stix2-patterns Documentation

Release 2.0.0

OASIS Open

Mar 31, 2022

Contents:

1 Installation **3**
- 1.1 Requirements . . . 3
- 1.2 Install Package . . . 3

2 Usage **5**
- 2.1 From Python Code . . . 5
- 2.2 User Input . . . 5
- 2.3 File Input . . . 6

3 Developer's Guide **7**
- 3.1 Setting up a development environment . . . 7
- 3.2 Code style . . . 8
- 3.3 Updating the Grammar . . . 8
- 3.4 Testing . . . 8
- 3.5 Adding a dependency . . . 9

4 CHANGELOG **11**
- 4.1 2.0.0 - Released 2022-03-31 . . . 11
- 4.2 1.3.2 - Released 2020-12-10 . . . 11
- 4.3 1.3.1 - Released 2020-07-09 . . . 11
- 4.4 1.3.0 - Released 2020-03-04 . . . 11
- 4.5 1.2.1 - Released 2019-11-26 . . . 11
- 4.6 1.2.0 - Released 2019-11-22 . . . 12
- 4.7 1.1.0 - Released 2018-11-20 . . . 12
- 4.8 1.0.0 - Released 2018-07-18 . . . 12
- 4.9 0.6.0 - Released 2017-11-13 . . . 12
- 4.10 0.5.0 - Released 2017-07-12 . . . 12
- 4.11 0.4.1 - Released 2017-05-19 . . . 12
- 4.12 0.4.0 - Released 2017-05-19 . . . 13
- 4.13 0.3.0 - Released 2017-05-04 . . . 13
- 4.14 0.2.2 - Released 2017-03-01 . . . 13
- 4.15 0.2.0 - Released 2017-02-24 . . . 13

5 stix2patterns **15**
- 5.1 stix2patterns package . . . 15

6 Indices and tables **17**

Python Module Index 19

Index 21

The STIX 2 Pattern Validator is a software tool for checking the syntax of the Cyber Threat Intelligence (CTI) STIX Pattern expressions, which are used within STIX to express conditions (prepresented with the Cyber Observable data model) that indicate particular cyber threat activity. The repository contains source code, an ANTLR grammar, and automated tests for the tool. The validator can be used as a command-line tool or as a Python library which can be included in other applications.

CHAPTER 1

Installation

1.1 Requirements

- Python 2.7 or 3.4+
- ANTLR grammar runtime (4.7 or newer):
  - antlr4-python2-runtime (Python 2.7)
  - antlr4-python3-runtime (Python 3)
- six
- typing (Python 3.4)

1.2 Install Package

Using pip is highly recommended:

```
$ pip install stix2-patterns
```

For more information about installing Python packages, see the Python Packaging User Guide.

CHAPTER 2

Usage

The STIX Pattern Validator provides an executable script (`validate-patterns`) in addition to being an importable Python library.

The `validate-patterns` script accepts patterns from either direct user input or a file passed as an option.

2.1 From Python Code

The `run_validator` function can be called on any Python string. It returns a list of errors encountered while parsing the pattern.

```
from stix2patterns.validator import run_validator

pattern = "[file-object:hashes.md5 = '79054025255fb1a26e4bc422aef54eb4']"
errors = run_validator(pattern)
```

2.2 User Input

When prompted, enter a pattern to validate and press enter. The validator will supply whether the pattern has passed or failed. If the pattern fails the test, the validator will supply where the first syntax error occurred. The validator will continue to prompt for patterns until Ctrl-C is pressed. Example:

```
$ validate-patterns

Enter a pattern to validate: [file-object:hashes.md5 =
↪'79054025255fb1a26e4bc422aef54eb4']

PASS: [file-object:hashes.md5 = '79054025255fb1a26e4bc422aef54eb4']
```

2.3 File Input

```
$ validate-patterns -f <path_to_file>
```

Use <path_to_file> to specify the path to a file containing a set of patterns to validate. Each pattern must be on a separate line of the file so that the validator may determine where the pattern begins and ends. The validator will supply the PASS/FAIL result of each pattern.

CHAPTER 3

Developer's Guide

We're thrilled that you're interested in contributing to stix2-patterns! Here are some things you should know:

- contribution-guide.org has great ideas for contributing to any open-source project (not just this one).
- All contributors must sign a Contributor License Agreement. See CONTRIBUTING.md in the project repository for specifics.
- If you are planning to implement a major feature (vs. fixing a bug), please discuss with a project maintainer first to ensure you aren't duplicating the work of someone else, and that the feature is likely to be accepted.

Now, let's get started!

3.1 Setting up a development environment

We recommend using a virtualenv.

1. Clone the repository. If you're planning to make pull request, you should fork the repository on GitHub and clone your fork instead of the main repo:

```
$ git clone https://github.com/yourusername/cti-pattern-validator.git
```

2. Install develoment-related dependencies:

```
$ cd cti-pattern-validator
$ pip install -r requirements.txt
```

3. Install pre-commit git hooks:

```
$ pre-commit install
```

At this point you should be able to make changes to the code.

3.2 Code style

All code should follow PEP 8. We allow for line lengths up to 160 characters, but any lines over 80 characters should be the exception rather than the rule. PEP 8 conformance will be tested automatically by Tox and Travis-CI (see below).

3.3 Updating the Grammar

The ANTLR pattern grammar is maintained in the stix2-json-schemas repository. If the grammar changes, the code in this repository should be updated to match. To do so, use the Java ANTLR package to generate new Python source files. (The .jar file is not needed for normal use of the validator).

1. Download antlr-4.7.1-complete.jar from http://www.antlr.org/download/
2. Clone the stix2-json-schemas repository or download the STIXPattern.g4 file.
3. Change to the directory containing the STIXPattern.g4 file.
4. Run the following command (for STIX v2.1)

   ```
   $ java -jar "/path/to/antlr-4.7.1-complete.jar" -Dlanguage=Python2 STIXPattern.g4 -visitor -o /p
   ```

5. Commit the resulting files to git.

3.4 Testing

Note: All of the tools mentioned in this section are installed when you run `pip install -r requirements.txt`.

This project uses pytest for testing. We encourage the use of test-driven development (TDD), where you write (failing) tests that demonstrate a bug or proposed new feature before writing code that fixes the bug or implements the features. Any code contributions should come with new or updated tests.

To run the tests in your current Python environment, use the `pytest` command from the root project directory:

```
$ pytest
```

This should show all of the tests that ran, along with their status.

You can run a specific test file by passing it on the command line:

```
$ pytest stix2patterns/test/v21/test_<xxx>.py
```

You can also test against the examples provided in the supplied example file. Note that you must specify which version to test.

```
$ validate-patterns -v 2.1 -f stix2patterns/test/v21/spec_examples.txt
```

To ensure that the test you wrote is running, you can deliberately add an `assert False` statement at the beginning of the test. This is another benefit of TDD, since you should be able to see the test failing (and ensure it's being run) before making it pass.

tox allows you to test a package across multiple versions of Python. Setting up multiple Python environments is beyond the scope of this guide, but feel free to ask for help setting them up. Tox should be run from the root directory of the project:

```
$ tox
```

We aim for high test coverage, using the coverage.py library. Though it's not an absolute requirement to maintain 100% coverage, all code contributions must be accompanied by tests. To run coverage and look for untested lines of code, run:

```
$ pytest --cov=stix2patterns
$ coverage html
```

then look at the resulting report in `htmlcov/index.html`.

All commits pushed to the `master` branch or submitted as a pull request are tested with Travis-CI automatically.

3.5 Adding a dependency

One of the pre-commit hooks we use in our develoment environment enforces a consistent ordering to imports. If you need to add a new library as a dependency please add it to the *known_third_party* section of *.isort.cfg* to make sure the import is sorted correctly.

CHAPTER 4

CHANGELOG

4.1 2.0.0 - Released 2022-03-31

- #85 Update to ANTLR 4.9 (@chisholm)
- #88 Default to STIX 2.1 version patterns

4.2 1.3.2 - Released 2020-12-10

- #79 Fix bug to prevent crashing on '*' selector in hashes (@chisholm)
- #81 Fix bug with bracket checking to allow for nested parentheses (@chisholm)

4.3 1.3.1 - Released 2020-07-09

- #75 Fix bug with SSDEEP hashes in STIX 2.1 (@emmanvg)

4.4 1.3.0 - Released 2020-03-04

- #68 Update to ANTLR 4.8 (@chisholm)
- #69 Add "visit()" methods to Pattern classes (@chisholm)
- #71 Fix bug with multiple qualifiers in a pattern (@chisholm)

4.5 1.2.1 - Released 2019-11-26

- Fix some imports for backwards compatibility

4.6 1.2.0 - Released 2019-11-22

- #59 Fixed bug where malformed hashes would pass (@JohannKT)
- #63, #64 Fixed bugs with leading and trailing whitespace (@squioc)
- Support STIX 2.1 patterns
- Add testing for Python 3.8

4.7 1.1.0 - Released 2018-11-20

- Add a visitor to the ANTLR parser
- Add testing for Python 3.7

4.8 1.0.0 - Released 2018-07-18

- #34 - Add documentation on ReadTheDocs: https://stix2-patterns.readthedocs.io/
- #39 - Raise error for unexepected unused character values.
- #41 - Raise error for negative REPEAT values.
- #42 - Improved Timestamp validation.
- #43 - Validate Base64 binary literals.
- #48 - Make pattern qualifier and operator keywords case-sensitive.
- Drop support for Python 2.6 and 3.3.

4.9 0.6.0 - Released 2017-11-13

- #32 - Added a public walk() method to the Pattern class. (@chisholm)
- Make repository structure match other projects. (@emmanvg)

4.10 0.5.0 - Released 2017-07-12

- Separate object and path components in inspector.
- Support "NOT" qualifier on all comparison operators.

4.11 0.4.1 - Released 2017-05-19

- Repackaged to not use a Wheel distribution

4.12 0.4.0 - Released 2017-05-19

- Encapsulated parsed patterns in a new Pattern class

4.13 0.3.0 - Released 2017-05-04

- Update for STIX 2.0 WD02.
- Add "inspector" module to extract features from patterns.
- Improve error messages.
- Update to ANTLR 4.7
- Add testing for Python 2.6 and 3.6

4.14 0.2.2 - Released 2017-03-01

- Update packaging to install correct ANTLR4 runtime depending on Python version.

4.15 0.2.0 - Released 2017-02-24

- Initial public version.

CHAPTER 5

stix2patterns

5.1 stix2patterns package

5.1.1 Subpackages

stix2patterns.grammars package

Submodules

stix2patterns.grammars.STIXPatternLexer module

stix2patterns.grammars.STIXPatternListener module

stix2patterns.grammars.STIXPatternParser module

Module contents

stix2patterns.test package

Submodules

stix2patterns.test.test_inspector module

stix2patterns.test.test_validator module

Module contents

5.1.2 Submodules

5.1.3 stix2patterns.inspector module

exception stix2patterns.inspector.**InspectionException**
Bases: Exception

Represents a error that occurred during inspection.

5.1.4 stix2patterns.pattern module

5.1.5 stix2patterns.validator module

Validates a user entered pattern against STIXPattern grammar.

stix2patterns.validator.**main**()
Continues to validate patterns until it encounters EOF within a pattern file or Ctrl-C is pressed by the user.

stix2patterns.validator.**run_validator**(*pattern*, *stix_version='2.1'*)
Validates a pattern against the STIX Pattern grammar. Error messages are returned in a list. The test passed if the returned list is empty.

stix2patterns.validator.**validate**(*user_input*, *stix_version='2.1'*, *ret_errs=False*, *print_errs=False*)
Wrapper for run_validator function that returns True if the user_input contains a valid STIX pattern or False otherwise. The error messages may also be returned or printed based upon the ret_errs and print_errs arg values.

5.1.6 Module contents

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

stix2patterns, 16
stix2patterns.grammars, 15
stix2patterns.grammars.STIXPatternLexer, 15
stix2patterns.grammars.STIXPatternListener, 15
stix2patterns.grammars.STIXPatternParser, 15
stix2patterns.inspector, 16
stix2patterns.pattern, 16
stix2patterns.test, 16
stix2patterns.validator, 16

Index

I

InspectionException, 16

M

main() (*in module stix2patterns.validator*), 16

R

run_validator() (*in module stix2patterns.validator*), 16

S

stix2patterns (*module*), 16
stix2patterns.grammars (*module*), 15
stix2patterns.grammars.STIXPatternLexer (*module*), 15
stix2patterns.grammars.STIXPatternListener (*module*), 15
stix2patterns.grammars.STIXPatternParser (*module*), 15
stix2patterns.inspector (*module*), 16
stix2patterns.pattern (*module*), 16
stix2patterns.test (*module*), 16
stix2patterns.validator (*module*), 16

V

validate() (*in module stix2patterns.validator*), 16