【攻防世界】十七 --- shrine

原创

通地塔 于 2020-12-28 23:06:53 发布 164 收藏 3

分类专栏： 攻防世界 文章标签： ctf 网络安全

本文链接：https://blog.csdn.net/qq_43168364/article/details/111873910

题目 — shrine

一、writeup

打开主页，映入眼帘的一段python代码，审计看看

```python
import flask
import os

# app是Flask的实例，它接收 包 或者 模块 的名字作为参数，但一般都是传递__name__
# __name__ --- 是本 文本文件的名字
app = flask.Flask(__name__)

# 自定义配置
app.config['FLAG'] = os.environ.pop('FLAG')

# __file__ --- 表示显示文件当前的位置
@app.route('/')
def index():
    return open(__file__).read()


@app.route('/shrine/<path:shrine>')
def shrine(shrine):
    # 用来过滤的函数
    def safe_jinja(s):

     # 替换掉所有的圆括号
        s = s.replace('(', '').replace(')', '')

        # 一个黑名单
        blacklist = ['config', 'self']

        # 返回 {{set config=None}} {{set self=None}} 用户输入
        # 过滤了 config 和 self
        return ''.join(['{{% set {}=None%}}'.format(c) for c in blacklist]) + s

    # 进行渲染
    return flask.render_template_string(safe_jinja(shrine))


if __name__ == '__main__':
    app.run(debug=True)
```

关键点：

- `/shrine/<path:shrine>` 路径存在ssti，但是进行了过滤
- 过滤了 () ---- 括号，那就无法使用`os模块`执行命令，`<type 'file'>`也无法使用。感觉通过命令执行来看flag这条路被堵死了
- config 和 self 无法使用。如果这道题没有过滤config 和 self 的话，可以执行：`{{config.FLAG}}` 和 `{{self.__dict__}}`

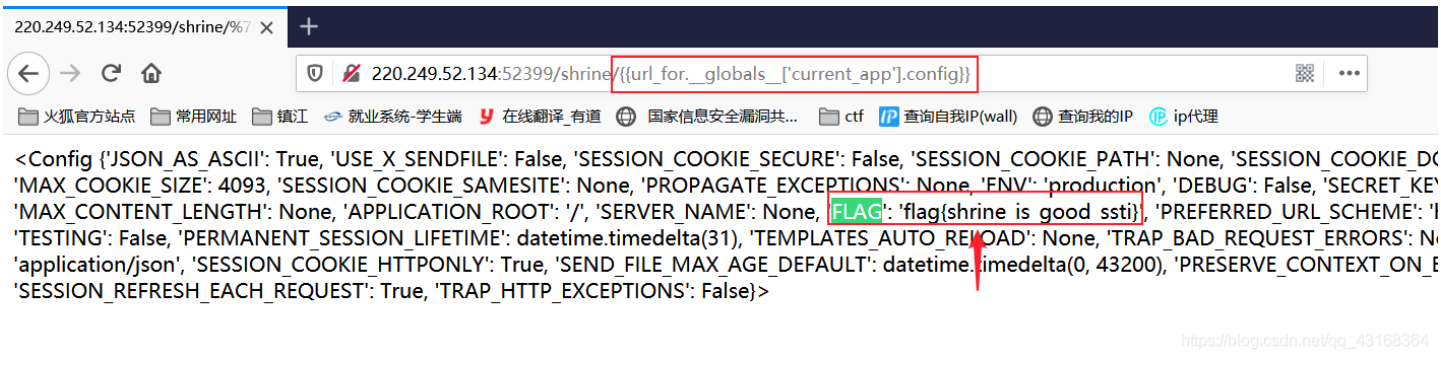
当前页面是存在SSTI的

这里绕过的过滤的方法是使用 **url_for()** 和 **get_flashed_messages()** 函数

- **url_for()** ---- 一般我们通过一个**url**即可执行到一个**函数**，如果知道一个函数，如何去获得url呢？url_for函数可以实现这个功能。url_for() 接收两个及两个以上的参数，以函数名作为第一个参数，后面的参数是url的命名规则
- **get_flashed_messages()** ---- 返回之前在Flask中通过 **flash()** 传入的闪现信息列表在渲染模板时，不需要手动分配
- 可以直接在 模板 中使用的 模板变量 及 函数：**config**、**request**、**url_for()**、**get_flashed_messages()**

利用**url_for**函数查看全局变量字典，发现了**Flask**：`/shrine/{{url_for.__globals__}}` — **'current_app': <Flask 'app'>**

payload 如下即可得到flag：`/shrine/{{url_for.__globals__['current_app'].config}}` ---- **'FLAG': 'flag{shrine_is_good_ssti}'**



二、知识点

- 在Flask中，有一些**特殊的变量**和**方法**是可以在模板文件中直接访问的。