mozvpn Documentation

Release 0.1.0

Ralph Heinkel

May 26, 2021

CONTENTS:

MozVPN is an alternative command line interface (CLI) and and graphical user interface (GUI) client for MozillaVPN.

CHAPTER

ONE

MOZVPN

MozVPN is an alternative CLI and GUI client for MozillaVPN.

When MozillaVPN showed up in May 2021 Mozilla published clients for Ubuntu Linux only, which didn't run on my OpenSuse machine. This was the motivation to implement this alternative client.

1.1 Short Usage

The following instructions assume that everything is installed and setup (incl. `wireguard` and `wireguard-tools`) and you have a subscription for MozillaVPN. For details see the complete documentation on https://mozvpn.readthedocs.io.

1.1.1 Graphical User Interface (GUI)

To start the GUI for mozvpn just run:

```
$ mozvpn gui
(or alternatively)
$ xmozvpn
```

A window should open and allow you to select the desired VPN server endpoint from a choice of cities in various countries. Then just click the `connect` button, and you should have a running VPN.

1.1.2 Command Line Interface (CLI)

The command line interface can be used to connect or disconnect to MozillaVPN from a linux or windows shell. Also the current status of the connection can be obtained.

Examples:

```
$ mozvpn status
Not connected
$ mozpvn up de4-wireguard   # must match files in /etc/wireguard/*.conf
Connected to: de4-wireguard
$ mozvpn status
Connected to: de4-wireguard
$ mozpvn down de4-wireguard
```

(continues on next page)

(continued from previous page)

```
Disconnected from: de4-wireguard
$ mozvpn status
Not connected
```

1.2 License

- Free software: GNU General Public License v3
- Documentation: https://mozvpn.readthedocs.io. (to be done).

1.3 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER

TWO

INSTALLATION

Note: This installation was tested on Linux (opensuse) only so far. In principle it should work on most Linux variants without modifications. Also there is a good chance that MacOS could be supported.
 If you find problems please file an issue at the github repo, or even better submit a pull request with a decent solution to the problem.

Warning: MozVPN should per se also work on Windows. The installation for required packages `MozWire`, `WireGuard`, and `MozVPN` should work just fine under Windows.
 However the setup of `MozVPN`, i.e. the point where `mozpvn setup` is run (see below) currently only supports Linux-like systems.

If you run `mozpvn setup --dry-run` you can see what commands would be run on Linux. If you could provide something similar for Windows this would be awesome.

2.1 Installing helper tools MozWire and wireguard

MozVPN is based on two other applications that need to be installed first, namely MozWire and wireguard.

2.1.1 MozWire

This tool has basically to be run once only. It connects to your MozillaVPN account (subscription) and downloads a set of approx. 400 configuration files, each one providing connection details and credentials for a VPN connection to a server somewhere in the world. (MozVPN will do the actual download for you, so no worry here).

Installing MozWire

MozWire can be downloaded or downloaded as explained on https://github.com/NilsIrl/MozWire. There you'll find binaries for Linux, MacOS, and Windows. Download the binary for your operating system and install it in a place where it can be found when being called from the command line, e.g. `/usr/local/bin` or `~/bin` in case of Linux or MacOS. Just make sure that this path is contained in your shell's PATH variable. Do the corresonding setup for Windows if that is your OS.

If nothing fits the MozWire homepage also explains how to compile it from sources. This is basically a one line command and worked like a charm in my case. The only tool required for compiling is `cargo` which is usually installed on Linux machines with `sudo zypper install cargo` (OpenSuse) or `sudo apt install cargo` (Debian/Ubuntu). Further explanations about installing cargo can be found at https://doc.rust-lang.org/cargo/getting-started/installation.html.

Once installed you should find the `mozwire` executable in your PATH.

2.1.2 WireGuard

Because MozillaVPN is based on the WireGuard protocol the corresponding binary and helper tools have to be installed as well.

WireGuard uses the configuration files provided by MozWire and uses those to make the actual connection to the VPN servers.

Installing WireGuard

The page https://www.wireguard.com/install/ describes this for more than 20 operating systems. For Linuxes it is usually a command like `sudo zypper install wireguard-tools` (OpenSuse) or `sudo apt install wireguard` (Debian/Ubuntu).

Once installed you should find the two executables `wg` and `wg-quick` in your PATH.

2.2 Installing a stable release of MozVPN

After having installed MozWire and WireGuard we finally can install MozVPN itself.

To install MozVPN, run this command in your terminal:

```
$ pip install mozvpn
```

This is the preferred method to install mozvpn, as it will always install the most recent stable release. MozVPN requires Python3.6 or newer.

You can either install it in a local virtual environment somewhere in your home directory, or system-wide (on a Linux like OS you need to be root or use sudo, e.g. `sudo pip install mozvpn`).

If you don't have pip installed, this Python installation guide can guide you through the process.

2.3 Installing MozVPN from sources

The sources for MozVPN can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/ralhei/mozvpn
```

Or download the tarball:

```
$ curl -OJL https://github.com/ralhei/mozvpn/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

Congratulation! Now the software part is done. The next step will be to setup the configuration, as explained in the next section *Setup*.

CHAPTER

THREE

SETUP

3.1 Get a subscription for MozillaVPN

Now it is time to get a subscription for MozillaVPN, if you don't have one already. Without a subscription you won't be able to setup VPN connections via MozillaVPN. Details can be found at https://www.mozilla.org/en-US/products/vpn/.

The following step will require you to enter your username and password for MozillaVPN, so make sure you saved them somewhere (e.g. in a password manager).

3.2 Setting up MozillaVPN access Linux

Luckily MozVPN has built in a setup feature which does the following steps for you. This setup has to be run only once. It might be required to be run later again, e.g. if your MozillaVPN subscription has expired and you have renewed it. But for now running this once is sufficient.

The command to run is:

```
mozvpn setup
```

or:

```
mozvpn setup --verbose
```

if you'd like to see the details.

You are required to enter two sets of credentials in this process (one for MozillaVPN and one for *root*), so please read the following descriptions (if you are interested in the details) or just follow the what is happening in your browser and in the shell where you started `mozvpn setup`.

The steps performed by `mozpvn setup` are:

1. It runs MozWire, which itself opens a page in your Firefox (or possibly other browser) asking for your MozillaVPN credentials. **Type them into the MozillaVPN login page if your are not already logged in.**

   This stage of the setup will download approx. 400 WireGuard configuration and credential files into a local directory on your computer.

2. Because these WireGuard configuration files do not contain detailed information about their server geo-locations a little script is run to determine those locations for each WireGuard configuration file. This geo-location information is stored in one single `locations.csv` file for you. No interaction required here from your side.

3. The WireGuard configuration files and the `locations.csv` file have to be installed in `/etc/wireguard` on Linux-like or MacOS systems. This requires root permissions.

4. To allow MozVPN to run as normal user `sudo`-privileges have to be setup for the WireGuard tools. This is achieved by creating a new Linux group `mozvpn` and adding those users to it who should be allowed to run MozVPN on your computer. Note that - after this is done - all affected users have to logout and login again to activate this new group membership for them. You will be told later when to do this.
5. A new `sudo`-file will be created in `/etc/sudoers.d/mozvpn` giving all members of group `mozvpn` the privileges to run `wg-quick` with root privileges. This is the tool that actually sets up and tears down the VPN connection under the hood.

You can check the commands for all five steps beforehand by running `mozpvn setup --dry_run`. This will print the commands to the shell only without actually executing them. If you have security concerns you can then run those commands yourself without going through `mozpvn setup`. This also allows you do adapt them if you prefer or require an alternative setup on your computer.

3.3 Testing

Once everything is in place you should be able to activate and tear down a connection to one of Mozilla's VPN servers through the command line, simply by running the following commands from the shell:

```
$ mozvpn status
Not connected
$ mozpvn up de4-wireguard   # must match a file in /etc/wireguard/*.conf
Connected to: de4-wireguard
$ mozvpn status
Connected to: de4-wireguard
$ mozpvn down de4-wireguard
Disconnected from: de4-wireguard
$ mozvpn status
Not connected
```

CHAPTER

FOUR

USAGE

4.1 Using the graphical User Interface (GUI)

To start the GUI for mozvpn just run:

```
$ mozvpn gui
(or alternatively)
$ xmozvpn
```

A window should open and allow you to select the desired VPN server endpoint from a choice of cities in various countries. Then just click the `connect` button, and you should have a running VPN.

4.2 Command Line Interface (CLI)

The command line interface can be used to connect or disconnect to MozillaVPN from a linux or windows shell. Also the current status of the connection can be obtained.

Examples:

```
$ mozvpn status
Not connected
$ mozpvn up de4-wireguard   # must match files in /etc/wireguard/*.conf
Connected to: de4-wireguard
$ mozvpn status
Connected to: de4-wireguard
$ mozpvn down de4-wireguard
Disconnected from: de4-wireguard
$ mozvpn status
Not connected
```

CHAPTER

FIVE

MOZVPN

5.1 mozvpn package

5.1.1 Submodules

5.1.2 mozvpn.cli module

5.1.3 mozvpn.mozvpn module

Main module.

mozvpn.mozvpn.**determine_ip_location**(*ip: str*) → Dict
Determine location of IP address.

Parameters **ip** – IP address

Returns dict containing (among others) fields country, region, city

mozvpn.mozvpn.**find_vpn_server_locations**(*wg_config_files: List[str]*)
Find geographic locations of wireguard VPN server endpoints.

Parameters **wg_config_files** – list of wireguard config files/directories

Returns list of dictionaries containing configuration/location data.

5.1.4 mozvpn.mozvpn_gui module

5.1.5 mozvpn.wireguard module

Functions for interacting with wireguard command line tools.

exception mozvpn.wireguard.**CommandError**(*msg*, *cmd*)
Bases: Exception

property msg

exception mozvpn.wireguard.**ControlledExit**
Bases: Exception

Raise when error handling is finished and program can gracefully exit.

exception mozvpn.wireguard.**WireguardError**
Bases: Exception

Raised when external wireguard or wg-quick command reported an error.

mozvpn.wireguard.**check_wireguard_commands**() → dict

Check absolute path to 'wg' and 'wg-quick' commands if they are installed and executable.

Returns '/usr/bin/wg', 'wg-quick': '/usr/bin/wg-quick'}

Return type {'wg'

Raises **RuntimeError if path to wireguard command could not be determined.** –

mozvpn.wireguard.**connect**(*conf_or_if: str*)

Establish connection to VPN server via wg-quick command.

Parameters **conf_or_if** – Either - Name of wireguard conf file, e.g. "/path/to/us122-wireguard.conf" - Name of wireguard interface, e.g. "us122-wireguard"

In this case the corresponding configuration file has to exist in the /etc/wireguard/ directory.

mozvpn.wireguard.**disconnect**(*conf_or_if: str*)

Shut down connection to VPN server via wg-quick command.

Parameters **conf_or_if** – Either - Path to wireguard conf file, e.g. "/path/to/us122-wireguard.conf" - Name of wireguard interface, e.g. "us122-wireguard"

In this case the corresponding configuration file has to exist at /etc/wireguard/INTERFACE.conf.

mozvpn.wireguard.**interface**() → str

Return interface of VPN connection, if available.

Returns Name of connected interface (e.g. 'de12-wireguard), otherwise None, if not connected.

mozvpn.wireguard.**ipinfo**()

Obtain externally visible IP information from https://ipinfo.io

Returns: JSON like

{ "ip": "185.213.155.160", "city": "Frankfurt am Main", "region": "Hesse", "country": "DE", "loc": "50.1155,8.6842", "org": "AS39351 31173 Services AB", "postal": "60311", "timezone": "Europe/Berlin", "readme": "https://ipinfo.io/missingauth"

}

mozvpn.wireguard.**mullvad_info**()

Obtain externally visible IP information from https://am.i.mullvad.net/json

Note: Mullvad is the provider behind MozillaVPN.

Returns: JSON like

{ "ip": "212.14.256.33", "country": "Germany", "city": "Stadt", "longitude": 8.2, "latitude": 44.4, "mullvad_exit_ip": false, "blacklisted": { ... }, "organization": "Telecom"

}

mozvpn.wireguard.**run_command**(*cmd: str*, *shell: bool = False*, *verbose: bool = False*, *dry_run: bool = False*) → str

Run external command, and collect results or errors.

Parameters

- **cmd** – The command to be executed

- **shell** – if True run command via shell.
- **verbose** – if True print command to stdout.
- **dry_run** – if True then the commands will only be written to stdout only. and not executed.

Raises **CommandError in case of failling command execution.** –

mozvpn.wireguard.**setup_wireguard_configuration**(*user: str*, *verbose: bool*, *dry_run: bool*, *limit: int*)
Setup configurations needed to operate wireguard.

Parameters

- **user** – name of primary user who should be allowed to use MozVPN.
- **verbose** – if True print command to stdout.
- **dry_run** – if True then the commands will only be written to stdout only.
- **limit** – Limit the number of servers downloaded via mozwire and not executed.

mozvpn.wireguard.**status**(*ip: bool = False*) → str
Show status of VPN connection.

Parameters **ip** – if given add currenlty visible external IP address to connection status.

Returns String telling if VPN connection is up, and if so, which server is currently is used. The IP address will optionally be added. Examples: - 'Not connected' - 'Connected to de10-wireguard' - 'Connected to de10-wireguard, ip: 234.12.642.0'

5.1.6 Module contents

Top-level package for mozvpn.

CHAPTER

SIX

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

6.1 Types of Contributions

6.1.1 Report Bugs

Report bugs at https://github.com/ralhei/mozvpn/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

6.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

6.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

6.1.4 Write Documentation

mozvpn could always use more documentation, whether as part of the official mozvpn docs, in docstrings, or even on the web in blog posts, articles, and such.

6.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/ralhei/mozvpn/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

6.2 Get Started!

Ready to contribute? Here's how to set up *mozvpn* for local development.

1. Fork the *mozvpn* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/mozvpn.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv mozvpn
$ cd mozvpn/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 mozvpn tests
$ python setup.py test or pytest
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

6.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/ralhei/mozvpn/pull_requests and make sure that the tests pass for all supported Python versions.

6.4 Tips

To run a subset of tests:

```
$ pytest tests.test_mozvpn
```

6.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

SEVEN

CREDITS

7.1 Development Lead

- Ralph Heinkel <rh@ralph-heinkel.com>

7.2 Contributors

None yet. Why not be the first?

CHAPTER

EIGHT

HISTORY

8.1 0.2.0 (2021-05-24)

- Added ‘setup’ functionality to mozvpn
- Extended documentation, added installation docs

8.2 0.1.0 (2021-05-06)

- First commit on github.

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

INDEX