Expanding the scope of the VPN to include additional machines on either the client or server subnet.

Including multiple machines on the server side when using a routed VPN (dev tun)

Once the VPN is operational in a point-to-point capacity between client and server, it may be desirable to expand the scope of the VPN so that clients can reach multiple machines on the server network, rather than only the server machine itself.

For the purpose of this example, we will assume that the server-side LAN uses a subnet of 10.66.0.0/24 and the VPN IP address pool uses 10.8.0.0/24 as cited in the server directive in the OpenVPN server configuration file.

First, you must *advertise* the 10.66.0.0/24 subnet to VPN clients as being accessible through the VPN. This can easily be done with the following server-side config file directive:

```
push "route 10.66.0.0 255.255.255.0"
```

Next, you must set up a route on the server-side LAN gateway to route the VPN client subnet (10.8.0.0/24) to the OpenVPN server (this is only necessary if the OpenVPN server and the LAN gateway are different machines).

Make sure that you've enabled IP and TUN/TAP forwarding on the OpenVPN server machine.

Including multiple machines on the server side when using a bridged VPN (dev tap)

One of the benefits of using ethernet bridging is that you get this for free without needing any additional configuration.

Including multiple machines on the client side when using a routed VPN (dev tun)

In a typical road-warrior or remote access scenario, the client machine connects to the VPN as a single machine. But suppose the client machine is a gateway for a local LAN (such as a home office), and you would like each machine on the client LAN to be able to route through the VPN.

For this example, we will assume that the client LAN is using the 192.168.4.0/24 subnet, and that the VPN client is using a certificate with a common name of client2 . Our goal is to set up the VPN so that any machine on the client LAN can communicate with any machine on the server LAN through the VPN.

Before setup, there are some basic prerequisites which must be followed:

- The client LAN subnet (192.168.4.0/24 in our example) must not be exported to the VPN by the server or any other client sites which are using the same subnet. Every subnet which is joined to the VPN via routing must be unique.
- The client must have a unique Common Name in its certificate ("client2" in our example), and the duplicate

currently reference a client configuration directory, add one now:

```
client-config-dir ccd
```

In the above directive, ccd should be the name of a directory which has been pre-created in the default directory where the OpenVPN server daemon runs. On Linux this tends to be /etc/openvpn and on Windows it is usually \Program Files\OpenVPN\config. When a new client connects to the OpenVPN server, the daemon will check this directory for a file which matches the common name of the connecting client. If a matching file is found, it will be read and processed for additional configuration file directives to be applied to the named client.

The next step is to create a file called client2 in the ccd directory. This file should contain the line:

```
iroute 192.168.4.0 255.255.255.0
```

This will tell the OpenVPN server that the 192.168.4.0/24 subnet should be routed to client2.

Next, add the following line to the main server config file (not the ccd/client2 file):

```
route 192.168.4.0 255.255.255.0
```

Why the redundant route and iroute statements, you might ask? The reason is that route controls the routing from the kernel to the OpenVPN server (via the TUN interface) while iroute controls the routing from the OpenVPN server to the remote clients. Both are necessary.

Next, ask yourself if you would like to allow network traffic between client2's subnet (192.168.4.0/24) and other clients of the OpenVPN server. If so, add the following to the server config file.

```
client-to-client
push "route 192.168.4.0 255.255.255.0"
```

This will cause the OpenVPN server to *advertise* client2's subnet to other connecting clients.

The last step, and one that is often forgotten, is to add a route to the server's LAN gateway which directs 192.168.4.0/24 to the OpenVPN server box (you won't need this if the OpenVPN server box *is* the gateway for the server LAN). Suppose you were missing this step and you tried to ping a machine (not the OpenVPN server itself) on the server LAN from 192.168.4.8? The outgoing ping would probably reach the machine, but then it wouldn't know how to route the ping reply, because it would have no idea how to reach 192.168.4.0/24. The rule of thumb to use is that when routing entire LANs through the VPN (when the VPN server is not the same machine as the LAN gateway), make sure that the gateway for the LAN routes all VPN subnets to the VPN server machine.

Similarly, if the client machine running OpenVPN is not also the gateway for the client LAN, then the gateway for the client LAN must have a route which directs all subnets which should be reachable through the VPN to the OpenVPN client machine.

Including multiple machines on the client side when using a bridged VPN (dev tap)

a DHCP server on the OpenVPN server side of the VPN.

Updates & Announcements

Cyber Shield Released

Release Notes 2.11.3

Access Server

Release Notes

Documentation

Plugins

CloudConnexa™

Features

Cyber Shield

Quick Start Guide

Documentation

Resources

Support Center

What is a VPN?

Resource Center

Vulnerability Reporting

Compliance

Security Advisories

Company

About Us

Careers

Blog

Contact

In The News

Partner with us