
HawkREST Documentation

Release 0.0.1

Kumar McMillan

Jul 07, 2017

Contents

1	Installation	3
2	Topics	5
2.1	Usage	5
2.1.1	Django Configuration	5
2.1.2	Protecting API views with Hawk	7
2.1.3	Verification tool	8
2.2	Developers	8
2.2.1	Run the tests	8
2.2.2	Set up an environment	8
2.2.3	Build the docs	8
2.2.4	Publish a release	9
3	Bugs	11
4	Changelog	13
5	Indices and tables	15

Hawk HTTP Authorization for [Django Rest Framework](#). Hawk lets two parties securely communicate with each other using messages signed by a shared key. It is based on [HTTP MAC access authentication](#) (which was based on parts of [OAuth 1.0](#)).

HawkREST uses the [mohawk](#) module to add Hawk authorization to your REST API views.

This guide will help you set everything up but you should also read through [mohawk security considerations](#) to get familiar with the security aspects of Hawk.

CHAPTER 1

Installation

Requirements:

- Python 2.7+ or 3.4+
- Django 1.8 through 1.11
- Django Rest Framework 3.4 or 3.5
- mohawk

(Older versions of these libraries may work, but support is not guaranteed.)

Using `pip`, install the module like this:

```
pip install hawkrest
```

This will also install all necessary dependencies. You'll most likely put this in a `requirements` file within your Django app.

The source code is available at <https://github.com/kumar303/hawkrest>

Usage

Django Configuration

After *installation*, you'll need to configure your Django app with some variables in your `settings.py` file.

Make sure the module is installed as an app:

```
INSTALLED_APPS = (  
    ...  
    'hawkrest',  
)
```

Make sure the middleware is installed:

```
MIDDLEWARE_CLASSES = (  
    ...  
    'hawkrest.middleware.HawkResponseMiddleware',  
)
```

To protect all your REST views with Hawk, you can make hawkrest the default:

```
REST_FRAMEWORK = {  
    'DEFAULT_AUTHENTICATION_CLASSES': (  
        'hawkrest.HawkAuthentication',  
    ),  
    ...  
}
```

Set up the allowed access credentials. Each dict key will be a Hawk ID for a user who is allowed to connect to your API. For example, an incoming request with an ID named `script-user` would sign its request using the secret `this should be a long secret string` to make a successful connection. The credentials dict in your settings file would look like this:

```
HAWK_CREDENTIALS = {
    'script-user': {
        'id': 'script-user',
        'key': 'this should be a long secret string',
        'algorithm': 'sha256'
    },
}
```

You can add each Hawk credential to this dict.

If you need an alternative method for looking up credentials you can set up a lookup function under the `HAWK_CREDENTIALS_LOOKUP` setting. This function receives a Hawk ID as a parameter and returns a dict containing the credentials. For example, if you have a `HawkUser` model with a `key` attribute then you can write a function `hawk_credentials_lookup` as follows:

```
def hawk_credentials_lookup(id):
    user = HawkUser.objects.get(some_id=id)
    return {
        'id': id,
        'key': user.key,
        'algorithm': 'sha256'
    }
```

and then you would configure it in your settings:

```
HAWK_CREDENTIALS_LOOKUP = 'yourapi.auth.hawk_credentials_lookup'
```

Alternately, you can subclass `HawkAuthentication` and override the `hawk_credentials_lookup()` method. For example:

```
from hawkrest import HawkAuthentication

class YourHawk(HawkAuthentication):
    def hawk_credentials_lookup(self, id):
        user = HawkUser.objects.get(some_id=id)
        return {
            'id': id,
            'key': user.key,
            'algorithm': 'sha256'
        }
```

and then specify your new class instead in the authentication backend list:

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'yourapi.auth.YourHawk',
    ),
    ...
}
```

By default, a generic `HawkAuthenticatedUser` instance is returned when valid Hawk credentials are found. If you need another user model, you can set up a lookup function under the `HAWK_USER_LOOKUP` setting. This function receives the request and the matched credentials dict as parameters and returns a `(user, auth)` tuple as per [custom authentication](#). For example, with a `HawkUser` model whose `user_id` is included in the credentials dict, you can write a function `hawk_user_lookup` as follows:

```
def hawk_user_lookup(request, credentials):
    return HawkUser.objects.get(some_id=credentials['id'])
```

and then you would configure it in your settings:

```
HAWK_USER_LOOKUP = 'yourapi.auth.hawk_user_lookup'
```

Alternately, you can subclass `HawkAuthentication` and override the `hawk_user_lookup()` method. For example:

```
from hawkrest import HawkAuthentication

class YourHawk(HawkAuthentication):
    def hawk_user_lookup(self, request, credentials):
        return HawkUser.objects.get(some_id=credentials['id'])
```

and then specify your new class instead in the authentication backend list:

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'yourapi.auth.YourHawk',
    ),
    ...
}
```

This setting is the number of seconds until a Hawk message expires:

```
HAWK_MESSAGE_EXPIRATION = 60
```

To prevent replay attacks, Hawkrest uses the Django cache framework for nonce lookups. You should configure Django with something like `memcache` in production. By default, Django uses in-memory caching and by default nonce checking will be activated. If you need to *disable* it for some reason, set this:

```
USE_CACHE_FOR_HAWK_NONCE = False # only disable this if you need to
```

Protecting API views with Hawk

To protect all API views with Hawk by default, put this in your settings:

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'hawkrest.HawkAuthentication',
    ),
    'DEFAULT_PERMISSION_CLASSES': (
        'rest_framework.permissions.IsAuthenticated',
    ),
}
```

To protect a specific view directly, define it like this:

```
from rest_framework.permissions import IsAuthenticated
from rest_framework.views import APIView

from hawkrest import HawkAuthentication

class ExampleView(APIView):
```

```
authentication_classes = (HawkAuthentication,)
permission_classes = (IsAuthenticated,)
```

Verification tool

Hawkrest ships with a management command you can use to verify your own Hawk API or any other Hawk authorized resource.

Run this from a Django app with Hawkrest installed for more info:

```
./manage.py hawkrequest --help
```

If you had secured your Django app using the credentials dict with key `script-user` you could test it out like this:

```
./manage.py hawkrequest --url http://127.0.0.1:8000/your/view \  
--creds script-user -X POST -d foo=bar
```

Developers

Grab the source from Github: <https://github.com/kumar303/hawkrest>

Run the tests

You can run the full test suite with the `tox` command:

```
tox
```

To just run Python 2.7 unit tests type:

```
tox -e py27-django1.8-drf3.2
```

To just run doctests type:

```
tox -e docs
```

Set up an environment

Using a `virtualenv` you can set yourself up for development like this:

```
pip install -r requirements/dev.txt
python setup.py develop
```

Note that this won't install any libraries that are tested at different versions. You need `tox` for that.

Build the docs

In your development `virtualenv`, you can build the docs like this:

```
make -C docs/ html doctest  
open docs/_build/html/index.html
```

Publish a release

To publish a new release on [PyPI](#), make sure the changelog is up to date and make sure you bumped the module version in `setup.py`. Tag master as that version. For example, something like:

```
git tag 0.0.5  
git push --tags
```

Run this from the repository root to publish on [PyPI](#) as both a source distribution and wheel:

```
rm -rf dist/*  
python setup.py sdist bdist_wheel  
twine upload dist/*
```


CHAPTER 3

Bugs

You can report issues at <https://github.com/kumar303/hawkrest>

Important: If you're upgrading from a version prior to 0.0.6, be sure to use `rest_framework.permissions.IsAuthenticated` on your views *as documented*

- **1.0.0 (2017-04-05)**
 - Added support for a `HAWK_USER_LOOKUP` setting. See *Usage*.
 - Added hooks to make subclassing `HawkAuthentication` easier. See *Usage*.
 - Dropped support for Django 1.6/1.7.
 - Dropped support for `django-rest-framework` 3.2/3.3.
 - Confirmed support for `django-rest-framework` 3.4/3.5.
 - Added support for Django 1.10/1.11.
 - Started using *semantic versioning*.
- **0.0.10 (2016-06-01)**
 - Adds support for Django 1.9.
- **0.0.9 (2016-01-07)**
 - Adds more specific `AuthenticationFailed` errors.
- **0.0.8 (2015-10-01)**
 - Fixes [issue #11](#) where exception info was leaked to the response, potentially revealing sensitive information.
- **0.0.7 (2015-09-30)**
 - Fixes [issue #9](#) where using `rest_framework.permissions.IsAuthenticated` on your `Hawk` protected view caused an unexpected traceback.
- **0.0.6 (2015-09-08)**

- **IMPORTANT:** If migrating to this version from an earlier version of `hawkrest`, your Django Rest Framework API views *must* require an authenticated user *as documented*. In other words, older versions of `hawkrest` would reject any request that didn't have a Hawk authentication header but this version does not (see the bug fix below).
- Fixed bug where other HTTP authorization schemes could not be supported at the same time as Hawk. Thanks to [Mauro Doglio](#) for the patch.
- Fixed incorrect statement in docs that Python 2.6 was supported. Only 2.7 or greater is supported at this time.
- Sends `WWW-Authenticate: Hawk` header in 401 responses now.
- **0.0.5** (2015-07-21)
 - Added `HAWK_CREDENTIALS_LOOKUP` setting which is a *callable*. Thanks to [Felipe Otamendi](#) for the patch.
- **0.0.4** (2015-06-24)
 - Fixed nonce callback support for [mohawk 0.3.0](#). Thanks to Josh Wilson for the patches.
- **0.0.3** (2015-01-05)
 - Fixed traceback when cache setting is undefined. Thanks to [wolfgangmeyers](#) for the patch.
- **0.0.2** (2014-03-03)
 - Added support for Python 3.3 and greater
 - Added support for Python 2.6
- **0.0.1** (2014-02-27)
 - Initial release, extracted from <https://github.com/mozilla/apk-signer>

CHAPTER 5

Indices and tables

- `genindex`
- `modindex`
- `search`