rcsh Documentation

Release 0.2.3

Freark van der Bos

Jun 08, 2021

Contents

Contents:

rcsh

Non-interactive command whitelisted shell

This is a work-in-progress little script intended to be used as a shell for Linux user accounts which are allowed to run a limited set of commands over SSH non-interactively and nothing else. The commands which are allowed are based on a whitelist of exact command invocation strings and/or a list of regular expressions which they should match.

- Free software: BSD license

- Documentation: https://rcsh.readthedocs.io. (not much documentation yet, please stand by. . .)

1.1 Features

- Allow execution of commands based on an exact or regular expression whitelist

- Log invocation using syslog's LOG_AUTH facilities

1.2 Credits

This package was inspired by lshell and bdsh:

- https://github.com/ghantoos/lshell

- https://raymii.org/s/software/bdsh.html

- https://github.com/RaymiiOrg/boa-diminish-restricted-shell/

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

Installation

2.1 Stable release

To install rcsh, run this command in your terminal:

```
$ pip install rcsh
```

This is the preferred method to install rcsh, 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 rcsh can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/SafPlusPlus/rcsh
```

Or download the tarball:

```
$ curl  -OL https://github.com/SafPlusPlus/rcsh/tarball/master
```

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

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

Usage

After adding rcsh as the shell of a user account you can whitelist commands for that user by editing the following files:

- `/etc/rcsh.d/<username>.exact`
- `/etc/rcsh.d/<username>.regex`

3.1 Exact whitelist

Every line in `/etc/rcsh.d/<username>.exact` represents a command which this user is allowed to execute over ssh.

3.2 Regular expression whitelist

Every line in `/etc/rcsh.d/<username>.regex` must be a valid Python regular expression and must start with '^' and end with '$'. If the user tried to execute a command which matches one of these, it will be allowed.

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/SafPlusPlus/rcsh/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

rcsh could always use more documentation, whether as part of the official rcsh 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/SafPlusPlus/rcsh/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 *rcsh* for local development.

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

```
$ git clone git@github.com:your_name_here/rcsh.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 rcsh
$ cd rcsh/
$ 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 rcsh 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 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/SafPlusPlus/rcsh/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_rcsh
```

Indices and tables

- genindex
- modindex
- search