Easy-RSA

The first step when setting up **OpenVPN** is to create a **Public Key Infrastructure (PKI)**. In summary, this consists of:

- A public master **Certificate Authority (CA)** certificate and a private key.
- A separate public certificate and private key pair for each server.
- A separate public certificate and private key pair for each client.

One can think of the key-based authentication in terms similar to that of how **SSH keys** work with the added layer of a signing authority (the CA). OpenVPN relies on a bidirectional authentication strategy, so the client must authenticate the server's certificate and in parallel, the server must authenticate the client's certificate. This is accomplished by the 3rd party's signature (the CA) on both the client and server certificates. Once this is established, further checks are performed before the authentication is complete. For more details, see **secure-computing's guide (https://www.secure-computing.net/openvpn/howto.php#pki)**.

Note:

- The process outlined below requires users to securely transfer private key files to/from machines. For the purposes of this guide, using scp is shown, but readers may employ alternative methods as well. Since the Arch default is to deny the root user over ssh,

using scp requires transferring ownership of the files to be exported to a non-root user called *foo* throughout the guide.
- Avoid generating keys on devices without a good entropy source. See **[1] (https://community.openvpn.net/openvpn/wiki/GettingStartedwithOVPN#Configuringencryption)**.
- Due to a **bug (https://github.com/OpenVPN/easy-rsa/issues/119)**, the `vars` file is not sourced if it is not specified explicitly.

Contents

Certificate Authority (CA)

For security purposes, it is recommended that the CA machine be separate from the machine running OpenVPN.

On the **CA machine**, install `easy-rsa (https://www.archlinux.org/packages/?name=easy-rsa)`, initialize a new PKI and generate a CA keypair that will be used to sign certificates:

```
# cd /etc/easy-rsa
# export EASYRSA=$(pwd)
# easyrsa init-pki
# easyrsa build-ca
```

OpenVPN server files

A functional OpenVPN server requires the following (in alphabetical order):

1. The CA's public certificate

2. The Diffie-Hellman (DH) parameters file (needed for TLS mode which is recommended).
3. The server key pair (a public certificate and a private key).
4. The Hash-based Message Authentication Code (HMAC) key.

Upon completing the steps outlined in this article, users will have generated the following files on the server:

1. `/etc/openvpn/server/ca.crt`
2. `/etc/openvpn/server/dh.pem`
3. `/etc/openvpn/server/servername.crt` and `/etc/openvpn/server/servername.key`
4. `/etc/openvpn/server/ta.key`

CA public certificate

The CA public certificate `/etc/easy-rsa/pki/ca.crt` generated in the previous step needs to be copied over to the machine that will be running OpenVPN.

On the **CA machine**:

```
# scp /etc/easy-rsa/pki/ca.crt foo@hostname-of-openvpn-server:/tmp/ca.crt
```

On the **OpenVPN server machine**:

```
# mv /tmp/ca.crt /etc/openvpn/server/
# chown root:root /etc/openvpn/server/ca.crt
```

Server certificate and private key

On the **OpenVPN server machine**, install `easy-rsa (https://www.archlinux.org/packages/?name=easy-rsa)` and generate a key pair for the server:

```
# cd /etc/easy-rsa
# easyrsa init-pki
# easyrsa gen-req servername nopass
# cp /etc/easy-rsa/pki/private/servername.key /etc/openvpn/server/
```

This will create two files:

`/etc/easy-rsa/pki/reqs/servername.req`
`/etc/easy-rsa/pki/private/servername.key`

Diffie-Hellman (DH) parameters file

On the **OpenVPN server machine**, create the initial dh.pem file:

```
# openssl dhparam -out /etc/openvpn/server/dh.pem 2048
```

Note: Although values higher than 2048 (4096 for example) may be used, they take considerably more time to generate and offer little benefit in security but advisable to have the DH prime number length to match the length of the RSA key. See **[2] (https://community.openvpn.net/openvpn/wiki/GettingStartedwithOVPN#Configuringencryption)**

Hash-based Message Authentication Code (HMAC) key

On the **OpenVPN server machine**, create the HMAC key:

```
# openvpn --genkey --secret /etc/openvpn/server/ta.key
```

This will be used to add an additional HMAC signature to all SSL/TLS handshake packets. In addition any UDP packet not having the correct HMAC signature will be immediately dropped, protecting against:

- Portscanning.
- DOS attacks on the OpenVPN UDP port.
- SSL/TLS handshake initiations from unauthorized machines.
- Any eventual buffer overflow vulnerabilities in the SSL/TLS implementation.

OpenVPN client files

Client certificate and private key

Any machine can generate client files provided that `easy-rsa (https://www.archlinux.org/packages/?name=easy-rsa)` is installed.

If the pki is not initialized, do so via:

```
# cd /etc/easy-rsa
# easyrsa init-pki
```

Generate the client key and certificate:

```
# cd /etc/easy-rsa
# easyrsa gen-req client1 nopass
```

This will create two files:

`/etc/easy-rsa/pki/reqs/client1.req` `/etc/easy-rsa/pki/private/client1.key`

The gen-req set can be repeated as many times as needed for additional clients.

Sign the certificates and pass them back to the server and clients

Obtain and sign the certificates on the CA

The server and client(s) certificates need to be signed by the CA then transferred back to the OpenVPN server/client(s).

On the **OpenVPN server** (or the box used to generate the certificate/key pairs):

```
# cp /etc/easy-rsa/pki/reqs/*.req /tmp
# chown foo /tmp/*.req
```

Securely transfer the files to the CA machine for signing:

```
$ scp /tmp/*.req foo@hostname-of-CA:/tmp
```

On the **CA machine**, import and sign the certificate requests:

```
# cd /etc/easy-rsa
# easyrsa import-req /tmp/servername.req servername
# easyrsa import-req /tmp/client1.req client1
# easyrsa sign-req server servername
# easyrsa sign-req client client1
```

This will create the following signed certificates which can be transferred back to their respective machines:

`/etc/easy-rsa/pki/issued/servername.crt`
`/etc/easy-rsa/pki/issued/client1.crt`

The leftover .req files can be safely deleted:

```
# rm -f /tmp/*.req
```

Pass the signed certificates back to the server and client(s)

On the **CA machine**, copy the signed certificates and transfer them to the server/client(s):

```
# cp /etc/easy-rsa/pki/issued/*.crt /tmp
# chown foo /tmp/*.crt
$ scp /tmp/*.crt foo@hostname-of-openvpn_server:/tmp
```

On the **OpenVPN server**, move the certificates in place and reassign ownership. For the server:

```
# mv /tmp/servername.crt /etc/openvpn/server/
# chown root:root /etc/openvpn/server/servername.crt
```

For the client:

```
# mkdir /etc/easy-rsa/pki/signed
# mv /tmp/client1.crt /etc/easy-rsa/pki/signed
```

That is it. To generate the client profile. See: **OpenVPN#ovpngen**.

Revoking certificates and alerting the OpenVPN server

Revoke a certificate

Over time, it may become necessary to revoke a certificate thus denying access to the affected user(s). This example revokes the "client1" certificate.

On the **CA machine**:

```
# cd /etc/easy-rsa
# easyrsa revoke client1
# easyrsa gen-crl
```

This will produce the CRL file `/etc/easy-rsa/pki/crl.pem` that needs to be transferred to the OpenVPN server and made active there.

Alert the OpenVPN server

On the **CA machine**:

```
# cp /etc/easy-rsa/pki/crl.pem /tmp
# chown foo /tmp/crl.pem
```

On the **OpenVPN machine**, copy `crl.pem` and inform the server to read it:

```
# mv /tmp/crl.pem /etc/openvpn/server/
# chown root:root /etc/openvpn/server/crl.pem
```

Edit `/etc/openvpn/server/server.conf` uncommenting the crl-verify directive, then **restart** openvpn-server@server.service to re-read it:

```
/etc/openvpn/server/server.conf
```

```
.
crl-verify /etc/openvpn/server/crl.pem
.
```

Abbreviated example specifically for containerized Openvpn

This section is specifically for users wanting to run Openvpn in a Linux container (**LXC**). The code below is designed to be pasted into a root shell; the standard hash has been omitted to allow for easy copy/paste operations. It is recommended to have two different shell windows open, one for the host and one for the container.

Note:

- It is assumed that the CA machine is the host and the server machine is the container.
- Both the host and container need to have both `openvpn (https://www.archlinux.org/packages/?name=openvpn)` and `easy-rsa (https://www.archlinux.org/packages/?name=easy-rsa)` installed.
- The container needs to be running.
- Define the name of the container in the CONTAINERNAME variable below.

On the host:

```
CONTAINERNAME=foo
/etc/easy-rsa
easyrsa init-pki && easyrsa build-ca
cp /etc/easy-rsa/pki/ca.crt /var/lib/lxc/$CONTAINERNAME/rootfs/etc/openvpn/server/
```

Note: One may substitute other names in the 2nd line of this code (the for loop). At a minimum, one needs to generate a key for the server and for at least 1 client. The generic words "server" and "client" are shown, but in reality, these can by any words such as the hostname of the container or the name of the intended user. As well, one can add additional words to the for loop if more than 2 keys are needed. If that is the case, just be sure to add corresponding lines to the subsequent steps for each of them.

In the container:

```
cd /etc/easy-rsa && easyrsa init-pki
for i in server client; do easyrsa gen-req $i nopass; done
cp /etc/easy-rsa/pki/private/server.key /etc/openvpn/server/
openssl dhparam -out /etc/openvpn/server/dh.pem 2048
openvpn --genkey --secret /etc/openvpn/server/ta.key
```

Back on the host:

```
easyrsa import-req /var/lib/lxc/$CONTAINERNAME/rootfs/etc/easy-rsa/pki/reqs/junk.req junk
easyrsa import-req /var/lib/lxc/$CONTAINERNAME/rootfs/etc/easy-rsa/pki/reqs/client.req client
easyrsa sign-req client client
easyrsa sign-req server server
```

```
mkdir /var/lib/lxc/$CONTAINERNAME/rootfs/etc/easy-rsa/pki/issued/
mkdir /var/lib/lxc/$CONTAINERNAME/rootfs/etc/easy-rsa/pki/signed/
cp /etc/easy-rsa/pki/issued/*.crt /var/lib/lxc/$CONTAINERNAME/rootfs/etc/easy-rsa/pki/issued/
```

That will provide the needed files to make an OpenVPN compatible tunnel profile for the client, and the needed server key files for the server. To generate a client profile, refer to **OpenVPN#ovpngen**.

See also

- **README.quickstart (https://github.com/OpenVPN/easy-rsa/blob/master/README.quickstart.md)**.
- **EASYRSA-Advanced (https://github.com/OpenVPN/easy-rsa/blob/master/doc/EasyRSA-Advanced.md)**.

Retrieved from "https://wiki.archlinux.org/index.php?title=Easy-RSA&oldid=508525"