Running a Single Application Through a VPN

24 February 2018

In its simplest and most common usage, VPNs route all internet traffic from your device. However, it's possible to route only traffic that is going to a particular IP address using OS/VPN-app level features. Furthermore, it's also possible to route all traffic from a particular application. This last possibility is what we will focus on.

Linux has the concept of namespaces for processes, users, storage devices, and other things. One of these namespaces is for networks. These can be used to isolate applications within a new networking stack of devices and routing tables.

In this article we will create simple way to run only particular applications through a VPN. It is based on VPN in a Nutshell by Thomas Gläßle. This was the best resource that I could find on this topic, and it is definitely worth a read as it goes into more detail than I will.

Installing OpenVPN

First, let's install and make sure OpenVPN is working. We will have to obtain the client config files from our VPN provider (these can vary but they are generally three files - a `.crt`, `.pem`, and a `.ovpn`/`.conf` file).

```
$ sudo pacman -S --needed openvpn
# Copy the OpenVPN client config files
$ sudo cp ca.rsa.4096.crt /etc/openvpn/client
$ sudo cp crl.rsa.4096.pem /etc/openvpn/client
$ sudo cp CONFIG.ovpn /etc/openvpn/client
```

Note: You may need to restart your system if you upgraded your kernel in order to get OpenVPN to run.

☰

Note: If your VPN provider requires a username/password, skip the next section to configure authentication.

Start OpenVPN for testing:

```
$ sudo openvpn --cd /etc/openvpn/client/ --config CONFIG.ovpn
```

Test that we have a different IP address now:

```
$ curl ifconfig.co
```

Configuring Authentication

We will configure OpenVPN so that it can automatically connect using a username and password from a file. The file will be secured under the `root` user.

The OpenVPN configuration will be done in a separate config file, to allow us to add/modify settings separately to the client config provided by the VPN provider.

1. `sudoedit /etc/openvpn/client/override.conf`, and add the line:

   ```
   auth-user-pass auth.txt
   ```

2. `sudoedit /etc/openvpn/client/auth.txt`, and add two lines:

   ```
   <YOUR USERNAME HERE>
   <YOUR PASSWORD HERE>
   ```

Start OpenVPN for testing:

```
$ sudo openvpn --cd /etc/openvpn/client/ --config CONFIG.ovpn --config override.conf
```

Test that we have a different IP address now:

```
$ curl ifconfig.co
```

Running OpenVPN through netns

Thomas Gläßle provides the necessary scripts which more-or-less work out of the box!

1. `sudoedit /etc/openvpn/client/override.conf` with:

   ```
   # Configure interface later:
   ifconfig-noexec

   # Don't route all traffic on this machine through VPN:
   route-noexec

   # Enable up-script
   script-security 2
   up   move-to-netns.sh
   down move-to-netns.sh
   ```

2. Add the `/etc/openvpn/client/move-to-netns.sh` script:

☰

```
#!/usr/bin/bash

up() {
    # create network namespace
    ip netns add vpn || true

    # bring up loop device
    ip netns exec vpn ip link set dev lo up

    # move VPN tunnel to netns
    ip link set dev "$1" up netns vpn mtu "$2"

    # configure tunnel in netns
    ip netns exec vpn ip addr add dev "$1" \
            "$4/${ifconfig_netmask:-30}" \
            ${ifconfig_broadcast:+broadcast "$ifconfig_broadcast"}
    if [ -n "$ifconfig_ipv6_local" ]; then
            ip netns exec vpn ip addr add dev "$1" \
                    "$ifconfig_ipv6_local"/112
    fi

    # set route in netns
    ip netns exec vpn ip route add default via "$route_vpn_gateway"
}

down() { true; }

"$script_type" "$@"

# Update DNS servers in netns - normal version:
#if [ -x /etc/openvpn/update-resolv-conf ]; then
#     ip netns exec vpn /etc/openvpn/update-resolv-conf "$@"
#fi
# systemd-resolvd version:
#if [ -x /etc/openvpn/update-systemd-resolved ]; then
#     ip netns exec vpn /etc/openvpn/update-systemd-resolved "$@"
#fi
```

Start OpenVPN for testing:

```
$ sudo openvpn --cd /etc/openvpn/client/ --config CONFIG.ovpn --config override.conf
```

Test we can connect to the internet in the network namespace:

```
$ sudo ip netns exec vpn sudo -u $(whoami) -- ping 8.8.8.8
```

When OpenVPN is not running you should not be able to obtain a connection.

Configuring DNS

The script above has commented-out lines at the end which handle the configuration to resolve DNS queries.

Check whether you are using `systemd-resolved` with `sudo systemctl status systemd-resolved`.

1. If you are not obtain the appropriate scripts and uncomment the first three lines.
2. If you are, uncomment the last three lines.

Start OpenVPN for testing:

```
$ sudo openvpn --cd /etc/openvpn/client/ --config CONFIG.ovpn --config override.conf
```

Test domain resolution with:

```
$ sudo ip netns exec vpn sudo -u $(whoami) -- curl ifconfig.co
```

Verify DNS queries are routed through the VPN:

```
$ sudo ip netns exec vpn sudo -u $(whoami) -- drill +short whoami.akamai.net
```

This should match the IP address through the VPN - otherwise you have a DNS leak!

Running Applications Through the VPN

We will create a script that allows applications to be started within the network namespace.

1. `sudoedit /usr/local/bin/vpnbox` with:

   ```
   #!/usr/bin/bash

   # check if there is a default route in the netns going over tun0:
   # NOTE: 'tun0' may not be the correct interface name
   vpn_online() {
       sudo ip netns exec vpn sudo -u $(whoami) -- ip route \
           | grep default | grep tun0
   }

   if ! vpn_online; then
       # Execute openvpn in daemon mode:
       sudo /bin/openvpn --cd /etc/openvpn/client/ --config CONFIG.ovpn --config overri

       # Wait for completion. Otherwise routes/DNS information may not be
       # setup when the main program starts:
       echo "Waiting for route."
       while ! vpn_online; do
           sleep 0.1
       done
   fi

   # Execute the actual command as before:
   sudo ip netns exec vpn sudo -u $(whoami) -- "$@"
   ```

2. `sudo chmod 755 /usr/local/bin/vpnbox`.

3. Let normal users run the command `sudo visudo -f /etc/sudoers.d/vpnbox`:

   ```
   # Allow <USER> to use vpnbox command without a password
   <USER> ALL=(ALL:ALL) NOPASSWD: /usr/bin/ip netns exec vpn sudo -u <USER> -- *
   <USER> ALL=(ALL:ALL) NOPASSWD: /usr/bin/openvpn --cd /etc/openvpn/client/ --config C
   ```

Test that everything works:

```
$ vpnbox curl ifconfig.co
```

We no longer need to start OpenVPN manually. It will automatically start in the background on-demand.

Some applications may require environment variables, in which case, the following *additional* line in sudoers *may* be helpful. I am not sure how dangerous it is, but if you are the only user it doesn't make much difference.

```
<USER> ALL=(ALL:ALL) SETENV:NOPASSWD: /usr/bin/ip netns exec vpn sudo -E -u <USER> --
```

Conclusion

We have:

1. Install OpenVPN connecting to a provider's VPN.
2. Sandboxed OpenVPN inside a separate network namespace.
3. Configured DNS resolution, and checked there is no DNS leak.
4. Added a dead simple command to run an application through the VPN: `vpnbox`.

Aside: Firefox

1. First create the profile `vpn` in Firefox `about:profiles`.

2. Then create a script to launch a separate instance of Firefox that uses the VPN:

```
#!/bin/sh
# Start Firefox in a VPN

vpnbox firefox -P vpn -no-remote -new-instance -private-window "${1-https://duckduckg
```

Aside: Application Servers

If the app is a server listening on `localhost`, then you can forward some ports with `socat` on the host machine (i.e. not inside the network namespace):

```
$ sudo socat tcp-listen:8080,fork,reuseaddr exec:'ip netns exec vpn socat STDIO tcp-c
```

You'll probably just want to use a proper proxy though *somehow*.

Read and write comments on the discussion.

☰