# django-sudo Documentation

***Release 2.1.0***

**Matt Robenolt**

July 26, 2016

# Contents

`django-sudo` is an implementation of GitHub's Sudo Mode for Django.

CHAPTER 1

# What is this for?

`django-sudo` provides an extra layer of security for after a user is already logged in. Views can be decorated with `@sudo_required`, and then a user must re-enter their password to view that page. After verifying their password, that user has elevated permissions for the duration of `SUDO_COOKIE_AGE`. This duration is independent of the normal session duration allowing short elevated permission durations, but retain long user sessions.

# CHAPTER 2

# Installation

```
$ pip install django-sudo
```

CHAPTER 3

# Compatibility

- Django 1.4-1.9
- Python 2.6-3.5
- pypy

CHAPTER 4

# Contents

## 4.1 Getting Started

### 4.1.1 Installation

First, install the `django-sudo` library with pip.

```
$ pip install django-sudo
```

Next, we need to add the `sudo` application to our `INSTALLED_APPS`. Installing the application will automatically register the `user_logged_in` and `user_logged_out` signals that are needed.

```
INSTALLED_APPS = (
    # ...
    'sudo',
)
```

Now we need to install the required middleware into `MIDDLEWARE_CLASSES`:

```
MIDDLEWARE_CLASSES = (
    # ...
    'sudo.middleware.SudoMiddleware',
)
```

**Note:** `sudo.middleware.SudoMiddleware` **must** be installed after `django.contrib.session.middleware.SessionMiddleware`.

Proceed to the Configuration documentation.

## 4.2 Configuration

### 4.2.1 Settings

By default, all of the settings are optional and define sane and secure defaults.

**`SUDO_URL`** The url or view name for the sudo view. *Default: sudo.views.sudo*

**`SUDO_REDIRECT_URL`** Default url to be redirected to after elevating permissions. *Default: /*

**`SUDO_REDIRECT_FIELD_NAME`** The querystring argument to be used for redirection. *Default: next*

**SUDO_COOKIE_AGE** How long should sudo mode be active for? Duration in seconds. *Default: 10800*

**SUDO_COOKIE_DOMAIN** The domain to bind the sudo cookie to. *Default: current exact domain*.

**SUDO_COOKIE_HTTPONLY** Should the cookie only be accessible via http requests? *Default: True*

> **Note:** If this is set to `False`, any JavaScript files have the ability to access this cookie, so this should only be changed if you have a good reason to do so.

**SUDO_COOKIE_NAME** The name of the cookie to be used for sudo mode. *Default: sudo*

**SUDO_COOKIE_PATH** Restrict the sudo cookie to a specific path. *Default: /*

**SUDO_COOKIE_SECURE** Only transmit the sudo cookie over https if True. *Default: matches current protocol*

> **Note:** By default, we will match the protocol that made the request. So if your sudo page is over https, we will set the `secure` flag on the cookie so it won't be transmitted over plain http. It is highly recommended that you only use `django-sudo` over https.

**SUDO_COOKIE_SALT** An extra salt to be added into the cookie signature. *Default: ''*

**SUDO_REDIRECT_TO_FIELD_NAME** The name of the session attribute used to preserve the redirect destination between the original page request and successful sudo login. *Default: sudo_redirect_to*

### 4.2.2 Set up URLs

We need to hook up one url to use `django-sudo` properly. At minimum, you need something like the following:

```
(r'^sudo/$',  # Whatever path you want
    'sudo.views.sudo',  # Required
    {'template_name': 'sudo/sudo.html'}  # Optionally change the template to be used
)
```

### 4.2.3 Required Template

To get up and running, we last need to create a template for the sudo page to render. By default, the package will look for `sudo/sudo.html` but can easily be overwritten by setting the `template_name` when defining the url definition as seen above.

#### sudo/sudo.html

This template gets rendered with the the following context:

**form** An instance of `SudoForm`.

**SUDO_REDIRECT_FIELD_NAME** The value of `?next=/foo/`. If `SUDO_REDIRECT_FIELD_NAME` is `name`, then expect to find `{{ next }}` in the context, with the value of `/foo/`.

After configuring things, we can now start securing pages.

## 4.3 Usage

Once we have `django-sudo` installed and configured, we need to decide which views should be secured.

`sudo.decorators.`**`sudo_required`**`()`

The meat of `django-sudo` comes from decorating your views with `@sudo_required` much in the same way that `@login_required` works.

Let's pretend that we have a page on our site that has sensitive information that we want to make extra sure that a user is allowed to see it:

```python
from sudo.decorators import sudo_required

@login_required  # Make sure they're at least logged in
@sudo_required  # On top of being logged in, are you in sudo mode?
def super_secret_stuff(request):
    return HttpResponse('your social security number')
```

That's it! When a user visits this page and they don't have the correct permission, they'll be redirected to a page and prompted for their password. After entering their password, they'll be redirected back to this page to continue on what they were trying to do.

**class** `sudo.mixins.`**`SudoMixin`**

`SudoMixin` provides an easy way to sudo a class-based view. Any view that inherits from this mixin is automatically wrapped by the `@sudo_required` decorator.

This works well with the `LoginRequiredMixin` from django-braces:

```python
from django.views import generic
from braces.views import LoginRequiredMixin
from sudo.mixins import SudoMixin

class SuperSecretView(LoginRequiredMixin, SudoMixin, generic.TemplateView):
    template_name = 'secret/super-secret.html'
```

`request.`**`is_sudo`**`()`

Returns a boolean to indicate if the current request is in sudo mode or not. This gets added on by the *`SudoMiddleware`*. This is an shortcut for calling *`has_sudo_privileges()`* directly.

**class** `sudo.middleware.`**`SudoMiddleware`**

By default, you just need to add this into your `MIDDLEWARE_CLASSES` list.

**`has_sudo_privileges`**`(`*self*, *request*`)`

Subclass and override *`has_sudo_privileges()`* if you'd like to override the default behavior of *`request.is_sudo()`*.

**`process_request`**`(`*self*, *request*`)`

Adds *`is_sudo()`* to the request.

**`process_response`**`(`*self*, *request*, *response*`)`

Controls the behavior of setting and deleting the sudo cookie for the browser.

`sudo.utils.`**`grant_sudo_privileges`**`(`*request*, *max_age=SUDO_COOKIE_AGE*`)`

Assigns a random token to the user's session that allows them to have elevated permissions.

```python
from sudo.utils import grant_sudo_privileges
token = grant_sudo_privileges(request)
```

sudo.utils.**revoke_sudo_privileges**(*request*)

Revoke sudo privileges from a request explicitly

```
from sudo.utils import revoke_sudo_privileges
revoke_sudo_privileges(request)
```

sudo.utils.**has_sudo_privileges**(*request*)

Check if a request is allowed to perform sudo actions.

```
from sudo.utils import has_sudo_privileges
has_sudo = has_sudo_privileges(request)
```

## 4.4 Contributing

### 4.4.1 Getting the source

You will first want to clone the source repository locally with `git`:

```
$ git clone git@github.com:mattrobenolt/django-sudo.git
```

### 4.4.2 Setup Environment

I would recommend using virtualenv to set up a dev environment. After creating an environment, install all dep dependencies with:

```
$ pip install -r dev-requirements.txt
```

### 4.4.3 Running Tests

Tests are run using pytest and can be found inside `tests/*`.

Tests can simply be run using:

```
$ py.test
```

This will discover and run the test suite using your default Python interpreter. To run tests for all supported platforms, we use tox.

```
$ tox
```

### 4.4.4 Submitting Patches

Patches are accepted via Pull Requests on GitHub.

**Note:** If you are submitting a security patch, please see our Security page for special instructions.

#### Tests

All new code and changed code must come with **100%** test coverage to be considered for acceptance.

## 4.5 How does this work?

`django-sudo` works by setting an additional cookie that must match a secret value in your session. This cookie is ideally set to a shorter TTL than the normal session. When not in sudo mode, any view that is decorated with `@sudo_required` will require the user to re-enter their password. Once in sudo mode, they won't be prompted to enter their password for the next `SUDO_COOKIE_AGE` seconds.

In practice, we want to serve this sudo cookie over https only to avoid a man-in-the-middle attack where someone hijacks this cookie. This can be utilized safely in situations where the sessionid cookie is being transmitted over http, but we want to make sure that secure areas of our site are not accessible with just the sessionid.

- When logging in, `django-sudo` automatically elevates your permission to `sudo mode`.
- A second cookie is sent to your browser (by default this cookie is named `sudo` but can be set to anything with `SUDO_COOKIE_NAME`). This cookie contains a randomly generated string of characters.
- The same randomly generated string of characters is stored in the user's session.
- On subsequent requests, the cookie value must match the value that was stored in the session. If the values do not match, or the cookie is not sent at all, the user will be redirected to a page to re-enter their password.
- If they re-enter their password successfully, a new cookie is set and their permissions are again elevated.

**Note:** The best way to secure your site and your users is to use https. `django-sudo` won't be able to help you if it's being served over http.

## 4.6 Security

We take the security of `django-sudo` seriously. If you believe you've identified a security vulnerability, please report it to `matt@ydekproductions.com`.

You may encrypt the message using the PGP fingerprint: `D9D7 15C4 32A9 8C40 F794 35AB A54A 6FD1 5E45 9C77`.

Once you've submitted an issue via email, you should receive an acknowledgement within 48 hours, and depending on the action to be taken, you may receive further follow-up emails.

## 4.7 Changelog

### 4.7.1 2.1.0

- Vendored a more secure `is_safe_url` implementation from latest Django, instead of relying on a potentially insecure bundled version. See #17.

### 4.7.2 2.0.1

- Added `sudo.views.SudoView` class based view. This is now more extensible and should be preferred over the older `sudo.views.sudo` view function.
- Removed `SUDO_FORM` setting. It's now suggested to subclass `sudo.views.SudoView` and override `form_class`.

- Added `SUDO_URL` setting to set the url for the sudo page.

### 4.7.3 2.0.0

- Bad release. :( Don't install.

### 4.7.4 1.3.0

- Store `redirect_to` value in session. See #10.

### 4.7.5 1.2.1

- Pass along `request` to template context See #8.
- Verified compatibility with Django 1.9

### 4.7.6 1.2.0

- Verified compatibility with python 3.5 and pypy3
- Verified compatibility with Django 1.8
- Dropped support for python 3.2
- Better support for custom User models. See #4.
- Added a `SudoMixin` for use with class based views. See #5.

### 4.7.7 1.1.3

- Use `constant_time_compare` when verifying the correct sudo token.
- Make sure to check against all `AUTHENTICATION_BACKENDS` for the `SudoForm`. See #3.

### 4.7.8 1.1.2

- Added new setting, `SUDO_FORM` which allows you to override the default form that is used. See #2.

### 4.7.9 1.1.1

- Fixed a bug when using the new `SUDO_COOKIE_SALT`. If specifying a non-default salt, all cookies would be marked incorrectly as invalid.
- Don't use `request.REQUEST` anymore since that's deprecated in modern Django. Always use `request.GET` instead since we never POSTed the `next` variable anyways.

### 4.7.10 1.1.0

- Switch to using signed cookies for the sudo cookie, see #1.
- Added new `SUDO_COOKIE_SALT` setting to go along with the signed cookie.

### 4.7.11 1.0.0

- Initial release

# Python Module Index

# Index