

XCTF系列 // Web ez题 Writeup

原创

[Ga1axy_z](#) 于 2020-07-11 13:04:32 发布 254 收藏

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/qq_45805420/article/details/107208232

版权



[CTF 专栏收录该内容](#)

13 篇文章 1 订阅

订阅专栏

1` robots

robots 协议也叫 robots.txt 是一种存放于网站根目录下的ASCII编码的文本文件, 它用来告知搜索引擎 (或网络爬虫) 哪些内容能被抓取, 哪些内容不能被抓取。因为一些系统中的URL是大小写敏感的, 所以 robots.txt 的文件名应统一为小写。

进入本题场景, 网页中什么内容也没有, 不过没有关系, 既然题目为 robots, 故可以先访问 robots.txt 看看。

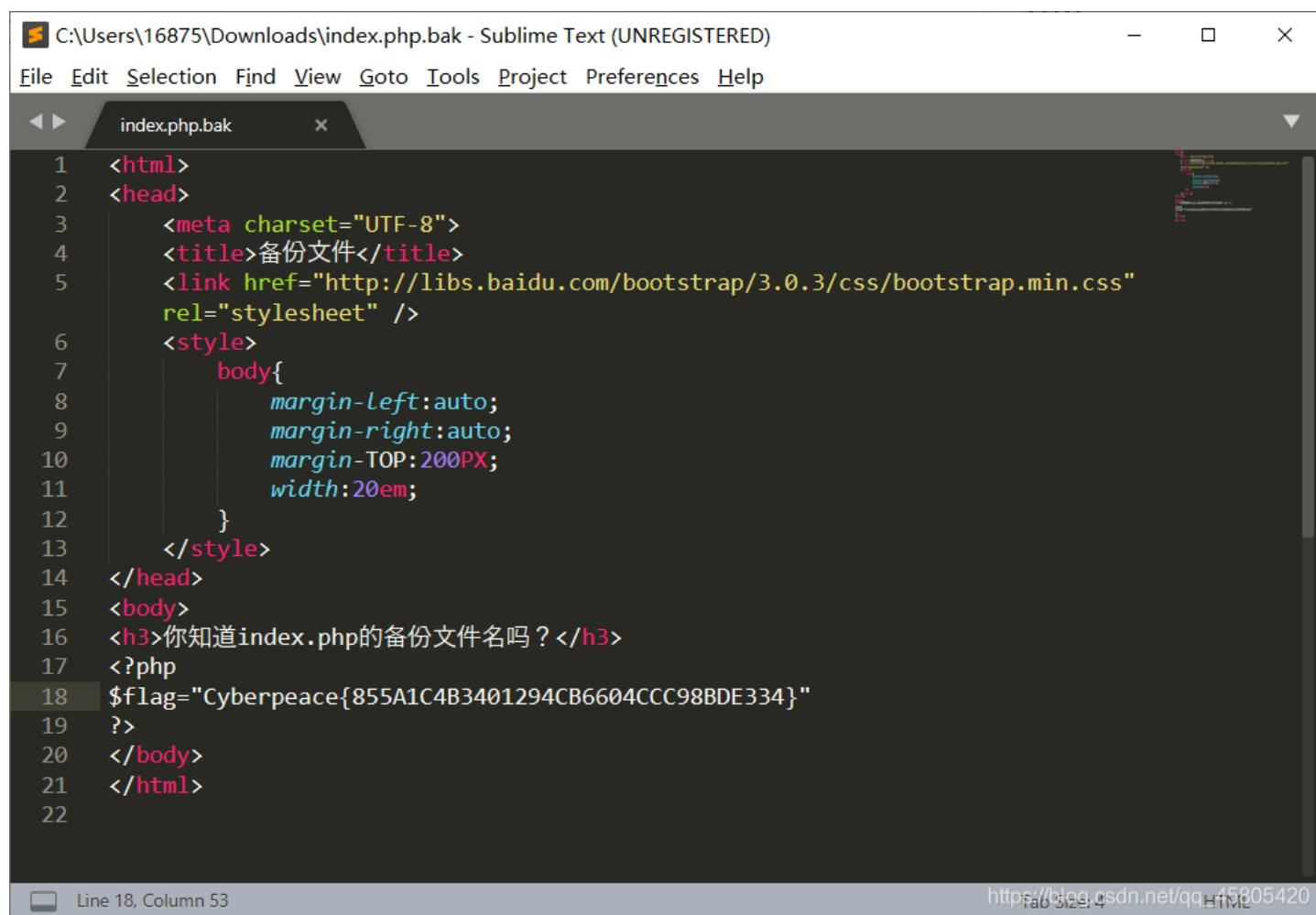
← → ↻ ⓘ 不安全 | 220.249.52.133:45694/robots.txt

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

在该页面可以看到明显的提示, 故直接访问 `flag_1s_h3re.php` 页面就可以得到flag。

2` backup

本题题目描述中给出了“备份文件”这个信息，由此想到相关后缀名应为.bak。既然题目要求访问 index.php 的备份文件，故访问 index.php.bak 浏览器即可下载得到该文件。用Sublime打开该文件，即可得到flag。



```
C:\Users\16875\Downloads\index.php.bak - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

index.php.bak
1 <html>
2 <head>
3   <meta charset="UTF-8">
4   <title>备份文件</title>
5   <link href="http://libs.baidu.com/bootstrap/3.0.3/css/bootstrap.min.css"
6     rel="stylesheet" />
7   <style>
8     body{
9       margin-left:auto;
10      margin-right:auto;
11      margin-top:200PX;
12      width:20em;
13    }
14  </style>
15 </head>
16 <body>
17 <h3>你知道index.php的备份文件名吗？</h3>
18 <?php
19 $flag="Cyberpeace{855A1C4B3401294CB6604CCC98BDE334}"
20 ?>
21 </body>
22 </html>
```

Line 18, Column 53 https://blog.csdn.net/qq_45805420

3` weak_auth

打开本题网页，发现需要登录，在该页面没有发现什么线索，所以先随便输入一个账号密码试试。

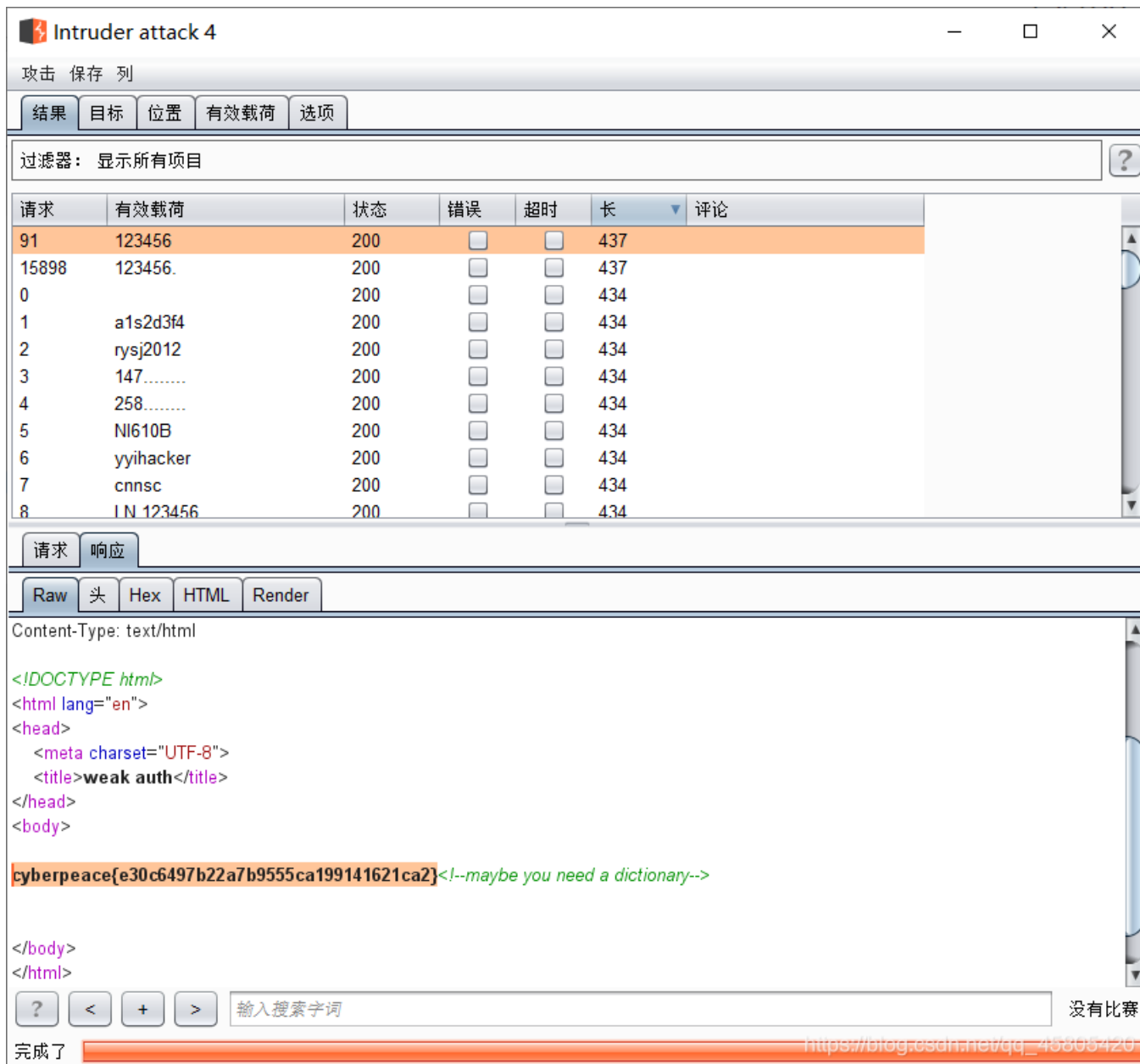


可以看到，网站弹出提示，要求我们使用 admin 的用户名来登录，同时我们可以查看该页面的源码，发现提示，建议使用字典来爆破密码。

```
<html lang="en">
  <head>...</head>
  <body>
    <script>alert('please login as admin');</script>
    <!--maybe you need a dictionary--> == $0
    <div class="xl-chrome-ext-bar" id="xl_chrome_ext_{4DB361DE-01F7-4376-B494-639E489D19ED}" style="display: none;">...</div>
  </body>
```

```
</body>
</html>
```

因此，使用 Burpsuite 爆破密码即可得到 flag。（这题使用 Burpsuite 自带的字典不太好用。。。在这里贴一个还算比较全的字典吧：https://github.com/rootphantomer/Blasting_dictionary）



The screenshot shows the Burp Suite interface for an intruder attack. The top bar indicates the attack is complete. Below the filter bar, a table lists the requests. The first request (ID 91) is highlighted, showing a status of 200 and a length of 437. The response content is displayed below the table, showing the HTML structure of the page. The response content is as follows:

```
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>weak auth</title>
</head>
<body>

cyberpeace{e30c6497b22a7b9555ca199141621ca2}<!--maybe you need a dictionary-->

</body>
</html>
```

The response content is displayed in the 'Render' view. The status bar at the bottom indicates '完成了' (Completed) and provides a URL: https://blog.csdn.net/qq_43868429.

4` simple_php

本题浏览源码可知，考察点在于 PHP 的弱类型比较。故按照题目要求，构造 $a=a$ ， $b=1235a$ 即可得到 flag。

在这里总结记录一下碰到过的 PHP 弱类型比较。

首先介绍一下相关概念，HTML 表单并不传递整数、浮点数或者布尔值，它们只传递字符串。要想检测一个字符串是不是数字，可以使用 `is_numeric()` 函数。在 PHP 中有两种比较是否相等的符号 `==` 与 `===`。

等于 `==`，当等号两边为相同类型时，直接比较值是否相等，当等号两边类型不同时，先转换为相同的类型，再对转换后的值进行比较。

全等 `===`，在进行比较的时候，首先判断等号两边的类型是否相等，如果不同，则直接返回 `false`，如果相同，再比较值是否相等。

举一个简单的例子，变量 `a` 为数字 `1`，变量 `b` 为字符 `'1'`，在 PHP 中，`$a == $b` 会返回 `true`，而 `$a === $b` 则会返回 `false`。

当字符串与数字比较时，首先将字符串转换为数字，不能转换为数字的字符串或 `NULL`，被转换为 `0`。例如，“`abc`”是不能转换为数字的字符串，而“`123`”、“`123a`”、“`0x12`”或“`2e3`”是可以转换为数字的字符串。PHP 手册中相关描述如下：

如果比较一个数字和字符串或者比较涉及到数字内容的字符串，则字符串会被转换为数值并且比较按照数值来进行。

一个字符串的开始部分决定了它转化成数值后的值，如果该字符串以合法的数值开始，则使用该数值，否则其值为 `0`。

在进行比较运算时，如果遇到了“`0e\d+`”这种字符串，PHP 就会将这种字符串解析为科学计数法。

例子	字符串转化后的数值	结果
<code>"admin"==0</code>	0	true
<code>"1admin"==1</code>	1	true
<code>"admin1"==1</code>	0	false
<code>"0e123456"=="0e456789"</code>	0，字符串解析为科学计数法	true
<code>"0e123456a"=="0e456789a"</code>	字符串与字符串类型直接比较，无转换	false
<code>"1e3b31"==1000</code>	1000	true
<code>"2e3"==2000</code>	2000	true
<code>"2E3"==2</code>	2000	false
<code>"0x12"==0</code>	0	true
<code>0x12==18</code>	—	true

MD5绕过就是以上知识的一种实际应用，举个例子，假如我们需要找到两个字符串，它们不相等，但是md5的值却弱相等，这种字符串要怎么找呢，一种方法是我们只需要找到对应md5的值都以0e开头后接数字的两个字符串即可。

例如：

```
md5("s878926199a")==md5("s155964671a")为 true。
md5("s878926199a")为 "0e545993274517709034328855841020";
md5("s155964671a")为 "0e342768416822451524974117254469";
两个字符串均为"0e\d+"格式，均被解析为科学计数法，数值都等于0。
```

其它比较运算与等于类似，在这里附上 PHP 手册中相关内容的地址：

PHP：PHP 类型比较表

PHP：比较运算符

5` xff_referer

根据题目信息可知，本题与 *X-Forwarded-For* 和 *Referer* 有关。进入网页，发现题目要求登录ip地址必须为123.123.123.123，于是使用 Burpsuite 进行抓包，在请求头添加一项 *X-Forwarded-For:123.123.123.123* 后重发，发现如下回显：



```
    }
  </style>
</head>
<body>
<p id="demo">ip地址必须为123.123.123</p>
<script>document.getElementById("demo").innerHTML="必须来自https://www.google.com";
</script></body>
</html>
```

既然题目要求请求必须来自谷歌，于是再在请求头中添加一项 *referer:https://www.google.com* 后重发即可，在网页响应中即可看到 flag。

X-Forwarded-For:

简称XFF头，它代表客户端，也就是 HTTP 的请求端真实的IP，换句话说，XFF头就是告诉服务器当前请求者的IP地址的 HTTP 请求头字段。它的标准格式为 *X-Forwarded-For: client1, proxy1, proxy2*，从标准格式可以看出，XFF头信息可以有多个，中间用逗号分隔，第一项为真实的客户端ip，剩下的就是曾经经过的代理或负载均衡的ip地址，经过几个就会出现几个。

Referer:

HTTP Referer是 header 的一部分，当浏览器向 WEB 服务器发送请求的时候，一般会带上 Referer，告诉服务器该网页是从哪个页面链接过来的，服务器因此可以获得一些信息用于处理。

6` command_execution

本题的考察点为命令执行漏洞。当应用需要调用一些外部程序去处理内容的情况下，就会用到一些执行系统命令的函数。如 PHP 中的 `system`、`exec`、`shell_exec` 等，当用户可以控制命令执行函数中的参数时，将可以注入恶意系统命令到正常命令中，造成命令执行攻击。

进入本题网页，首先试试输入 127.0.0.1 来观察一下回显。

PING

127.0.0.1

PING

```
ping -c 3 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.048 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.041 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.053 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.041/0.047/0.053/0.007 ms
```

https://blog.csdn.net/qq_45805420

这里涉及到另外一个知识点，即如何从 PING 命令的回显来判断操作系统的类型。一般来说，我们可以通过查看 TTL 值从而判断操作系统类型。确定目标系统后，即可使用对应系统的相关系统命令来寻找 flag。

- TTL=128，这是 *WINNT/2KXP*。
- TTL=32，这是 *WIN95/98/ME*。
- TTL=256，这是 *UNIX*。
- TTL=64，这是 *LINUX*。

本题目标系统为 LINUX，因此我们可以使用 LINUX 的查找命令以 flag 为关键词试试搜索相关文件。

```
find / -name "flag*"
```

关于 **LINUX** 的常用系统命令可参考以下博客：[Linux系统常用基本命令总结](#)

关于 **WINDOWS** 的常用系统命令可参考以下博客：[windows系统常用命令](#)

PING

```
127.0.0.1 && find / -name "flag*"
```

PING

```
ping -c 3 127.0.0.1 && find / -name "flag*"
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.086 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.059 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.056 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.056/0.067/0.086/0.013 ms
/home/flag.txt
/proc/sys/kernel/sched_domain/cpu0/domain0/flags
/proc/sys/kernel/sched_domain/cpu0/domain1/flags
```

可以发现题目没有对用户输入的参数进行过滤，在回显中我们可以明显的看到 flag 文件的路径，接下来使用 cat 命令显示该文件内容即可得到 flag。

```
cat /home/flag.txt
```

本题题目把用户输入的参数不经过过滤直接放在 PING 命令之后，因此攻击者可以在输入的参数后用连接符截断再加上新的系统命令，进而执行该新命令。常见的 **LINUX** 命令间的连接符如下：

连接符	举例	效果
;	A;B	按顺序依次执行，先执行命令A，再执行命令B
	A B	管道符，命令A执行成功后，命令A的输出会交给命令B继续处理。若命令A执行失败，则会报错，若命令B无法处理命令A的输出，也会报错。
&&	A&&B	逻辑与关系，命令A执行成功后，才会执行命令B；命令A执行失败，则命令B不被执行。
	A B	逻辑或关系，命令A执行失败后，才会执行命令B；命令A执行成功，则命令B不被执行