Django Axes Documentation

Release 2.2.0

jazzband

Jul 20, 2017

Contents

CHAPTER 1

Contents

Installation

You can install the latest stable package running this command:

```
$ pip install django-axes
```

Configuration

Just add *axes* to your `INSTALLED_APPS`:

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    ...
    'axes',
    ...
)
```

Remember to run `python manage.py migrate` to sync the database.

Customizing Axes

You have a couple options available to you to customize `django-axes` a bit. These should be defined in your `settings.py` file.

- `AXES_LOGIN_FAILURE_LIMIT`: The number of login attempts allowed before a record is created for the failed logins. Default: `3`

- AXES_LOCK_OUT_AT_FAILURE: After the number of allowed login attempts are exceeded, should we lock out this IP (and optional user agent)? Default: True
- AXES_USE_USER_AGENT: If True, lock out / log based on an IP address AND a user agent. This means requests from different user agents but from the same IP are treated differently. Default: False
- AXES_COOLOFF_TIME: If set, defines a period of inactivity after which old failed login attempts will be forgotten. Can be set to a python timedelta object or an integer. If an integer, will be interpreted as a number of hours. Default: None
- AXES_LOGGER: If set, specifies a logging mechanism for axes to use. Default: 'axes.watch_login'
- AXES_LOCKOUT_TEMPLATE: If set, specifies a template to render when a user is locked out. Template receives cooloff_time and failure_limit as context variables. Default: None
- AXES_LOCKOUT_URL: If set, specifies a URL to redirect to on lockout. If both AXES_LOCKOUT_TEMPLATE and AXES_LOCKOUT_URL are set, the template will be used. Default: None
- AXES_VERBOSE: If True, you'll see slightly more logging for Axes. Default: True
- AXES_USERNAME_FORM_FIELD: the name of the form field that contains your users usernames. Default: username
- AXES_LOCK_OUT_BY_COMBINATION_USER_AND_IP: If True prevents the login from IP under a particular user if the attempt limit has been exceeded, otherwise lock out based on IP. Default: False
- AXES_ONLY_USER_FAILURES : If True only locks based on user id and never locks by IP if attempts limit exceed, otherwise utilize the existing IP and user locking logic Default: False
- AXES_NEVER_LOCKOUT_WHITELIST: If True, users can always login from whitelisted IP addresses. Default: False
- AXES_IP_WHITELIST: A list of IP's to be whitelisted. For example: AXES_IP_WHITELIST=['0.0.0.0']. Default: []
- AXES_BEHIND_REVERSE_PROXY: If True, it will look for the IP address from the header defined at AXES_REVERSE_PROXY_HEADER. Please make sure if you enable this setting to configure your proxy to set the correct value for the header, otherwise you could be attacked by setting this header directly in every request. Default: False
- AXES_REVERSE_PROXY_HEADER: If AXES_BEHIND_REVERSE_PROXY is True, it will look for the IP address from this header. Default: HTTP_X_FORWARDED_FOR
- AXES_NUM_PROXIES: If AXES_BEHIND_REVERSE_PROXY is True, use this value to calculate the end user IP address from the end of the list of IPs in header AXES_REVERSE_PROXY_HEADER. For example, if you have one (1) proxy configured and set AXES_NUM_PROXIES = 1 we, choose IP [ip.strip() for ip in request.META.get(AXES_REVERSE_PROXY_HEADER).split(',')][-1]. For X-Forwarded-For: a, b, client-ip this would pick the value client-ip. This configuration is used to prevent X-Forwarded-For (XFF) header spoofing or injection by the end user, because the X-Forwarded-For headers can be added to the request by the end user, circumventing the IP locking mechanisms in Axes. If you are running with Apache, nginx, or Elastic Load Balancer, you should set this to 1. It is by default configured to 0 for backwards compatibility. Default: 0
- AXES_DISABLE_ACCESS_LOG: If True, disable all access logging, so the admin interface will be empty.
- AXES_DISABLE_SUCCESS_ACCESS_LOG: If True, successful logins will not be logged, so the access log shown in the admin interface will only list unsuccessful login attempts.

Usage

Using `django-axes` is extremely simple. All you need to do is periodically check the Access Attempts section of the admin.

By default, django-axes will lock out repeated attempts from the same IP address. You can allow this IP to attempt again by deleting the relevant `AccessAttempt` records in the admin.

You can also use the `axes_reset` management command using Django's `manage.py`.

- `manage.py axes_reset` will reset all lockouts and access records.
- `manage.py axes_reset ip` will clear lockout/records for ip

In your code, you can use `from axes.utils import reset`.

- `reset()` will reset all lockouts and access records.
- `reset(ip=ip)` will clear lockout/records for ip
- `reset(username=username)` will clear lockout/records for a username

Requirements

`django-axes` requires a supported Django version. The application is intended to work around the Django admin and the regular `django.contrib.auth` login-powered pages. Look here https://github.com/jazzband/django-axes/blob/master/.travis.yml to check if your django / python version are supported.

Development

You can contribute to this project forking it from github and sending pull requests.

This is a Jazzband project. By contributing you agree to abide by the Contributor Code of Conduct and follow the guidelines.

Running tests

Clone the repository and install the django version you want. Then run:

```
$ ./runtests.py
```

Issues

Not being locked out after failed attempts

You may find that Axes is not capturing your failed login attempts. It may be that you need to manually add watch_login to your login url.

For example, in your urls.py:

```
...
from my.custom.app import login
from axes.decorators import watch_login
...
urlpatterns = patterns('',
    (r'^login/$', watch_login(login)),
...
```

Locked out without reason

It may happen that you have suddenly become locked out without a single failed attempt. One possible reason is that you are using some custom login form and the username field is named something different than "username", e.g. "email". This leads to all users attempts being lumped together. To fix this add the following to your settings:

AXES_USERNAME_FORM_FIELD = "email"

Using a captcha

Using https://github.com/mbi/django-simple-captcha you do the following:

1. Change axes lockout url in `settings.py`:

```
AXES_LOCKOUT_URL = '/locked'
```

2. Add the url in `urls.py`:

```
url(r'^locked/$', locked_out, name='locked_out'),
```

3. Create a captcha form:

```
class AxesCaptchaForm(forms.Form):
    captcha = CaptchaField()
```

4. Create a captcha view for the above url that resets on captcha success and redirects:

```
def locked_out(request):
    if request.POST:
        form = AxesCaptchaForm(request.POST)
        if form.is_valid():
            ip = get_ip_address_from_request(request)
            reset(ip=ip)
            return HttpResponseRedirect(reverse_lazy('signin'))
    else:
        form = AxesCaptchaForm()

    return render_to_response('locked_out.html', dict(form=form), context_
↪instance=RequestContext(request))
```

5. Add a captcha template:

```
<form action="" method="post">
    {% csrf_token %}

    {{ form.captcha.errors }}
```

```
    {{ form.captcha }}

    <div class="form-actions">
        <input type="submit" value="Submit" />
    </div>
</form>
```

CHAPTER 2

Indices and tables

- search