esis Documentation

Release 0.2.0

Javier Collado

Mar 05, 2018

Contents

Contents:

CHAPTER 1

Elasticsearch Index & Search

1.1 Features

- Index content for every SQLite database row in Elasticsearch
- Search indexed content

1.2 Why?

esis is based on the code used in a mobile forensics product. An important use case of such a product is to extract data from a mobile device and provide a way for investigators to search relevant information in that data. Since most of that data is stored in SQLite databases, it makes sense to figure out a way to perform that operation in an efficient way and Elasticsearch has been a good solution to that problem so far.

The tool was initially released as a companion to the presentation how to search extracted data that was given at DFRWS EU 2015

CHAPTER 2

Installation

At the command line:

```
$ easy_install esis
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv esis
$ pip install esis
```

CHAPTER 3

Usage

3.1 As a python library

To use *esis* in a project just import the *Client* object from the *esis* package and call any of the following methods: *index*, *count*, *search* and *query*.

```
from esis import Client

client = Client()
client.index(directory)
client.count()
client.search(query)
client.clean()
```

Note: The client method names match the command line tool subcommand names as it can be seen in next section.

3.2 As a command line tool

- Index every SQLite database row under a given directory (recursively)

```
esis index <directory>
```

- Search for a given string in the indexed data

```
esis search <query>
```

- Get information about the number of indexed documents

```
esis count
```

- Delete all indexed documents

```
esis clean
```

3.3 In docker containers

3.3.1 Using docker-compose

Docker files are included in the source code to run esis and elasticsearch in their own containers. To build/pull the images needed to run esis and start the elasticsearch server, use the following commands:

```
docker-compose build
docker-compose start
```

After that, to launch esis in a container run:

```
docker-compose run esis <subcommand>
```

where <*subcommand*> is any of the subcommands in the previous section (*index*, *search*, *count* or *clean*).

Note:

- If *docker-compose run* is executed too quickly, then a connection error might be returning meaning that elasticsearch is still initializing.
- The entry point in the esis container uses the *–host* command line option to connect to the linked container where elasticsearch is running.
- The user home directory is mounted in the esis container as */data*. This must be taken into account when passing a directory to the *index* subcommand using a path in the container, not in the host machine.

3.3.2 Using docker

If you want to use *docker* instead of *docker-compose*, you just need to follow the steps below.

- Download the images

```
docker pull elasticsearch
docker pull jcollado/esis
```

- Start Elasticsearch

```
docker run --name elasticsearch elasticsearch
```

where the *–name* option is used to have a known container name that can be used later to link the Elasticsearch container to the esis one.

The entry point for the esis image assumes that the hostname for Elasticsearch is elasticsearch in the container. Take this into account when using the *–link* option as in the examples below.

- Run esis
  - Index data under directory

```
docker run --link elasticsearch:elasticsearch -v <directory>:/data --rm -
↪t -i jcollado/esis index /data
```

where <*directory*> is a location in the host filesystem and */data* is an arbitrary directory used to index data in the container.

- Count indexed documents

```
docker run --link elasticsearch:elasticsearch --rm -t -i jcollado/esis␣
↪count
```

- Run search query

```
docker run --link elasticsearch:elasticsearch --rm -t -i jcollado/esis␣
↪search <string>
```

where <*string*> is the string to be used in the query against Elasticsearch.

- Remove indexed documents

```
docker run --link elasticsearch:elasticsearch --rm -t -i jcollado/esis␣
↪clean
```

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/jcollado/esis/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" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

4.1.4 Write Documentation

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

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

```
$ git clone git@github.com:your_name_here/esis.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 esis
$ cd esis/
$ 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 esis tests
$ python setup.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. Check https://travis-ci.org/jcollado/esis/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

Use dot notation to run any subset of test cases included in a module, class or even by selecting a single test case by its method:

```
$ python -m unittest tests.test_es.ClientTest.test_index
```

CHAPTER 5

Credits

5.1 Development Lead

- Javier Collado <jcollado@nowsecure.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.0 (2015-03-23)

- First release on PyPI.

6.2 0.2.0 (2015-05-14)

- All documents indexed under the same index name.
- Docker files allow using the tool in a container.

CHAPTER 7

License

Developer documentation:

CHAPTER 8

esis package

8.1 Submodules

8.2 esis.cli module

Command Line Interface.

esis.cli.**clean**(*args*)
Remove all indexed documents.

esis.cli.**configure_logging**(*log_level*)
Configure logging based on command line argument.

Parameters **log_level** (*int*) – Log level passed form the command line

esis.cli.**count**(*args*)
Print indexed documents information.

esis.cli.**index**(*args*)
Index database information into elasticsearch.

esis.cli.**main**(*argv=None*)
Entry point for the esis.py script.

esis.cli.**parse_arguments**(*argv*)
Parse command line arguments.

Returns Parsed arguments

Return type argparse.Namespace

esis.cli.**search**(*args*)
Send query to elasticsearch.

esis.cli.**valid_directory**(*path*)
Directory validation.

8.3 esis.db module

Database related tools.

class esis.db.**DBReader**(*database*)
Bases: object

Iterate through all db tables and rows easily.

Parameters **database** (esis.db.Database) – Database to traverse

FTS_SUFFIXES = ('content', 'segdir', 'segments', 'stat', 'docsize')

tables()
Generator that traverses all tables in a database.

Returns Table name

Return type str

class esis.db.**Database**(*db_filename*)
Bases: object

Generic database object.

Parameters **db_filename** (*str*) – Path to the sqlite database file

connect()
Create connection.

disconnect()
Close connection.

reflect(*table_names*)
Get table metadata through reflection.

sqlalchemy already provides a reflect method, but it will stop at the first failure, while this method will try to get as much as possible.

Parameters **table_names** (*list(str)*) – Table names to inspect

run_quick_check()
Check database integrity.

Some files, especially those files created after carving, might not contain completely valid data.

class esis.db.**DatetimeDecorator**(**args*, ***kwargs*)
Bases: sqlalchemy.sql.type_api.TypeDecorator

A datetime class that translates data to ISO strings.

The reason ISO strings are used instead of datetime objects or integer timestamps is because is what elasticsearch handles as a datetime value. Internally it seems to store it as an integer timestamp, but that's transparent to the user.

impl
alias of TEXT

process_result_value(*value*, *_dialect*)
Translate datetime/timestamp to ISO string.

class esis.db.**IntegerDecorator**(**args*, ***kwargs*)
Bases: sqlalchemy.sql.type_api.TypeDecorator

An integer class that translates 'null' values to None.

This is needed because some tables use 'null' instead of NULL and elastic search fails to index documents with strings where integers should be found.

impl
alias of INTEGER

process_result_value(*value*, *_dialect*)
Translate 'null' to None if needed.

class esis.db.**TableReader**(*database*, *table_name*)
Bases: *esis.db.TypeCoercionMixin*

Iterate over all rows easily.

Parameters

- **database** (esis.db.Database) – Database being explored
- **table** (*sqlalchemy.sql.schema.Table*) – Database table

get_schema()
Return table schema.

Returns Column names and their type

Return type dict(str, sqlalchemy.types.*)

rows()
Generator that traverses all rows in a table.

Returns All rows in the table

Return type generator(sqlalchemy.engine.result.RowProxy)

class esis.db.**TypeCoercionMixin**
Bases: object

A mixin to transform database values.

This is useful to get safe values from sqlalchemy when data types are not very well defined in SQLite.

COERCIONS = {<class 'sqlalchemy.sql.sqltypes.BOOLEAN'>: <class 'esis.db.IntegerDecora

8.4 esis.es module

Elasticsearch related funcionality.

class esis.es.**Client**(*host*, *port*)
Bases: object

Elasticsearch client wrapper.

Parameters

- **host** (*str*) – Elasticsearch host
- **port** (*int*) – Elasticsearch port

INDEX_NAME = 'sqlite'

clean()
Remove all indexed documents.

count()
Return indexed documents information.

Returns Indexed documents information

Return type dict

index(*directory*)
Index all the information available in a directory.

In elasticsearch there will be an index for each database and a document type for each table in the database.

Parameters **directory** (*str*) – Directory that should be indexed

search(*query*)
Yield all documents that match a given query.

Parameters **query** (*str*) – A simple query with data to search in elasticsearch

Returns Records that matched the query as returned by elasticsearch

Return type list(dict)

class esis.es.**Mapping**(*document_type*, *table_schema*)
Bases: object

ElasticSearch mapping.

Parameters

- **document_type** (*str*) – Document type user for the database table
- **table_schema** (*dict(str, sqlalchemy.types.*)*) – Database table schema from sqlalchemy

SQL_TYPE_MAPPING = {<class 'sqlalchemy.sql.sqltypes.DATETIME'>: 'date', <class 'sqlal

esis.es.**get_document**(*db_filename*, *table_name*, *row*)
Get document to be indexed from row.

Parameters

- **db_filename** (*str*) – Path to the database file
- **table_name** – Database table name
- **row** (*sqlalchemy.engine.result.RowProxy*) – Database row

esis.es.**get_index_action**(*index_name*, *document_type*, *document*)
Generate index action for a given document.

Parameters

- **index_name** (*str*) – Elasticsearch index to use
- **document_type** – Elasticsearch document type to use
- **document** – Document to be indexed

Returns Action to be passed in bulk request

Return type dict

8.5 esis.fs module

Filesystem functionality.

class esis.fs.**TreeExplorer**(*directory*, *blacklist=None*)

Bases: object

Look for sqlite files in a tree and return the valid ones.

Parameters

- **directory** (*str*) – Base directory for the tree to be explored.
- **blacklist** (*list(str)*) – List of relative directories to skip

paths()

Return paths to valid databases found under directory.

Returns Paths to valid databases

Return type list(str)

8.6 esis.util module

Utility functionality.

esis.util.**datetime_to_timestamp**(*datetime_obj*)

Return a timestamp for the given datetime object.

Parameters **datetime_obj** (*datetime.datetime*) – datetime object to be converted

Returns timestamp from the passed datetime object

Return type int

8.7 Module contents

Elastic Search Index & Search.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

e

Index