edeposit.amqp.antivirus

Release 1.0.1

February 05, 2015

Contents

This module provides wrappers over ClamAV antivirus for edeposit project.

CHAPTER 1

Installation

Module is hosted at PIP, so you can install it easily with following command:

```
sudo pip install edeposit.amqp.antivirus
```

This will install the module and necessary requirements with one exception - the ClamAV itself. That can be installed manually or using package manager from your distribution.

Ubuntu/Debian:

```
sudo apt-get install clamav clamav-daemon
```

OpenSuse:

```
sudo zypper install clamav
```

1.1 Initialization

After installation of the `ClamAV` and `edeposit.amqp.antivirus`, run the *edeposit_clamd_init.py* script (should be in your path), which will configure ClamAV and create all necessary files and directories.

You may also want to check `settings` module, to change some of the paths using JSON configuration files.

1.1.1 Database update

You should update the signature database from time to time.

You can do it by running `freshclam` command, or by sending `UpdateDatabase` structure over AMQP.

I think, that the best way is to put the `freshclam` command to cron.

Usage

To check some file, encode it to `base64`, put it into `ScanFile` structure and send it over AMQP to `reactToAMQPMessage()`.

Here is example without AMQP communication, but at the AMQP level of abstraction:

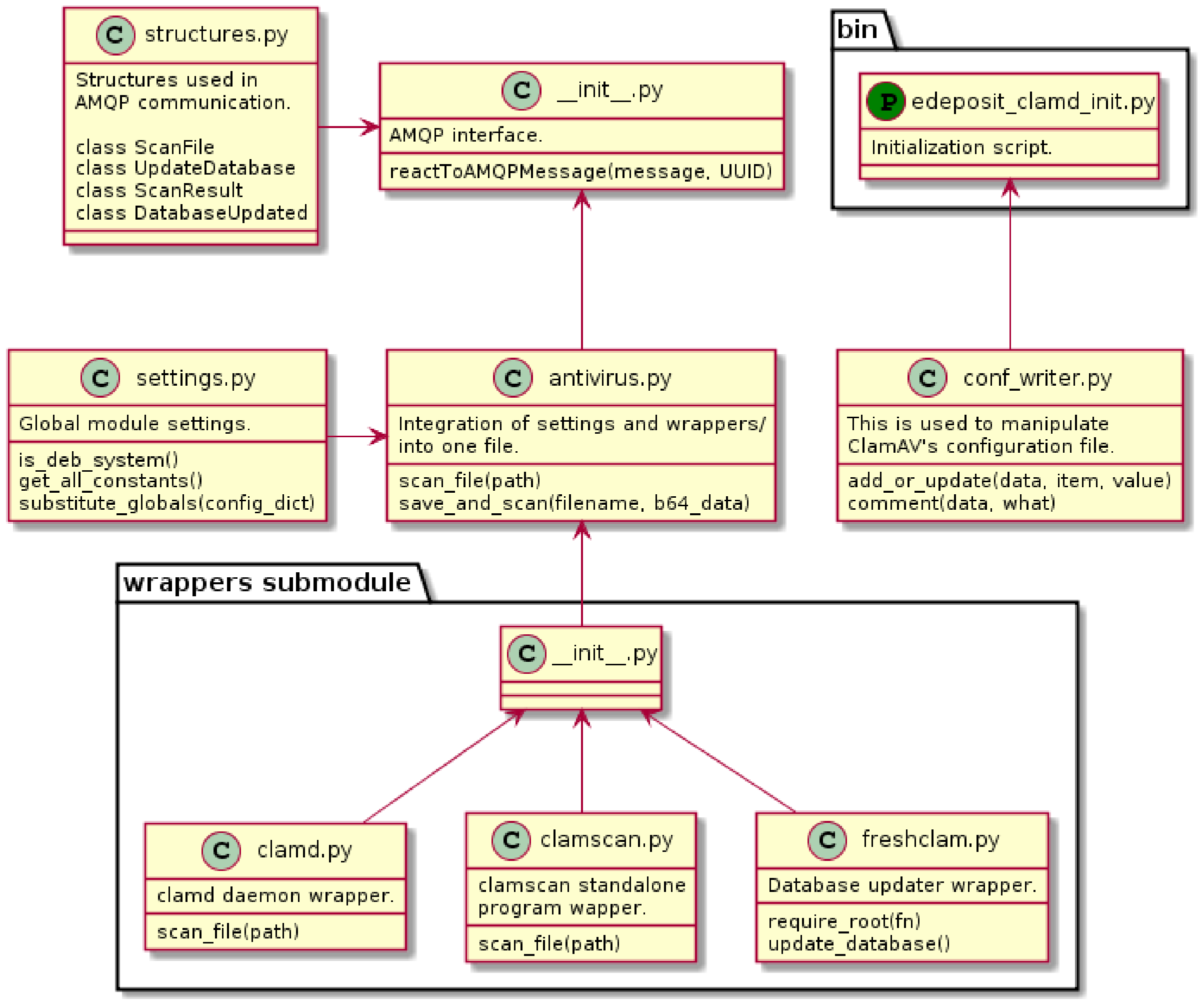
```
$ python
>>> import base64
>>> import antivirus as av
>>> fn = "test_file.exe"
>>> data = open(fn).read()
>>> av.reactToAMQPMessage(
...     av.structures.ScanFile(fn, base64.b64encode(data)),
...     "UUID"
... )
ScanResult(filename='test_file.exe', result={})  # result is blank -> file is ok
```

Or positive detection:

```
$ python
>>> import base64
>>> import antivirus as av
>>> fn = "eicar.com"
>>> data = open(fn).read()
>>> av.reactToAMQPMessage(
...     av.structures.ScanFile(fn, base64.b64encode(data)),
...     "UUID"
... )
ScanResult(filename='test_file.exe', result={u'/tmp/tmpuCA2fe_eicar.com': ('FOUND', 'Eicar-Test-Signa
```

Content

Parts of the module can be divided into two subcategories - script and API.



3.1 Standalone script

Script can be found in `bin/` folder and it should be automatically put into your path, so you can just simply run `edeposit_clamd_init.py` from shell.

3.1.1 Initializer script

Initialization script used to set necessary settings in ClamAV configuration file and correct permissions.

`edeposit_clamd_init.`**`REQUIRED_SETTINGS`** **= {'MaxThreads': '2', 'LocalSocket': '/var/run/clamav/clamd.ctl', 'AllowS**

All required settings is there, rest is not important.

`edeposit_clamd_init.`**`get_username`**`()`

Return username depending on type of system (deb/suse).

`edeposit_clamd_init.`**`update_configuration`**`(`*configuration*`)`

Set all configuration specified in `REQUIRED_SETTINGS`.

Parameters **configuration** (*str*) – Configuration file content.

Returns Updated configuration.

Return type str

`edeposit_clamd_init.`**`create_config`**`(`*cnf_file*, *uid*, *overwrite*`)`

Creates configuration file and the directory where it should be stored and set correct permissions.

Parameters

- **cnf_file** (*str*) – Path to the configuration file.
- **uid** (*int*) – User ID - will be used for chown.
- **overwrite** (*bool*) – Overwrite the configuration with `CLEAN_CONFIG`.

`edeposit_clamd_init.`**`create_log`**`(`*log_file*, *uid*`)`

Create log file and set necessary permissions.

Parameters

- **log_file** (*str*) – Path to the log file.
- **uid** (*int*) – User ID - will be used for chown.

`edeposit_clamd_init.`**`get_service_name`**`()`

Return name of the daemon depending on the system type.

`edeposit_clamd_init.`**`main`**`(`**args*, ***kwargs*`)`

Create configuration and log file. Restart the daemon when configuration is done.

Parameters

- **conf_file** (*str*) – Path to the configuration file.
- **overwrite** (*bool*) – Overwrite the configuration file with *clean* config?

Usage

```
$ ./edeposit_clamd_init.py -h
usage: edeposit_clamd_init.py [-h] [-v] [-o] [-c CONFIG]

edeposit.amqp.antivirus ClamAV initializer.

optional arguments:
  -h, --help            show this help message and exit
  -v, --verbose         Print logging messages.
  -o, --overwrite       Overwrite default configuration file. Don't worry,
                        your original file will be stored in backup_.
  -c CONFIG, --config CONFIG
                        Path to the configuration file. Default
                        /etc/clamav/clamd.conf.
```

3.2 API

3.2.1 antivirus package

There are two levels of abstraction - AMPQ API and python API.

AQMP API is higlevel API, where you send some structure, something happens in magick box and you get back another structure.

Python API is just collection of *"lowlevel"* python wrappers over ClamAV.

AMQP API

AMQP interface used by edeposit.amqp package.

`antivirus.`**`reactToAMQPMessage`**(*message*, *send_back*)

React to given (AMQP) message. *message* is expected to be `collections.namedtuple()` structure from `structures` filled with all necessary data.

Parameters

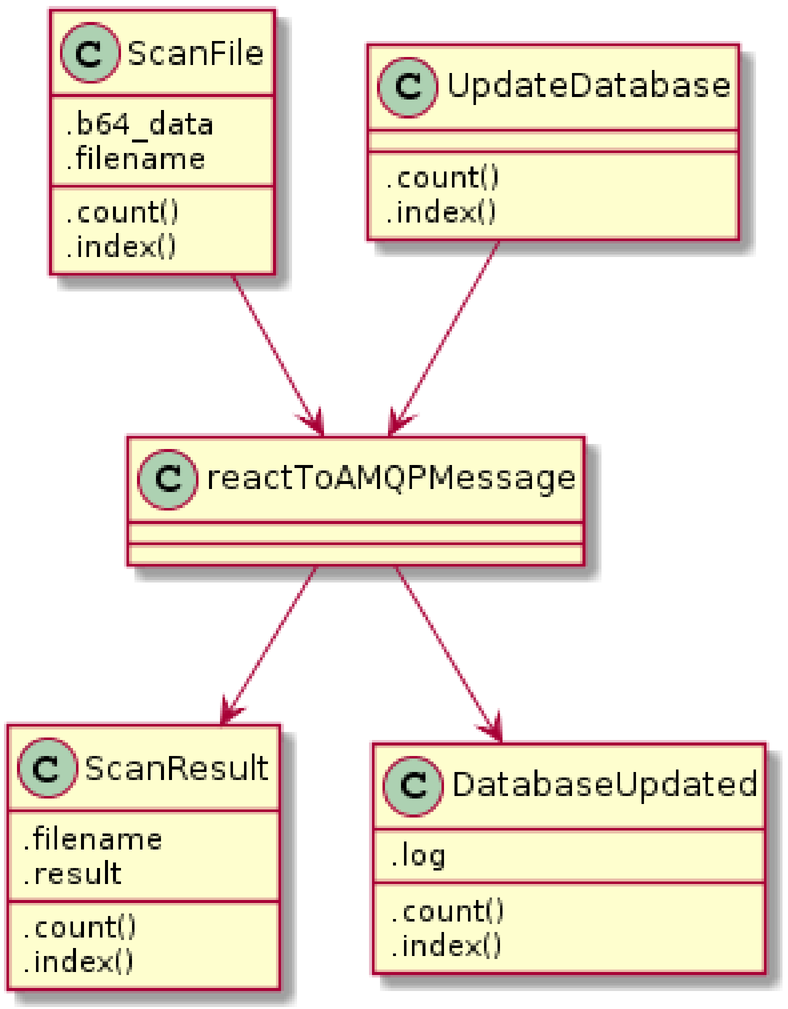
- **message** (*object*) – One of the request objects defined in `structures`.
- **send_back** (*fn reference*) – Reference to function for responding. This is useful for progress monitoring for example. Function takes one parameter, which may be response structure/namedtuple, or string or whatever would be normally returned.

Returns Response class from `structures`.

Return type object

Raises `ValueError` – if bad type of *message* structure is given.

All AMQP communication structures can be found in `structures` submodule.



Python API

Antivirus wrapper

ClamAV wrapper to scan files for malware.

`antivirus.antivirus.`**`scan_file`**(*path*)

Scan *path* for viruses using `clamd` or `clamscan` (depends on `settings.USE_CLAMD`.

Parameters **path** (*str*) – Relative or absolute path of file/directory you need to scan.

Returns `{filename: ("FOUND", "virus type")}` or blank dict.

Return type dict

Raises

- `ValueError` – When the server is not running.
- `AssertionError` – When the internal file doesn't exists.

`antivirus.antivirus.`**`save_and_scan`**(*filename*, *b64_data*)

Save *b64_data* to temporary file and scan it for viruses.

Parameters

- **filename** (*str*) – Name of the file - used as basename for tmp file.
- **b64_data** (*str*) – Content of the file encoded in base64.

Returns `{filename: ("FOUND", "virus type")}` or blank dict.

Return type dict

Configuration writer

This module is used to write and update configuration for ClamAV daemon.

`antivirus.conf_writer.`**`add_or_update`**`(`*data*, *item*, *value*`)`
Add or update value in configuration file format used by proftpd.

Parameters

- **data** (*str*) – Configuration file as string.
- **item** (*str*) – What option will be added/updated.
- **value** (*str*) – Value of option.

Returns updated configuration

Return type str

`antivirus.conf_writer.`**`comment`**`(`*data*, *what*`)`
Comments line containing *what* in string *data*.

Parameters

- **data** (*str*) – Configuration file in string.
- **what** (*str*) – Line which will be commented out.

Returns Configuration file with commented *what*.

Return type str

Wrappers

wrappers package This package contains lowlevel wrappers over `clamd`, `clamscan` and `freshclam`.

Submodules

clamd wrapper API for scanning files using `clamd` daemon.

`antivirus.wrappers.clamd.`**`scan_file`**`(`*path*`)`
Scan *path* for viruses using `clamd` antivirus daemon.

Parameters **path** (*str*) – Relative or absolute path of file/directory you need to scan.

Returns `{filename: ("FOUND", "virus type")}` or blank dict.

Return type dict

Raises

- `ValueError` – When the server is not running.

- `AssertionError` – When the internal file doesn't exists.

clamscan wrapper API for scanning files using `clamscan` standalone program.

`antivirus.wrappers.clamscan.`**`scan_file`**`(`*path*`)`
Scan *path* for viruses using `clamscan` program.

Parameters **path** (*str*) – Relative or absolute path of file/directory you need to scan.

Returns `{filename: ("FOUND", "virus type")}` or blank dict.

Return type dict

Raises `AssertionError` – When the internal file doesn't exists.

freshclam wrapper Wrapper over `freshclam` program to update database over amqp.

`antivirus.wrappers.freshclam.`**`require_root`**`(`*fn*`)`
Decorator to make sure, that user is root.

`antivirus.wrappers.freshclam.`**`update_database`**`(`**args*, ***kwargs*`)`
Run `freshclam`. Make sure, that user is root.

clamd wrapper API for scanning files using `clamd` daemon.

`antivirus.wrappers.clamd.`**`scan_file`**`(`*path*`)`
Scan *path* for viruses using `clamd` antivirus daemon.

Parameters **path** (*str*) – Relative or absolute path of file/directory you need to scan.

Returns `{filename: ("FOUND", "virus type")}` or blank dict.

Return type dict

Raises

- `ValueError` – When the server is not running.
- `AssertionError` – When the internal file doesn't exists.

clamscan wrapper API for scanning files using `clamscan` standalone program.

`antivirus.wrappers.clamscan.`**`scan_file`**`(`*path*`)`
Scan *path* for viruses using `clamscan` program.

Parameters **path** (*str*) – Relative or absolute path of file/directory you need to scan.

Returns `{filename: ("FOUND", "virus type")}` or blank dict.

Return type dict

Raises `AssertionError` – When the internal file doesn't exists.

freshclam wrapper Wrapper over `freshclam` program to update database over amqp.

`antivirus.wrappers.freshclam.`**`require_root`**`(`*fn*`)`
Decorator to make sure, that user is root.

`antivirus.wrappers.freshclam.`**`update_database`**`(`**args*, ***kwargs*`)`
Run `freshclam`. Make sure, that user is root.

Package configuration

If you wish to change behavior or paths to some of the files, you can use do it in `settings` submodule.

3.2.2 Antivirus wrapper

ClamAV wrapper to scan files for malware.

`antivirus.antivirus.`**`scan_file`**`(`*path*`)`

Scan *path* for viruses using `clamd` or `clamscan` (depends on `settings.USE_CLAMD`.

Parameters **path** (*str*) – Relative or absolute path of file/directory you need to scan.

Returns `{filename:  ("FOUND", "virus type")}` or blank dict.

Return type dict

Raises

- `ValueError` – When the server is not running.
- `AssertionError` – When the internal file doesn't exists.

`antivirus.antivirus.`**`save_and_scan`**`(`*filename*, *b64_data*`)`

Save *b64_data* to temporary file and scan it for viruses.

Parameters

- **filename** (*str*) – Name of the file - used as basename for tmp file.
- **b64_data** (*str*) – Content of the file encoded in base64.

Returns `{filename:  ("FOUND", "virus type")}` or blank dict.

Return type dict

3.2.3 AMQP communication structures

Definitions of the communication structures used in edeposit.amqp.antivirus project.

class `antivirus.structures.`**`ScanFile`**

Bases: `antivirus.structures.ScanFile`

Request to scan file.

Parameters

- **filename** (*str*) – Path of the file at your system. It will be used in result structure.
- **b64_data** (*str*) – Base64 encoded content of the file.

Returns `ScanResult`

Return type object

class `antivirus.structures.`**`UpdateDatabase`**

Bases: `antivirus.structures.UpdateDatabase`

Request to update clamav database (= to run `freshclam` program).

Returns `DatabaseUpdated`

Return type object

class `antivirus.structures.`**`ScanResult`**

Bases: `antivirus.structures.ScanResult`

Result of the file scan.

Parameters

- **filename** (*str*) – Name of the file as was specified in `ScanFile` request.
- **result** (*dict*) – Dictionary in following format:

```
{
    "local_path": ("RESULT", "TYPE")
}
```

Where *RESULT* is "FOUND" or string like that and *TYPE* is name of the malware.

Note: When no malware is found, `result` is blank dict.

class `antivirus.structures.`**`DatabaseUpdated`**

Bases: `antivirus.structures.DatabaseUpdated`

Response to `UpdateDatabase`.

Attr: log (str): Log of the `freshclam` run.

3.2.4 Settings and configuration

Module is containing all necessary global variables for the package.

Module also has the ability to read user-defined data from two paths:

- `$HOME/_SETTINGS_PATH`
- `/etc/_SETTINGS_PATH`

See `_SETTINGS_PATH` for details.

Note: If the first path is found, other is ignored.

Example of the configuration file (`$HOME/edeposit/antivirus.json`):

```
{
    "USE_CLAMD": false
}
```

Attributes

`antivirus.settings.`**`is_deb_system`**`()`

Badly written test whether the system is deb/apt based or not.

`antivirus.settings.`**`USE_CLAMD`** **= True**

True - `clamd` daemon will be used, False - `clamscan` will be used. `clamscan` takes much less memory, but takes a LOT more time to scan. `clamd` takes huge amouts of memory (500MB min), but scans in fractions of seconds.

`antivirus.settings.`**`DEB_CONF_PATH`** **= '/etc/clamav/'**

Configuration file directory at debian systems.

`antivirus.settings.`**`SUSE_CONF_PATH`** **= '/etc/'**
Configuration file directory at suse systems.

`antivirus.settings.`**`CONF_FILE`** **= 'clamd.conf'**
Name of the configuration file.

`antivirus.settings.`**`CONF_PATH`** **= '/etc/clamav/clamd.conf'**
Path to the configuration file.

`antivirus.settings.`**`LOCALSOCKET`** **= '/var/run/clamav/clamd.ctl'**
Path to the local unix socket - don't change this if you are not sure (it will break things).

`antivirus.settings.`**`PIDFILE`** **= '/var/run/clamav/clamd.pid'**
Path to the pid file - don't change this if you are not sure (it will break things).

`antivirus.settings.`**`LOGFILE`** **= '/var/log/clamav/clamav.log'**
Path to the log file.

`antivirus.settings.`**`get_all_constants`**`()`
Get list of all uppercase, non-private globals (doesn't start with _).

Returns Uppercase names defined in *globals()* (variables from this module).

Return type list

`antivirus.settings.`**`substitute_globals`**`(`*config_dict*`)`
Set global variables to values defined in *config_dict*.

Parameters **config_dict** (*dict*) – dictionary with data, which are used to set *globals*.

Note: *config_dict* have to be dictionary, or it is ignored. Also all variables, that are not already in globals, or are not types defined in `_ALLOWED` (str, int, float) or starts with _ are silently ignored.

3.2.5 wrappers package

This package contains lowlevel wrappers over `clamd`, `clamscan` and `freshclam`.

Submodules

clamd wrapper

API for scanning files using `clamd` daemon.

`antivirus.wrappers.clamd.`**`scan_file`**`(`*path*`)`
Scan *path* for viruses using `clamd` antivirus daemon.

Parameters **path** (*str*) – Relative or absolute path of file/directory you need to scan.

Returns `{filename:  ("FOUND", "virus type")}` or blank dict.

Return type dict

Raises

- `ValueError` – When the server is not running.
- `AssertionError` – When the internal file doesn't exists.

clamscan wrapper

API for scanning files using `clamscan` standalone program.

`antivirus.wrappers.clamscan.`**`scan_file`**`(`*path*`)`
Scan *path* for viruses using `clamscan` program.

Parameters **path** (*str*) – Relative or absolute path of file/directory you need to scan.

Returns `{filename:  ("FOUND", "virus type")}` or blank dict.

Return type dict

Raises `AssertionError` – When the internal file doesn’t exists.

freshclam wrapper

Wrapper over `freshclam` program to update database over amqp.

`antivirus.wrappers.freshclam.`**`require_root`**`(`*fn*`)`
Decorator to make sure, that user is root.

`antivirus.wrappers.freshclam.`**`update_database`**`(`**args*, ***kwargs*`)`
Run `freshclam`. Make sure, that user is root.

CHAPTER 4

Source code

The project is opensource (GPL) and source codes can be found at GitHub:

- https://github.com/edeposit/edeposit.amqp.antivirus

Testing

Almost every feature of the project is tested in unit/integration tests. You can run this tests using provided `run_tests.sh` script, which can be found in the root of the project.

5.1 Requirements

Test script expects that pytest is installed. In case you don't have it yet, it can be easily installed using following command:

```
pip install --user pytest
```

or for all users:

```
sudo pip install pytest
```

5.2 Options

Script provides three options - to run just unittests (`-u` switch), to run integration tests (`-i` switch) or to run both (`-a` switch).

Integration tests requires that ClamAV is installed, running and that the test script has **root permissions**.

Example of the success output from the test script:

```
$ sudo service clamav-daemon start
[sudo] password for bystrousak:
 * Starting ClamAV daemon clamd

$ ./run_tests.sh -a
============================== test session starts ===============================
platform linux2 -- Python 2.7.5 -- py-1.4.20 -- pytest-2.5.2
collected 7 items

src/edeposit/amqp/antivirus/tests/integration/test_antivirus.py .....
src/edeposit/amqp/antivirus/tests/unittests/test_amqp.py ..

=========================== 7 passed in 44.04 seconds ============================
```

CHAPTER 6

Indices and tables

- *genindex*
- *modindex*
- *search*

Python Module Index

Index