OpenRelay Documentation

Release 0.3

Steven Richards, Roberto Rosario

November 28, 2011

CONTENTS

OpenRelay is an Open source, peer to peer based web hosting project. This is the initial Django based implementation of the protocols that make up the OpenRelay network.

Website http://peer.to/peer

Source https://github.com/Captainkrtek/OpenRelay

CHAPTER

ONE

CONTENTS

1.1 Installation

Requirements:

To get started, you will need to install `virtualenv`, `git`, and `pip`.

in Ubuntu/Debian:

```
$ sudo apt-get install python-virtualenv python-pip git-core
```

and in Fedora/RedHat:

```
$ sudo yum install git-core
$ sudo yum install python-virtualenv.noarch
$ sudo yum install pyton-pip.noarch
```

Initialize a `virtualenv` to deploy the project:

```
$ virtualenv --no-site-packages OpenRelay
```

Or clone the latest development version straight from github:

```
$ cd OpenRelay
$ git clone git://github.com/Captainkrtek/OpenRelay.git
```

To install the python dependencies using the production dependencies file included:

```
$ source bin/activate
$ cd OpenRelay
$ pip install -r requirements/production.txt
$ git submodule init
$ git submodule update
```

Populate the database with the project's schema doing:

```
$ ./manage.py syncdb
```

Launch the server daemon doing:

```
$ ./runserver.sh start
```

Open a browser and point it to http://127.0.0.1:8000

To stop the server daemon process do:

```
$ ./runserver.sh stop
```

1.2 Settings

1.2.1 Resources

RESOURCES_STORAGE_BACKEND
Default: `FileBasedStorage` class

As Django supports file storage abstraction, adminitrators of nodes could choose instead to use other storage means such as NAS, SANs, Cloud based (S3), FTP, Samba, etc.

RESOURCES_FILESTORAGE_LOCATION
Default: `resource_storage`

This setting relates exclusively to the FileBaseStorage class and detemines where in the physical disk are the node files going to be stored.

1.2.2 Core

CORE_KEYSERVERS
Default: `['peer.to']`

The list of key server that the node will query for public keys to verify resources. This setting option may be eliminated in the future when OpenRelay supports storing and replicating of public keys without using centralize key servers.

1.2.3 API

SERVER_IPADDRESS
Default: Computer's current IP address

TCP/IP port where the UI and API can be accessed.

SERVER_PORT
Default: `8000`

TCP/IP port where the UI and API can be accessed.

SERVER_HEARTBEAT_QUERY_INTERVAL
Default: `10`

Time interval in seconds to perform heartbeat checks on other OpenRelay nodes.

SERVER_INVENTORY_QUERY_INTERVAL
Default: `11`

Time interval in seconds to perform an inventory hash checks on other OpenRelay nodes.

SERVER_HEARTBEAT_FAILURE_THRESHOLD
Default: `30`

Amount of failed heartbeat attempts before removing a node for the active list.

1.3 Development

OpenRelay is under active development, and contributions are welcome.

1.3.1 Source Control

The **OpenRelay** source is managed with Git

The project is publicly accessible, hosted and can be cloned from **GitHub** using:

```
$ git clone git://github.com/Captainkrtek/OpenRelay.git
```

Git branch structure

OpenRelay follows the model layout by Vincent Driessen in his Successful Git Branching Model blog post. Git-flow is a great tool for managing the repository in this way.

`development` The "next release" branch, likely unstable.

`master` Current production release (0.3).

`feature/<feature-name>` Unfinished/ummerged feature.

Each release is tagged and available for download on the Downloads section of the **OpenRelay** repository on GitHub.

When submitting patches, please place your feature/change in its own branch prior to opening a pull request on GitHub. To familiarize yourself with the technical details of the project read the *internals* section.

1.3.2 Getting Involved

OpenRelay, as mentioned earlier, is under active development. If you're interested in getting involved, clone the repo and send us a pull request with your update. If you're interested in becoming a maintainer, email steve@peer.to with your qualifications.

1.3.3 Documentation

The documentation is written in reStructured Text format.

The documentation lives in the `docs` directory. In order to build it, you will first need to install Sphinx.

```
$ pip install sphinx
```

Then, to build an HTML version of the documentation, simply run the following from the **docs** directory:

```
$ make html
```

Your `docs/_build/html` directory will then contain an HTML version of the documentation, ready for publication on most web servers.

You can also do a recursive wget of http://peer.to/docs to obtain the latest documentation build of OpenRelay

```
$ wget -r http://peer.to/git/docs/_build/html/
```

Then navigate to the _build/html directory and open index.html in your browser to browse the documentation.

You can also generate the documentation in format other than HTML.

1.4 Changelog

1.4.1 Version 0.3

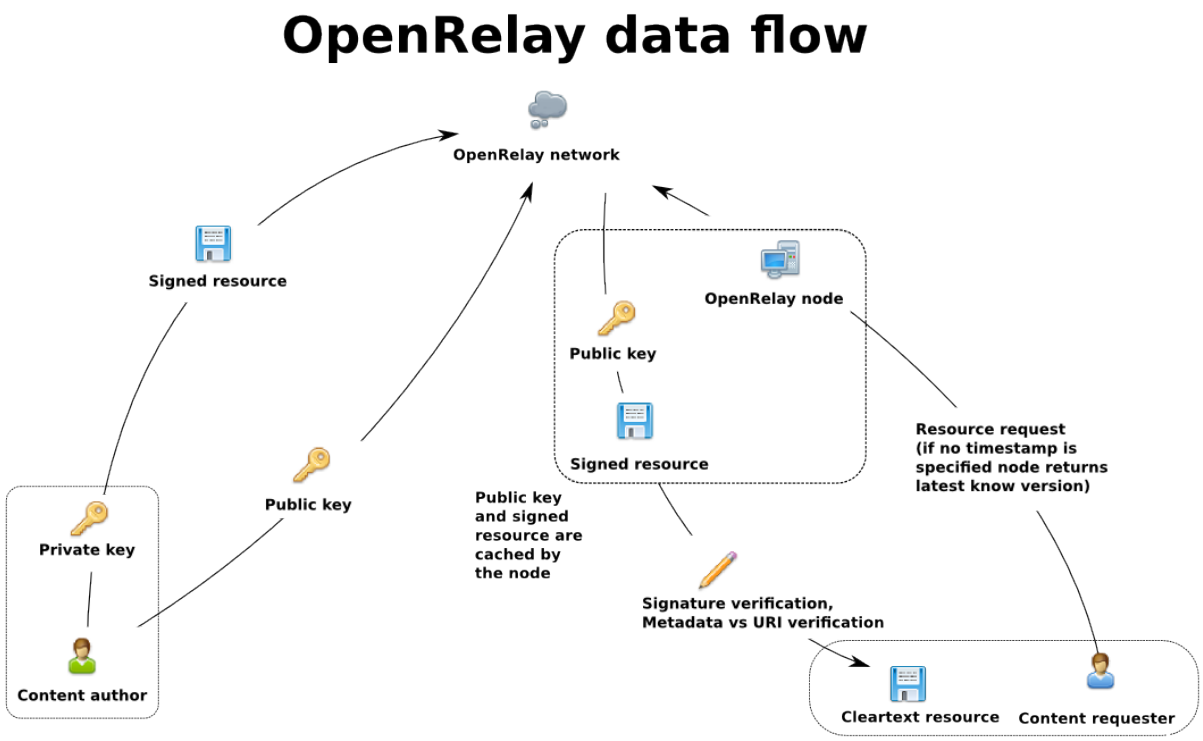
- More control over the key creation parameters
- Queue manager app backported from Mayan EDMS to improve background tasks efficiency
- Slash '/' character can now be part of a resources name in effect allowing filesystem path emulation
- The ammount of technical details shown to the user has been replaced with informative views
- Improved node indentity by using a private and public per node
- Better key handling, key publishing view and background key generation
- New configuration option: SERVER_HEARTBEAT_FAILURE_THRESHOLD, to determine when to delete a node from the siblings list
- Better remote node status detection and reporting
- Interactive locale switching
- Translation: Spanish, Czech, Klingon and Ukrainian
- Better caching of remote resources information
- Refactoring of the entire API
- Resource model class refactoring
- Project wide improved error and exception handling
- Filesystem storage improvements (deletion, filename sanitation)
- Adding of contact information to the about template, cleaned up the markup
- Improvements in template appearance (base, about, home and forms, image centering, etc)
- Favicon added
- New SERVER_INVENTORY_QUERY_INTERVAL configuration option to setup the time interval between nodes inventory query API call
- Node resource sharing API call implemented
- Node resource inventory sharing API call implemented
- Node resource inventory hash sharing API call implemented
- Updated the default keyserver of the project to just 'peer.to'
- Documentation updates, added development, installation, internals, license and settings pages
- New SERVER_HEARTBEAT_QUERY_INTERVAL configuration options to enable the setting of the heartbeat query interval
- CPU load calculator for the heartbeat API call
- lock_manager app from Mayan EDMS to avoid race conditions in scheduled task execution
- Added background scheduled tasks support using APScheduler

1.5 Planned features

- Validated API calls
- Routing
- Inter node communication encryption
- Node blacklisting
- HTTP Proxy support
- Load balancing
- Zip file upload support
- ARP-like resource locating algorithm
- More caching improvements

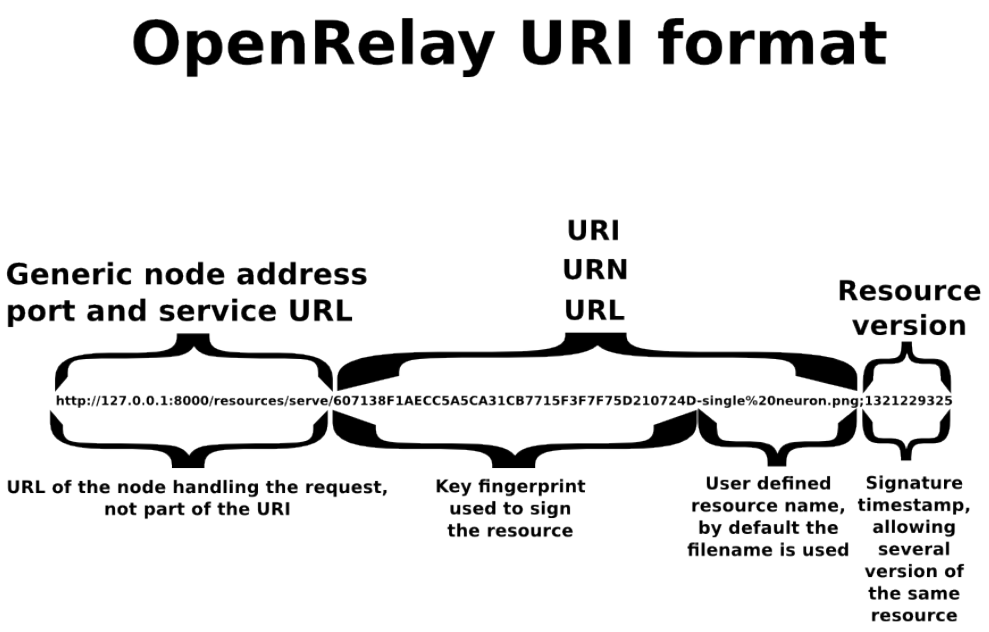
1.6 Internals

1.6.1 Basic overview



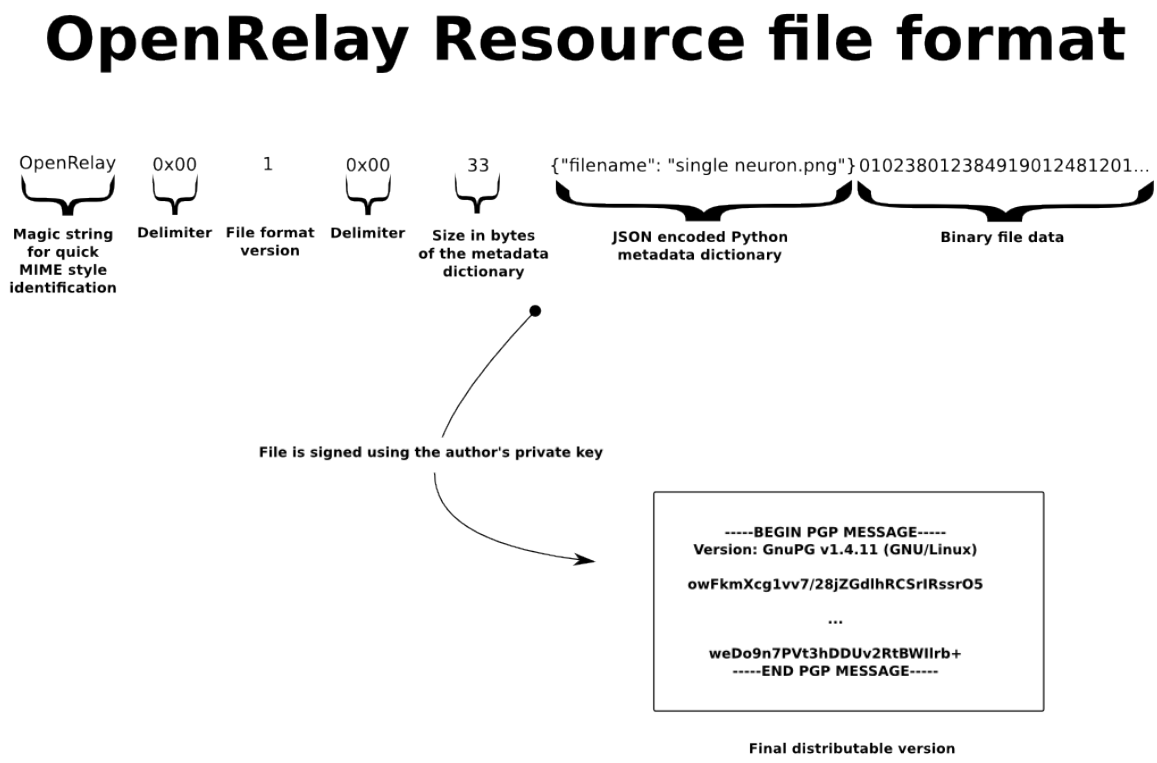
OpenRelay makes extensive use of Public-key cryptography to sign content uploaded to the network to ensure that such content, by the nature of being distributed and replicated hasn't been tampered. Traditional peer to peer file sharing networks provide increased content survivability and availability by creating and distributing copies, this works very good when users accessing content expect such content to never change, this mode of operation resembles that of a static repository. This mode of operation create a problem for content publishers where previously uploaded content will have a very different identification code than a more recently uploaded content with the publisher being unable announce to users of the network that a newer version of the same content is available. In effect the two versions are completely different instances of content replicated in the peer to peer network without any relation to each other. This is the reason existing peer to peer file sharing networks are not suitable for more dynamic uses such as web hosting. **OpenRelay** solves this problem, creating authenticated universally addressable versioned resources. When a user publishes content a unique identifier containing the user's personal key id and a name is created. This new content converted into an **OpenRelay** resource is also cryptographically time stamped, and this time stampbecomes the resource version number. If the user wishes to update the resource file, he just publishes new content with the same key and the same name, but this new resource will have a more recent timestamp, hence both resource have the same identifier but can be addressed individually if necesary. If another user wishes to access the resource file published by the first user, only the resource identifier needs to be specified with the network fetching the most recent version. Both versions of the resource exists on the network but as the older version is less and less requested, it will eventually disapear from the network as the individual nodes erase the less requested content from their respective caches.

1.6.2 URI specification



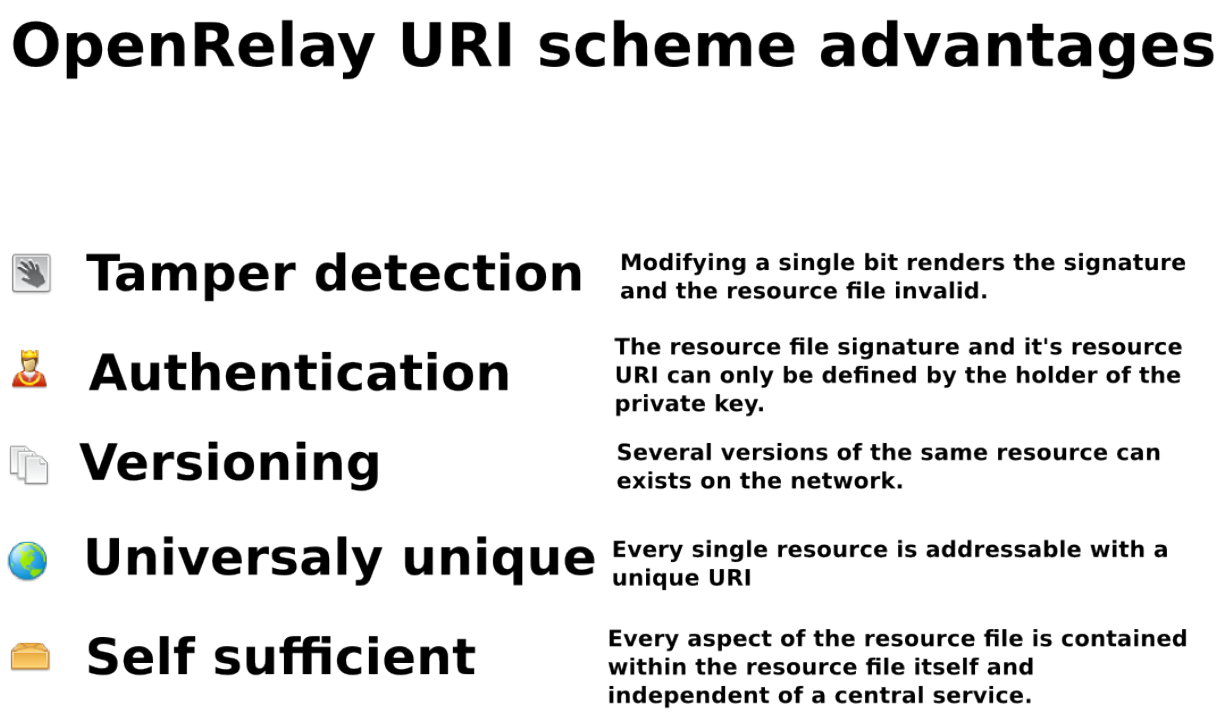
An **OpenRelay** resource identifier is crafted using the publisher's key fingerprint and appending a name to it, by default the original filename is used but the publisher can choose any string of characters. The resource identifier doesn't include any actual server location, because the same resource can be requested from any node in the network.

1.6.3 Resource file format



This is the actual content of the resource file, the filename is the only required metadata field and is used by recipients of the resource file for validation by appending it to the key fingerprint used during signing.

1.6.4 Advantages



1.7 Credits

- OpenRelay
  - Copyright (c) 2011 Steven Richards & Roberto Rosario
  - Site designed by Dustin Cogburn, with use of Twitter's Bootstrap
- Python
  - Copyright (c) 2001-2010 Python Software Foundation.
  - Copyright (c) 2000 BeOpen.com.
  - Copyright (c) 1995-2001 Corporation for National Research Initiatives.
  - Copyright (c) 1991-1995 Stichting Mathematisch Centrum, Amsterdam.
- Django - A high-level Python Web framework that encourages rapid development and clean, pragmatic design.
  - Copyright Django Software Foundation
  - http://www.djangoproject.com/
- Twitter bootstrap - HTML, CSS, and JS toolkit from Twitter

  - http://twitter.github.com/bootstrap
- Werkzeug - The Swiss Army knife of Python web development
  - Copyright Armin Ronacher (armin.ronacher@active-4.com)
  - http://werkzeug.pocoo.org/
- django-extensions - Extensions for Django
  - Copyright Bas van Oostveen (v.oostveen@gmail.com)
  - http://code.google.com/p/django-command-extensions/
- AutonomoTorrent - Fork of ABTorrent, a minimal, pure python BitTorrent client implementation using Twisted
  - https://github.com/joshsziegler/AutonomoTorrent
- Logo image - "network neurons 1" by gerard79
  - http://www.sxc.hu/photo/1043922
  - http://www.digital-delight.ch