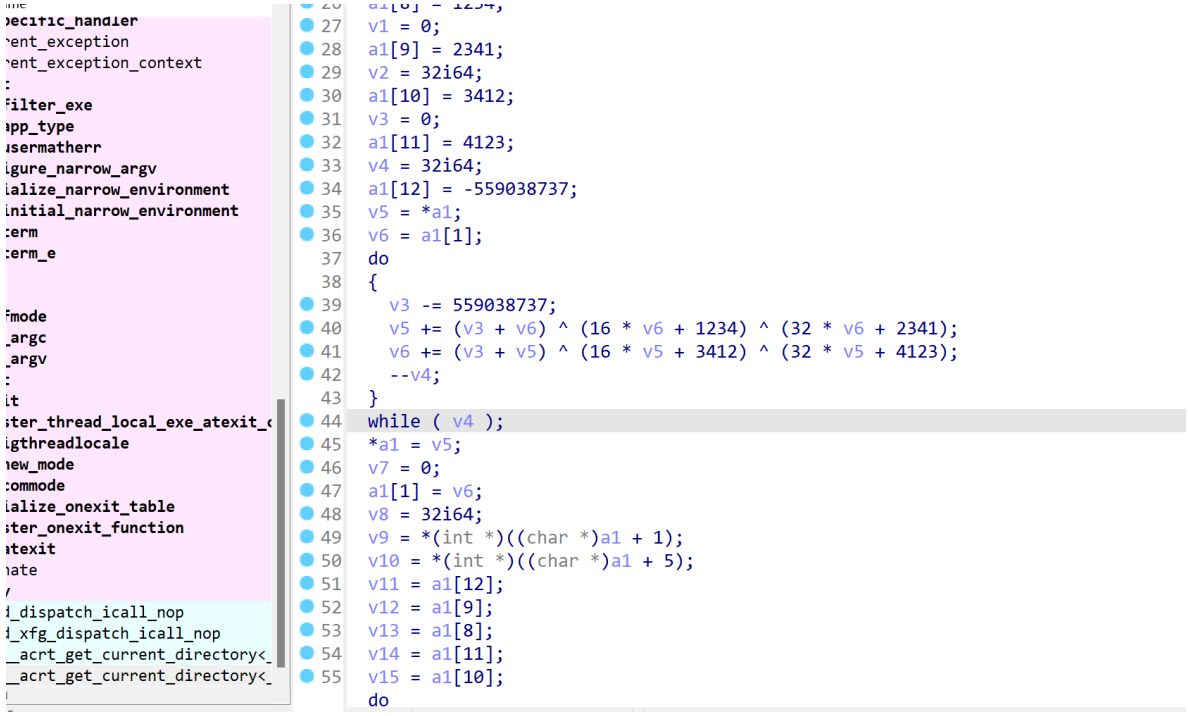
week2

web

reverse

ezcpp

IDA进去



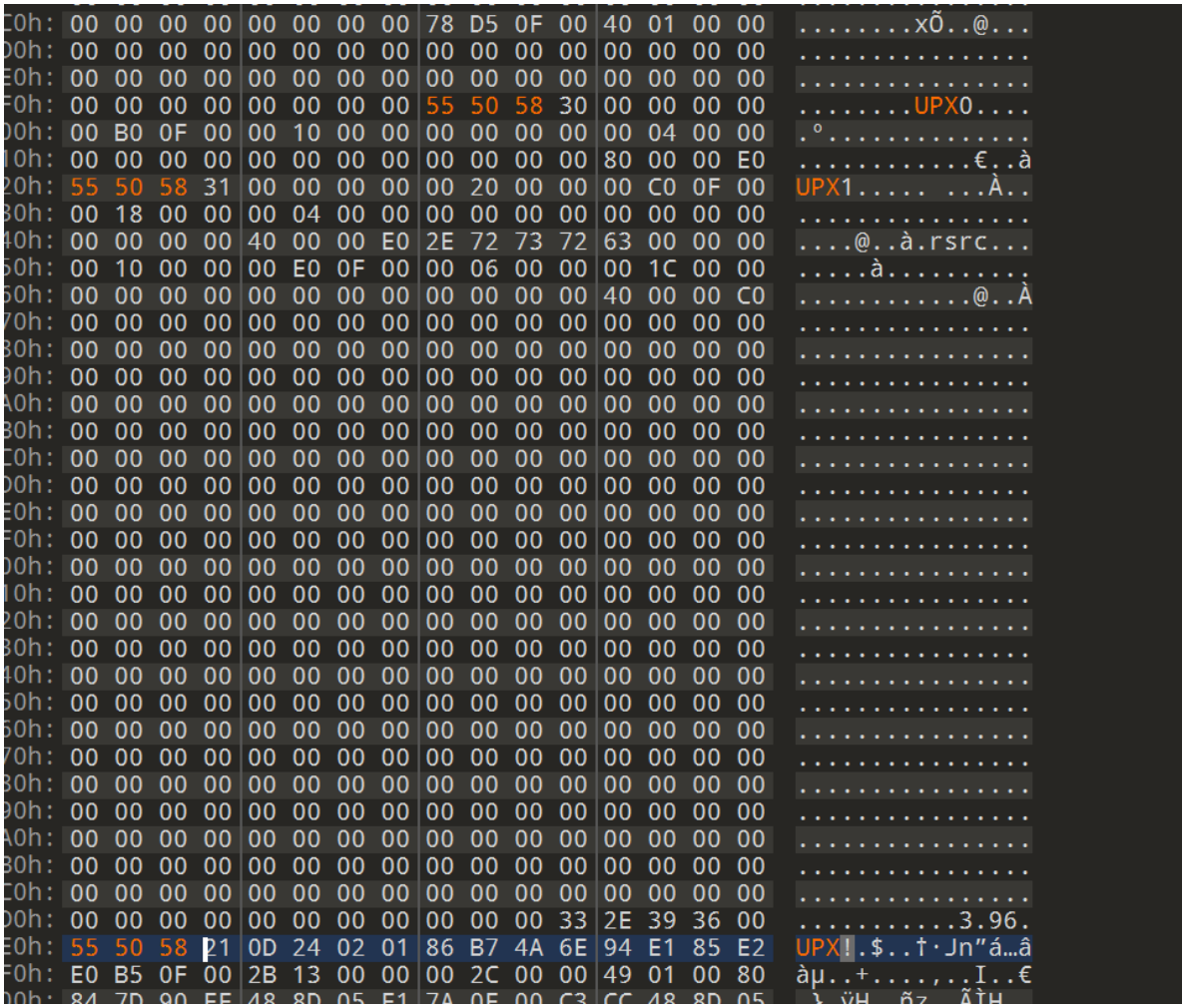
tea，前几个字节在加密

```
#include <stdio.h>
#include <stdint.h>
#include<stdlib.h>
void decrypt(uint32_t* v, uint32_t* k) {
    uint32_t delta = 0xdeadbeef;
    uint32_t v0 = v[0] , v1 = v[1] , sum = delta * 32, i;
    uint32_t k0 = k[0], k1 = k[1], k2 = k[2], k3 = k[3];
    for (i = 0; i < 32; i++) {
        v1 -= (v0 + sum) ^ (k2 + (v0 << 4)) ^ (k3 + (v0 << 5));
        v0 -= (v1 + sum) ^ (k0 + (v1 << 4)) ^ (k1 + (v1 << 5));
        sum -= delta;
    }
    v[0] = v0; v[1] = v1;
}
int main()
{
    uint32_t j;
    int8_t v[33] = { 136, 106, 176, 201, 173, 241, 51, 51, 148, 116, 181, 105,
115, 95, 48, 98, 74, 51, 99, 84, 95, 48, 114, 49, 101, 110, 84, 101, 68, 63, 33,
125 };
```

```
    uint32_t k[4] = { 1234,2341,3412,4123 };
    decrypt((uint32_t*)&v[3] ,k);
    decrypt((uint32_t*)&v[2] ,k);
    decrypt((uint32_t*)&v[1] ,k);
    decrypt((uint32_t*)&v[0] ,k);
    printf(v);
    system("pause");
    return 0;
}
```

arithmetic

IDA开不了，查壳upx，但脱不了，010得知被改特征



改回特征后可以脚本脱，IDA打开是一个三角形最大路径向下算法

最后的那个数字就是最大路径

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

int max(int a, int b) {
    return a > b ? a : b;
}

int fun(int n, int i, int j, int **a, int **op) {
    if (i == n) {
```

```
        return 0;
    }
    if (op[i][j] != 0)
        return op[i][j];

    op[i][j] = a[i][j] + max(fun(n, i + 1, j, a, op), fun(n, i + 1, j + 1, a,
op));
    return op[i][j];
}

void Find_path(int **op, int **a, int n) {
    int j = 0;
    printf("路径如下: \n");
    for (int i = 1; i < n; i++) {
        int node = op[i - 1][j] - a[i - 1][j];
        if (node == op[i][j + 1]){
            printf("%d", 2);
            j++;
        }
        else
            printf("%d", 1);
    }
}

int main() {
    int n;
    printf("Enter the size of the triangle: ");
    scanf("%d", &n);

    int **data = (int **)malloc(sizeof(int *) * n);
    for (int i = 0; i < n; i++) {
        data[i] = (int *)malloc(sizeof(int) * n);
    }

    for (int i = 0; i < n; i++) {
        for (int j = 0; j <= i; j++) {
            data[i][j] = 0;
        }
    }

    FILE *file = fopen("out.txt", "r");
    if (file == NULL) {
        fprintf(stderr, "Error opening the file.\n");
        return 1;
    }

    for (int i = 0; i < n; i++) {
        for (int j = 0; j <= i; j++) {
            fscanf(file, "%d", &data[i][j]);
        }
    }
    fclose(file);

    int **op = (int **)malloc(sizeof(int *) * n);
    for (int i = 0; i < n; i++) {
        op[i] = (int *)malloc(sizeof(int) * n);
    }
    for (int x = 0; x < n; x++) {
```

```
            for (int y = 0; y < n; y++)
                op[x][y] = 0;
        }

        int result = fun(n, 0, 0, data, op);
        printf("最大路径是：%d\n", result);
        Find_path(op, data, n);
        system("pause");
        return 0;
    }
```

得到路径后md5加密就是flag

babyAndroid

apk文件jadx打开

```
/* loaded from: classes.dex */
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    private ActivityMainBinding binding;
    private Button enter;
    private EditText password;
    private EditText username;

    public native boolean check2(byte[] bArr, byte[] bArr2);

    static {
        System.loadLibrary("babyandroid");
    }

    /* JADX INFO: Access modifiers changed from: protected */
    @Override // androidx.fragment.app.FragmentActivity, androidx.activity.ComponentActivity,
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        ActivityMainBinding inflate = ActivityMainBinding.inflate(getLayoutInflater());
        this.binding = inflate;
        setContentView(inflate.getRoot());
        this.username = (EditText) findViewById(R.id.username);
        this.password = (EditText) findViewById(R.id.password);
        Button button = (Button) findViewById(R.id.enter);
        this.enter = button;
        button.setOnClickListener(this);
    }

    @Override // android.view.View.OnClickListener
    public void onClick(View view) {
        byte[] bytes = this.username.getText().toString().getBytes();
        byte[] bytes2 = this.password.getText().toString().getBytes();
        if (new Check1(getResources().getString(R.string.key).getBytes()).check(bytes)) {
            if (check2(bytes, bytes2)) {
                Toast.makeText(this, "Congratulate!!!^_^", 0).show();
                return;
            } else {
                Toast.makeText(this, "password wrong!!!>_<", 0).show();
                return;
            }
        }
        Toast.makeText(this, "username wrong!!!>_<", 0).show();
    }
}
```

两个函数校验username和password，先看check1

```
public class Check1 {
    private byte[] S = new byte[256];
    private int i;
```

```
    private int j;

    public Check1(byte[] bArr) {
        for (int i = 0; i < 256; i++) {
            this.S[i] = (byte) i;
        }
        int i2 = 0;
        for (int i3 = 0; i3 < 256; i3++) {
            byte[] bArr2 = this.S;
            i2 = (i2 + bArr2[i3] + bArr[i3 % bArr.length]) & 255;
            swap(bArr2, i3, i2);
        }
        this.i = 0;
        this.j = 0;
    }

    private void swap(byte[] bArr, int i, int i2) {
        byte b = bArr[i];
        bArr[i] = bArr[i2];
        bArr[i2] = b;
    }

    public byte[] encrypt(byte[] bArr) {
        byte[] bArr2 = new byte[bArr.length];
        for (int i = 0; i < bArr.length; i++) {
            int i2 = (this.i + 1) & 255;
            this.i = i2;
            int i3 = this.j;
            byte[] bArr3 = this.S;
            int i4 = (i3 + bArr3[i2]) & 255;
            this.j = i4;
            swap(bArr3, i2, i4);
            byte[] bArr4 = this.S;
            bArr2[i] = (byte) (bArr4[(bArr4[this.i] + bArr4[this.j]) & 255] ^
bArr[i]);
        }
        return bArr2;
    }

    public boolean check(byte[] bArr) {
        return Arrays.equals(new byte[]{-75, 80, 80, 48, -88, 75, 103, 45, -91,
89, -60, 91, -54, 5, 6, -72}, encrypt(bArr));
    }
}
```

rc4，找到key对密文重新加密一遍就是解密

用apktools提取文件，在res/value/string.xml找到key

```
<string name="hide bottom view on scroll behavior">com
<string name="icon content description">Dialog Icon</stri
<string name="item view role description">Tab</string>
<string name="key">3e1fel</string>
<string name="m3 exceed max badge text suffix">%1$s%2
<string name="m3 ref typeface brand medium">sans-serif
<string name="m3 ref typeface brand regular">sans-serif<
<string name="m3 ref typeface plain medium">sans-serif-
```

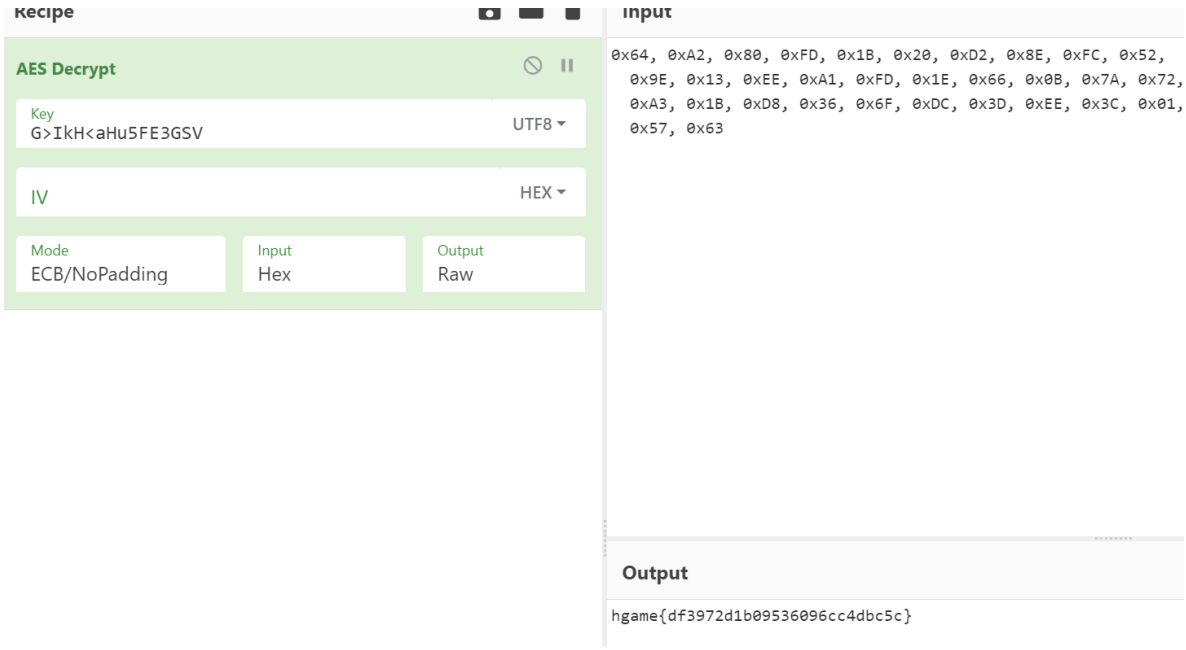
```
package package0;

import java.util.Arrays;

/* loaded from: classes.dex */
public class Check1 {
    private byte[] S = new byte[256];
    private int i;
    private int j;

    public Check1(byte[] bArr) {
        for (int i = 0; i < 256; i++) {
            this.S[i] = (byte) i;
        }
        int i2 = 0;
        for (int i3 = 0; i3 < 256; i3++) {
            byte[] bArr2 = this.S;
            i2 = (i2 + bArr2[i3] + bArr[i3 % bArr.length]) & 255;
            swap(bArr2, i3, i2);
        }
        this.i = 0;
        this.j = 0;
    }

    private void swap(byte[] bArr, int i, int i2) {
        byte b = bArr[i];
        bArr[i] = bArr[i2];
        bArr[i2] = b;
    }

    public byte[] encrypt(byte[] bArr) {
        byte[] bArr2 = new byte[bArr.length];
        for (int i = 0; i < bArr.length; i++) {
            int i2 = (this.i + 1) & 255;
            this.i = i2;
            int i3 = this.j;
            byte[] bArr3 = this.S;
            int i4 = (i3 + bArr3[i2]) & 255;
            this.j = i4;
            swap(bArr3, i2, i4);
            byte[] bArr4 = this.S;
            bArr2[i] = (byte) (bArr4[(bArr4[this.i] + bArr4[this.j]) & 255] ^
bArr[i]);
        }
        return bArr2;
```

```
    }

    public void check(byte[] bArr) {
        System.out.println(Arrays.toString(encrypt(bArr)));
    }


    public static void main(String[] args) {
        String key = "3e1fel";
        byte[] keyBytes = key.getBytes();
        Check1 rc4 = new Check1(keyBytes);
        byte[] encryptedData = {-75, 80, 80, 48, -88, 75, 103, 45, -91, 89, -60,
91, -54, 5, 6, -72};
        byte[] decryptedData = rc4.encrypt(encryptedData);
        System.out.println(Arrays.toString(decryptedData));
    }
}
#G>IkH<aHu5FE3GSV
```

回到主方法中可以发现check2是native层的方法，IDA打开lib里的.so文件

```
    *(v14 - 3) = *(v14 - 19) ^ v20;
    *(v14 - 2) = *(v14 - 18) ^ v21;
    LOBYTE(v22) = *(v14 - 17) ^ v22;
    *(v14 - 1) = v22;
    LOBYTE(v23) = *(v14 - 16) ^ v23;
    *v14 = v23;
    v14 += 16;
  }
  v24 = (*(*a1 + 1472LL))(a1, a5, 0LL, v22, v23, byte_7C0);
  if ( v12 <= 0 )
  {
    sub_1260(src, &v66);
    sub_1260(src, &v66);
    goto LABEL_60;
  }
  v25 = v24;
  if ( v12 < 8 || src < v24 + v12 && v24 < &src[v12] )
  {
    v26 = 0LL;
LABEL_10:
    v27 = v12 + ~v26;
    v28 = v12 & 3;
    if ( (v12 & 3) != 0 )
    {
      do
      {
        src[v26] = *(v25 + v26);
```

aes加密，那么猜测username就是key，cyberchef解密



babyre

虚拟机逆向，IDA打开分析

```
 9
10  v9[2] = __readfsqword(0x28u);
11  sub_5630D3194708();
12  if ( !__sigsetjmp(env, 1) )
13  {
14    signal(8, handler);
15    for ( i = 0; i <= 5; ++i )
16      *(&dword_5630D31970A0 + i) ^= 0x11u;
17  }
18  sem_init(&sem, 0, 1u);
19  sem_init(&stru_5630D3197280, 0, 0);
20  sem_init(&stru_5630D31972A0, 0, 0);
21  sem_init(&stru_5630D31972C0, 0, 0);
22  pthread_create(&newthread, 0LL, start_routine, 0LL);
23  pthread_create(&v7, 0LL, sub_5630D319440D, 0LL);
24  pthread_create(&v8, 0LL, sub_5630D319450C, 0LL);
25  pthread_create(v9, 0LL, sub_5630D3194609, 0LL);
26  for ( j = 0; j <= 3; ++j )
27    pthread_join(*(&newthread + j), 0LL);
28  sub_5630D3194803();
29  return 0LL;
30 }
```

首先查看 `sub_5630D3194708()` 函数

```
  v3 = __readfsqword(0x28u);
  puts("plz input your answer:");
  __isoc99_scanf("%s", s);
  if ( strlen(s) != 32 )
  {
    puts("length error!");
    exit(0);
  }
  for ( i = 0; i <= 31; ++i )
    flag[i] = s[i];
  dword_5630D3197240 = 249;
  return v3 - __readfsqword(0x28u);
}
```

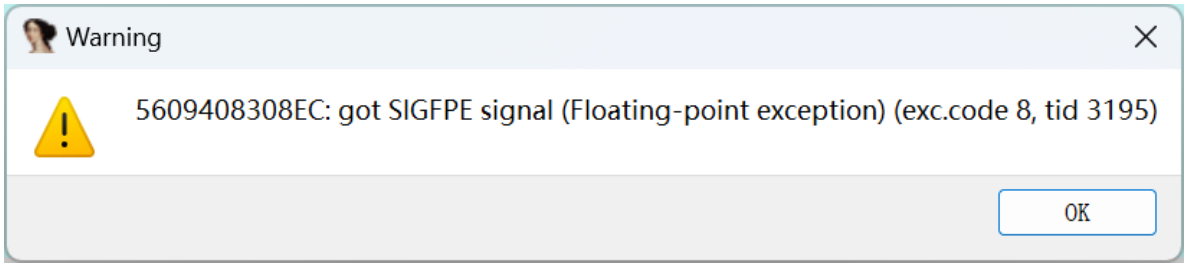
这里是输入flag，并且在flag后又增加一个值为249的内容

主函数中`signal(8, handler);`函数，是有浮点异常时执行handler

```
void __noreturn handler()
{
  ++dword_5630D3197240;
  siglongjmp(env, 1);
}
```

handler函数中，是对刚刚flag最后追加的249执行+1的操作，也就是说抛出异常时249就会+1

那么动调起来看看有没有异常



果然有异常，那么flag后追加的内容就是250

回到main函数继续分析

四个线程对flag进行加密，最后`sub_5630D3194803()`校验flag

这里先对几个函数进行解释

`sem_init`:对由sem指定的信号量进行初始化

`sem`:为指向信号量结构的一个指针

`sem_post`:用来增加信号量的值

`sem_wait`:用来阻塞当前线程直到信号量sem的值大于0，解除阻塞后将sem的值减一

那么现在先进入第一个加密函数进行分析

```
{
  while ( 1 )
  {
    sem_wait(&sem);
    if ( dword_5630D3197244 > 31 )
      break;
    flag[dword_5630D3197244] += *(&dword_5630D31970A0 + (dword_5630D3197244 + 1) % 6) * flag[dword_5630D3197244 + 1];
    ++dword_5630D3197244;
    sem_post(&stru_5630D3197280);
  }
  sem_post(&stru_5630D3197280);
  pthread_exit(0LL);
}
```

第一个加密函数先用`sem_wait`函数对sem进行-1然后执行加密

加密结束后使用`sem_post`来增加下一个加密函数的信号量，即进行下一个函数的加密

那么依次加密，我们看到最后一个加密函数

```
{
  while ( 1 )
  {
    sem_wait(&stru_5630D31972C0);
    if ( dword_5630D3197244 > 31 )
      break;
    flag[dword_5630D3197244] ^= flag[dword_5630D3197244 + 1] - *(&c
    ++dword_5630D3197244;
    sem_post(&sem);
  }
  sem_post(&sem);
  pthread_exit(0LL);
}
```

最后一个加密函数又使用`sem_post`增加第一个加密函数的信号量

因此构成循环，逐个对flag进行加密，总共32字节

其中每个加密函数中`&dword_5630D31970A0`是一个key数组，并且真正的key需要动调才可以拿到

在主函数中有对key的异或加密，因此需要对key进行异或后才是真正的key

从前往后加密，从后往前解密

exp:

```
#include<stdio.h>
#include <stdlib.h>
char key[] = {119, 116, 120, 102, 101, 105};
int main(){
    int enc[33] = {12052, 78, 20467, 109, 13016, 109, 27467, -110, 9807, 91,
21243, -100, 11121, 20, 10863, -107, 10490, 29, 10633, -101, 10420, 78, 17670,
-38, 6011, -4, 16590, 125, 10723, 15, 7953, 255, 250};
    for (int i = 28; i >= 0; i -= 4)
    {
```

```
        enc[i + 3] = enc[i + 3] ^ (enc[i + 4] - key[(i + 4) % 6]);
        enc[i + 2] = enc[i + 2] / (enc[i + 3] + key[(i + 3) % 6]);
        enc[i + 1] = enc[i + 1] + (enc[i + 2] ^ key[(i + 2) % 6]);
        enc[i + 0] = enc[i + 0] - (enc[i + 1] * key[(i + 1) % 6]);
    }
    for (int i = 0; i < 32; i++)
    {
        printf("%c", enc[i]);
    }
    system("pause");
    return 0;
}
```

pwn

crypto

midRSA

midRSA.py

```
from Crypto.Util.number import *
from secret import flag

def padding(flag):
    return flag+b'\xff'*(64-len(flag))

flag=padding(flag)
m=bytes_to_long(flag)
p=getPrime(512)
q=getPrime(512)
e=3
n=p*q
c=pow(m,e,n)
m0=m>>208

print(f'n={n}')
print(f'c={c}')
print(f'm0={m0}')

"""
n=120838778421252867808799302603972821425274682456261749029016472234934876266617
266346399909705742862458970575637664059189613618956880430078774892479256301209695
323302787221508556481196281420676074116272495278097275927604857336484564777404497
9791457260629981038498741259484407193554669081990692025400404539158542 7
c=118961547254465282603128910126369011072248057317653811110746611348016137361383
017921465395766977129601435508590006599755740818071303929227578504412967513468921
191689357367045286190040251695094706564443721393216185563727951256414649625597950
957960429709583109707961019498084511008637686004730015209939219983527
m0=132921474085670873515807320829616401305433137422104094324716252817023277489632
74496942276607
```

```
"""
```

m高位泄露，直接解了

```
from Crypto.Util.number import long_to_bytes

n=120838778421252867808799302603972821425274682456261749029016472234934876266617
266346399909705742862458970575637664059189613618956880430078774892479256301209695
532330278722150855648119628142067607411627249527809727592760485733648456477740449
979145726062998103849874125948440719355466908199069202540040453915854 27
c=118961547254465282603128910126369011072248057317653811110746611348016137361383
017921465395766977129601435508590006599755740818071303929227578504412967513468921
119168935736704528619004025169509470656444372139321618556372795125641464962559795
095796042970958310970796101949808451100863768600473001520993921998352 7
m_high=132921474085670873515807320829616401305433137422104094324716252817023277 4
8963274496942276607
m_high <<= 208
e = 3

R.<x> = PolynomialRing(Zmod(n))
m = m_high + x
f = m^e - c
f = f.monic()

x = f.small_roots(X = 2^208,beta = 0.4)
if x:
    m = m_high + x[0]
    print(long_to_bytes(int(m)))
```

midRSA revenge

attachment.py

```
from Crypto.Util.number import *
from secret import flag
m=bytes_to_long(flag)
p=getPrime(1024)
q=getPrime(1024)
e=5
n=p*q
c=pow(m,e,n)
m0=m>>128

print(f'n={n}')
print(f'c={c}')
print(f'm0={m0}')

"""
```

```
n=27814334728135671995890378154778822687713875269624843122353458059697288888640572922486287556431241786461159513236128914176680497775619694684903498070577307810263677280294114135929708745988406963307279767028969515305895207028282193547356414827419008393701158467818535109517213088920890236300281646288761697842280633285355376389468360033584102258243058885174812018295460196515483819254913183079496947309574392848378504246991546781252139861876509894476420525317251695953355755164789878602945615879965709871975770823484418665634050103852564819575756950047691205355599004786541600213204423145854859214897431430282333052121
c=45622131411586708863820720303449463624470661111162172357784872909606923006795813266301862566144713150175868450263938320833284468193969812445918857181352714977229246413953073671761974170494592607563206407212536151643563112184575318655929799335527077981805770297378339158985115911402931029655170145674869891423134483518791755930544026956061332689320474812799925490210291960537036388958113672416409687957317387028080662045408746697035899865473675525702322507814701853710101
m0=99999002810033577734203106811693308232665325338039056370
"""
```

同上，exp：

```
from Crypto.Util.number import long_to_bytes

n=27814334728135671995890378154778822687713875269624843122353458059697288888640572922486287556431241786461159513236128914176680497775619694684903498070577307810263677280294114135929708745988406963307279767028969515305895207028282193547356414827419008393701158467818535109517213088920890236300281646288761697842280633285355376389468360033584102258243058885174812018295460196515483819254913183079496947309574392848378504246991546781252139861876509894476420525317251695953355755164789878602945615879965709871975770823484418665634050103852564819575756950047691205355599004786541600213204423145854859214897431430282333052121
c=45622131411586708863820720303449463624470661111162172357784872909606923006795813266301862566144713150175868450263938320833284468193969812445918857181352714977229246413953073671761974170494592607563206407212536151643563112184575318655929799335527077981805770297378339158985115911402931029655170145674869891423134483518791755930544026956061332689320474812799925490210291960537036388958113672416409687957317387028080662045408746697035899865473675525702322507814701853710101
m_high=99999002810033577734203106811693308232665325338039056370
m_high <<= 128
e = 5

R.<x> = PolynomialRing(Zmod(n))
m = m_high + x
f = m^e - c
f = f.monic()

x = f.small_roots(X = 2^208,beta = 0.4)
if x:
    m = m_high + x[0]
    print(long_to_bytes(int(m)))
```

backpack

背包密码

attachment.py

```
from Crypto.Util.number import *
import random
from secret import flag
a=[getPrime(32) for _ in range(20)]
p=random.getrandbits(32)
assert len(bin(p)[2:])==32
bag=0
for i in a:
    temp=p%2
    bag+=temp*i
    p=p>>1

enc=bytes_to_long(flag)^p

print(f'enc={enc}')
print(f'a={a}')
print(f'bag={bag}')
"""
enc=871114172567853490297478570113449366988793760172844644007566824913350088148162949968812541218339
a=[3245882327, 3130355629, 2432460301, 3249504299, 3762436129, 3056281051,
3484499099, 2830291609, 3349739489, 2847095593, 3532332619, 2406839203,
4056647633, 3204059951, 3795219419, 3240880339, 2668368499, 4227862747,
2939444527, 3375243559]
bag=45893025064
"""
```

exp:

```
import libnum

enc =
871114172567853490297478570113449366988793760172844644007566824913350088148162949968812541218339
M = [3245882327, 3130355629, 2432460301, 3249504299, 3762436129, 3056281051,
3484499099, 2830291609, 3349739489, 2847095593, 3532332619, 2406839203,
4056647633, 3204059951, 3795219419, 3240880339, 2668368499, 4227862747,
2939444527, 3375243559]
S = 45893025064

n = len(M)
Ge = Matrix.identity(n)
last_row = [0 for x in range(n)]
Ge_last_row = Matrix(ZZ, 1, len(last_row), last_row)

last_col = M[:]
last_col.append(S)
Ge_last_col = Matrix(ZZ, len(last_col), 1, last_col)

Ge = Ge.stack(Ge_last_row)
Ge = Ge.augment(Ge_last_col)

X = Ge.LLL()[-1]
X = X[:-1]

p = ""
```

```
for i in X:
    if abs(i) == 1:
        p += "1"
    if abs(i) == 0:
        p += "0"

print(p)
m = int(p,2) ^^ enc
print(m)
flag = bytes.fromhex(hex(int(m))[2:])
print(flag)
```

非预期：

```
from Crypto.Util.number import long_to_bytes
enc=871114172567853490297478570113449366988793760172844644007566824913350088148162949968812541218339
print(long_to_bytes(enc))
```

babyRSA

attachment.py

```
from Crypto.Util.number import *
from secret import flag,e
m=bytes_to_long(flag)
p=getPrime(64)
q=getPrime(256)
n=p**4*q
k=getPrime(16)
gift=pow(e+114514+p**k,0x10001,p)
c=pow(m,e,n)
print(f'p={p}')
print(f'q={q}')
print(f'c={c}')
print(f'gift={gift}')
"""
p=14213355454944773291
q=6184356205162070038634855117537193048606497844115920076561833974376400103329 7
c=1050021387224669464959366386560382140000434757516390250852551139650887492724619068925866162502649223481924965979864527862811511564362295740651939654228 41
gift=9751789326354522940
"""
```

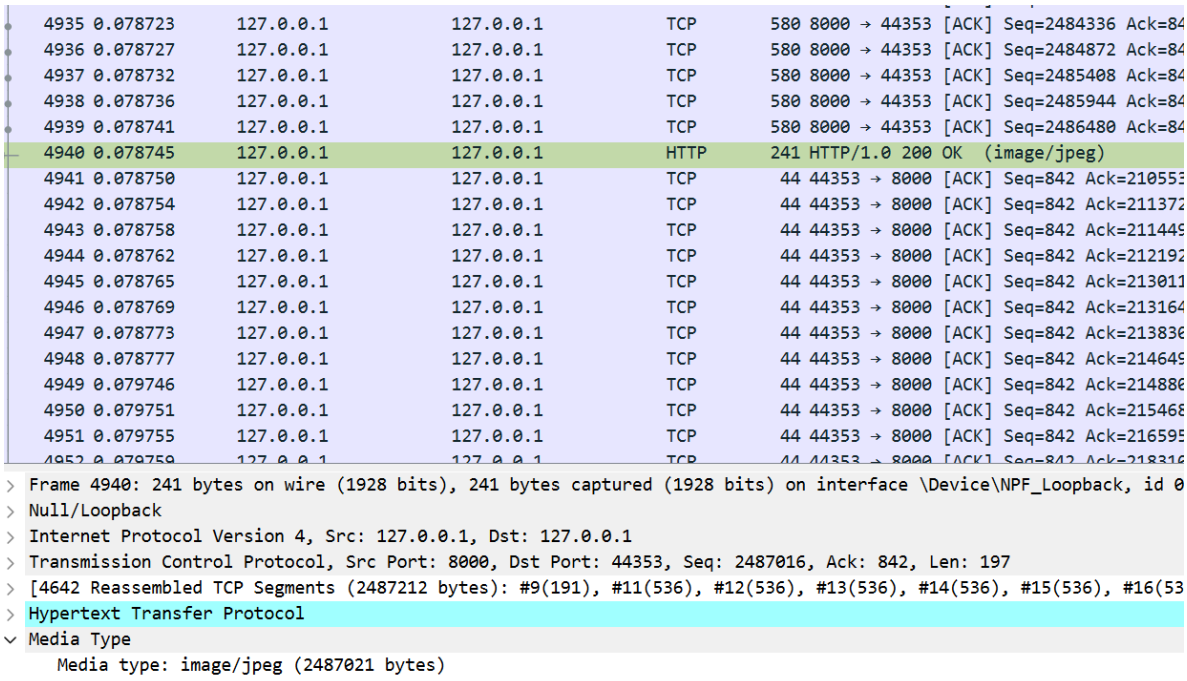
exp：

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

p = 14213355454944773291
q =
6184356205162070038634855117537193048606497844115920076561833974376400103329 7
```

```
c = 
105002138722466946495936638656038214000043475751639025085255113965088749272461906892586616250264922348192496597986452786281151156436229574065193965422841
gift = 9751789326354522940

n = p**4*q
d = gmpy2.invert(65537,p-1)
temp = pow(gift,d,p)
e = temp - 114514
res = Zmod(n)(c).nth_root(e, all=True)

for m in res:
    flag = long_to_bytes(int(m))
    if b"hgame" in flag:
        print(flag)
        break
```

misc

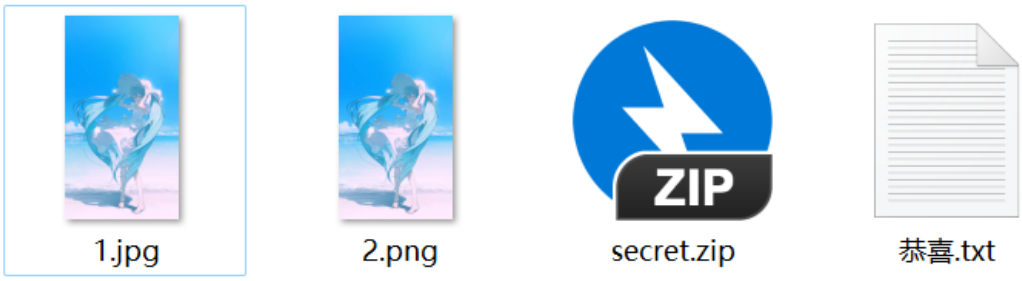
ek1ng_want_girlfriend



追踪http，导出jpg，得到flag

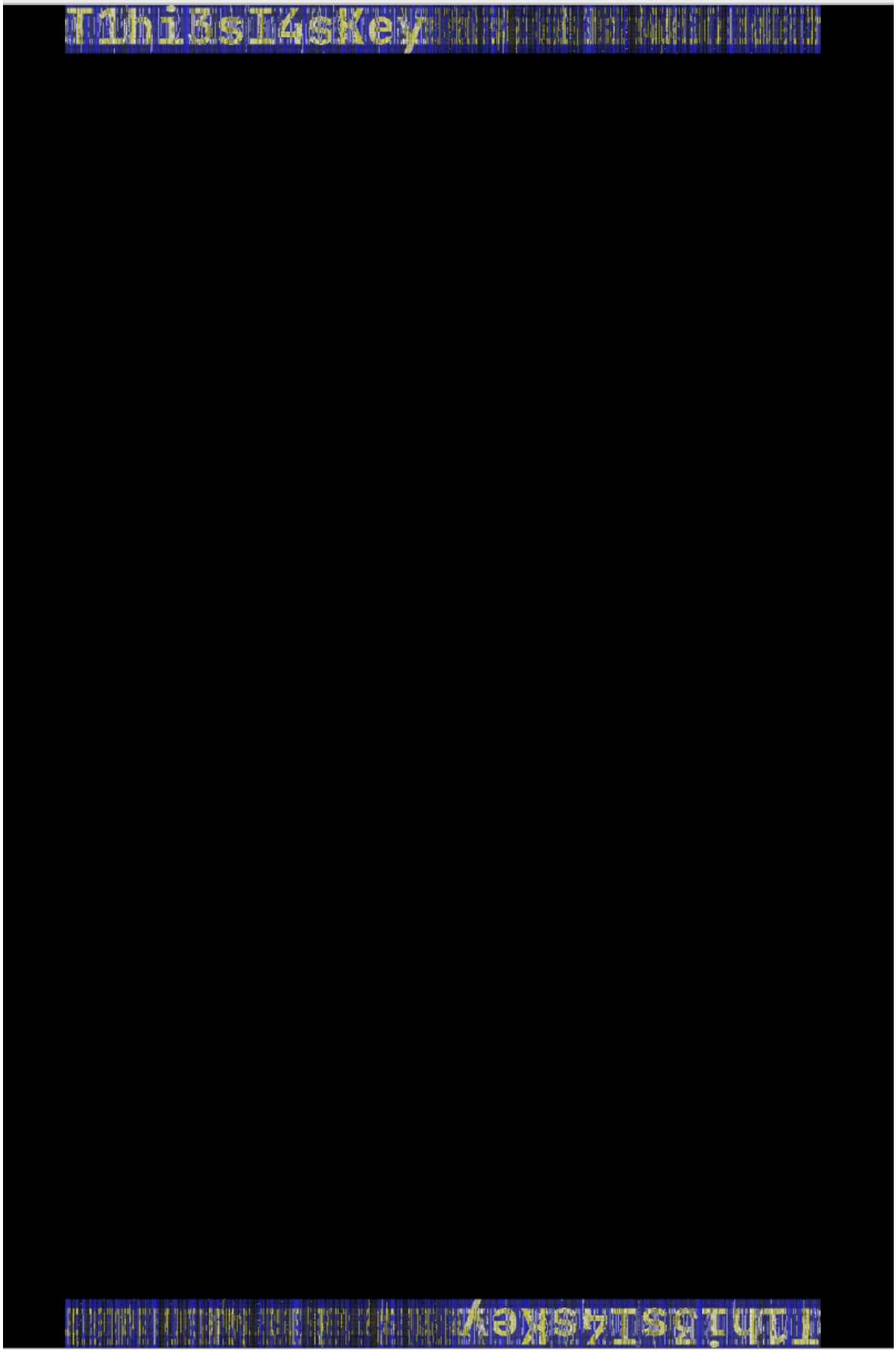
ezWord

010得知是个压缩包，改zip后解压找到关键文件



根据hint是盲水印，解出来后得到压缩包密码（这里第一次做盲水印题，对老脚本里的随机函数需要进行修改，卡了好久）

flag.jpg



解开压缩包后难道一堆乱七八糟的英文，查出来是fake加密，解密后得到一堆中文乱码，刚开始有想到Unicode，但是解不出来，后面根据出题人提示得知rot8000，解出flag