PCap Filter Documentation

Release __version__ = '0.1.7'

Nahuel Defossé

Nov 23, 2018

Contents:

CHAPTER 1

PCap Filter

Python package for packet filtering and manipulation using scapy

- Free software: MIT license
- Documentation: https://pcapfilter.readthedocs.io.

1.1 Features

- Define your filters in Python
- Pipe your output to Wireshark or TShark for visualization
- Manipulate the payloads using Python's 3 byte regxes
- Save results to pcap files
- Work in progress *live reload* of your filter file

1.2 Credits

This project relies on the power of Scapy for either filtering or payload modification. This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install PCap Filter, run this command in your terminal:

```
$ pip install pcapfilter
```

This is the preferred method to install PCap Filter, as it will always install the most recent stable release.

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

2.2 From sources

The sources for PCap Filter can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/D3f0/pcapfilter
```

Or download the tarball:

```
$ curl  -OL https://github.com/D3f0/pcapfilter/tarball/master
```

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

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

2.3 Docker Container

You can obtain this Python package as a slim docker container (30MB). To get the image run:

```
docker pull pcapfilter
```

CHAPTER 3

Usage

3.1 pcapfilter

Read packet capture data (pcap) stream from stdin, apply a function and write to stdout. Example (capture from INTERFACE and display in Wireshark): tcpdump -i INTERFACE -s0 -w - | pcapfilter -m myfiltermodule.py | wireshark -k -i -

```
pcapfilter [OPTIONS]
```

Options

-m, --module <module>
A python module name that contains a packet_filter(packet). You can create one with -w example.py

-s, --silent <silent>
Hide log messages from STDERR)

-o, --oldpcap
Use old pcap for input

-r, --reload
Reloads the module upon changes

-w, --create-template <create_template>
Creates an example file

-d, --docker-help
Shows help when running from docker

The basic usage is:

```
INPUT | pcapfilter [options] | OUTPUT
```

Where INPUT is a package capture provider, such as *tcpdump* and OUTPUT is some program able to conusme pcap from stdin.

For example, capturing packets from interface en0 (INPUT) and showing the results in wireshark (OUTPUT):

```
tcpdump -i en0  -s0 -w - | pcapfilter -vm myfilter.py | wireshark -k -i -
```

You can use ssh packet capture from your router and display it in Wireshark

```
ssh router "tcpdump -i eth1.2 -i br-lan -s0 -w - " | pcapfilter -vm main.py |␣
↪wireshark -k -i -
```

3.2 As a docker image

You need to provide a volume where your filter file is defined:

```
INPUT | docker run --rm -i -v $(pwd):/shared pcapfilter pcapfilter -vm /shared/main.
↪py | OUTPUT
```

3.3 Defining a filter

A filter is a python module (or file ending in .py) that has a function called *packet_filter*. This funcion receives an argument named *pkg* If *None* is returned, then the package is **discarded**.

If the packet is returned (modified or not) it will be sent to the OUTPUT.

CHAPTER 4

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/D3f0/pcapfilter/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.

4.1.2 Fix Bugs

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

4.1.3 Implement Features

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

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/D3f0/pcapfilter/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 :)

4.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/pcapfilter.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 pcapfilter
$ cd pcapfilter/
$ 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 pcapfilter tests
$ python setup.py test or py.test
$ 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.

4.3 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 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/D3f0/pcapfilter/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_pcapfilter
```

4.5 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:

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

Travis will then deploy to PyPI if tests pass.

CHAPTER 5

Credits

5.1 Development Lead

- Nahuel Defossé <nahuel.defosse+pip@gmail.com>

5.2 Contributors

- [Lucas Krmpotic](https://github.com/LucasKrmpotic)

CHAPTER 6

History

6.1 0.1.0 (2018-10-23)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Index

Symbols

P