


What do you think? Discuss, post comments, or ask questions at the end of this article [\[More about me\]](#) 

# My VPN Linux killswitch: securely disable all traffic except through VPN tunnel

Created by Jay Faria, last modified on Jul 29, 2020

For the security and privacy conscious, a trusted VPN is a great tool that provides encrypted communications and assured privacy while accessing unsecured networks (the internet in particular).

I use several VPNs, I have my own [private OpenVPN server](#) which I run on my main (personal) server, and also use a well known [VPN service](#). I do use both (not at the same time) for various purposes and for different use cases which won't be discussed here.

One of the important reasons for using a VPN is to obfuscate (hide) your ip address. It's important to note that even the best VPN setups are subject to occasional drops and disconnects. When this occurs, your system will likely fallback to using its standard network interfaces (which it would use without a VPN) and incidentally expose your actual ip address. In sensitive environments (or countries) this can potentially lead to some very undesirable situations (including being able to find your actual location).

Enter a **Kill Switch**. A kill switch is basically some system which ensures your ip address is not leaked, and ideally blocks all traffic that is not through the VPN. Hence if your VPN connection drops, your system doesn't fall back to standard network interfaces.

Many VPN services provide built in Kill Switch functionality. However, I generally prefer to roll my own, with an implementation that I trust (because I did it).

## Guide

- [Requirements and setup](#)
- [vpnkillschitch scripts](#)
- [Usage](#)
- [Examples](#)
- [Integrating killswitch into OpenVPN client ovpn](#)

## Requirements and setup

This implementation doesn't require much. Basically you need

- bash
- sudo access
- ufw (uncomplicated firewall)

All Linux distros should have these (mostly likely by default).

Make sure UFW is setup and installed. For distros that use [systemd](#), you would simply do

```
sudo systemctl enable ufw
sudo ufw enable
```

## vpnkillschitch scripts

My killswitch consists of a single bash script. This script supports both starting the killswitch and stopping the killswitch

(which resets the firewall), plus another option to open an outgoing route to a specific [CIDR](#):

**vpnkillswitch** (sourced from <https://gitlab.jaytaala.com/j.taala/killswitch.git>)



```
1  #!/bin/bash
2
3  # process arguments
4  while getopts ":edt:" opt; do
5      case ${opt} in
6          e)
7              # ENABLE KILLSWITCH
8              # Default policies
9              sudo /usr/bin/ufw default deny incoming
10             sudo /usr/bin/ufw default deny outgoing
11
12             # Openvpn interface (adjust interface accordingly to your confi
13             sudo /usr/bin/ufw allow out on tun0
14
15             # Openvpn (adjust port accordingly to your vpn setup)
16             sudo /usr/bin/ufw allow out to any port 1194
17             ;;
18         d)
19             # DISABLE KILLSWITCH
20             sudo /usr/bin/ufw --force reset
21             sudo /usr/bin/ufw enable
22
23             # delete backUP rules from reset
24             sudo /usr/bin/rm /etc/ufw/*.rules.*
25
26             # reset to defaults and enable
27             sudo /usr/bin/ufw default deny incoming
28             sudo /usr/bin/ufw default allow outgoing
29             ;;
30         t)
31             # ADD OUTGOING RULE
32             echo "allow outgoing traffic to $OPTARG"
33             sudo /usr/bin/ufw allow out to $OPTARG
34             ;;
35     esac
36 done
37
38 if (( $OPTIND == 1 )); then
39     echo " "
40     echo -e "Please provide at least one of the following options:\n    -e"
41     echo "        enable killswitch"
42     echo "    -d"
43     echo "        disable killswitch"
44     echo "    -t [CIDR]"
```

```
45     echo -e "          open outgoing ufw rule to a specific CIDR (ip address
46     fi
```

You'll need to make the script executable and ideally put it in some folder on your path. To make them executable, cd into whatever folder contains the scripts and:

```
chmod +x vpnkillswitch
```

## Usage

Basic usage is to simply call the script with a valid option. The supported options are:

```
-e
    enable killswitch (turn it on)

-d
    disable killswitch (turn it off)

-t [CIDR]
    open outgoing (ufw) route to a specific CIDR (ip address or range)
```

Note that the `-t` option will likely be needed if you also run a web server (for example) on the same server that is running your VPN and want to access web server resources.

## Examples

Turn on (enable) the killswitch

```
vpnkillswitch -e
```

Turn on (enable) the killswitch and allow outgoing route to to some server, e.g. 93.184.216.34:

```
vpnkillswitch -et 93.184.216.34
```

Turn off (disable) the killswitch

```
vpnkillswitch -d
```


## Integrating killswitch into OpenVPN client ovpn

If you (or your VPN provider) uses OpenVPN you can integrate the killswitch script into your client `.ovpn` file so that when you connect the killswitch script is automatically run. This comes in handy especially if you run your own OpenVPN server (which I do) - and also use that server for other things (such as a web server etc.). In these types of use cases you can also open an outgoing route to your server (for example to access server web resources while also being connected to said server via OpenVPN).


You can simply add a `up` or `route-up` directive with the `vpnkillswitch` command. For example, in my

client .ovpn I simply added:

```
script-security 2
route-up "/home/user/.local/bin/vpnkillswitch -et <IP-ADDRESS-OF-OPENVPN-SERVER>"
```

 Replace <IP-ADDRESS-OF-OPENVPN-SERVER> with the IP address of the outgoing route you want to allow.

I use **route-up** here as OpenVPN will execute the script after routes to the VPN server has been added. This is after any **up** commands have been run (which you might use in your client config to set [DNS addresses](#) etc.).






 Don't forget that you'll need to disable the kill switch after disconnecting with from OpenVPN. You can do this by executing:

```
vpnkillswitch -d
```

## References

1. <https://thetinh.com/tutorials/misc/linux-vpn-drop-protection-firewall.html>

## Related articles

-  [Continuous connection to Cisco AnyConnect VPN with linux Openconnect](#)
-  [My VPN Linux killswitch: securely disable all traffic except through VPN tunnel](#)
-  [Using ipset to block IP addresses - firewall](#)
-  [Make ip-tables \(firewall\) rules persistent](#)
-  [Continuous and automated VPN connection with FortiClient \(CLI only\) using bash and expect scripting](#)

[vpn](#) [ufw](#) [firewall](#) [kb-how-to-article](#) [killswitch](#)

## One Comment

Type Comment Here (at least 3 chars)

Name (optional)

E-mail (optional)

Website (optional)

Preview

Submit



Anonymous • 3 months ago

I found I needed to add the following to the -d DISABLE portion of the script:

## **sudo systemctl restart firewalld**

Otherwise my Samba server remains unreachable.

^ | v [Reply](#)