How to Protect Sensitive Data in Django?

O Django já dispões de diversas seguranças dentro dele próprio, entretanto não adianta em nada se não temos a intenção de usá-lo.

Muitas coisas são armazenadas no arquivo settings.py. Nós podemos protege-la usando a biblioteca Python Decouple Library, elas separa parâmetros do sorce code. Para isso, criaremos um arquivo (.env ou .ini) e guardar os parâmetros dentro dele.

Por que fazer isso?

Suponhamos que coloquemos o código no git (público), qualquer pessoa poderá entrar no nosso Django Settings e ver dados sensíveis que deveriam ser secretos. Isso pode ser usado como ataque para os desenvolvedores. Para evitar essa situação não é recomendado dar commit, em vez disso deixar em um arquivo local dentro do .gitignore.

Para instalar a biblioteca Python Decouple Library:

```
pip install python-decouple
```

Como usar a biblioteca para uma aplicação Django?

Primero é necessário criar um arquivo na pasta principal chamada .env. Dentro desse arquivo colocamos os dados importantes (SECRET_KEY, DEBUG, DB_NAME, DB_USER. DB_PASSWORD, DB_HOST) como se fossem variáveis, como mostra abaixo.

```
SECRET_KEY= 6hv(7)$yiu&fzo#qb&*s-=$u14p5emb+ycamu7l@i1c!^hos!
DEBUG=True
DB_NAME= MY_DJANGO
DB_USER=My_Django
DB_PASSWORD=256hv(7)$y
DB_HOST=127.0.0.1
```

Agora no arquivo settings.py é necessário importar a biblioteca:

```
from decouple import config
```

Depois disso substituir no settings.py os dados pelas nomes das variáveis do arquivo .env. Para isso é basta colocar "config('VARIAVEL')"

```
import os

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
SECRET_KEY = config('SECRET_KEY')
DEBUG = config('DEBUG', cast=bool)
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': config('DB_NAME'),
        'USER': config('DB_USER'),
        'PASSWORD': config('DB_PASSWORD'),
        'HOST': config('DB_HOST'),
        'PORT': '80',
    }
}
```

Depois desta etapa colocar o arquivo .env no gitignore para não subir para o repositório.

Casting Data

Exemplo: o DEBUG espera um valor Boolean e o EMAIL_PORT um inteiro. Para isso é preciso colocar o cast na mesma estrutura.

```
DEBUG = config('DEBUG', cast=bool)
EMAIL_PORT = config('EMAIL_PORT', cast=int)
```

Sobre o ALLOWED_HOST:

É possível colocá-lo dessa forma:

```
ALLOWED_HOSTS=.localhost, .herokuapp.com
```

Para acessar o ALLOWED_HOST podemos usar o CSV helper (biblioteca que ajuda na leitura) assim:

```
from decouple import config, Csv

ALLOWED_HOSTS = config('ALLOWED_HOSTS', cast=Csv())
```

Django Default Values

Podemos adicionar mais um argumento para a função config, o valor default.

```
DEBUG = config('DEBUG', default=True, cast=bool)
```

Fonte:

How to Protect Sensitive Data in Django - Python Decouple - Django

Django database password security. django decouple . The Internet is full of data, which is used for legal and illegal purposes. As a developer, we need to keep our system protect and secure. Securing applications will help our system

https://studygyaan.com/django/how-to-protect-sensitive-data-in-django