REVERSE

1. babyre

未加壳，拖入 ida 分析，发现有很多不认识和函数，去学习了一波（一开始没发现 wait 会减少信号量，卡了很久）。跟进四个进程发现，进程一二三四依次进行。

```
void __fastcall __noreturn start_routine(void *a1)
{
  while ( 1 )
  {
    sem_wait(&num1);
    if ( count > 31 )
      break;
    input[count] += *((char *)&key + (count + 1) % 6) * input[count + 1];
    ++count;
    sem_post(&num2);
  }
  sem_post(&num2);
  pthread_exit(0LL);
}
```

根据第一个进程可以分析出，进程 1 会对 input[count]进行一次操作，进程一运行一次后，对全局的 count 加 1；然后唤醒第二个进程，第一个线程开始休眠等待信号，进入第二个进程对下一个元素进行操作，以此类推，程序的加密逻辑就是这样。

接下来就是关于 key 的问题，可以看到在内存地址里面 key 被截成了两截，于是在对 key 异或的时候，在被截断的地方会触发 signal 函数，然后进入 handler，对最后的一个元素做递增，并且使得 key 的后三字节没被异或修改，然后进入加密。这里也可以通过下断点直接获取 key 的值

```
key dd 66787477h                        ; DATA XREF: sub_556B3A3492E9+8↑w
                                        ; start_routine+AC↑o
                                        ; second+AC↑o
                                        ; third+AC↑o
                                        ; fourth+AC↑o
                                        ; main+78↑o
                                        ; main+8D↑o
word_556B3A34C0A4 dw 6965h              ; DATA XREF: sub_556B3A3492E9+12↑w
byte_556B3A34C0A6 db 0                  ; DATA XREF: sub_556B3A3492E9+1B↑w
```

那么就可以编写脚本了，脚本如下:

```
#define _CRT_SECURE_NO_WARNINGS 1
#include<stdio.h>

int main(void)
{

    void total(unsigned int*, unsigned char*);


     unsigned char enc[] = { 0x14, 0x2F, 0x00, 0x00, 0x4E, 0x00, 0x00, 0x00, 0xF3,
0x4F,
  0x00, 0x00, 0x6D, 0x00, 0x00, 0x00, 0xD8, 0x32, 0x00, 0x00,
  0x6D, 0x00, 0x00, 0x00, 0x4B, 0x6B, 0x00, 0x00, 0x92, 0xFF,
  0xFF, 0xFF, 0x4F, 0x26, 0x00, 0x00, 0x5B, 0x00, 0x00, 0x00,
  0xFB, 0x52, 0x00, 0x00, 0x9C, 0xFF, 0xFF, 0xFF, 0x71, 0x2B,
  0x00, 0x00, 0x14, 0x00, 0x00, 0x00, 0x6F, 0x2A, 0x00, 0x00,
  0x95, 0xFF, 0xFF, 0xFF, 0xFA, 0x28, 0x00, 0x00, 0x1D, 0x00,
  0x00, 0x00, 0x89, 0x29, 0x00, 0x00, 0x9B, 0xFF, 0xFF, 0xFF,
  0xB4, 0x28, 0x00, 0x00, 0x4E, 0x00, 0x00, 0x00, 0x06, 0x45,
  0x00, 0x00, 0xDA, 0xFF, 0xFF, 0xFF, 0x7B, 0x17, 0x00, 0x00,
  0xFC, 0xFF, 0xFF, 0xFF, 0xCE, 0x40, 0x00, 0x00, 0x7D, 0x00,
  0x00, 0x00, 0xE3, 0x29, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00,
  0x11, 0x1F, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00 ,0xFA, 0x00,0x00,0x00};
    unsigned char key[] = { 0x77,0x74,0x78,0x66,0x65,0x69 };
    total((unsigned int*)enc, key);
    for(int i=0;i<sizeof(enc);i++)
        printf("%c", enc[i]);
}

void total(unsigned int* enc, unsigned char* key)
{
    void change(unsigned char*);
    int i = 31;

    while (i >= 0)
```

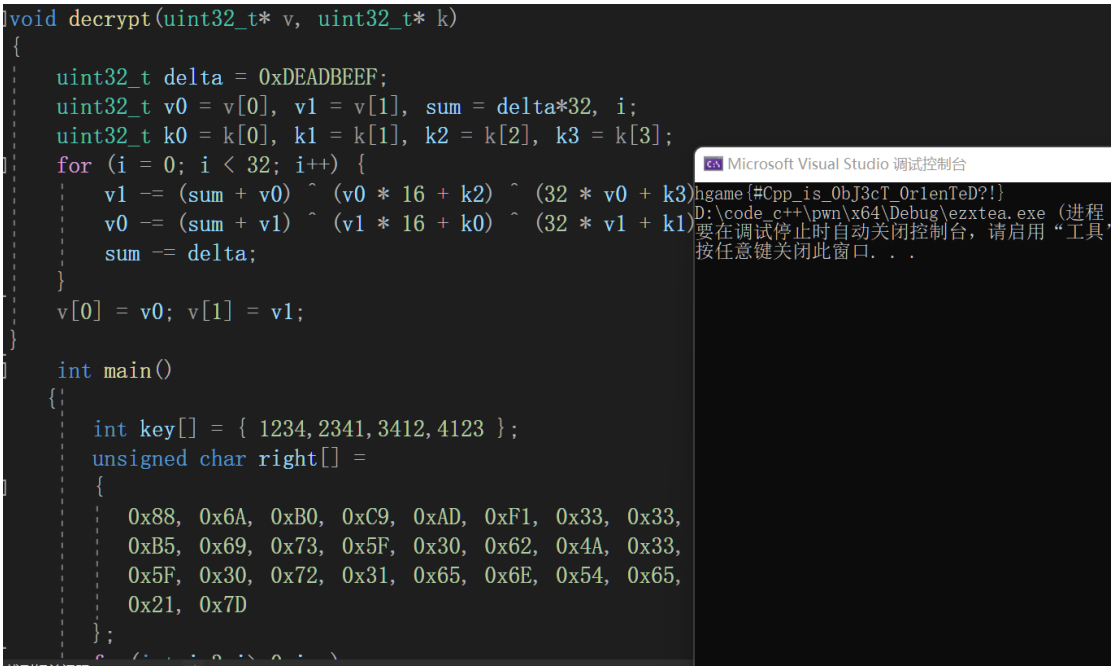
```
    {

        enc[i] ^= enc[i + 1] - key[(i + 1) % 6];
        i--;
        enc[i] /= enc[i + 1] + key[(i + 1) % 6];
        i--;
        enc[i] += enc[i + 1] ^ key[(i + 1) % 6];
        i--;
        enc[i] -= enc[i + 1] * key[(i + 1) % 6];
        i--;
        }
}
```

```
#define _CRT_SECURE_NO_WARNINGS 1
#include<stdio.h>

int main(void)
{

    void total(unsigned int*, unsigned char*);


     unsigned char enc[] = { 0x14, 0x2F, 0x00,
  0x00, 0x00, 0x6D, 0x00, 0x00, 0x00, 0xD8, 0x3
  0x6D, 0x00, 0x00, 0x00, 0x4B, 0x6B, 0x00, 0x0
  0xFF, 0xFF, 0x4F, 0x26, 0x00, 0x00, 0x5B, 0x0
  0xFB, 0x52, 0x00, 0x00, 0x9C, 0xFF, 0xFF, 0xF
  0x00, 0x00, 0x14, 0x00, 0x00, 0x00, 0x6F, 0x2
  0x95, 0xFF, 0xFF, 0xFF, 0xFA, 0x28, 0x00, 0x0
  0x00, 0x00, 0x89, 0x29, 0x00, 0x00, 0x9B, 0xF
  0xB4, 0x28, 0x00, 0x00, 0x4E, 0x00, 0x00, 0x0
  0x00, 0x00, 0xDA, 0xFF, 0xFF, 0xFF, 0x7B, 0x1
  0xFC, 0xFF, 0xFF, 0xFF, 0xCE, 0x40, 0x00, 0x0
  0x00, 0x00, 0xE3, 0x29, 0x00, 0x00, 0x0F, 0x0
  0x11, 0x1F, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x0
    unsigned char key[] = { 0x77,0x74,0x78,0x66
    total((unsigned int*)enc, key);
    for(int i=0;i<sizeof(enc);i++)
        printf("%c", enc[i]);
}

void total(unsigned int* enc, unsigned char* ke
{
    void change(unsigned char*);
    int i = 31;
```

```
Microsoft Visual Studio 调试控制台
hgame{you_are_3o_c1ever2_3Olve!}
D:\code_c++\真正的一般调试文件\x64\Debug\真正的一
要在调试停止时自动关闭控制台，请启用“工具”->“
按任意键关闭此窗口. . .
```

2. ezcpp

很明显的简单的魔改后的 tea 加密，把 delta 值改成了 0XDEADBEEF，然后 key 值是 1234，2341，3412，4132，其它地方基本没有改变，写个脚本解密完了就行。



3. babyeAndroid

找到 Mainactivity，点进 check1 函数，发现关键部分

```
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;
}
```

显然这是一个 RSA 加密，回到上层找 key，点进去，发现一串 16 进制数据

```
/* loaded from: classes.dex */
public static final class string {
    public static int app_name = 0x7f0f001c;
    public static int key = 0x7f0f0030;
    /* JADX INFO: Added by JADX */
```

原先以为这就是 RSA 的密钥，直接解密发现出的乱码，觉得很奇怪，经过资料搜索发现这其实是 key 的 id，其具体值要去资源文件里寻找，于是前往寻找

```
<string name="icon_content_description">Di
<string name="item_view_role_description">
<string name="key">3e1fel</string>
<string name="m3_exceed_max_badge_text_suf
<string name="m3_ref_typeface_brand_medium
```

于是找到了 key，用脚本进行 RSA 解密

```
    unsigned char temp = s[i];
    s[i] = s[j];
    s[j] = temp;
    unsigned char key_byte = s[(s[i] + s[j]) % 256];
    data[n] ^= key_byte;
}


main() {
unsigned char key[] = "3e1fel";
unsigned char data[] = { -75, 80, 80, 48, -88, 75, 103, 45, -91, 89, -60, 91, -54, 5, 6,
unsigned long key_length = strlen(key);
unsigned long data_length = sizeof(data);
rc4(key, key_length, data, data_length);

for (unsigned long i = 0; i < data_length; i++) {
    printf("%c ", data[i]);
}
printf("\n");
```

```
Microsoft Visual Studio 调试控制台
G > I k H < a H u 5 F E 3 G S V

D:\code_c++\Project3\x64\Debug\babyanzhuo.exe
要在调试停止时自动关闭控制台，请启用“工具”->
按任意键关闭此窗口. . .
```

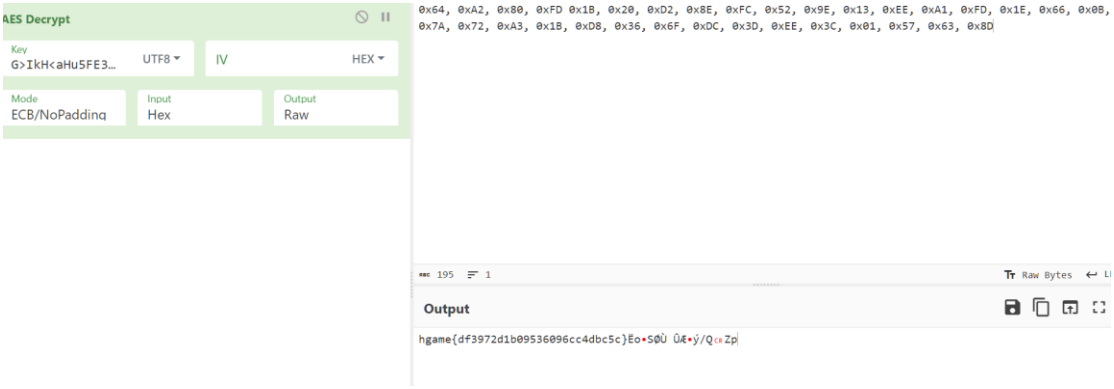
做到这发现字符串应该是解对了，但还是觉得很怪，但不管，先去看 check2，但是发现 check2 点不进去，于是解包 apk，拖到 ida 里看 native 层，点开 JNI_Onload

```
IDA View-A | Pseudocode-C | Pseudocode-B | Pseudocode-A | Hex View-1 | Structures | Enu
 1 int __cdecl JNI_OnLoad(int a1)
 2 {
 3   int v1; // esi
 4   int v2; // edi
 5   int v3; // eax
 6   int v5; // [esp+Ch] [ebp-20h] BYREF
 7   __int64 v6; // [esp+10h] [ebp-1Ch] BYREF
 8   int (__cdecl *v7)(int, int, __int128, __int128, __int128, __int128); // [esp+18h] [ebp-14h]
 9   unsigned int v8; // [esp+1Ch] [ebp-10h]
10
11   v8 = __readgsdword(0x14u);
12   v5 = 0;
13   v1 = -1;
14   if ( !(*(int (__cdecl **)(int, int *, int))(*(_DWORD *)a1 + 24))(a1, &v5, 65542) )
15   {
16     v2 = v5;
17     v3 = (*(int (__cdecl **)(int, const char *))(*(_DWORD *)v5 + 24))(v5, "com/feifei/babyandroid/MainActivity");
18     v1 = 65542;
19     if ( v3 )
20     {
21       v7 = sub_B60;
22       v6 = *(_QWORD *)off_2BD4;
23       (*(void (__cdecl **)(int, int, __int64 *, int))(*(_DWORD *)v2 + 860))(v2, v3, &v6, 1);
24     }
25   }
26   return v1;
27 }
```

用 findcrypto 发现是 AES 加密算法，那么 check1 得到的应该就是 AES 的密钥，那么找到待解密的数据

```
F+                                      ; DATA XREF: JNI_OnLoad+51↓o
aCheck2 db 'check2',0                   ; DATA XREF: .data.rel.ro:off_2BD4↓o
 db 64h, 0A2h, 80h, 0FDh
1+byte_597 db 1Bh, 20h, 0D2h, 8Eh, 0FCh, 52h, 9Eh, 13h, 0EEh, 0A1h, 0FDh, 1Eh, 66h, 0Bh, 7Ah, 72h, 0A3h, 1Bh, 0D8h
6+                                      ; DATA XREF: sub_B60+5F7↓o
 db 36h, 6Fh, 0DCh, 3Dh, 0EEh, 3Ch, 1, 57h, 63h, 8Dh
unk_5B4 db    1                         ; DATA XREF: sub_B60+8F↓o
 db    2
 db    4
 db    8
 db  10h
 db  20h
```

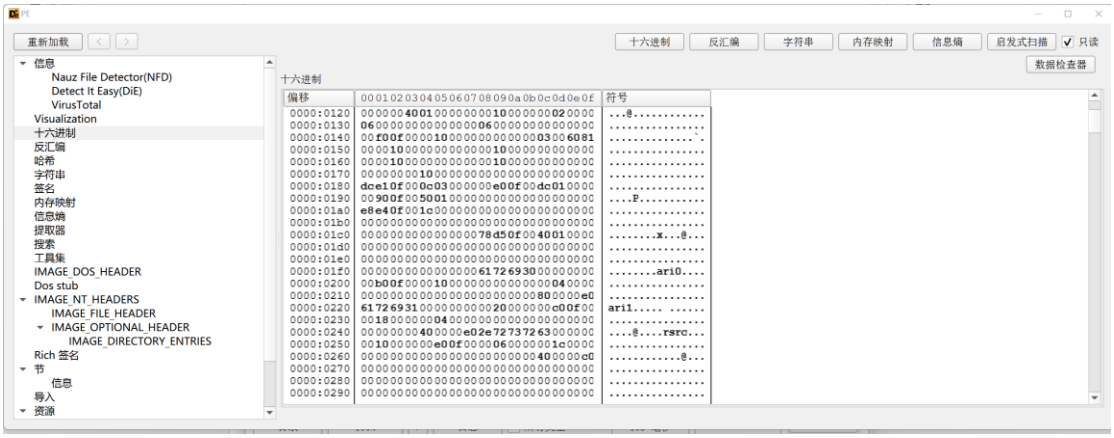
接下来直接去网站解就行了



4. arithmetic

（真解不出来，看看脱壳）

先尝试用脱壳机，发现脱不开壳，在 die 里面发现 UPX 的签名被改成 ari 了，用 010editor 改成 UPX 就能用脱壳机脱掉了


重新加载
十六进制
反汇编
字符串
内存映射
信息熵
启发式扫描
只读
数据检查器
信息
Nauz File Detector(NFD)
Detect It Easy(DiE)
VirusTotal
Visualization
十六进制
反汇编
哈希
字符串
签名
内存映射
信息熵
提取器
搜索
工具集
IMAGE_DOS_HEADER
Dos stub
IMAGE_NT_HEADERS
IMAGE_FILE_HEADER
IMAGE_OPTIONAL_HEADER
IMAGE_DIRECTORY_ENTRIES
Rich 签名
节
信息
导入
资源
十六进制
偏移
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
符号
0000:0120 00000040010000000010000000020000 ...@............
0000:0130 06000000000000000600000000000000 ....
0000:0140 00f00f0000010000000000000300608l ....`.
0000:0150 0000100000000000010000000000000 ....
0000:0160 0000100000000000010000000000000 ....
0000:0170 0000000010000000000000000000000 ....
0000:0180 dce10f000c030000e00f00dc0100000 ....
0000:0190 00900f0050010000000000000000000 ....P...........
0000:01a0 e8e40f001c000000000000000000000 ....
0000:01b0 0000000000000000000000000000000 ....
0000:01c0 0000000000000078d50f0040010000 ....x...@...
0000:01d0 0000000000000000000000000000000 ....
0000:01e0 0000000000000000000000000000000 ....
0000:01f0 0000000000000061726930000000000 ....ari0....
0000:0200 00b00f0000010000000000000040000 ....
0000:0210 0000000000000000000000800000e0 ....
0000:0220 6172693100000000020000000c00f00 ari1..... ......
0000:0230 0018000000040000000000000000000 ....
0000:0240 0000000040000e02e72737263000000 ....@....rsrc...
0000:0250 0010000000e00f0000060000001e0000 ....
0000:0260 0000000000000000000000400000e0 ....@....
0000:0270 0000000000000000000000000000000 ....
0000:0280 0000000000000000000000000000000 ....
0000:0290 0000000000000000000000000000000 ....