# django-restricted-sessions Documentation

***Release 0.1.4***

## django-restricted-sessions

November 03, 2016

Contents

Contents:

CHAPTER 1

# django-restricted-sessions

Restricts Django sessions to IP and/or user agent.

If the IP or user agent changes after creating the session, the a 400 response is given to the request, the session is flushed (all session data deleted, new session created) and a warning is logged. The goal of this middleware is to make it harder for an attacker to use a session ID they obtained. It does not make abuse of session IDs impossible.

For compatibility with IPv6 privacy extensions, by default only the first 64 bits of an IPv6 address are checked.

## 1.1 Documentation

The full documentation is at https://django-restricted-sessions.readthedocs.org.

## 1.2 Quickstart

Install django-restricted-sessions:

```
pip install django-restricted-sessions
```

Then add it to your middleware after SessionMiddleware:

```
MIDDLEWARE_CLASSES = [
    ....
    'django.contrib.sessions.middleware.SessionMiddleware',
    # 'django.contrib.auth.middleware.AuthenticationMiddleware',
    'restrictedsessions.middleware.RestrictedSessionsMiddleware',
    ....
]
```

When `RESTRICTEDSESSIONS_AUTHED_ONLY` setting enabled ensure this middleware is added after `AuthenticationMiddleware` such that the `request.user` is present.

CHAPTER 2

# Installation

At the command line:

```
$ easy_install django-restricted-sessions
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django-restricted-sessions
$ pip install django-restricted-sessions
```

CHAPTER 3

# Usage

## 3.1 Setup

Add the RestrictedSessionsMiddleware to your MIDDLEWARE_CLASSES settings, after SessionMiddleware:

```
MIDDLEWARE_CLASSES = [
    ....
    'django.contrib.sessions.middleware.SessionMiddleware',
    'restrictedsessions.middleware.RestrictedSessionsMiddleware',
    ....
]
```

When RestrictedSessionsMiddleware sees a request with a new session, it will store the client's current IP and user agent in the session. Upon further requests, it will validate the IP and user agent. If changes have occured, a 400 response is returned, the session is flushed and a warning is logged.

By default, for IPv4 addresses, the address must remain exactly the same. For IPv6 address, the first 64 bits must remain the same. The latter 64 bits may change rapidly in clients that have IPv6 privacy extensions (RFC4941) enabled, so rigorous checks on those will only annoy users. User agents must also remain exactly the same.

When encountering requests without a session, or in which the client IP could not be found, the middleware takes no action.

## 3.2 Settings

The following settings are available:

- `RESTRICTEDSESSIONS_RESTRICT_IP` (bool, default: True): restrict sessions based on IP address.
- `RESTRICTEDSESSIONS_RESTRICT_UA` (bool, default: True): restrict sessions based on user agent.
- `RESTRICTEDSESSIONS_REMOTE_ADDR_KEY` (string, default: 'REMOTE_ADDR'): key in request.META under which the real IP address can be found. This may differ depending on the setup of the web and WSGI server.
- `RESTRICTEDSESSIONS_IPV4_LENGTH` (int, default: 32): number of bits to consider when comparing IPv4 addresses. 32 means the full address must be equal, 24 would mean that a change from 192.0.2.1 to 192.0.2.200 is allowed, but not to 192.0.3.1.

- `RESTRICTEDSESSIONS_IPV6_LENGTH` (int, default: 64): number of bits to consider when comparing IPv6 addresses. 128 means the full address must be equal. 64 means that a change from 2001:db8::1 to 2001:db8::3 is allowed, but not to 2001:db9::1. Setting this to 128 is not recommended, as it will cause frequent session invalidation if clients use IPv6 privacy extensions.
- `RESTRICTEDSESSIONS_REDIRECT_VIEW` (string, default: None): when this value is set to be a known view

configured within the project's ROOT_URLCONF, then any failure of the session validation will redirect to this location after the session is cleared/flushed. * `RESTRICTEDSESSIONS_FAILURE_STATUS` (int, default: 400) the HTTP status code to return when

**not utilizing RESTRICTEDSESSIONS_REDIRECT_VIEW setting such that any failure of the session validation** will return this status code.

- `RESTRICTEDSESSIONS_AUTHED_ONLY` (bool, default: False) when set to true, only restricts the sessions

for authenticated users. Utilizes the *django.contrib.auth.logout* method to invalidate the session when enabled.

## 3.3 How much added security does this offer?

In a case where an attacker is able to obtain a session ID and tries to reuse it, this middleware will prevent this if the attacker is not careful. Once an attacker has the session ID, it is fairly likely that they also know the original user agent, which they could spoof. If they are in the same location as the victim, they may also be using the same IPv4 address or IPv6 block. Therefore, this middleware adds an extra hurdle for session ID abuse at very low cost, but may not help against careful attackers in the right situations.

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/erikr/django-restricted-sessions/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

django-restricted-sessions could always use more documentation, whether as part of the official django-restricted-sessions 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/erikr/django-restricted-sessions/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 *django-restricted-sessions* for local development.

1. Fork the *django-restricted-sessions* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-restricted-sessions.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 django-restricted-sessions
$ cd django-restricted-sessions/
$ 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 restrictedsessions 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 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/erikr/django-restricted-sessions/pull_requests and make sure that the tests pass for all supported Python versions.

## 4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_middleware
```

CHAPTER 5

# Credits

## 5.1 Development Lead

- Erik Romijn <eromijn@solidlinks.nl>

## 5.2 Contributors

- Matt Davis <matteius@gmail.com>
- Rainer Koirikivi <rainer@koirikivi.fi>

CHAPTER 6

# History

## 6.1 0.1.4 (2016-07-02)

- Fixed an exception that could occur when non-utf8 bytes were included in user agent strings.

## 6.2 0.1.3.1 (2016-05-26)

- Version bump to avoid PyPI's duplicate filename ban.

## 6.3 0.1.3 (2016-05-26)

- Added support to redirect to known view, or use custom status code settings.
- Added support for ignoring unauthenticated sessions.
- Fixed short circuit when REMOTE_ADDR was unknown.
- Dropped support for older Python versions: now requires 2.7, 3.3 or newer, with Django 1.8.

## 6.4 0.1.2 (2014-03-20)

- Resolved exception being raised when session switches from IPv4 to IPv6
- Python 3.4 support

## 6.5 0.1.1 (2014-02-18)

- Added missing netaddr requirement to setup.py.

## 6.6 0.1.0 (2014-02-17)

- First release on PyPI.