Django 渗透测试与代码安全漫谈（一）

PHP的说了很多了，是时候谈谈Django了，总结一下Django中可能存在的一些安全问题，以及如何去挖掘。废话就不多说了。

简单介绍Djnago

Django是企业用的最多的Python开发框架，大家在日常工作中应该也或多或少遇到过。我们可以直接使用`pip install django`来安装最新版本的Djnago。

一个完整的Django项目是由一个个的app组成了。最常见的app就是django自带的"Django admin"了，你们平时看到的Django后台登录页面就是这个app输出的。我们基于Django做开发，也是写一个或数个app。这些app包括django自带的一些app最终组成了一个完整的项目。

使用`django-admin startproject djtest`来创建一个新的项目，然后进入项目目录djtest，再创建一个app：`python3 ./manage.py startapp djapp`。

```
# shiyu @ shiyudeMBP in /tmp [23:54:26]
$ django-admin startproject djtest

# shiyu @ shiyudeMBP in /tmp [23:54:35]
$ cd djtest

# shiyu @ shiyudeMBP in /tmp/djtest [23:54:41]
$ python3 manage.py startapp djapp

# shiyu @ shiyudeMBP in /tmp/djtest [23:54:50]
$ tree .
.
├── djapp
|   ├── __init__.py
|   ├── admin.py
|   ├── apps.py
|   ├── migrations
|   |   └── __init__.py
|   ├── models.py
|   ├── tests.py
|   └── views.py
├── djtest
|   ├── __init__.py
|   ├── __pycache__
|   |   ├── __init__.cpython-35.pyc
|   |   └── settings.cpython-35.pyc
|   ├── settings.py
|   ├── urls.py
|   └── wsgi.py
└── manage.py

4 directories, 14 files
```

如上图，djtest目录是项目文件，其中包含配置文件、路由文件等。djapp是我创建的一个app，其中包含model、view等文件。

有的古老的教程上经常用django-admin来创建所有东西，道理也一样。如果app太多的话，有的开发会把所有app放在一个目录中，比如这样：

```
.
├── app
```

```
│   ├── front
│   ├── manager
│   └── ucenter
├── manage.py
├── requirements.txt
├── djtest
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── templates
```

上述结构有三个app，分别是front、manager、和ucenter。模板文件理论上可以放在任何地方，我通常就放在主目录下面的templates文件夹中。

适当了解一些目录结构，以后遇到文件下载、文件读取、文件上传覆盖之类的漏洞就容易多了。

Django有两种常见的运行方法，第一种是简单的执行`python3 manage.py runserver`就可以启动一个其内置的服务器，默认监听8000端口；第二种是使用gunicorn之类的uwsgi服务器，对接djtest/wsgi.py 文件中的application变量，通常在生产环境使用。这两种方法都可能在外层套一个nginx中间件进行流量转发。

Django官方推荐使用Postgres作为其数据库，也支持mysql、sqlite、sqlserver等数据库。但假如你什么也没配置，Django默认是使用sqlite数据库，数据库文件在主目录下，默认为`db.sqlite3`。

Django内置的缓存方法有内存、数据库、文件或memcached，并没有redis，通常使用memcached的比较多。但通过安装第三方模块，也可以使用redis做缓存。

Django如果配合Celery（异步任务模块）使用，通常在djtest目录下都会创建一个celery.py，在其中创建celeryapp。而要监控Celery的话，通常配合一个开源web应用，叫flower。Celery通常和RabbitMQ或redis配合使用。

这些应用的统一管理，通常会使用docker或supervisor，supervisor也自带一个web管理页面。

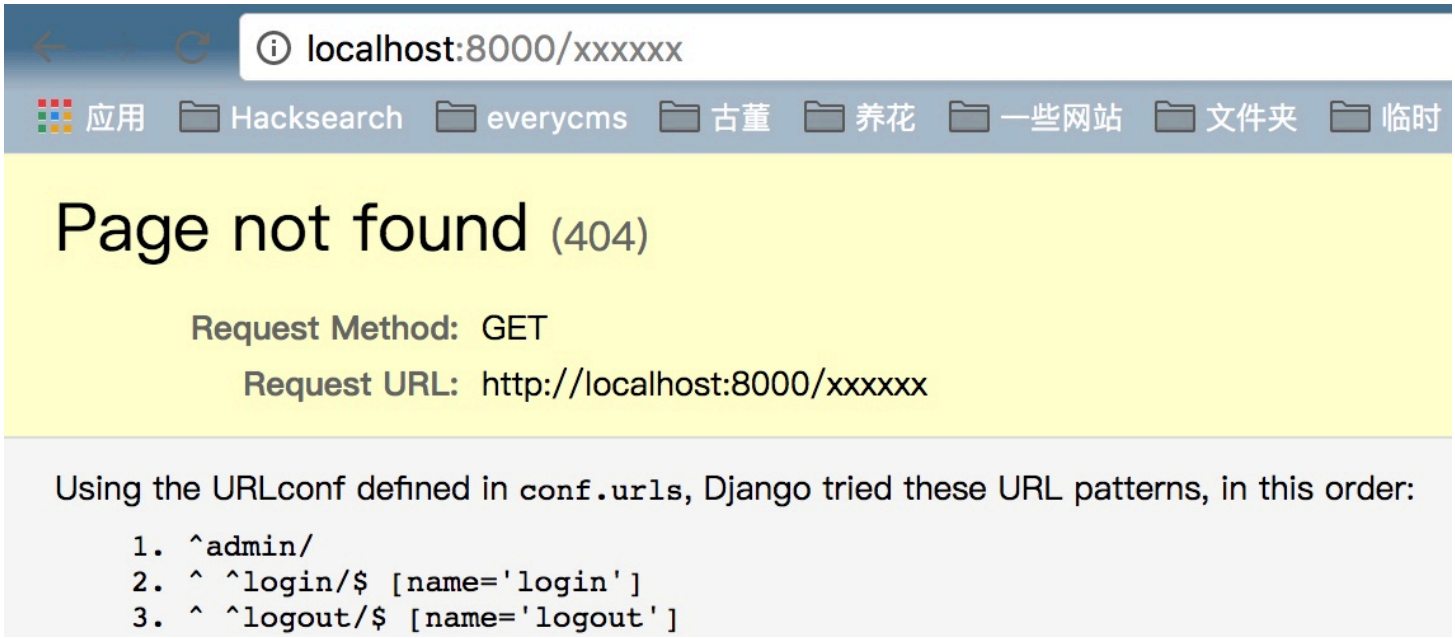
我上述列举了一些Django技术栈可能会涉及到的应用和概念，了解这些概念，对于挖掘Django的系统有很大帮助，不管是黑盒还是白盒。

判断一个站是否是Django

在黑盒测试的情况下，如何判断一个站是否是Django开发的？以下这些方法，很多都能在我的博客（https://www.leavesongs.com）得到印证。

利用Debug模式异常页面判断

DEBUG模式开启时，访问不存在的页面或出错的页面会有特殊的异常抛出。像这样的页面，就可以确定是Django



CSRF Token名称确认

访问一个包含表单的页面，表单中会有一个隐藏的input，用来做CSRF检测的Token，其名字比较独特，csrfmiddlewaretoken：

```
    <hr>
    <div class="form-group pull-right">
      <div class="col-md-12">
        <button type="button" data-toggle="modal" data-target="#previewModel"
                class="btn btn-success btn-sm">预览</button>
        <button type="submit" class="btn btn-primary btn-sm">提交审核</button>
        <input type='hidden' name='csrfmiddlewaretoken' value='3t4kcrgxAODFvkXY8WrqGjecgoNWc9dJVQMQzYiOt5ZWsBjFQnfuoWHRffbMztMr' />
      </div>
    </div>
    <div class="clearfix"></div>
  </form>
  </div>
</div>
```

遇到有这个名字的表单，基本可以确定是Django。

假如开发者将`csrfmiddlewaretoken`这个名字换了，怎么办?

随便向目标的某个页面POST一个数据包，因为缺少CSRF TOKEN，如果目标网站是Django，它将给你一个颇具其特色的错误页面：

利用后台确认

Django默认安装后会自带一个后台，地址是/admin：

遇到这个样式的后台界面，可以确定是Django。

利用HTTP头

有的Django站点会返回Server头：

虽然不能100%确定是Djnago，但范围就缩的很小了。

通过一些细节判断

有些细节虽然不能100%确定是django，但多个细节组成在一起就可以基本确定了。

比如，Django输出的html中通常会有很多空白行，因为这些位置放的是逻辑语句，Django不像jinja2中会提供 `{%-` 这样清除空白行的方法：

```
<!DOCTYPE html>

<html lang="zh-hans" >
<head>
<title>登录 | Django 站点管理员</title>
<link rel="stylesheet" type="text/css" href="/static/admin/css/base.css" />
<link rel="stylesheet" type="text/css" href="/static/admin/css/login.css" />




<meta name="csrfmiddlewaretoken" content="87iJht4LlEHLPiUVyVVC3KnyKKfFsJlEgJOqiak0RlWAXbNUncRSGQE6HbtFq6MN">
<script src="/static/js/admin.js" type="application/javascript"></script>

<meta name="robots" content="NONE,NOARCHIVE" />
</head>

<body class=" login"
  data-admin-utc-offset="28800">

<!-- Container -->
<div id="container">


    <!-- Header -->
    <div id="header">
        <div id="branding">

<h1 id="site-name"><a href="/admin/">Django 管理</a></h1>

        </div>


    </div>
    <!-- END Header -->




    <!-- Content -->
    <div id="content" class="colM">
```

再比如，Django默认找回密码的链接是 `/password_reset/` ，邮件发送成功是 `/password_reset/done/` ，找回密码链接是 `reset/(?P<uidb64>[0-9A-Za-z_\-]+)/(?`

`P<token>[0-9A-Za-z]{1,13}-[0-9A-Za-z]{1,20})/`，找回密码成功是`/reset/done/`，正常修改密码是`/password_change/`，修改成功是`/password_change/done/`。

不过这些链接都可以改，只能用作参考。

再比如，django文件上传的目录通常叫`media`，注册时密码要求8位以上数字加字母，分页通常是`?page=2`而不会是`/page/2/`，表单输入框的id通常是`id_xxxx`，中文的情况下还会有一些特定的翻译语句，如`请上传一张有效的图片。您所上传的文件不是图片或者是已损坏的图片。`、`CSRF验证失败. 相应中断.`等。

通过一些第三方模块的特点判断

Django之所以好用，因为其代码耦合性很低，所以有丰富的第三方模块可以直接使用。通过这些模块的特点也能判断目标网站是否是Django。

常用的第三方模块有django-rest-framework、django-debug-toolbar、django-bootstrap3、django-filter、django-cron、django-allauth、django-simple-captcha等。

比如，django-rest-framework默认包含一个登陆页面，`/api-auth/login/`：

再比如，django-simple-captcha生成的验证码会包含一个名字是`captcha_0`，值为40位hex的隐藏输入框。

这些第三方库的特点也可以辅助你判断，就是需要收集与细心观察了。

必杀技：用静态文件分析

有的网站可能修改了Django的后台地址，但Django后台所使用的静态文件地址通常没有修改，也较难修改。访问这些静态文件地址，看看内容是否是Django的这一套，就可以确定目标是否为Django：

如 https://www.leavesongs.com/static/admin/css/dashboard.css 、 http://www.wuzheng.org/static/admin/css/dashboard.css 、 http://static.fuwo.com/static/admin/css/dashboard.css 、 http://www.lintcode.com/static/admin/css/dashboard.css


```
/* DASHBOARD */

.dashboard .module table th {
    width: 100%;
}

.dashboard .module table td {
    white-space: nowrap;
}

.dashboard .module table td a {
    display: block;
    padding-right: .6em;
}

/* RECENT ACTIONS MODULE */

.module ul.actionlist {
    margin-left: 0;
}

ul.actionlist li {
    list-style-type: none;
}

ul.actionlist li {
    overflow: hidden;
    text-overflow: ellipsis;
    -o-text-overflow: ellipsis;
}
```

我就不罗列了。

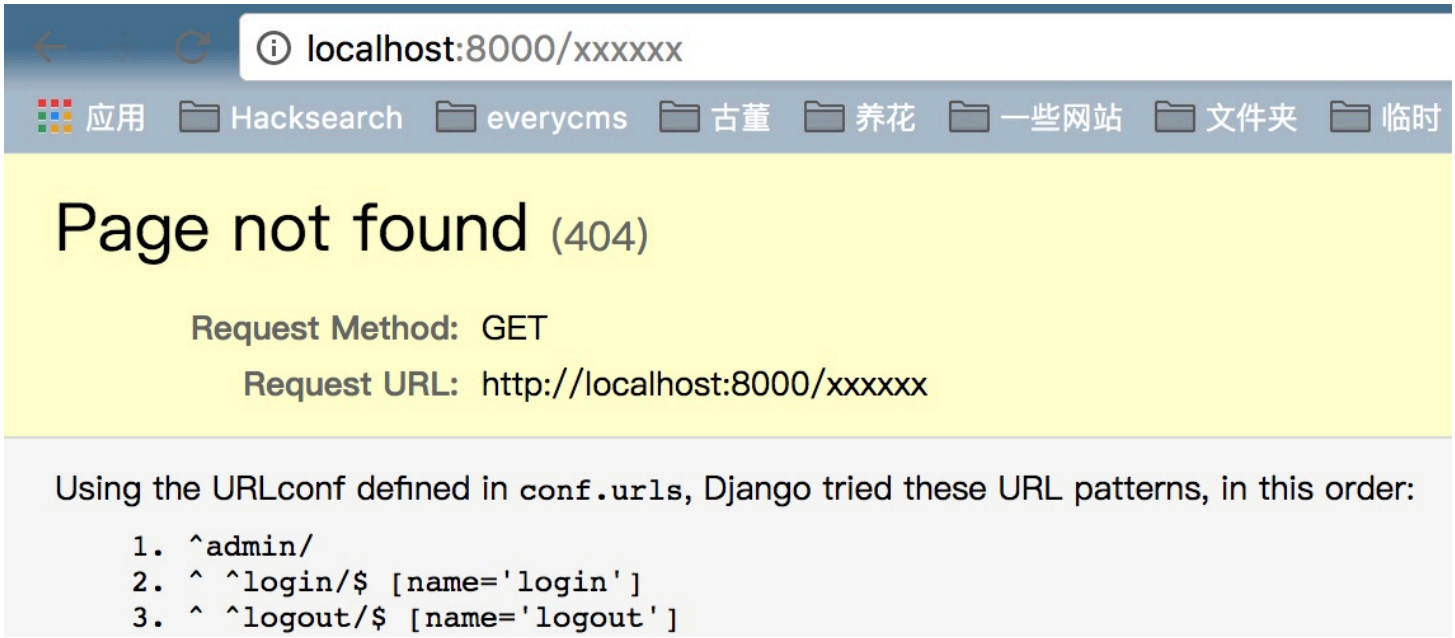
但这个方法有个局限，如果目标网站没有使用Django自带的django-admin（未将其包含在settings.py的INSTALLED_APPS中），就没有这个静态文件了。

DEBUG模式

最后说一下DEBUG模式。Django默认创建的项目是开启了DEBUG开关的，也就是在settings.py中设置了 `DEBUG = True` 。

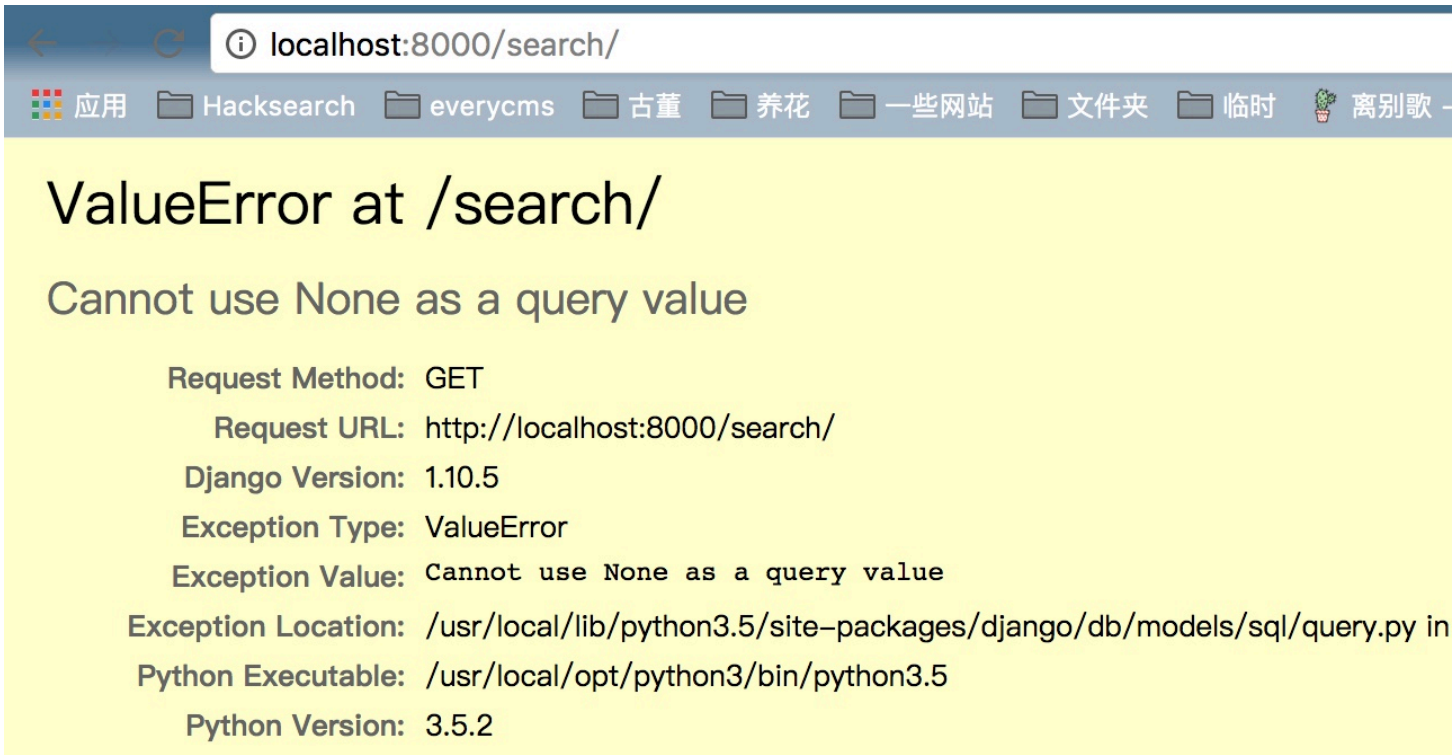
在开启DEBUG模式后，访问出现错误的时候会抛出异常，常见的异常有两种，一是404，二是500。

在用户访问了不存在的url时会抛出404异常，前面已经说过了：

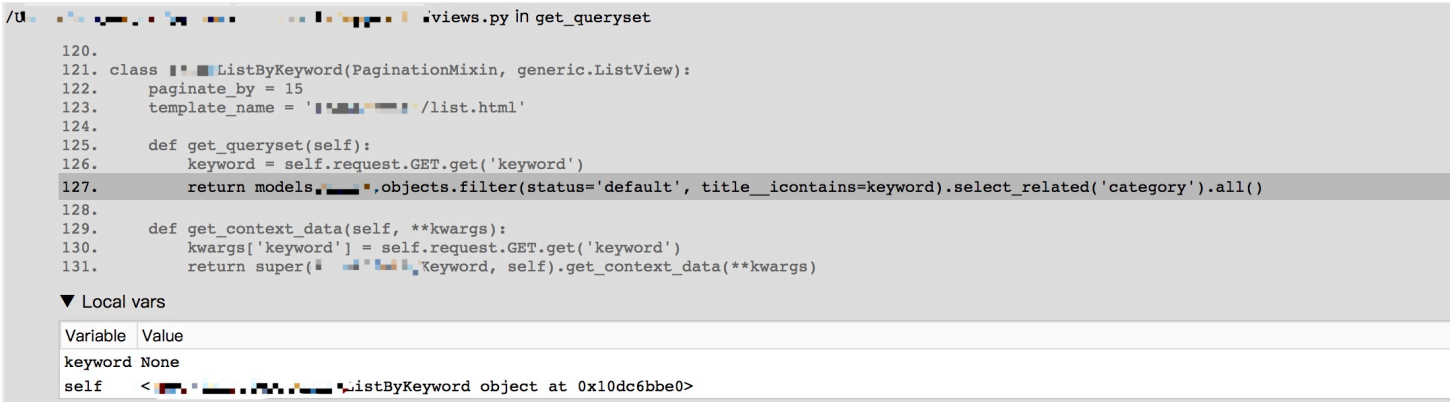


通过这个异常信息，可以找到很多没有暴露出来的url。

500页面是Python抛出异常但没有捕捉到的时候显示的，Python的异常大家应该领教过，很容易出现异常。比如，正常搜索url是`http://localhost:8000/search/?keyword=xxx`，我将`keyword=`去掉，就可能导致抛出异常：



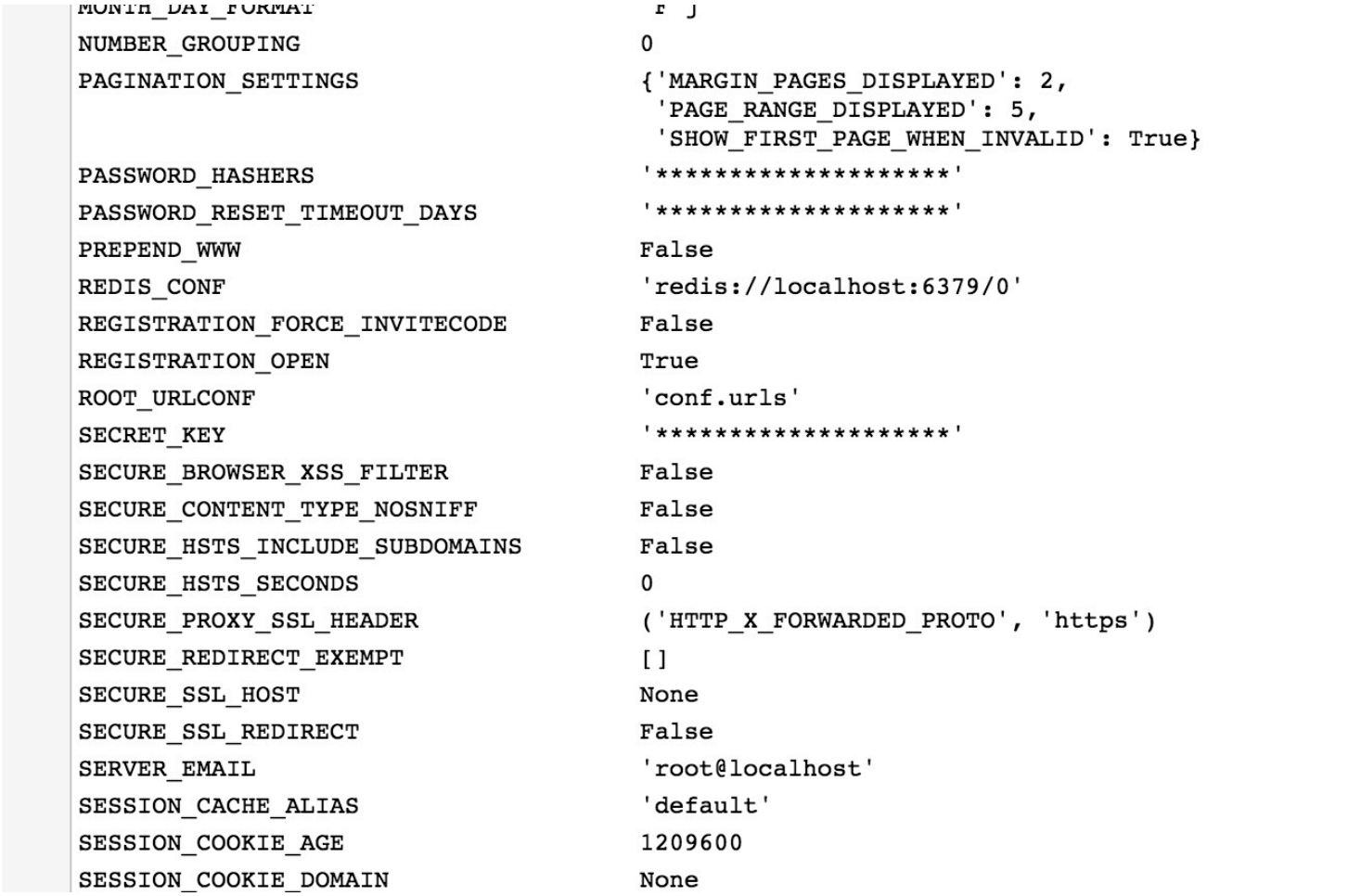
异常信息中将包含一些敏感信息，如少部分代码（可以尝试挖掘其他漏洞）：



如上图，你就能看出当前View执行了哪些东西。

假如出错的view或回溯到的view里刚好有的变量中包含敏感信息，如password，那你就赚大了：

老版本的DEBUG中甚至包含一些敏感信息，如SECRET_KEY，但新版本全部做了星号替换：



可以看到，在Django源码（ django/views/debug.py ）中，有一个简单的正则：

```
HIDDEN_SETTINGS = re.compile('API|TOKEN|KEY|SECRET|PASS|SIGNATURE', flags=re
.IGNORECASE)

CLEANSED_SUBSTITUTE = '********************'
```

所以，只要名称中包含API、TOKEN、KEY、SECRET、PASS、SIGNATURE的配置项都会被替换成一堆星号。

不过一些数据库配置不会包含这些关键字，比如使用了这个库后 https://github.com/kennethreitz/dj-database-url ，Mysql配置就可以写为`DATABASE_URL = 'mysql://root:password@localhost:3306/db'`，这样，Mysql的密码就暴露了。

（未完待续）