP ROUTING TABLE**

![IP Routing Table]

The network destination 172.16.0.0 with mask 255.255.255.0 is the route that has been pushed from the OpenVPN server.

5.3 Check the traffic on the OpenVPN Connection

Ping the OpenVPN Server address from the TransPort WR:

![Administration - Execute a command]

Command: ping 10.8.0.1

Command result:

Pinging Addr [10.8.0.1]

<table>
<thead>
<tr>
<th>sent</th>
<th>received</th>
<th>success</th>
<th>average RTT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>100%</td>
<td>0.00 seconds</td>
</tr>
</tbody>
</table>

OK
Ping the OpenVPN Server LAN address from the TransPort WR:

![Administration - Execute a command]

Both Ping will be successful and will be sent via the OpenVPN interface OVPN0.
### 6 FIRMWARE VERSIONS

#### 6.1 Digi TransPort WR

<table>
<thead>
<tr>
<th>Module</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digi TransPort WR21-U22B-DE1-XX Ser#:237416</td>
<td></td>
</tr>
<tr>
<td>Software Build Ver5.2.19.6.</td>
<td>Aug 23 2017 11:05:52 WW</td>
</tr>
<tr>
<td>ARM Bios Ver 7.61u v43 454MHz B987-M995-F80-08140,0 MAC:00042d039f68</td>
<td></td>
</tr>
<tr>
<td>Async Driver</td>
<td>Revision: 1.19 Int clk</td>
</tr>
<tr>
<td>Ethernet Port Isolate Driver</td>
<td>Revision: 1.11</td>
</tr>
<tr>
<td>Firewall</td>
<td>Revision: 1.0</td>
</tr>
<tr>
<td>EventEdit</td>
<td>Revision: 1.0</td>
</tr>
<tr>
<td>Timer Module</td>
<td>Revision: 1.1</td>
</tr>
<tr>
<td>(B)USBHOST</td>
<td>Revision: 1.0</td>
</tr>
<tr>
<td>L2TP</td>
<td>Revision: 1.10</td>
</tr>
<tr>
<td>PPTP</td>
<td>Revision: 1.00</td>
</tr>
<tr>
<td>TACPLUS</td>
<td>Revision: 1.00</td>
</tr>
<tr>
<td>MODBUS</td>
<td>Revision: 0.00</td>
</tr>
<tr>
<td>RealPort</td>
<td>Revision: 0.00</td>
</tr>
<tr>
<td>MultiTX</td>
<td>Revision: 1.00</td>
</tr>
<tr>
<td>LAPB</td>
<td>Revision: 1.12</td>
</tr>
<tr>
<td>X25 Layer</td>
<td>Revision: 1.19</td>
</tr>
<tr>
<td>MACRO</td>
<td>Revision: 1.0</td>
</tr>
<tr>
<td>PAD</td>
<td>Revision: 1.4</td>
</tr>
<tr>
<td>X25 Switch</td>
<td>Revision: 1.7</td>
</tr>
<tr>
<td>V120</td>
<td>Revision: 1.16</td>
</tr>
<tr>
<td>TPAD Interface</td>
<td>Revision: 1.12</td>
</tr>
<tr>
<td>GPS</td>
<td>Revision: 1.0</td>
</tr>
<tr>
<td>TELITUPD</td>
<td>Revision: 1.0</td>
</tr>
<tr>
<td>SCRIBATSK</td>
<td>Revision: 1.0</td>
</tr>
<tr>
<td>BASTSK</td>
<td>Revision: 1.0</td>
</tr>
<tr>
<td>PYTHON</td>
<td>Revision: 1.0</td>
</tr>
<tr>
<td>CLOUDSMS</td>
<td>Revision: 1.0</td>
</tr>
<tr>
<td>TCP (HASH mode)</td>
<td>Revision: 1.14</td>
</tr>
<tr>
<td>TCP Utils</td>
<td>Revision: 1.13</td>
</tr>
<tr>
<td>PPP</td>
<td>Revision: 5.2</td>
</tr>
<tr>
<td>WEB</td>
<td>Revision: 1.5</td>
</tr>
<tr>
<td>SMTP</td>
<td>Revision: 1.1</td>
</tr>
<tr>
<td>FTP Client</td>
<td>Revision: 1.5</td>
</tr>
<tr>
<td>FTP</td>
<td>Revision: 1.5</td>
</tr>
<tr>
<td>IKE</td>
<td>Revision: 1.0</td>
</tr>
<tr>
<td>PollANS</td>
<td>Revision: 1.2</td>
</tr>
<tr>
<td>PPPPOE</td>
<td>Revision: 1.0</td>
</tr>
<tr>
<td>BRIDGE</td>
<td>Revision: 1.1</td>
</tr>
<tr>
<td>MODEM CC (Huawei LTE)</td>
<td>Revision: 5.2</td>
</tr>
<tr>
<td>FLASH Write</td>
<td>Revision: 1.2</td>
</tr>
<tr>
<td>Command Interpreter</td>
<td>Revision: 1.38</td>
</tr>
<tr>
<td>SSLCLI</td>
<td>Revision: 1.0</td>
</tr>
<tr>
<td>OSPF</td>
<td>Revision: 1.0</td>
</tr>
<tr>
<td>BGP</td>
<td>Revision: 1.0</td>
</tr>
<tr>
<td>QOS</td>
<td>Revision: 1.0</td>
</tr>
<tr>
<td>PWRCTRL</td>
<td>Revision: 1.0</td>
</tr>
<tr>
<td>RADIUS Client</td>
<td>Revision: 1.0</td>
</tr>
</tbody>
</table>
6.2 Ubuntu OpenVPN Server

digi@Digi:~/openvpn-ca2/keys$ openvpn --version
OpenVPN 2.4.0 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11]
[MH/PKTINFO] [AEAD] built on Jun 22 2017
library versions: OpenSSL 1.0.2g 1 Mar 2016, LZO 2.08
Originally developed by James Yonan
Copyright (C) 2002-2017 OpenVPN Technologies, Inc. <sales@openvpn.net>
Compile time defines: enable_async_push=no enable_comp_stub=no enable_crypto=yes
enable_crypto_ofb_cfb=yes enable_debug=yes enable_def_auth=yes
enable_dependency_tracking=no enable_dlopen=unknown enable_dlopen_self=unknown
enable_dlopen_self_static=unknown enable_fast_install=needless enable_fragment=yes
enable_iproute2=yes enable_libtool_lock=yes enable_lz4=yes enable_lzo=yes
enable_maintainer_mode=no enable_management=yes enable_multi=yes enable_multihome=yes
enable_pam_dlopen=no enable_password_save=yes enable_pedantic=no enable_pf=yes
enable_pkcs11=yes enable_plugin_auth_pam=yes enable_plugin_down_root=yes
enable_plugins=yes enable_port_share=yes enable_selinux=no enable_server=yes
enable_shared=yes enable_shared_with_static_runtimes=no enable_silent_rules=no
enable_small=no enable_static=yes enable_strict=no enable_strict_options=no
enable_systemd=yes enable_werror=no enable_win32.dll=yes enable_x509_alt_username=yes
with_crypto_library=openssl with_gnu_ld=yes with_mem_check=no
with_plugindir='${prefix}/lib/openvpn' with_sysroot=no
7 CONFIGURATION FILES

7.1 Digi Transport WR

eth 0 IPaddr "172.16.1.1"
eth 0 ipanon ON
lapb 0 ans OFF
lapb 0 tinact 120
lapb 1 tinact 120
lapb 3 dtemode 0
lapb 4 dtemode 0
lapb 5 dtemode 0
lapb 6 dtemode 0
ip 0 cidr ON
def_route 0 ll_ent "ppp"
def_route 0 ll_add 1
dhcp 0 IPmin "192.168.1.100"
dhcp 0 respdelms 500
dhcp 0 mask "255.255.255.0"
dhcp 0 gateway "192.168.1.1"
dhcp 0 DNS "192.168.1.1"
sntp 0 server "time.devicecloud.com"
dyndns 0 ifent "default"
ppp 0 timeout 300
ppp 1 name "W-WAN (LTE)"
ppp 1 phonenum "*98*1#"
ppp 1 IPaddr "0.0.0.0"
ppp 1 timeout 0
ppp 1 use_modem 1
ppp 1 aodion 1
ppp 1 autoassert 1
ppp 1 r_chap OFF
ppp 3 defpak 16
ppp 4 defpak 16
web 0 prelogin_info ON
web 0 showgswiz ON
modemcc 0 info_asy_add 4
modemcc 0 init_str "+CGQREQ=1"
modemcc 0 init_str1 "+CGQMIN=1"
modemcc 0 apn "internet.t-d1.de"
modemcc 0 link_retries 10
modemcc 0 stat_retries 30
modemcc 0 sms_interval 1
modemcc 0 sms_access 1
modemcc 0 sms_concat 0
modemcc 0 init_str_2 "+CGQREQ=1"
modemcc 0 init_str1_2 "+CGQMIN=1"
modemcc 0 apn_2 "Your.APN.goes.here"
modemcc 0 link_retries_2 10
modemcc 0 stat_retries_2 30
modemcc 0 sms_access_2 1
modemcc 0 sms_concat_2 0
ana 0 anon ON
ana 0 l2on OFF
ana 0 xoton OFF
ana 0 lapdon 0
ana 0 lapbon 0
ana 0 maxdata 1500
ana 0 logsize 180
cmd 0 unitid "ss%s>"
cmd 0 cmdnua "99"
cmd 0 hostname "digi.router"
cmd 0 asyled_mode 2
cmd 0 rcihttp ON
user 0 access 0
user 1 name "username"
user 1 epassword "KD5lSVJDVWg="
user 1 access 0
user 2 access 0
user 3 access 0
user 4 access 0
user 5 access 0
user 6 access 0
user 7 access 0
user 8 access 0
user 9 access 0
local 0 transaccess 2
sslcli 0 certfile "certcli1.pem"
sslcli 0 keyfile "privcli1.pem"
sslcli 0 verify 10
sslsvr 0 certfile "cert01.pem"
sslsvr 0 keyfile "privrsa.pem"
ssh 0 hostkey1 "privSSH.pem"
ssh 0 nb_listen 5
ssh 0 v1 OFF
ovpn 0 descr "toUbuntuServer"
ovpn 0 dest "10.104.1.125"
ovpn 0 autoup ON
ovpn 0 ipanon ON
ovpn 0 pullip ON
ovpn 0 pullroute ON
ovpn 0 pulldns ON
ovpn 0 pingint 10
ovpn 0 pingto 120
ovpn 0 cipher "AES-256-CBC"
ovpn 0 tlskeyfile "ta.key"
ovpn 0 tlskeydir 2
templog 0 mo_autooff ON
cloud 0 ssl ON
7.2 Ubuntu OpenVPN Server

### Sample OpenVPN 2.0 config file for multi-client server.

# This file is for the server side of a many-clients <-> one-server OpenVPN configuration.

# OpenVPN also supports single-machine <-> single-machine configurations (See the Examples page on the web site for more info).

# This config should work on Windows or Linux/BSD systems. Remember on Windows to quote pathnames and use double backslashes, e.g.:

# "C:\Program Files\OpenVPN\config\foo.key"

# Comments are preceded with '#' or ';'

# Which local IP address should OpenVPN listen on? (optional)
local 10.104.1.125

# Which TCP/UDP port should OpenVPN listen on?
port 1194

# TCP or UDP server?
;proto tcp
proto udp

# "dev tun" will create a routed IP tunnel,
# "dev tap" will create an ethernet tunnel.
# Use "dev tap0" if you are ethernet bridging
# and have precreated a tap0 virtual interface
# and bridged it with your ethernet interface.
# If you want to control access policies over the VPN, you must create firewall rules for the the TUN/TAP interface.
# On non-Windows systems, you can give an explicit unit number, such as tun0.
# On Windows, use "dev-node" for this.
# On most systems, the VPN will not function unless you partially or fully disable the firewall for the TUN/TAP interface.
;dev tap
dev tun

# Windows needs the TAP-Win32 adapter name
# from the Network Connections panel if you
# have more than one. On XP SP2 or higher,
# you may need to selectively disable the
# Windows firewall for the TAP adapter.
# Non-Windows systems usually don't need this.
;dev-node MyTap

# SSL/TLS root certificate (ca), certificate
# (cert), and private key (key). Each client
# and the server must have their own cert and
# key file. The server and all clients will
# use the same ca file.
#
# See the "easy-rsa" directory for a series
# of scripts for generating RSA certificates
# and private keys. Remember to use
# a unique Common Name for the server
# and each of the client certificates.
#
# Any X509 key management system can be used.
# OpenVPN can also use a PKCS #12 formatted key file
# (see "pkcs12" directive in man page).
ca ca.crt

key server.crt  # This file should be kept secret

# Diffie hellman parameters.
# Generate your own with:
# openssl dhparam -out dh2048.pem 2048

dh dh1024.pem

# Network topology
# Should be subnet (addressing via IP)
# unless Windows clients v2.0.9 and lower have to
# be supported (then net30, i.e. a /30 per client)
# Defaults to net30 (not recommended)
;topology subnet

# Configure server mode and supply a VPN subnet
# for OpenVPN to draw client addresses from.
# The server will take 10.8.0.1 for itself,
# the rest will be made available to clients.
# Each client will be able to reach the server
# on 10.8.0.1. Comment this line out if you are
# ethernet bridging. See the man page for more info.
server 10.8.0.0 255.255.255.0

# Maintain a record of client <-> virtual IP address
# associations in this file. If OpenVPN goes down or
# is restarted, reconnecting clients can be assigned
# the same virtual IP address from the pool that was
# previously assigned.
ifconfig-pool-persist ipp.txt

# Configure server mode for ethernet bridging.
# You must first use your OS's bridging capability
# to bridge the TAP interface with the ethernet
# NIC interface. Then you must manually set the
# IP/netmask on the bridge interface, here we
# assume 10.8.0.4/255.255.255.0. Finally we
# must set aside an IP range in this subnet
# (start=10.8.0.50 end=10.8.0.100) to allocate
# to connecting clients. Leave this line commented
# out unless you are ethernet bridging.
;server-bridge 10.8.0.4 255.255.255.0 10.8.0.50 10.8.0.100

# Configure server mode for ethernet bridging
# using a DHCP-proxy, where clients talk
# to the OpenVPN server-side DHCP server
# to receive their IP address allocation
# and DNS server addresses. You must first use
# your OS's bridging capability to bridge the TAP
# interface with the ethernet NIC interface.
# Note: this mode only works on clients (such as
# Windows), where the client-side TAP adapter is
# bound to a DHCP client.
;server-bridge

# Push routes to the client to allow it
# to reach other private subnets behind
# the server. Remember that these
# private subnets will also need
# to know to route the OpenVPN client
# address pool (10.8.0.0/255.255.255.0)
# back to the OpenVPN server.
;push "route 192.168.10.0 255.255.255.0"
push "route 172.16.0.0 255.255.255.0"

# To assign specific IP addresses to specific
# clients or if a connecting client has a private
# subnet behind it that should also have VPN access,
# use the subdirectory "ccd" for client-specific
# configuration files (see man page for more info).

# EXAMPLE: Suppose the client
# having the certificate common name "Thelonious"
# also has a small subnet behind his connecting
# machine, such as 192.168.40.128/255.255.255.248.
# First, uncomment out these lines:
;client-config-dir ccd
;route 192.168.40.128 255.255.255.248
# Then create a file ccd/Thelonious with this line:
#   iroute 192.168.40.128 255.255.255.248
# This will allow Thelonious' private subnet to
# access the VPN. This example will only work
# if you are routing, not bridging, i.e. you are
# using "dev tun" and "server" directives.

# EXAMPLE: Suppose you want to give
# Thelonious a fixed VPN IP address of 10.9.0.1.
# First uncomment out these lines:
;client-config-dir ccd
;route 10.9.0.0 255.255.255.252
# Then add this line to ccd/Thelonious:
#   ifconfig-push 10.9.0.1 10.9.0.2

# Suppose that you want to enable different
# firewall access policies for different groups
# of clients. There are two methods:
# (1) Run multiple OpenVPN daemons, one for each
#     group, and firewall the TUN/TAP interface
#     for each group/daemon appropriately.
# (2) (Advanced) Create a script to dynamically
#     modify the firewall in response to access
#     from different clients. See man
#     page for more info on learn-address script.
;learn-address ./script

# If enabled, this directive will configure
# all clients to redirect their default
# network gateway through the VPN, causing
# all IP traffic such as web browsing and
# and DNS lookups to go through the VPN
# (The OpenVPN server machine may need to NAT
# or bridge the TUN/TAP interface to the internet
# in order for this to work properly).
;push "redirect-gateway def1 bypass-dhcp"

# Certain Windows-specific network settings
# can be pushed to clients, such as DNS
# or WINS server addresses. CAVEAT:
# http://openvpn.net/faq.html#dhcpcaveats
# The addresses below refer to the public
# DNS servers provided by opendns.com.
;push "dhcp-option DNS 208.67.222.222"
push "dhcp-option DNS 208.67.220.220"

# Uncomment this directive to allow different
# clients to be able to "see" each other.
# By default, clients will only see the server.
# To force clients to only see the server, you
# will also need to appropriately firewall the
# server's TUN/TAP interface.
;client-to-client

# Uncomment this directive if multiple clients
# might connect with the same certificate/key
# files or common names. This is recommended
# only for testing purposes. For production use,
# each client should have its own certificate/key
# pair.
#
# IF YOU HAVE NOT GENERATED INDIVIDUAL
# CERTIFICATE/KEY PAIRS FOR EACH CLIENT,
# EACH HAVING ITS OWN UNIQUE "COMMON NAME",
# UNCOMMENT THIS LINE OUT.
;duplicate-cn
#
# The keepalive directive causes ping-like
# messages to be sent back and forth over
# the link so that each side knows when
# the other side has gone down.
# Ping every 10 seconds, assume that remote
# peer is down if no ping received during
# a 120 second time period.
keepalive 10 120
#
# For extra security beyond that provided
# by SSL/TLS, create an "HMAC firewall"
# to help block DoS attacks and UDP port flooding.
# Generate with:
#   openvpn --genkey --secret ta.key
#
# The server and each client must have
# a copy of this key.
# The second parameter should be '0'
# on the server and '1' on the clients.
tls-auth ta.key 0 # This file is secret
#
# Select a cryptographic cipher.
# This config item must be copied to
# the client config file as well.
# Note that 2.4 client/server will automatically
# negotiate AES-256-GCM in TLS mode.
# See also the ncp-cipher option in the manpage
cipher AES-256-CBC
#
# Enable compression on the VPN link and push the
# option to the client (2.4+ only, for earlier
# versions see below)
;compress lz4-v2
;push "compress lz4-v2"
#
# For compression compatible with older clients use comp-lzo
# If you enable it here, you must also
# enable it in the client config file.
;comp-lzo
#
# The maximum number of concurrently connected
# clients we want to allow.
;max-clients 100
# It's a good idea to reduce the OpenVPN
daemon's privileges after initialization.
#
# You can uncomment this out on
# non-Windows systems.
;user nobody
;group nogroup

# The persist options will try to avoid
# accessing certain resources on restart
# that may no longer be accessible because
# of the privilege downgrade.
persist-key
persist-tun

# Output a short status file showing
# current connections, truncated
# and rewritten every minute.
status openvpn-status.log

# By default, log messages will go to the syslog (or
# on Windows, if running as a service, they will go to
# the "\Program Files\OpenVPN\log" directory).
# Use log or log-append to override this default.
# "log" will truncate the log file on OpenVPN startup,
# while "log-append" will append to it. Use one
# or the other (but not both).
;log openvpn.log
;log-append openvpn.log

# Set the appropriate level of log
# file verbosity.
#
# 0 is silent, except for fatal errors
# 4 is reasonable for general usage
# 5 and 6 can help to debug connection problems
# 9 is extremely verbose
verb 3

# Silence repeating messages. At most 20
# sequential messages of the same message
# category will be output to the log.
;mute 20

# Notify the client that when the server restarts so it
# can automatically reconnect.
explicit-exit-notify 1