Django HMAC Documentation

Release 1.3.2

SOON_

January 26, 2016

Contents

This module provides a middleware for HMAC signature Django views. It's simply designed to check that a client is entitled to access routes, based on the fact that it must possess a copy of the secret key.

CHAPTER 1

Key features:

- HMAC Middleware
- HMAC View decorators
- Multiple keys for more services
- Service restricted access

CHAPTER 2

Small example

```
class SignedView(View):

    @decorators.auth
    def get(self, request):
        return HttpResponse("for all services")

    @decorators.auth(only=['userservice'])
    def post(self, request):
        return HttpResponse("Only for user service")
```

CHAPTER 3

Contents:

3.1 Instalation

Install package:

```
pip install djangohmac
```

3.1.1 Middleware

To secure all your app with HMAC you can use a middleware.

```
MIDDLEWARE_CLASSES = (
    # ...
    'djangohmac.middleware.HmacMiddleware',
)
```

Note: Middleware is applied on all views except the admin!

3.1.2 Decorators

You can specify views which are protected by HMAC by using decorators. You can also pass list of services which have access to the view. If the list is not given all services defined in settings have access.

```
class SignedView(View):

    @decorators.auth()
    def get(self, request):
        return HttpResponse("For all services")

    @decorators.auth(only=['serviceA'])
    def post(self, request):
        return HttpResponse("Only for service A")
```

3.1.3 Settings

Single key:

```
HMAC_SECRET = 'HMAC_SECRET'
```

Multiple keys:

```
HMAC_SECRETS = {
    'serviceA': 'HMAC_SERVICE_A_SECRET',
    'serviceB': 'HMAC_SERVICE_B_SECRET'
}
```

Other settings:

- HMAC_HEADER: HTTP header where signature is stored (Default: Signature)
- HMAC_DIGESTMOD: Digest mod (Default: hashlib.sha256)
- HMAC_DISABLE: Disable or enable HMAC True/False (Default: Enabled)

3.2 Examples

The signature is build from a secret key and a request body if exists.

3.2.1 Python

To send valid HMAC signature to a view you can use *shmac.make_hmac()*

```
from djangohmac.sign import shmac

sig = shmac.make_hmac()  # generate signature
response = requests.get(
    '/hmac_auth_view',
    headers={hmac.header: sig}
)
```

To generate signature for particular service:

```
sig = shmac.make_hmac_for('service', 'request body')
response = requests.get(
    '/hmac_auth_view',
    'request body',
    headers={hmac.header: sig}
)
```

3.2.2 HTTP

Valid signature is send:

Example request:

```
GET /api/v1/users HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Connection: keep-alive
Host: localdocker:8000
Signature: dXNlcnNlcnZpY2U6RDVyRm5TcnJUUTQyZUttcDIreWhXayttYzZPK0hjRHZjWWFwbW9MeFdjQT0=
User-Agent: HTTPie/0.9.2
```

Response response:

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Date: Thu, 15 Oct 2015 09:53:10 GMT
Server: WSGIServer/0.1 Python/2.7.10
Vary: Cookie
X-Frame-Options: SAMEORIGIN
```

Invalid signature is send:

Response request:

```
GET /api/v1/users HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Connection: keep-alive
Host: localdocker:8000
Signature: blabla
User-Agent: HTTPie/0.9.2
```

Response response:

```
HTTP/1.0 403 FORBIDDEN
Content-Type: text/html; charset=utf-8
Date: Thu, 15 Oct 2015 09:53:35 GMT
Server: WSGIServer/0.1 Python/2.7.10
Vary: Cookie
X-Frame-Options: SAMEORIGIN
```

3.3 Django Hmac

3.3.1 djangohmac package

Submodules

djangohmac.decorators module

djangohmac.decorators.**auth**(*func=None*, *only=None*)

Route decorator. Validates an incoming request can access the route function.

Keyword Args: only (list): Optional list of clients that can access the view

```
    class SignedView(View):

        @decorators.auth
```

```
        def get(self, request):
            return HttpResponse("For all services")

        @decorators.auth(only=['serviceA'])
        def post(self, request):
            return HttpResponse("Only for service A")
```

djangohmac.middleware module

class djangohmac.middleware.**HmacMiddleware**
Bases: object

Uses global signature *HMAC_SECRET* defined in settings

process_request(*request*)

djangohmac.sign module

class djangohmac.sign.**Hmac**
Bases: object

abort()
Called when validation failed.

Raises: PermissionDenied()

get_signature(*request*)
Get signature from djagno requests

Arguments: request: Django request

Returns: string: HMAC signature

Raises: SecretKeyIsNotSet

hmac_disarm

hmac_key

make_hmac(*data=''*, *key=None*)
Generates HMAC key

Arguments: data (str): HMAC message key (str): secret key of another app

make_hmac_for(*name*, *data=''*)
Generates HMAC key for named key

Arguments: name (str): key name from HMAC_SECRETS dict data (str): HMAC message

Raises: UnknownKeyName

validate_multiple_signatures(*key_name*, *signature*, *request*)
Validate signature from djagno request. But it takes key from *HMAC_SECRETS* list

Arguments: request (request): Django request class only (list): Restricted only for this list of service

Returns: boolen

Raises: InvalidSignature

validate_signature(*request*, *only=None*)
Validate signate in given request.

Arguments: request: Django request only: list of keys from HMAC_SECRETS to restrict signatures

Returns: boolean: True when signature is valid otherwice False

Raises: InvalidSignature SecretKeyIsNotSet

validate_single_signature(*request*)
Validate signature from djagno request

Arguments: request (request): Django request class

Returns: boolen

Raises: InvalidSignature

exception djangohmac.sign.**HmacException**
Bases: exceptions.Exception

exception djangohmac.sign.**InvalidSignature**
Bases: *djangohmac.sign.HmacException*

exception djangohmac.sign.**SecretKeyIsNotSet**
Bases: *djangohmac.sign.HmacException*

exception djangohmac.sign.**UnknownKeyName**
Bases: *djangohmac.sign.HmacException*

djangohmac.sign.**decode_string**(*value*)

djangohmac.sign.**encode_string**(*value*)

Module contents

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

d

A

D

E

G

H

I

M

P

S

U

V