# OpenVPN Bridge

This page describes how to create a network bridge on Arch Linux and host an OpenVPN server using a IP layer-2 based Ethernet bridge (TAP) rather than a IP layer-3 based IP tunnel (TUN). The general **OpenVPN** page describes setting up PAM authentication or OpenSSL security certificates in more detail.

## Contents

# Introduction

The **OpenVPN documentation (http://openvpn.net/index.php/open-source/documentation.html)** page gives a full overview of server-side and client-side options that OpenVPN supports. It is easier to set up OpenVPN in tunneling mode and control routing the traffic and it is generally advised to do so if it serves your purpose. However, some network applications, such as Windows file sharing, rely on network broadcasts at the Ethernet level and benefit from believing they are physically located on the same subnet, and software bridging serves this purpose.

There are multiple ways to set bridging up. The dynamic method is where OpenVPN will be managing its own bridge on the system and will start, stop and configure it itself. This is the quickest way to set bridging up, although it interrupts other network services when OpenVPN starts and stops. If the system is going to manage its own bridge, maybe because other virtual network adapters connect to the bridge besides just that of OpenVPN, then it is preferable to use the static method.

# Dynamic Bridge Installation

You will need to **install** OpenVPN and Linux bridging utilities which are available in the **`openvpn (https://www.archlinux.org/packages/?name=openvpn)`** and **`bridge-utils (https://www.archlinux.org/packages/?name=bridge-utils)`** packages.

# Dynamic Bridge Configuration

OpenVPN will create/destroy the TAP device automatically for the name specified in the config file. OpenVPN settings common to TUN or TAP are not shown in the example config file below, only settings that affect TAP mode. Make sure the 'up' and 'down' scripts are executable with 'chmod +x' after you write them.

/etc/openvpn/server.conf (sections common to TUN and TAP omitted)

```
# this uses a dhcp server, server-side
#  clients must support binding their dhcp client to their tap adapter
# do not append 'nogw' if using dhcp
server-bridge
# can specify interface, like tap0 or tap1
#  or use up/down routing scripts to handle
#  more than one, if needed
dev tap0
# needed to call scripts like up/down
#  which call external programs within the scripts
script-security 2
# user defined scripts for adding/removing tap to bridge
#  'dev mtu link_mtu ifconfig_local_ip ifconfig_remote_ip' are appended if set
# make sure 'user' has permission to run 'down' ('up' will be root)
up "up br0 eth0"
down "down br0 eth0"
# call 'down' before TUN/TAP close
down-pre
# drop root priveledges once connected
#  good idea, for servers running on linux
# 'up' script not affected, 'down' script is
;user nobody
;group nobody
```

/etc/openvpn/up

```
#!/bin/bash
br=$1
eth=$2
dev=$3
mtu=$4
cd /usr/bin/

# only if you start dhcpcd and leave it
#  running for eth
#dhcpcd -k $eth

# needed if script is run independently
# but when run through openvpn
# openvpn will do this automatically
#  could also use 'ip tuntap ..'
#openvpn --mktun --dev $dev

brctl addbr $br
# set forwarding delay to 0
#  otherwise dhcp called below would timeout
brctl setfd $br 0
brctl addif $br $eth
# order matters here.. right now there is only
#  one mac in the bridge's table
# if there were two.. there is no guarantee
#  which would be passed to the dhcp server
dhcpcd $br
brctl addif $br $dev

ip link set $eth up promisc on mtu $mtu
ip link set $dev up promisc on mtu $mtu
```

# /etc/openvpn/down

```
#!/bin/bash
br=$1
eth=$2
cd /usr/bin/

dhcpcd -k $br

ip link set $br down
brctl delbr $br

# needed if script is run independently
# but when run through openvpn
# openvpn will do this automatically
#  could also use 'ip tuntap ..'
#openvpn --rmtun --dev $dev

# only if you start dhcpcd and leave it
#  running for eth
#dhcpcd $eth
```

These examples are for using dhcp. If you are going to use static IP addresses, you will need to adjust accordingly.

## Using Systemd

The OpenVPN systemd script looks for <name>.conf files in the /etc/openvpn folder by default. So assuming you have a file named server.conf, you can **enable** and start `openvpn@server` .

Be careful about having dhcpcd enabled separately (ie. dhcpcd@eth0.service) at the same time. It is possible, though unlikely, for it to complete after OpenVPN and ruin your dhcp setup for OpenVPN. You could probably disable dhcpcd@eth0.service since you know openvpn@server.service will be resetting dhcp anyway.

**Warning:** The Static Bridge section does not describe a method using systemd at all. In addition, it may contain outdated information. It should be revised at some point.

## Static Bridge Installation

The first thing you want to do is **install** these packages: `openvpn (https://www.archlinux.org/packages/?name=openvpn)`, `bridge-utils (https://www.archlinux.org/packages/?name=bridge-utils)`, `netctl (https://www.archlinux.org/packages/?name=netctl)`.

# Static Bridge Configuration

Earlier versions of guides for OpenVPN provided by the OpenVPN team or various Linux packagers give example scripts for constructing a bridge when starting OpenVPN and destroying it when shutting OpenVPN down.

However, this is a somewhat deprecated approach, since OpenVPN as of 2.1.1 defaults to not allowing itself to call external scripts or programs unless explicitly enabled to, for security reasons.

Also, constructing the bridge is relatively slow compared to all other parts of the network initialization process. (In fact, so slow that dhcpcd will time out before the bridge is ready. See **#Static Bridge Troubleshooting**.) Also, when restarting OpenVPN after configuration changes, there is no reason to rebuild a working bridge, interrupting all your other network applications. So, setting up a static bridge configuration as follows is the recommended method.

To create an OpenVPN bridge for your server, you are going to have to use netctl and create two network profiles - one for the tap interface and one for the bridge.

Go to `/etc/netctl` and copy the tuntap example file to the directory:

```
# cd /etc/netctl/
# cp examples/tuntap openvpn_tap
```

Now edit openvpn_tap to create a tap interface. It may look like this:

```
/etc/netctl/openvpn_tap
```

```
Description='tuntap connection'
Interface=tap0
Connection=tuntap
Mode='tap'
User='nobody'
Group='nobody'
```

Do not configure the IP address here, this is going to be done for the bridge interface!

To create the bridge profile, copy the example file:

```
# cp examples/bridge openvpn_bridge
```

Now edit openvpn_bridge. It may look like this:

```
/etc/netctl/openvpn_bridge
```

```
Description="Bridge connection"
Interface=br0
Connection=bridge
BindsToInterfaces=(eth0 tap0)
IP=static
Address=('192.168.11.1/24')
Gateway='192.168.11.254'
DNS=('192.168.11.254')
```

For more information, for example how to use DHCP instead, check the **netctl** article.

Now enable and start both profiles with:

```
# netctl enable openvpn_tap
# netctl enable openvpn_bridge
# netctl start openvpn_tap
# netctl start openvpn_bridge
```

# Static Bridge Troubleshooting

Q: Why does starting the network [FAIL] ?

A:This is probably because you are using DHCP on the bridge and setting up the bridge takes longer than dhcpcd is willing to wait. You can fix this by setting the FWD_DELAY parameter in your bridge network profile (openvpn_bridge). Start with a value of 5 and decrease it until it works.

# More Resources

**OpenVPN** | General page on configuring OpenVPN, including setting up authentication methods.

---

Any additions, clarifications, reorganizations, feedback etc. etc. are more than appreciated.

---

Retrieved from "https://wiki.archlinux.org/index.php?title=OpenVPN_Bridge&oldid=467629"

---

- This page was last edited on 4 February 2017, at 15:02.
-