FreeBSD 8, ipfw and OpenVPN 2.1 server (bridged mode)

Tomaž Muraus (kami@k5-storitve.net / @KamiSLO)

October 2009

1. Table of contents

2. Introduction

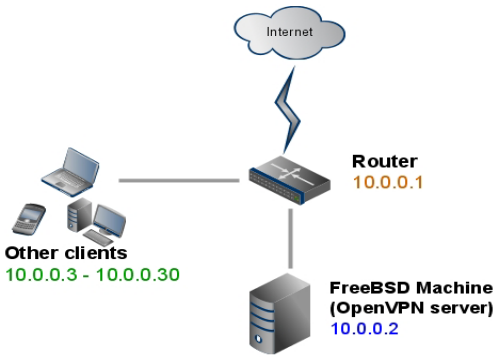
In this article, I will describe how to get OpenVPN 2.1 server up and running in bridged mode on FreeBSD 8.0.

Actually, when I was writing this article, FreeBSD 8.0 production a.k.a. stable release was not yet available, but in any case, this should work on the production release when it's out and most likely on FreeBSD 6 and 7 as well.

3. The setup

For the purpose of this article I will assume we are using the following setup.

Network:



What	IP address / range
Local gateway (router)	10.0.0.1
Local IP address (FreeBSD and VPN server)	10.0.0.2
VPN client address pool	10.0.0.100 - 10.0.0.50

Interfaces:

Interface	Value
Ethernet interface	bge0
Tap interface	tap0
Bridge interface	bridge0

Other server related settings:

Name	Value
OpenVPN server port	22222
OpenVPN protocol	UDP
OpenVPN config files directory	/usr/local/etc/openvpn
OpenVPN config file location	/usr/local/etc/openvpn/server.conf
Root certificate file location	/usr/local/etc/openvpn/ca.crt
Server certificate file location	/usr/local/etc/openvpn/server.crt
Server key file location	/usr/local/etc/openvpn/server.key
Diffie Hellman parameters file location	/usr/local/etc/openvpn/dh1024.pem

4. The difference between routing (tun) and bridged mode (tap)

I won't go into the details here, because the official documenation explains it pretty nicely.

The fundamental difference is, that when a client connects to a VPN server running in a bridged mode, it is assigned an IP address that is a part of the remote physical Ethernet subnet and is then able to interact with other machines on the remote subnet as if it were connected locally - each client's tap interface will be assigned an IP address that is a part of the server's LAN (that is why the VPN client address pool must be in the same subnet as the server LAN subnet).

Getting OpenVPN server to work on this setup in routing mode can be a bit trickier because it requires you to manually change the routing tables and the OpenVPN subnet must be different then the client private LAN subnet.

For example, if you use *192.168.0.0./24* as your VPN subnet and you try to connect to this VPN server from the place which uses the same subnet for its LAN (like Hotel, Internet Caffe or some open wireless network) you will encounter a routing conflict, because the client machine won't know if the *192.168.0.1* refers to the local place's gateway or the same address on the VPN.

In case like this, you should select such subnet for the VPN which you are less likely to encounter - for example *10.0.0.0/8* or *172.16.0.0/24*.

5. Step 1: Installing the OpenVPN server

First thing you need to do is to install the OpenVPN server. You could download the source from the openvpn.org and compile it by yourself or you could just use the FreeBSD port (that is what I will do).

cd /usr/ports/security/openvpn-devel/ && make install clean

Once the port has been compiled and installed, we need to create a directory for storing certificates, keys and two scripts which are executed when the OpenVPN server is started (start script which creates the bridge interface and bridges the two interfaces) and stopped (stop script removes the bridge between the interfaces and deletes the bridge interface).

6. Step 2: Configuring the server

If you want to use the bridged mode, you must enable the IP forwarding. You can do this so by putting the following line in the */etc/rc.conf* (this is always required if you want to route packets between interfaces):

```
gateway_enable="YES"
```

The setting will be activated the next time you reboot the computer. If you want changes to take the effect immediately, use the following command:

```
# sysctl net.inet.ip.forwarding=1
```

You can check if the IP forwarding was successfully enabled by executing the following command:

```
# sysctl -a | grep net.inet.ip.forwarding
```

You should receive output like this:

```
net.inet.ip.forwarding: 1
```

When the IP forwarding is enabled, you need to generate the necessary certificates and keys (master certificate authority certificate and key, server certificate and key and client certificates and keys).

I won't describe this here, because the process is pretty straight-forward and well described in the openvpn.net howto.

You could as well use ssl-admin port / package which makes generating the necessary certificate and key files and exporting them for OpenVPN even easier.

When you have generated the necessary files, you need to copy them to the OpenVPN config directory - */usr/local/etc/openvpn*.

Now we need to create the server config file, which should be named *server.conf* and located in */usr/local/etc/openvpn/*:

```
up /usr/local/etc/openvpn/up.sh
down /usr/local/etc/openvpn/down.sh

server-bridge 10.0.0.2 255.255.255.0 10.0.0.110 10.0.0.120
proto udp
port 22222
dev tap0
comp-lzo yes
keepalive 15 60
client-to-client
client-config-dir ccd

push "route 10.0.0.0 255.255.255.0"
push "dhcp-option DNS xxx.xxx.xxx.xxx"
push "dhcp-option DNS yyy.yyy.yyy.yyy"
push "dhcp-option WINS zzz.zzz.zzz.zzz"
push "redirect-gateway def1"

ca /usr/local/etc/openvpn/ca.crt
dh /usr/local/etc/openvpn/dh1024.pem
cert /usr/local/etc/openvpn/server.crt
key /usr/local/etc/openvpn/server.key

log /var/log/openvpn.log
status-version 2
status status.log
verb 3
mute 20
```

Here is an explanation of parameters which I consider important:

- up - location of the shell script which is executed when the server is started
- down - location of the shell script which is executed when the server is stopped
- server-bridge <local IP address> <vpn client address pool start> <vpn client address pool end>
    - local IP address - local IP address of the server
    - vpn client address pool start - the first IP address which is assigned to VPN clients
    - vpn client address pool end - the last IP address which is assigned to VPN clients
- proto - protocol, in our case UDP (TCP is also an option)
- port – listening port for the server
- dev - which device to use (tap - bridged mode, tun - routing mode)
- comp-lzo - enable / disable LZO compression (LZO compression can save some bandwidth if you are transferring mostly text). If you don't want to force the compression, "adaptive" is also an option.
- client-to-client - if enabled, OpenVPN will internally route client-to-client traffic rather than push all the client-originating traffic to the TAP interface and client will see the other clients which are currently connected.
- client-config-dir <directory> - directory where the client specific config files are located (config file should have the same name as the client's X509 common name - the parameter which you specify when creating the client key file). This is useful if you want to set client-specific configuration or push some specific client rules like routes or DNS servers.
- push - some special options which are pushed to the client

        - push "route 10.0.0.0 255.255.255.0" – this rule must be pushed to the client, so everything in this subnet is accessible throught the VPN
        - push "dhcp-option DNS xxx.xxx.xxx.xxx" - IP address of the first DNS server
        - push "dhcp-option DNS yyy.yyy.yyy.yyy" - IP address of the second DNS server
        - push "dhcp-option WINS zzz.zzz.zzz.zzz" - IP address of the WINS server
        - push "redirect-gateway def1" - if this rule is pushed to the client, it will cause all the outgoing IP traffic to be routed over the VPN (by default, only the traffic from and to OpenVPN server site will pass over the VPN - web browsing and other traffic is not routed over VPN and uses the client's default connection)
    - ca - location of the master certificate authority certificate
    - dh - Diffie Hellman parameters file location
    - cert - server certificate file location
    - key - server key file location
    - log - location of the log file
    - status <filename> - file which keeps a log of OpenVPN status
    - verb <number> - log verbosity (if you encounter problems, bump the value up to 9. For normal operation 3 should give you just enough information and not pollute the log file too much)
    - mute <number> - this limits the repetitive logging of similar messages

Note: VPN client address range should be outside of the local DHCP server IP address range.

7. Step 3. Up and down scripts

Only thing left to do server-side wise is to create the up and down scripts which are executed when the server is started or stopped.

/usr/local/etc/openvpn/up.sh:

```
#!/bin/sh
/sbin/ifconfig bridge0 create
/sbin/ifconfig bridge0 addm bge0 addm $dev up
/sbin/ifconfig $dev up
```

/usr/local/etc/openvpn/down.sh:

```
#!/bin/sh
/sbin/ifconfig bridge0 deletem $dev
/sbin/ifconfig bridge0 destroy
/sbin/ifconfig $dev destroy
```

Note: $dev is the first command line argument which is passed to this script by OpenSVN server - interface name (in our case, this will be **tap0**).

8. Step 4: Example client config file

Here is an example client config file, which you could use with this server configuration:

```
script-security 3

client
tls-client

dev tap
proto udp

remote <YOUR SERVER IP> 55555
resolv-retry infinite

nobind
persist-key
persist-tun

ca C:\\private\\ca.crt
cert C:\\private\client1-laptop.crt
key C:\\private\\client1-laptop.key

ns-cert-type server
cipher BF-CBC

comp-lzo
verb 3
```

9. Step 6. Auto starting the OpenVPN server on boot:

If you want OpenVPN server to be automatically started when the server is booted, you need to put the following lines in the */etc/rc.conf* file:

```
openvpn_enable="YES"
openvpn_configfile="/usr/local/etc/openvpn/server.conf"
openvpn_flags="--script-security 3"
openvpn_if="tap"
```

10. Step 7: Configuring the firewall (ipfw):

If you don't use ipfw you can skip this step, but if you don't use firewall at all, you should install one.

For the OpenVPN server to work, we need to configure our firewall so it allows traffic to pass between the bridged interfaces and open the UDP port for remote connections.

```
# ipfw -q add allow all from any to any via bge0
# ipfw -q add allow all from any to any via bridge0
# ipfw -q add allow all from any to any via tap0

# ipfw -q add allow udp from any to 10.0.0.2 22222 setup
```

11. Step 8: Starting and testing the server

If you have successfully completed all the steps above you should be able to start the server:

```
# /usr/local/etc/rc.d/openvpn start
```

If you didn't receive any error message, server should be up and running, quietly waiting (depends on your log verbosity level) for the clients to connect to it.

You can check if it's running by executing the following command */usr/local/etc/rc.d/openvpn status* or optionally check the log file located at */var/log/openvpn.log*.

On the client side, you can verify that all the traffic is being routed over the VPN by checking the routes (print route / netstat -r) or using the trace route command (for example: tracert google.com / traceroute google.com).

*Note: If you are using OpenVPN version >= **2.1-rc20** you need to provide "**--script-security 3**" flag when starting the server, else the server won't start (this is not needed, if you added the necessary parameters to the /etc/rc.conf file and you are using the /usr/local/etc/rc.d/openvpn script to start the server).*

Usefull links and resources:

- OpenVPN Howto
- Ethernet Bridging
- OpenVPN 2.1 manual
- GRC's OpenVPN HowTo Guide
- FreeBSD OpenVPN Server/Routed
- Ipfw Howto
- FreeBSD Handbook - Gateways and Routes