django-pgcryptoauth Documentation

Release 0.2.2

Drew Engelson

March 27, 2016

Contents

Django hasher for PostgreSQL pgcrypto encoded passwords.

`django-pgcryptoauth` is a custom Django password hasher which is intended to provide authentication continuity for legacy passwords that were encrypted with the Postgres pgcrypto extension.

Since we don't have access to the cleartext passwords, we instead just make Django understand and handle the legacy algorithm. When a user successfully logs in, Django will automatically upgrade the password to the preferred algorithm.

CHAPTER 1

Source code

https://github.com/tomatohater/django-pgcryptoauth

CHAPTER 2

Dependencies

Of course, you will need to be using a PostgreSQL database with the pgcrypto extension installed.

CHAPTER 3

Installation

1. Install the `django-pgcryptoauth` package:

```
pip install django-pgcryptoauth
```

2. Add `pgcryptoauth` to your `INSTALLED_APPS`:

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

3. Add `pgcryptoauth.hashers.PgCryptoPasswordHasher` to `PASSWORD_HASHERS` in your Django settings:

```
PASSWORD_HASHERS = (
    ...
    'pgcryptoauth.hashers.PgCryptoPasswordHasher',
)
```

Note: This hasher should probably at the bottom of the list so that other hashers take priority. See https://docs.djangoproject.com/en/1.4/topics/auth/#how-django-stores-passwords

CHAPTER 4

Configuration

By default, *pgcryptoauth* will use your *default* database connection. However, you may instruct it to use another connection by setting *PGCRYPTOAUTH_DATABASE* to something else in your Django settings.:

```
PGCRYPTOAUTH_DATABASE = 'another_database'
```

Of course, this other connection must be a valid Postgres database with the pgcrypto extension and listed in your *DATABASES* setting:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'my_database',
        'USER': 'my_user',
        'PASSWORD': 'my_pass',
        'HOST': '127.0.0.1',
        'PORT': '',
    },
    'another_database': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'another_database',
        'USER': 'another_user',
        'PASSWORD': 'another_pass',
        'HOST': '127.0.0.1',
        'PORT': '',
    }
}
```

This may be necessary if the pgcrypto extension is not (or can't be) installed on your primary database. Especially if your primary database is not PostgreSQL!

CHAPTER 5

Running test cases

Execute the unit test:

```
python manage.py test pgcryptoauth
```

CHAPTER 6

Loading legacy data

Note: Legacy pgcrypto hashed passwords look like `$1$BFw5nhna$XeiE8c4FInYGp3oND2l9n1`. When migrating these passwords, we simply need to prefix the hash with the `pgcrypto$` algorithm:

```
user.password = 'pgcrypto$$1$BFw5nhna$XeiE8c4FInYGp3oND2l9n1'
user.save()
```

If you review that users password via the Django `auth.user` admin, you should see:

```
algorithm: pgcrypto
hash: $1$BFw****************************************
```