Saruman Documentation

Release 0.3.0

Tycho Tatitscheff

January 05, 2016

Contents

A simply logic, configuration based, distributable and reliable extended-firewall.

Saruman is a extended firewall, meaning:

- firewall
- dns
- dhcp
- intruision detection
- reverse proxy)

build by a former Iresam. It targets I-Resam need's first but should be enough flexible to be used elsewhere.

saruman takes care of the boring bits for you.

Here's an overview of the documentation we have for you. First the documentation on *using* saruman:

CHAPTER 1

Saruman

A simply logic, configuration based, distributable and reliable extended-firewall.

Saruman is a extended firewall (meaning firewall + dns + dhcp +intruision detection + reverse proxy) build by a former Iresam. It targets I-Resam need's first but should be enough flexible to be used elsewhere.

It still unstable and yet brings not that much. Try at your own risks.

1.1 Most important Urls

- The full documentation is at saruman.readthedocs.org
- We are on Pypi so we're only an `pip install saruman` away from installation on your computer.
- The code is at github.com/tychota/saruman.

And... we're automatically being tested by Scrutinizer !

1.2 Technologies used

- Saruman **does require** Python 3, and if possible the newest version (**Python3.5** for now)
- It **does require** an Celery broker : take **RabbitMQ**, it is good, fast and reliable.
- It **does only works** on a recent linux machine : it requires **nftables** and **iproute2** so a linux 4+ kernel would be a necessity.

1.3 Available commands

Saruman gives you three commands to manage the worker and one to run your firewall. Worker's commands must be run in root since they manage main parts of your system. Firewall's one doesn't need this. The commands are:

- **saruman workers enable**: start the celery workers on the machine.
- **saruman workers disable**: start the celery workers on the machine.
- **saruman workers reload**: restart the celery workers on the machine.

- **saruman firewall start**: start the firewall

1.4 AMQP json-rpc api

Still infant

CHAPTER 2

Options

2.1 Command line options

These command line options are supported by the release commands (`saruman workers enable`, `saruman workers enable`, `saruman workers enable`, `saruman firewall start`).

-v, --verbose Run in verbose mode, printing a bit more, mostly only interesting for debugging.

-h, --help Display help text

CHAPTER 3

Assumptions

Saruman originated at Iresam so there are some assumptions build-in that might or might not fit you but I'm pretty sure it'll probably fit :-)

- In our case, *saruman* is run on a VM cluster so we have different VM handling different stuff For instance one for the netfilter firewall and router, one for dhcp, one for reverse proxy, one for admin site, one for AMQP broker so you have to tag the tasks you create so *saruman* could know which VM has to handle what.

That's just the style we started with. Pretty clear and useful.

CHAPTER 4

Further reading

Mighty fine documementation, the stuff you're reading now. But some other suggestions, ideas and a different tone might help you improve your firewall. So here are some pointers to other material.

And documentation on *saruman* as a project; for instance for reporting bugs and fixing the code:

CHAPTER 5

Improving saruman: report bugs, fork on github or email us

Did you find a bug? Do you have an improvement? Do you have questions? We run *saruman* as a proper open source project on github at https://github.com/tychota/saruman, so you have three basic options:

- Report bugs or problems at https://github.com/tychota/saruman/issues ! And feature requests too ! Normally you'll get a quick reply within a day or so, depending on our relative timezones. If you don't get an answer within a few days, please send off a quick email to remind us.
- Or make a fork, fix the bug or add something and open a pull request.

  If you are going to fork saruman, take a look at Information for developers of saruman for setup and test running information.
- You can mail Tycho Tatitscheff if you want to ask a question, too. Or if you want to tell us about an idea you have.

CHAPTER 6

Credits

- Tycho Tatitscheff is the originator and main author.
- Zest software for their releases manager and also for inspiration (copy paste of most docs).

CHAPTER 7

Information for developers of saruman

7.1 Running tests

We like to use Virtual env to get a simple environment and to use pytest to test. When you are in the root folder of your *saruman* checkout, do this:

```
$ virtualenv ~/venv/saruman --python=`which python3.5`  # Or a different python version.
$ source ~/venv/firewall/bin/activate
$ python setup.py test
```

7.2 Python versions

The tests currently pass on python 3.4 and 3.5. Travis continuous integration tests 3.4 and 3.5 for us automatically.

7.3 Necessary programs

To run the firewall and test, you need to have an AMQP broker ! On ubuntu:

```
$ sudo apt-get install rabbitmq
```

CHAPTER 8

Complete documentation from the source code

8.1 saruman package

8.1.1 Subpackages

saruman.actions package

Submodules

saruman.actions.start module

saruman.app package

Submodules

saruman.app.queue module

saruman.conf package

Submodules

saruman.conf.celery module

saruman.helpers package

Submodules

saruman.helpers.check module

saruman.helpers.check.**get_celery_worker_status**()

saruman.helpers.config module

saruman.helpers.error_handling module

saruman.helpers.error_handling.**error_handling**(**args*, ***kwds*)

saruman.helpers.exceptions module

exception saruman.helpers.exceptions.**FirewallGenericError**

Bases: exceptions.Exception

saruman.helpers.logger module

saruman.tasks package

Subpackages

saruman.tasks.iproute2 package

Submodules

saruman.tasks.iproute2.interfaces module

saruman.tasks.kernel package

Submodules

saruman.tasks.kernel.modprobe module

tasks.kernel.modprobe Contains all the functionalities that help loading or unloading a kernel module

class saruman.tasks.kernel.modprobe.**Check**

Bases: celery.app.task.Task

Tache de vérification de l'activation d'un module dans le kernel

Vérifie si le module *module_name* est activé dans le kernel. Se réfère à une liste des modules autorisés (ainsi, l'utilisateur ne peut pas supprimer le module du filesystem par exemple). La tache tourne dans un context (:py:func:error_handling) qui gère les erreurs

acks_late = False

ignore_result = False

name = 'kernel.modules.check'

rate_limit = None

request_stack = <celery.utils.threads._LocalStack object>

run(*module_name*)

Parameters **module_name** (*str*) – le nom du module à checker

Returns oui si le module est activé, non sinon

Return type bool

send_error_emails = False

serializer = 'json'

store_errors_even_if_ignored = False

track_started = False

class saruman.tasks.kernel.modprobe.**Add**
Bases: celery.app.task.Task

Tache de vérification de l'activation d'un module dans le kernel

Vérifie si le module *module_name* est activé dans le kernel. Se réfère à une liste des modules autorisés (ainsi, l'utilisateur ne peut pas supprimer le module du filesystem par exemple). La tache tourne dans un context (:py:func:error_handling) qui gère les erreurs

acks_late = False

ignore_result = False

name = 'kernel.modules.add'

rate_limit = None

request_stack = <celery.utils.threads._LocalStack object>

run(*module_name*)

Parameters **module_name** (*str*) – le nom du module à checker

send_error_emails = False

serializer = 'json'

store_errors_even_if_ignored = False

track_started = False

class saruman.tasks.kernel.modprobe.**AddWithArgs**
Bases: celery.app.task.Task

Tache de vérification de l'activation d'un module dans le kernel

Vérifie si le module *module_name* est activé dans le kernel. Se réfère à une liste des modules autorisés (ainsi, l'utilisateur ne peut pas supprimer le module du filesystem par exemple). La tache tourne dans un context (:py:func:error_handling) qui gère les erreurs

acks_late = False

ignore_result = False

name = 'kernel.modules.addWithArgs'

rate_limit = None

request_stack = <celery.utils.threads._LocalStack object>

run(*module_name*, *module_args*)

Parameters

- **module_name** (*str*) – le nom du module à checker
- **module_args** (*dict*) – un dictionnaire d'arguments

send_error_emails = False

serializer = 'json'

store_errors_even_if_ignored = False

track_started = False

class saruman.tasks.kernel.modprobe.**Remove**
Bases: celery.app.task.Task

acks_late = False

ignore_result = False

name = 'kernel.modules.remove'

rate_limit = None

request_stack = <celery.utils.threads._LocalStack object>

run(*module_name*)

send_error_emails = False

serializer = 'json'

store_errors_even_if_ignored = False

track_started = False

saruman.tasks.misc package

Submodules

saruman.tasks.misc.hello_world module

CHAPTER 9

Entrypoints for saruman

Saruman use an unique cli-entrypoint, that use click to parse command line arguments

CHAPTER 10

Changelog for Saruman

10.1 0.3.0 (2016-01-03)

- big modifications of the documentation structure

10.2 0.2.3 (2016-01-03)

- fixing badges and coverage in testing

10.3 0.2.2 (2016-01-03)

- testing works

10.4 0.2.1 (2016-01-03)

- fixing a lot of nasty issues

10.5 0.2.0 (2016-01-03)

- adding sphinx documentation
- adding CI coverage
- fixing nasty unpack in modprobe.py

10.6 0.1.2 (2016-01-03)

- fixing some typo.

10.7 0.1.1 (2016-01-03)

- remove download urls as we use sdist

10.8 0.1.0 (2016-01-03)

- add zest.release to perform check on release an better automation
- add some yaml config files

S

A

C

E

F

G

I

N

R

S

T