OpenVPN Setup & Certificates

1. Install Easy-RSA

The first step in building an OpenVPN 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, and
- a master Certificate Authority (CA) certificate and key which is used to sign each of the server and client certificates.

Install easy-rsa package on your Debian system with the following command:

```
# apt install easy-rsa
```

Create a directory where whole key structure will be stored:

```
# make-cadir /tmp/certs
# cd /tmp/certs
```

Edit the vars file to configure Certificate Authority (CA) variables:

```
#nano ./vars
-----------------------------------------
Uncomment:

set_var EASYRSA_KEY_SIZE        2048
set_var EASYRSA_REQ_COUNTRY     "NL"
set_var EASYRSA_REQ_PROVINCE    "Zuid-Holland"
set_var EASYRSA_REQ_CITY        "Rijnsburg"
set_var EASYRSA_REQ_ORG         "Oscar.de.Groot"
set_var EASYRSA_REQ_EMAIL       "oscar@oscardegroot.nl"
set_var EASYRSA_REQ_OU          "MySites"
```

Generate the required certificates and keys:

```
$ easyrsa init-pki
```

2. Create own CA certificate

```
$ easyrsa build-ca
```

3. Create Server Certificate & Key

Throughout this tutorial, the OpenVPN server's common name will be "MyServerName". Be sure to include the nopass option as well. Failing to do so will password-protect the request file, which could lead to permissions issues later on.

```
$ easyrsa gen-req MyServerName nopass
```

This will create a private key for the server and a certificate request file called server.req. Then sign the request by running easyrsa with the sign-req option, followed by the request type and the common name. The request type can either be client or server, so for the OpenVPN server's certificate request, be sure to use the server request type.

```
$ easyrsa sign-req server MyServerName
```

In the output, you'll be asked to verify that the request comes from a trusted source. Type yes and press ENTER to confirm this.

4. Generating Diffie-Hellman (DH) params

The Diffie–Hellman (DH) Algorithm is a key-exchange protocol that enables two parties communicating over public channel to establish a mutual secret without it being transmitted over the Internet. DH enables the two to use a public key to encrypt and decrypt their conversation or data using symmetric cryptography. After initializing a PKI, any entity can create DH params that needs them. DH key params can be generated with:

```
$ ./easyrsa gen-dh
```

This may take a few minutes to complete.

5. TLS-AUTH

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, port flooding, Port scanning, Buffer overflow vulnerabilities, etc.

Using tls-auth requires that you generate a shared-secret key that is used in addition to the standard RSA certificate/key. Generate an HMAC signature to strengthen the server's TLS integrity verification capabilities:

```
$ openvpn --genkey secret pki/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.

In the server configuration, add:

```
tls-auth ta.key 0
```

In the client configuration, add:

```
tls-auth ta.key 1
```

6. Create Client Certificate and Key Pair

n this step, you will first generate the client key and certificate pair. If you have more than one client, you can repeat this process for each one. Please note, though, that you will need to pass a unique name value to the script for every client. Throughout this tutorial, the first certificate/key pair is referred to as MyVPNClient.

```
$ easyrsa gen-req MyClientName nopass
```

This will create a private key for the client and a certificate request file called MyClientName.req. Then sign the request by running easyrsa with the sign-req option, followed by the request type and the common name. The request type can either be client or server, so for the OpenVPN server's certificate request, be sure to use the server request type.

```
$ easyrsa sign-req server MyClientName
```

In the output, you'll be asked to verify that the request comes from a trusted source. Type yes and press ENTER to confirm this.

7. Deploy Certificates & Keys

With that, all the certificate and key files needed by your server have been generated. You're ready to deploy the corresponding certificates and keys to both OpenVPN Server and Client systems.

Key Files

Now we will find our newly-generated keys and certificates in the keys subdirectory. Here is an explanation of the relevant files:

Filename	**Needed By**	**Purpose**
ca.crt	server + all clients	Root CA certificate
ca.key	key signing machine only	Root CA key
dh{n}.pem	server only	Diffie Hellman parameters
server.crt	server only	Server Certificate

Filename	Needed By	Purpose
server.key	server only	Server Key
client1.crt	client1 only	Client1 Certificate
client1.key	client1 only	Client1 Key

Links

- https://wiki.debian.org/OpenVPN#OpenVPN_Overview
- https://www.webhi.com/how-to/how-to-install-openvpn-server-on-linux-debian-11-12/
- https://www.digitalocean.com/community/tutorials/how-to-set-up-an-openvpn-server-on-debian-11