BUGKU-逆向(reverse)-writeup

转载

ahilll 于 2018-12-04 14:11:13 发布 2618 收藏 7

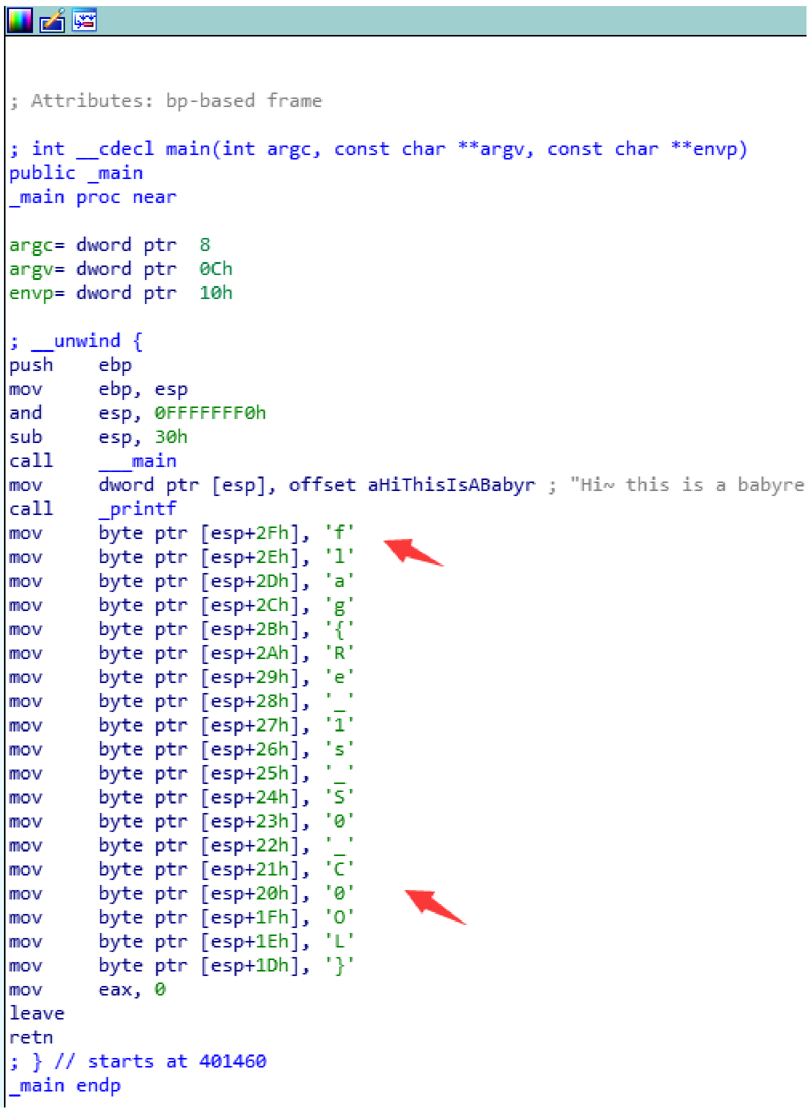
目录

- 入门逆向
- Easy_vb
- Easy_Re
- 游戏过关
- Timer(阿里CTF)
- 逆向入门
- love
- LoopAndLoop(阿里CTF)
- easy-100(LCTF)
- SafeBox(NJCTF)
- Mountain climbing

前言：在bugku上把能写的逆向都写了，由于大佬们的writeup太深奥或者说太简洁了让我(小白)看得云里雾里。所以我写了这个详细点的writeup（理解错的地方望指出），尽量让大家都看得懂。最近比较忙先写到了这里，未完待续

入门逆向

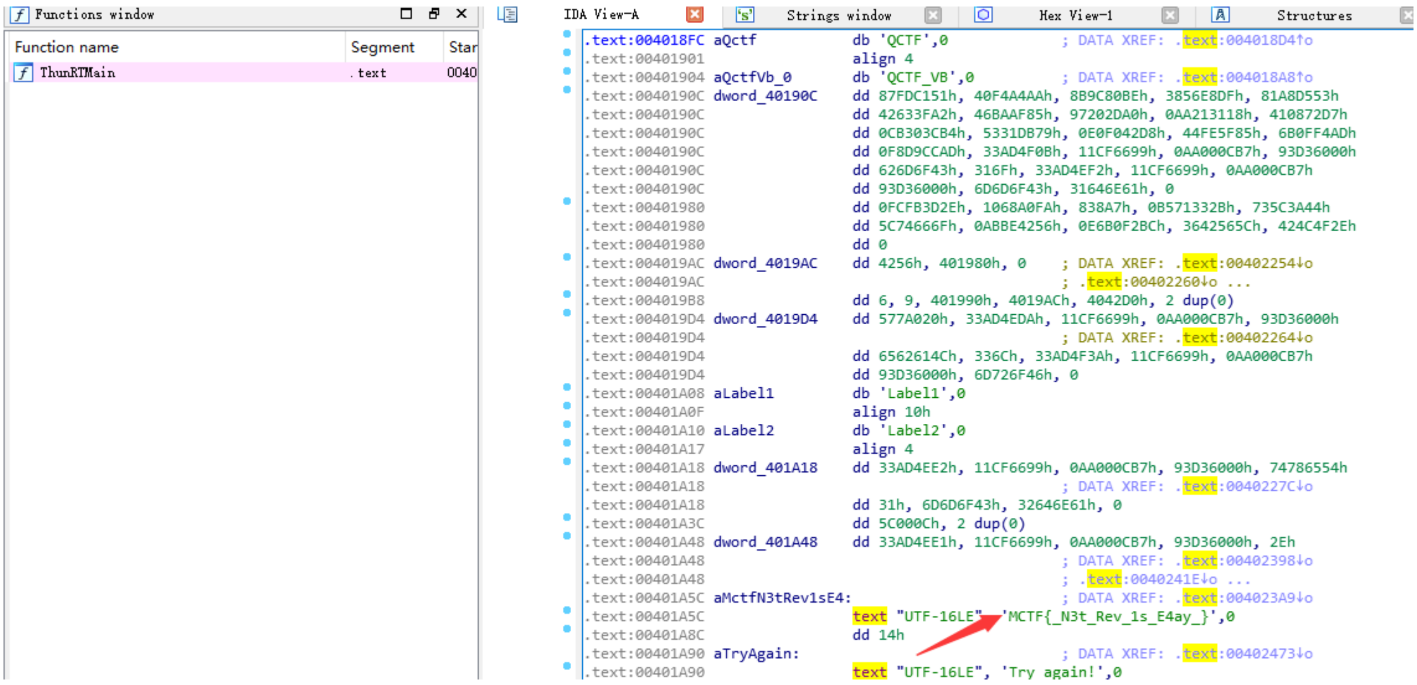
下载后ida打开,双击`_mail`函数里就有flag



```
; Attributes: bp-based frame

; int __cdecl main(int argc, const char **argv, const char **envp)
public _main
_main proc near

argc= dword ptr  8
argv= dword ptr  0Ch
envp= dword ptr  10h

; __unwind {
push    ebp
mov     ebp, esp
and     esp, 0FFFFFFF0h
sub     esp, 30h
call    ___main
mov     dword ptr [esp], offset aHiThisIsABabyr ; "Hi~ this is a babyre
call    _printf
mov     byte ptr [esp+2Fh], 'f'
mov     byte ptr [esp+2Eh], 'l'
mov     byte ptr [esp+2Dh], 'a'
mov     byte ptr [esp+2Ch], 'g'
mov     byte ptr [esp+2Bh], '{'
mov     byte ptr [esp+2Ah], 'R'
mov     byte ptr [esp+29h], 'e'
mov     byte ptr [esp+28h], '_'
mov     byte ptr [esp+27h], '1'
mov     byte ptr [esp+26h], 's'
mov     byte ptr [esp+25h], '_'
mov     byte ptr [esp+24h], 'S'
mov     byte ptr [esp+23h], '0'
mov     byte ptr [esp+22h], '_'
mov     byte ptr [esp+21h], 'C'
mov     byte ptr [esp+20h], '0'
mov     byte ptr [esp+1Fh], 'O'
mov     byte ptr [esp+1Eh], 'L'
mov     byte ptr [esp+1Dh], '}'
mov     eax, 0
leave
retn
; } // starts at 401460
_main endp
```

Easy_vb

下载后ida打开,往下翻里就有flag



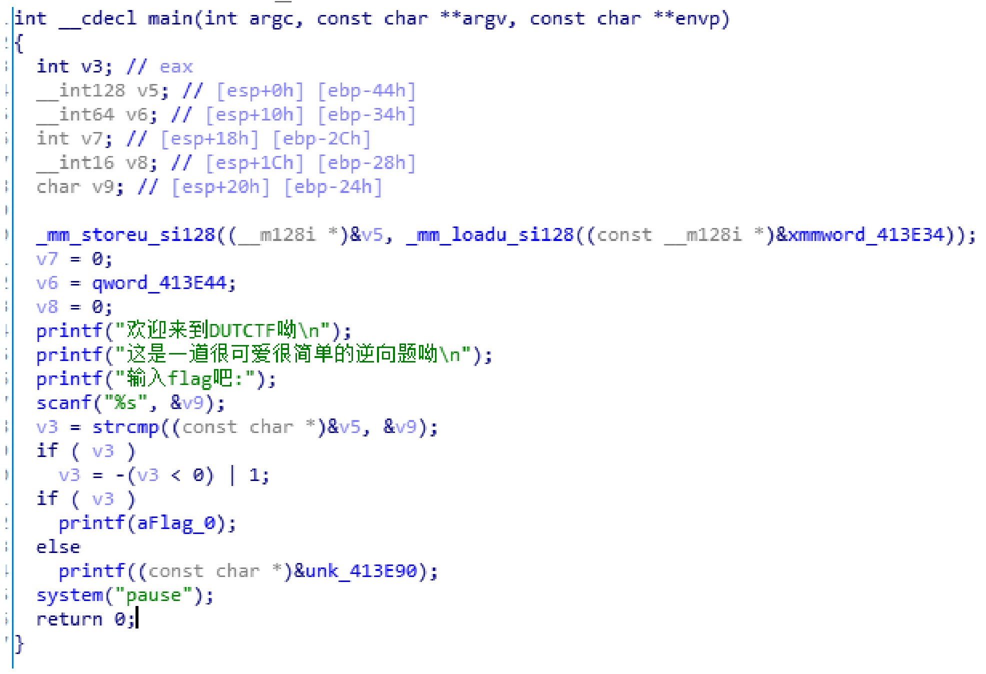
```
Functions window
Function name     Segment   Star
ThunRTMain        .text     0040
```

```
IDA View-A        Strings window        Hex View-1        Structures
.text:004018FC aQctf           db 'QCTF',0             ; DATA XREF: .text:004018D4↑o
.text:00401901                 align 4
.text:00401904 aQctfVb_0       db 'QCTF_VB',0          ; DATA XREF: .text:004018A8↑o
.text:0040190C dword_40190C    dd 87FDC151h, 40F4A4AAh, 8B9C80BEh, 3856E8DFh, 81A8D553h
.text:0040190C                 dd 42633FA2h, 46BAAF85h, 97202DA0h, 0AA213118h, 410872D7h
.text:0040190C                 dd 0CB303CB4h, 5331DB79h, 0E0F042D8h, 44FE5F85h, 6B0FF4ADh
.text:0040190C                 dd 0F8D9CCADh, 33AD4F0Bh, 11CF6699h, 0AA000CB7h, 93D36000h
.text:0040190C                 dd 626D6F43h, 316Fh, 33AD4EF2h, 11CF6699h, 0AA000CB7h
.text:0040190C                 dd 93D36000h, 6D6D6F43h, 31646E61h, 0
.text:00401980                 dd 0FCFB3D2Eh, 1068A0FAh, 838A7h, 0B571332Bh, 735C3A44h
.text:00401980                 dd 5C74666Fh, 0ABBE4256h, 0E6B0F2BCh, 3642565Ch, 424C4F2Eh
.text:00401980                 dd 0
.text:004019AC dword_4019AC    dd 4256h, 401980h, 0    ; DATA XREF: .text:00402254↓o
.text:004019AC                                         ; .text:00402260↓o ...
.text:004019B8                 dd 6, 9, 401990h, 4019ACh, 4042D0h, 2 dup(0)
.text:004019D4 dword_4019D4    dd 577A020h, 33AD4EDAh, 11CF6699h, 0AA000CB7h, 93D36000h
.text:004019D4                                         ; DATA XREF: .text:00402264↓o
.text:004019D4                 dd 6562614Ch, 336Ch, 33AD4F3Ah, 11CF6699h, 0AA000CB7h
.text:004019D4                 dd 93D36000h, 6D726F46h, 0
.text:00401A08 aLabel1         db 'Label1',0
.text:00401A0F                 align 10h
.text:00401A10 aLabel2         db 'Label2',0
.text:00401A17                 align 4
.text:00401A18 dword_401A18    dd 33AD4EE2h, 11CF6699h, 0AA000CB7h, 93D36000h, 74786554h
.text:00401A18                                         ; DATA XREF: .text:0040227C↓o
.text:00401A18                 dd 31h, 6D6D6F43h, 32646E61h, 0
.text:00401A3C                 dd 5C000Ch, 2 dup(0)
.text:00401A48 dword_401A48    dd 33AD4EE1h, 11CF6699h, 0AA000CB7h, 93D36000h, 2Eh
.text:00401A48                                         ; DATA XREF: .text:00402398↓o
.text:00401A48                                         ; .text:0040241E↓o ...
.text:00401A5C aMctfN3tRev1sE4:                        ; DATA XREF: .text:004023A9↓o
.text:00401A5C                 text "UTF-16LE", 'MCTF{_N3t_Rev_1s_E4ay_}',0
.text:00401A8C                 dd 14h
.text:00401A90 aTryAgain:                              ; DATA XREF: .text:00402473↓o
.text:00401A90                 text "UTF-16LE", 'Try again!',0
```

提交flag出错，将MCTF改成flag即可。

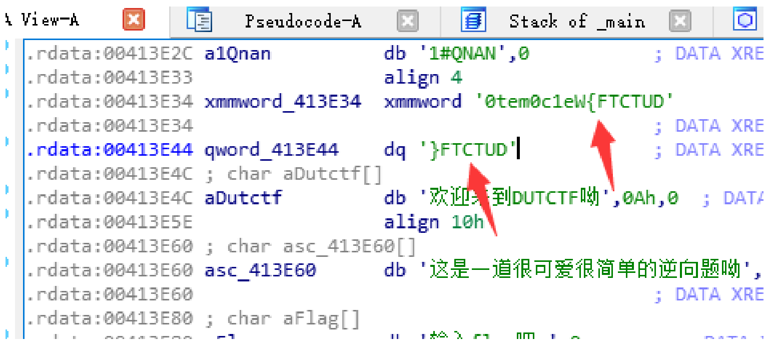
Easy_Re

下载后ida打开，双击`_mail`函数，F5翻译为伪C代码



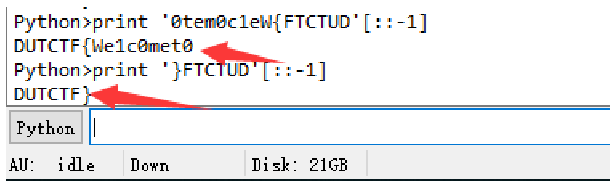
```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // eax
  __int128 v5; // [esp+0h] [ebp-44h]
  __int64 v6; // [esp+10h] [ebp-34h]
  int v7; // [esp+18h] [ebp-2Ch]
  __int16 v8; // [esp+1Ch] [ebp-28h]
  char v9; // [esp+20h] [ebp-24h]

  _mm_storeu_si128((__m128i *)&v5, _mm_loadu_si128((const __m128i *)&xmmword_413E34));
  v7 = 0;
  v6 = qword_413E44;
  v8 = 0;
  printf("欢迎来到DUTCTF呦\n");
  printf("这是一道很可爱很简单的逆向题呦\n");
  printf("输入flag吧:");
  scanf("%s", &v9);
  v3 = strcmp((const char *)&v5, &v9);
  if ( v3 )
    v3 = -(v3 < 0) | 1;
  if ( v3 )
    printf(aFlag_0);
  else
    printf((const char *)&unk_413E90);
  system("pause");
  return 0;
}
```

strcmp()对面输入的值是否等于xmmword_413E34位置的值，双击跟过去，发现了flag



```
.rdata:00413E2C a1Qnan          db '1#QNAN',0           ; DATA XRE
.rdata:00413E33                 align 4
.rdata:00413E34 xmmword_413E34  xmmword '0tem0c1eW{FTCTUD'
.rdata:00413E34                                         ; DATA XRE
.rdata:00413E44 qword_413E44    dq '}FTCTUD'            ; DATA XRE
.rdata:00413E4C ; char aDutctf[]
.rdata:00413E4C aDutctf         db '欢迎来到DUTCTF呦',0Ah,0  ; DATA
.rdata:00413E5E                 align 10h
.rdata:00413E60 ; char asc_413E60[]
.rdata:00413E60 asc_413E60      db '这是一道很可爱很简单的逆向题呦',
.rdata:00413E60                                         ; DATA XRE
.rdata:00413E80 ; char aFlag[]
```

小端存储的问题，看起来反了而已。

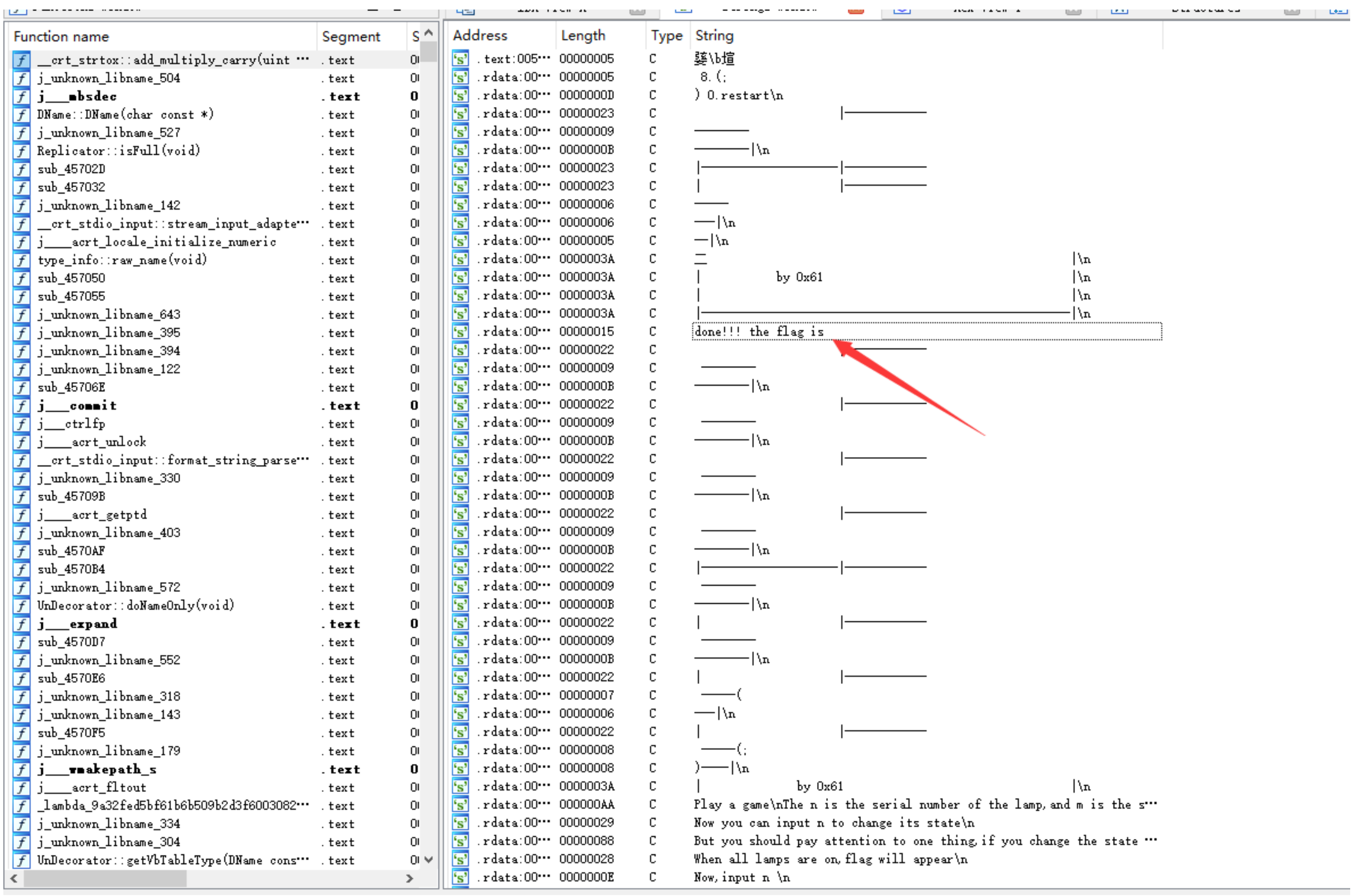


```
Python>print '0tem0c1eW{FTCTUD'[::-1]
DUTCTF{We1c0met0
Python>print '}FTCTUD'[::-1]
DUTCTF}
```

游戏过关

下载后ida打开，看到函数比较多，分享一种快速找关键函数的方法。

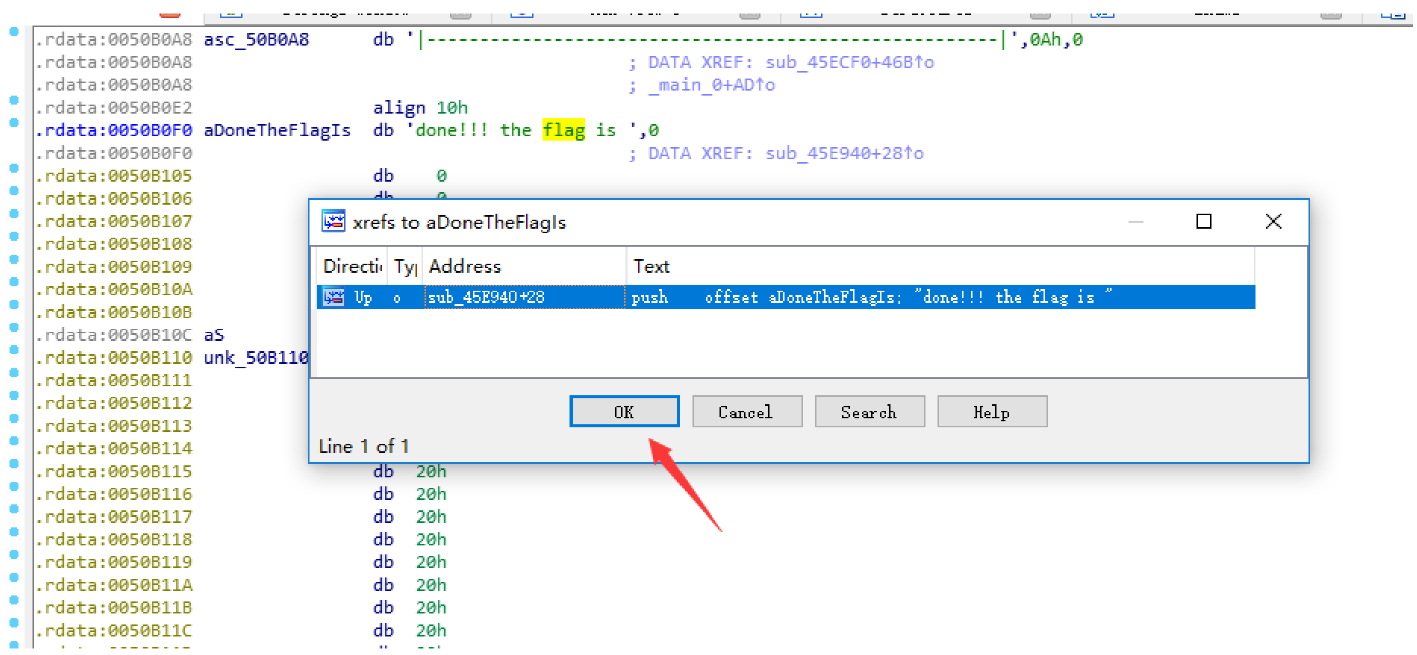
首先就是看运行遍程序，了解下程序流程以及关键字符串。然后打开ida
1.`Shift+F12`查看下字符串。



```
Function name                                  Segment  S
__crt_strtox::add_multiply_carry(uint …        .text    0
j_unknown_libname_504                          .text    0
j___mbsdec                                     .text    0
DName::DName(char const *)                     .text    0
j_unknown_libname_527                          .text    0
Replicator::isFull(void)                       .text    0
sub_45702D                                     .text    0
sub_457032                                     .text    0
j_unknown_libname_142                          .text    0
__crt_stdio_input::stream_input_adapte…        .text    0
j____acrt_locale_initialize_numeric            .text    0
type_info::raw_name(void)                      .text    0
sub_457050                                     .text    0
sub_457055                                     .text    0
j_unknown_libname_643                          .text    0
j_unknown_libname_395                          .text    0
j_unknown_libname_394                          .text    0
j_unknown_libname_122                          .text    0
sub_45706E                                     .text    0
j___commit                                     .text    0
j___ctrlfp                                     .text    0
j____acrt_unlock                               .text    0
__crt_stdio_input::format_string_parse…        .text    0
j_unknown_libname_330                          .text    0
sub_45709B                                     .text    0
j____acrt_getptd                               .text    0
j_unknown_libname_403                          .text    0
sub_4570AF                                     .text    0
sub_4570B4                                     .text    0
j_unknown_libname_572                          .text    0
UnDecorator::doNameOnly(void)                  .text    0
j___expand                                     .text    0
sub_4570D7                                     .text    0
j_unknown_libname_552                          .text    0
sub_4570E6                                     .text    0
j_unknown_libname_318                          .text    0
j_unknown_libname_143                          .text    0
sub_4570F5                                     .text    0
j_unknown_libname_179                          .text    0
j___wmakepath_s                                .text    0
j____acrt_fltout                               .text    0
_lambda_9a32fed5bf61b6b509b2d3f6003082…        .text    0
j_unknown_libname_334                          .text    0
j_unknown_libname_304                          .text    0
UnDecorator::getVbTableType(DName cons…        .text    0
```

```
Address        Length    Type  String
.text:005…     00000005  C     婁\b擅
.rdata:00…     00000005  C      8.(;
.rdata:00…     0000000D  C     ) 0.restart\n
.rdata:00…     00000023  C                     |──────────
.rdata:00…     00000009  C     ──────
.rdata:00…     0000000B  C     ──────|\n
.rdata:00…     00000023  C     |───────────────|──────────
.rdata:00…     00000023  C     |               |──────────
.rdata:00…     00000006  C     ────
.rdata:00…     00000006  C     ──|\n
.rdata:00…     00000005  C     ─|\n
.rdata:00…     0000003A  C     二                                            |\n
.rdata:00…     0000003A  C     |         by 0x61                            |\n
.rdata:00…     0000003A  C     |                                            |\n
.rdata:00…     0000003A  C     |────────────────────────────────────────────|\n
.rdata:00…     00000015  C     done!!! the flag is
.rdata:00…     00000022  C                     |──────────
.rdata:00…     00000009  C      ──────
.rdata:00…     0000000B  C     ──────|\n
.rdata:00…     00000022  C                     |──────────
.rdata:00…     00000009  C      ──────
.rdata:00…     0000000B  C     ──────|\n
.rdata:00…     00000022  C                     |──────────
.rdata:00…     00000009  C      ──────
.rdata:00…     0000000B  C     ──────|\n
.rdata:00…     00000022  C                     |──────────
.rdata:00…     00000009  C      ──────
.rdata:00…     0000000B  C     ──────|\n
.rdata:00…     00000022  C     |───────────────|──────────
.rdata:00…     00000009  C      ──────
.rdata:00…     0000000B  C     ──────|\n
.rdata:00…     00000022  C     |               |──────────
.rdata:00…     00000009  C      ──────
.rdata:00…     0000000B  C     ──────|\n
.rdata:00…     00000022  C     |               |──────────
.rdata:00…     00000007  C      ────(
.rdata:00…     00000006  C     ──|\n
.rdata:00…     00000022  C     |               |──────────
.rdata:00…     00000008  C      ────(;
.rdata:00…     00000008  C     )───|\n
.rdata:00…     0000003A  C     |          by 0x61                           |\n
.rdata:00…     000000AA  C     Play a game\nThe n is the serial number of the lamp,and m is the s…
.rdata:00…     00000029  C     Now you can input n to change its state\n
.rdata:00…     00000088  C     But you should pay attention to one thing,if you change the state …
.rdata:00…     00000028  C     When all lamps are on,flag will appear\n
.rdata:00…     0000000E  C     Now,input n \n
```

2.然后双击过去。

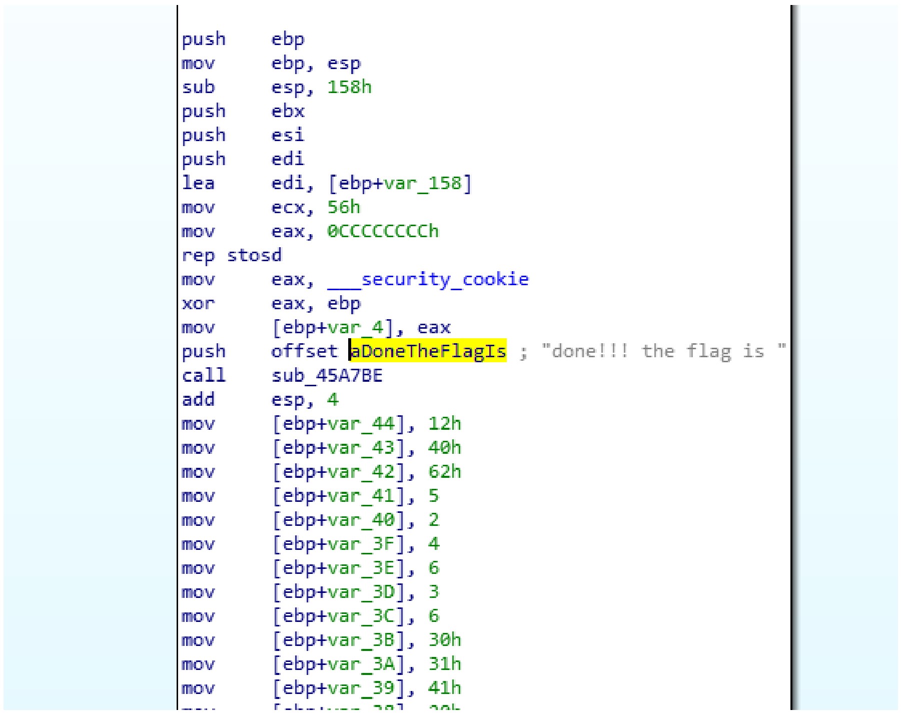


```
.rdata:0050B0A8 asc_50B0A8      db '|-----------------------------------------|',0Ah,0
.rdata:0050B0A8                                         ; DATA XREF: sub_45ECF0+46B↑o
.rdata:0050B0A8                                         ; _main_0+AD↑o
.rdata:0050B0E2                 align 10h
.rdata:0050B0F0 aDoneTheFlagIs  db 'done!!! the flag is ',0
.rdata:0050B0F0                                         ; DATA XREF: sub_45E940+28↑o
.rdata:0050B105                 db    0
.rdata:0050B106                 db    0
.rdata:0050B107
.rdata:0050B108
.rdata:0050B109
.rdata:0050B10A
.rdata:0050B10B
.rdata:0050B10C aS
.rdata:0050B110 unk_50B110
.rdata:0050B111
.rdata:0050B112
.rdata:0050B113
.rdata:0050B114
.rdata:0050B115                 db  20h
.rdata:0050B116                 db  20h
.rdata:0050B117                 db  20h
.rdata:0050B118                 db  20h
.rdata:0050B119                 db  20h
.rdata:0050B11A                 db  20h
.rdata:0050B11B                 db  20h
.rdata:0050B11C                 db  20h
```

xrefs to aDoneTheFlagIs

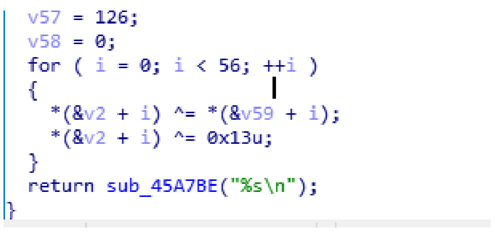
Directi	Ty	Address	Text
Up	o	sub_45E940+28	push offset aDoneTheFlagIs; "done!!! the flag is "

3.再按Cirt+X交叉引用显示调用位置



```
push    ebp
mov     ebp, esp
sub     esp, 158h
push    ebx
push    esi
push    edi
lea     edi, [ebp+var_158]
mov     ecx, 56h
mov     eax, 0CCCCCCCCh
rep stosd
mov     eax, ___security_cookie
xor     eax, ebp
mov     [ebp+var_4], eax
push    offset aDoneTheFlagIs ; "done!!! the flag is "
call    sub_45A7BE
add     esp, 4
mov     [ebp+var_44], 12h
mov     [ebp+var_43], 40h
mov     [ebp+var_42], 62h
mov     [ebp+var_41], 5
mov     [ebp+var_40], 2
mov     [ebp+var_3F], 4
mov     [ebp+var_3E], 6
mov     [ebp+var_3D], 3
mov     [ebp+var_3C], 6
mov     [ebp+var_3B], 30h
mov     [ebp+var_3A], 31h
mov     [ebp+var_39], 41h
```

然后F5看下伪代码



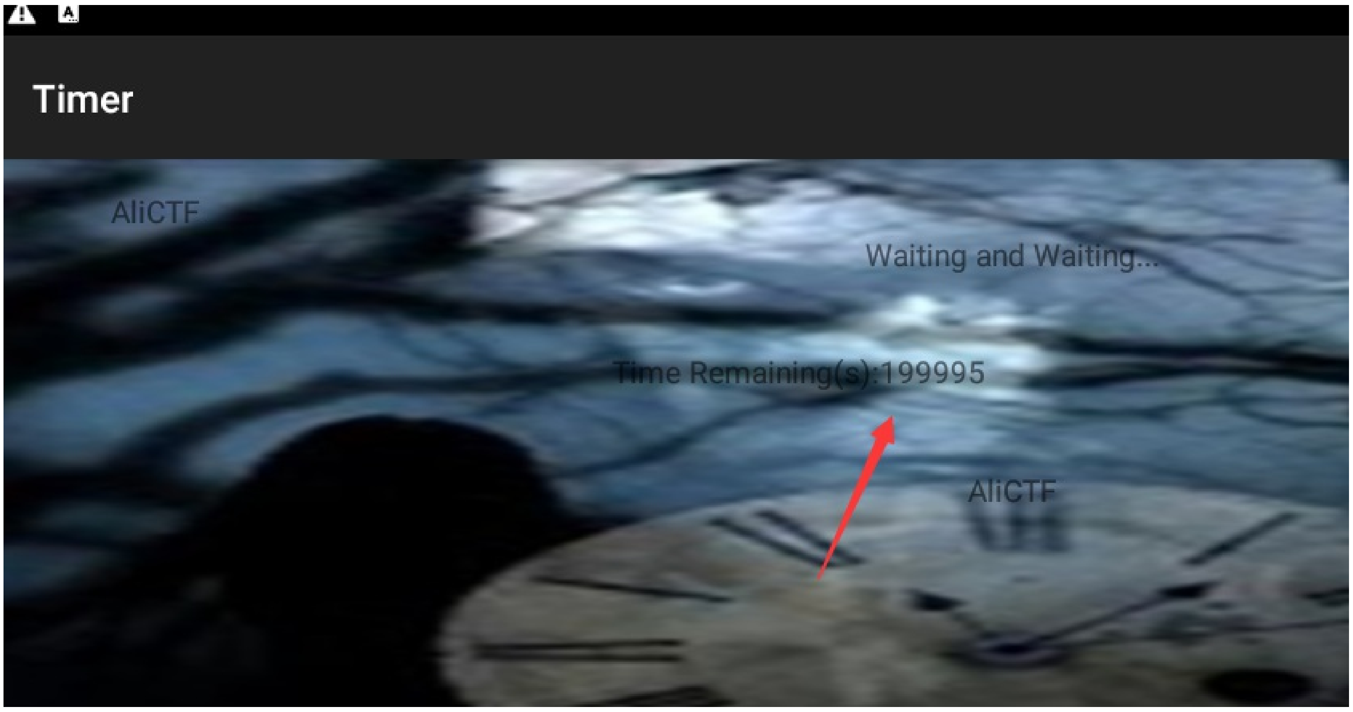
```
  v57 = 126;
  v58 = 0;
  for ( i = 0; i < 56; ++i )
  {
    *(&v2 + i) ^= *(&v59 + i);
    *(&v2 + i) ^= 0x13u;
  }
  return sub_45A7BE("%s\n");
}
```

打印出`done!!! the flag is` 然后有两个数组按位异或再和0x13异或生成flag

```
#!usr/bin/env python
#!coding=utf-8

__author__ = 'zhengjim'

array1 = [18,64,98,5,2,4,6,3,6,48,49,65,32,12,48,65,31,78,62,32,49,32,1,57,96,3,21,9,4,62,3,5,4,1,2,3,44,65
array2 = [123,32,18,98,119,108,65,41,124,80,125,38,124,111,74,49,83,108,94,108,84,6,96,83,44,121,104,110,32

flag = ''
for i in range(len(array1)):
    flag+= chr(array1[i] ^ array2[i] ^ 0x13 )
print flag
```

Timer(阿里CTF)

下载文件发现是apk，先安装运行下发现有一个倒计时，只是时间为200000秒。猜测是让时间走完获取flag。



用jadx-gui反编译，双击看MainActivity查看

```java
package net.bluelotus.tomorrow.easyandroid;

import android.os.Bundle;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    int beg = (((int) (System.currentTimeMillis() / 1000)) + 200000);
    int k = 0;
    int now;
    long t = 0;

    public native String stringFromJNI2(int i);

    public static boolean is2(int n) {
        if (n <= 3) {
            if (n > 1) {
                return true;
            }
            return false;
        } else if (n % 2 == 0 || n % 3 == 0) {
            return false;
        } else {
            int i = 5;
            while (i * i <= n) {
                if (n % i == 0 || n % (i + 2) == 0) {
                    return false;
                }
                i += 6;
            }
            return true;
        }
    }
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView((int) R.layout.activity_main);
        final TextView tv1 = (TextView) findViewById(R.id.textView2);
        final TextView tv2 = (TextView) findViewById(R.id.textView3);
        final Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
            public void run() {
                MainActivity.this.t = System.currentTimeMillis();
                MainActivity.this.now = (int) (MainActivity.this.t / 1000);
                MainActivity.this.t = 1500 - (MainActivity.this.t % 1000);
                tv2.setText("AliCTF");
                if (MainActivity.this.beg - MainActivity.this.now <= 0) {
                    tv1.setText("The flag is:");
                    tv2.setText("alictf{" + MainActivity.this.stringFromJNI2(MainActivity.this.k) + "}");
                }
                MainActivity mainActivity;
                if (MainActivity.is2(MainActivity.this.beg - MainActivity.this.now)) {
                    mainActivity = MainActivity.this;
                    mainActivity.k += 100;
                } else {
                    mainActivity = MainActivity.this;
                    mainActivity.k--;
                }
                tv1.setText("Time Remaining(s):" + (MainActivity.this.beg - MainActivity.this.now));
                handler.postDelayed(this, MainActivity.this.t);
            }
        }, 0);
    }

    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

    static {
        System.loadLibrary("lhm");
    }
}
```

首先初始化了beg为当前时间加上200000。`(System.currentTimeMillis() / 1000)`是获得系统的时间，单位为毫秒,转换为秒。

看`onCreate`方法，找到关键处

```
if (MainActivity.this.beg - MainActivity.this.now <= 0) {
                    tv1.setText("The flag is:");
                    tv2.setText("alictf{" + MainActivity.this.stringFromJNI2(MainActivity.this.k) + "}");
                }
```

所以`MainActivity.this.beg - MainActivity.this.now <= 0`就是过了得时间。如果过了200000秒则出现flag。flag是使用native层来打印。
思路：能不能直接跳过200000秒直接出现flag呢？
有一个关键变量`k`，往下看，看看k有没有什么运算。

```
 if (MainActivity.is2(MainActivity.this.beg - MainActivity.this.now)) {
                    mainActivity = MainActivity.this;
                    mainActivity.k += 100;
                } else {
                    mainActivity = MainActivity.this;
                    mainActivity.k--;
                }
```

将差值用is2函数判断，如果true，就k+100 ,如果false,就k-1。那就要看下is2函数

```
public static boolean is2(int n) {
        if (n <= 3) {
            if (n > 1) {
                return true;
            }
            return false;
        } else if (n % 2 == 0 || n % 3 == 0) {
            return false;
        } else {
            int i = 5;
            while (i * i <= n) {
                if (n % i == 0 || n % (i + 2) == 0) {
                    return false;
                }
                i += 6;
            }
            return true;
        }
    }
```

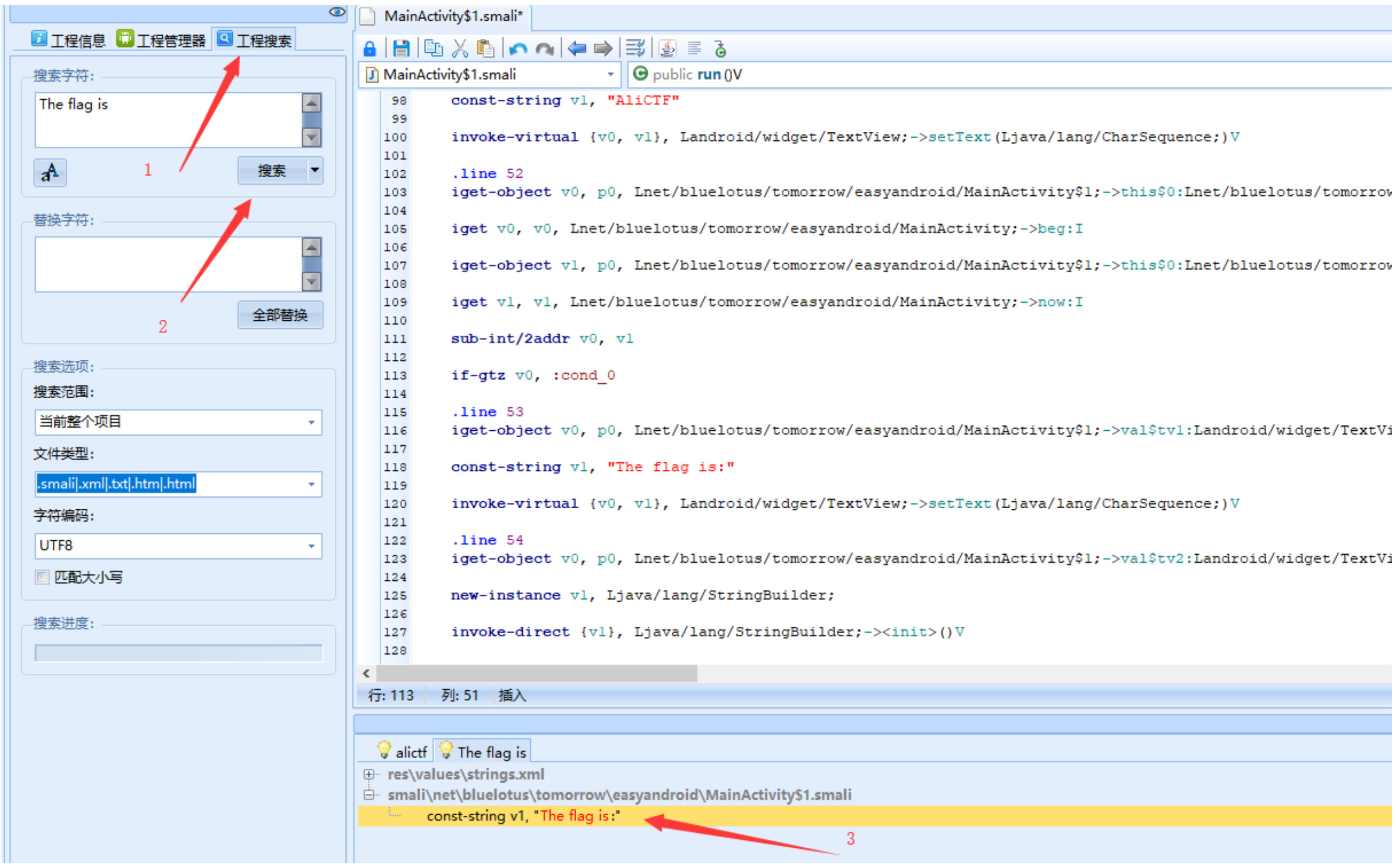
直接照着写一个即可，然后可以算出关键变量`k`
解密脚本

```
#!usr/bin/env python
#!coding=utf-8

__author__ = 'zhengjim'

def is2(n):
    if(n <= 3):
        if(n > 1):
            return True
        return False
    elif(n % 2 == 0 or n % 3 == 0):
        return False
    else:
        i = 5
        while(i * i <= n):
            if (n % i == 0 or n % (i + 2) == 0):
                return False
            i += 6
    return True

k=0

for i in xrange(200000,0,-1):
    k = k + 100 if is2(i) else k - 1
print k
```

算出`k = 1616384`

然后就可以绕过200000秒将k带入传入进去获取flag。

实现的话，用`Androidkiller`打开项目，因为跳转后输出了`The flag is`,所以搜索该字符串,双击跟过去。



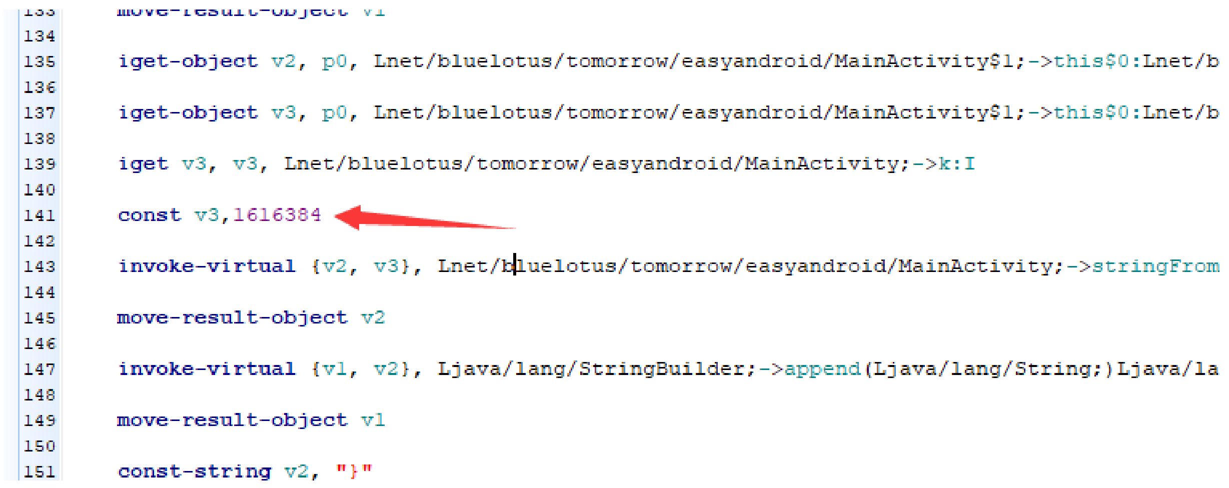
搜索字符: The flag is
替换字符:
搜索选项:
搜索范围: 当前整个项目
文件类型: .smali|.xml|.txt|.htm|.html
字符编码: UTF8
☐ 匹配大小写
搜索进度:

```
const-string v1, "AliCTF"

invoke-virtual {v0, v1}, Landroid/widget/TextView;->setText(Ljava/lang/CharSequence;)V

.line 52
iget-object v0, p0, Lnet/bluelotus/tomorrow/easyandroid/MainActivity$1;->this$0:Lnet/bluelotus/tomorrow

iget v0, v0, Lnet/bluelotus/tomorrow/easyandroid/MainActivity;->beg:I

iget-object v1, p0, Lnet/bluelotus/tomorrow/easyandroid/MainActivity$1;->this$0:Lnet/bluelotus/tomorrow

iget v1, v1, Lnet/bluelotus/tomorrow/easyandroid/MainActivity;->now:I

sub-int/2addr v0, v1

if-gtz v0, :cond_0

.line 53
iget-object v0, p0, Lnet/bluelotus/tomorrow/easyandroid/MainActivity$1;->val$tv1:Landroid/widget/TextVi

const-string v1, "The flag is:"

invoke-virtual {v0, v1}, Landroid/widget/TextView;->setText(Ljava/lang/CharSequence;)V

.line 54
iget-object v0, p0, Lnet/bluelotus/tomorrow/easyandroid/MainActivity$1;->val$tv2:Landroid/widget/TextVi

new-instance v1, Ljava/lang/StringBuilder;

invoke-direct {v1}, Ljava/lang/StringBuilder;-><init>()V
```

```
res\values\strings.xml
smali\net\bluelotus\tomorrow\easyandroid\MainActivity$1.smali
    const-string v1, "The flag is:"
```

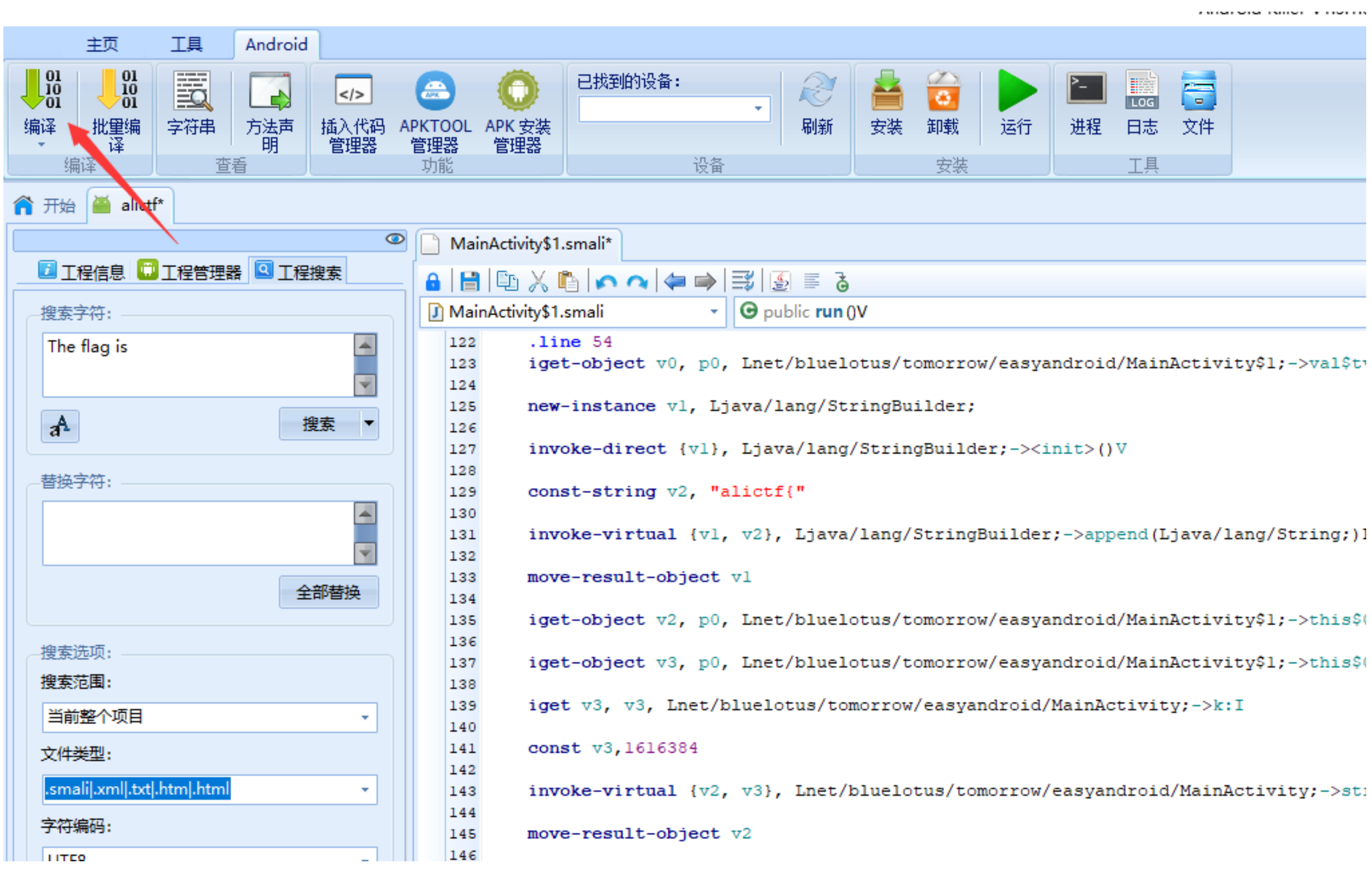
往上看第113行的if-gtz v0, :cond_0。if-ltz是如果大于0跳转，那改成如果小于0跳转就跳过了200000秒等待了。对应的语句为if-ltz v0, :cond_0。

然后要找到赋值k的位置，看第129行-149行，因为k的值是在alictf{和}之间传入的。
看到了139行的的iget v3, v3, Lnet/bluelotus/tomorrow/easyandroid/MainActivity;->k:I，知道v3是k的值。
于是在下面赋值const v3,1616384



```
133     move-result-object v1
134
135     iget-object v2, p0, Lnet/bluelotus/tomorrow/easyandroid/MainActivity$1;->this$0:Lnet/b
136
137     iget-object v3, p0, Lnet/bluelotus/tomorrow/easyandroid/MainActivity$1;->this$0:Lnet/b
138
139     iget v3, v3, Lnet/bluelotus/tomorrow/easyandroid/MainActivity;->k:I
140
141     const v3,1616384
142
143     invoke-virtual {v2, v3}, Lnet/bluelotus/tomorrow/easyandroid/MainActivity;->stringFrom
144
145     move-result-object v2
146
147     invoke-virtual {v1, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/la
148
149     move-result-object v1
150
151     const-string v2, "}"
```

然后保存，编译，安装运行就出现flag。



搜索字符:
The flag is

搜索

替换字符:

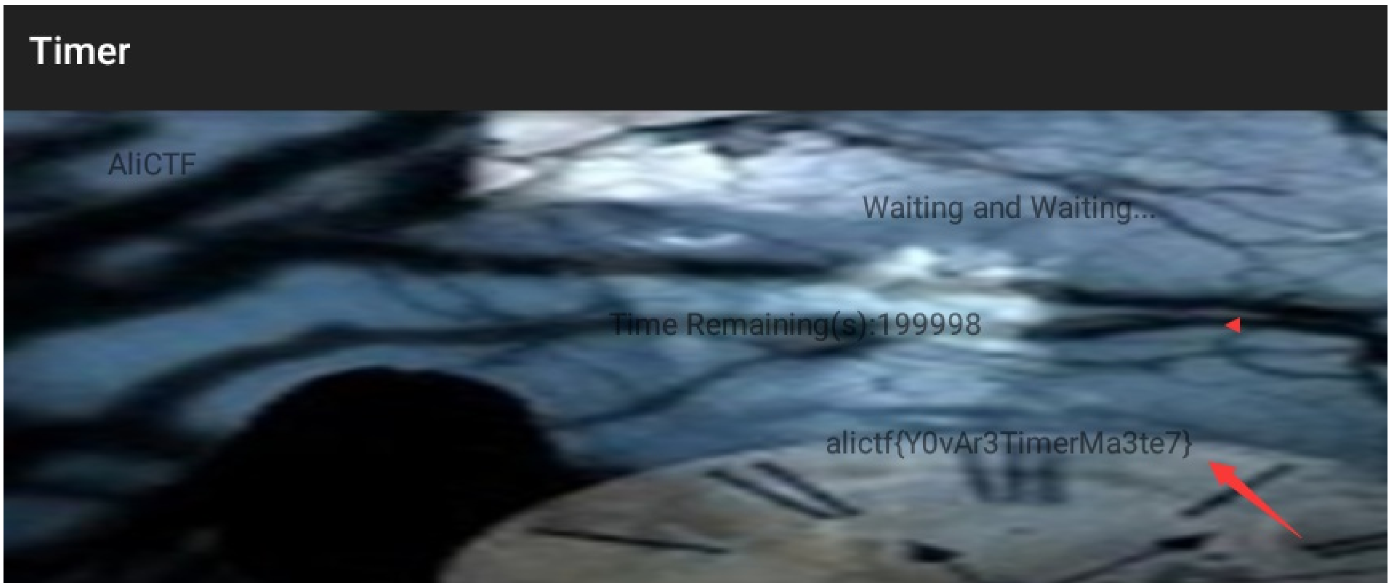
全部替换

搜索选项:
搜索范围:
当前整个项目
文件类型:
.smali|.xml|.txt|.htm|.html
字符编码:
UTF8

MainActivity$1.smali   public run()V

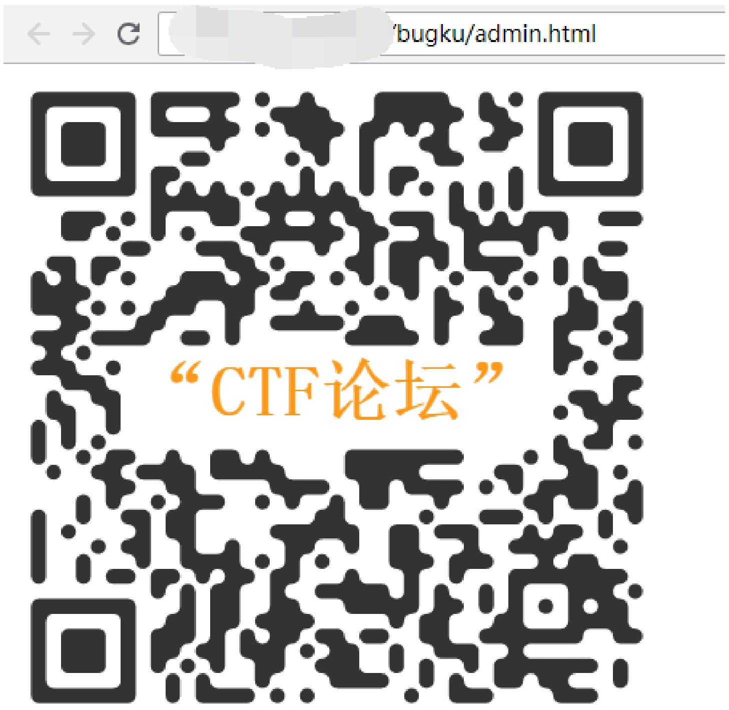
```
122     .line 54
123     iget-object v0, p0, Lnet/bluelotus/tomorrow/easyandroid/MainActivity$1;->val$t
124
125     new-instance v1, Ljava/lang/StringBuilder;
126
127     invoke-direct {v1}, Ljava/lang/StringBuilder;-><init>()V
128
129     const-string v2, "alictf{"
130
131     invoke-virtual {v1, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)L
132
133     move-result-object v1
134
135     iget-object v2, p0, Lnet/bluelotus/tomorrow/easyandroid/MainActivity$1;->this$
136
137     iget-object v3, p0, Lnet/bluelotus/tomorrow/easyandroid/MainActivity$1;->this$
138
139     iget v3, v3, Lnet/bluelotus/tomorrow/easyandroid/MainActivity;->k:I
140
141     const v3,1616384
142
143     invoke-virtual {v2, v3}, Lnet/bluelotus/tomorrow/easyandroid/MainActivity;->st
144
145     move-result-object v2
146
```

flag



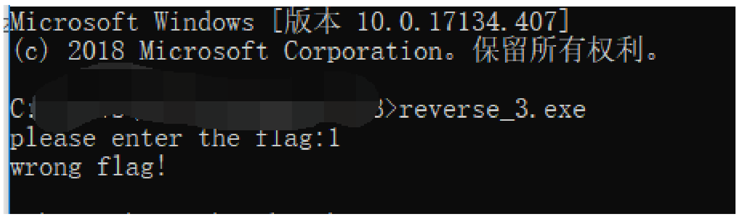
逆向入门

下载后发现不是pe文件，右键txt打开，看到`data:image/png;base64,iVBORXXXXXX...`开头的，为图像文件。
开头添加`<img scr="`，结尾添加`">`，html打开。有二维码扫描既可。



love

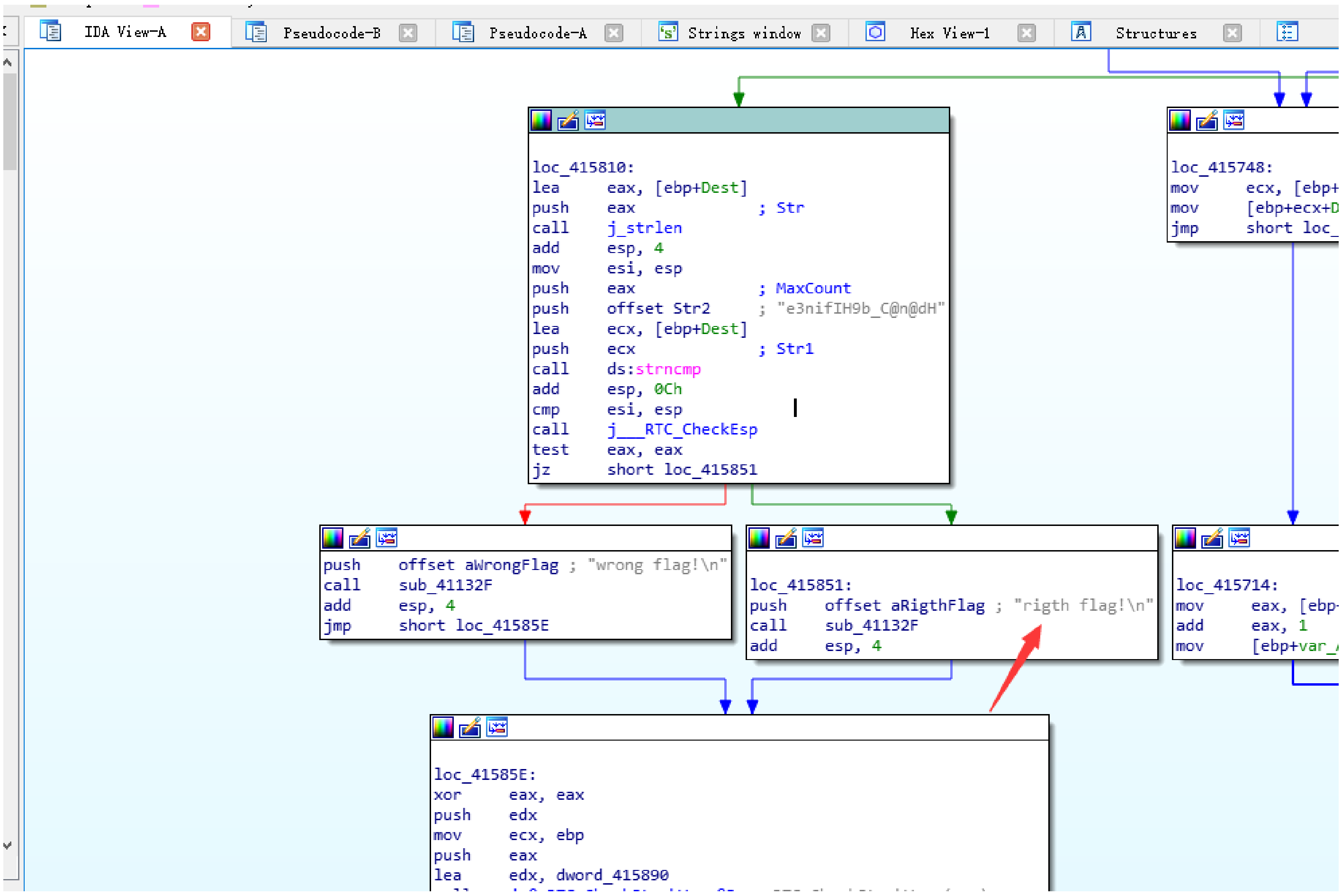
下载来用peid看是C++的，先运行下。



```
Microsoft Windows [版本 10.0.17134.407]
(c) 2018 Microsoft Corporation。保留所有权利。

C:[redacted]3>reverse_3.exe
please enter the flag:1
wrong flag!
```

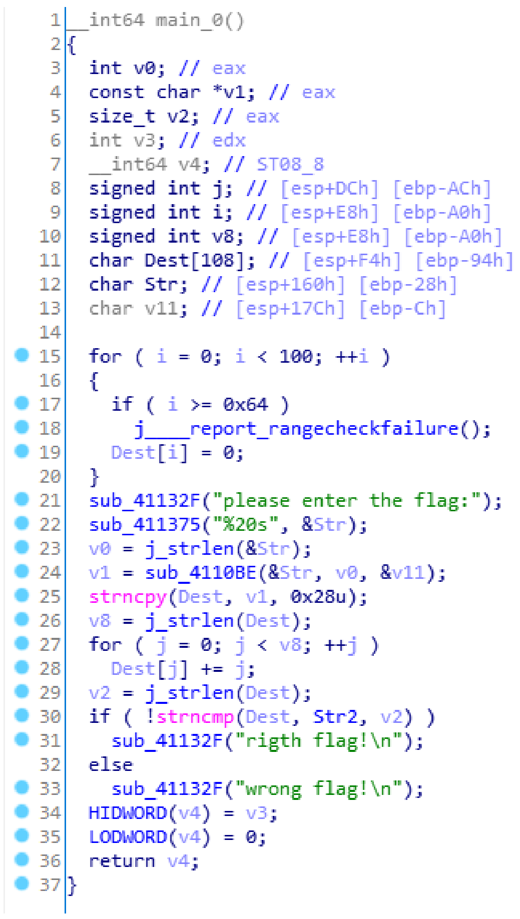
要输入flag。用ida打开

按之前说的方法，快速定位到关键函数



```
loc_415810:
lea     eax, [ebp+Dest]
push    eax             ; Str
call    j_strlen
add     esp, 4
mov     esi, esp
push    eax             ; MaxCount
push    offset Str2     ; "e3nifIH9b_C@n@dH"
lea     ecx, [ebp+Dest]
push    ecx             ; Str1
call    ds:strncmp
add     esp, 0Ch
cmp     esi, esp
call    j___RTC_CheckEsp
test    eax, eax
jz      short loc_415851

push    offset aWrongFlag ; "wrong flag!\n"
call    sub_41132F
add     esp, 4
jmp     short loc_41585E

loc_415851:
push    offset aRigthFlag ; "rigth flag!\n"
call    sub_41132F
add     esp, 4

loc_41585E:
xor     eax, eax
push    edx
mov     ecx, ebp
push    eax
lea     edx, dword_415890

loc_415748:
mov     ecx, [ebp+
mov     [ebp+ecx+D
jmp     short loc_

loc_415714:
mov     eax, [ebp+
add     eax, 1
mov     [ebp+var_
```

打F5查看伪代码



```
__int64 main_0()
{
  int v0; // eax
  const char *v1; // eax
  size_t v2; // eax
  int v3; // edx
  __int64 v4; // ST08_8
  signed int j; // [esp+DCh] [ebp-ACh]
  signed int i; // [esp+E8h] [ebp-A0h]
  signed int v8; // [esp+E8h] [ebp-A0h]
  char Dest[108]; // [esp+F4h] [ebp-94h]
  char Str; // [esp+160h] [ebp-28h]
  char v11; // [esp+17Ch] [ebp-Ch]

  for ( i = 0; i < 100; ++i )
  {
    if ( i >= 0x64 )
      j____report_rangecheckfailure();
    Dest[i] = 0;
  }
  sub_41132F("please enter the flag:");
  sub_411375("%20s", &Str);
  v0 = j_strlen(&Str);
  v1 = sub_4110BE(&Str, v0, &v11);
  strncpy(Dest, v1, 0x28u);
  v8 = j_strlen(Dest);
  for ( j = 0; j < v8; ++j )
    Dest[j] += j;
  v2 = j_strlen(Dest);
  if ( !strncmp(Dest, Str2, v2) )
    sub_41132F("rigth flag!\n");
  else
    sub_41132F("wrong flag!\n");
  HIDWORD(v4) = v3;
  LODWORD(v4) = 0;
  return v4;
}
```

可以看到有两步加密，第一步是先sub_4110BE(&Str, v0, &v11);用这个函数加密。然后再去循环加密

```
for ( j = 0; j < v8; ++j )
    Dest[j] += j;
```

然后把加密后的字符串与str2相比较。str2的值为e3nifIH9b_C@n@dH，先把循环逆向了。

```
#!usr/bin/env python
#!coding=utf-8

__author__ = 'zhengjim'

str2 ='e3nifIH9b_C@n@dH'
flag =''

for i in range(len(str2)):
    flag += chr(ord(str2[i])- i)
print flag
```

得到e21fbDB2ZV95b3V9
然后看sub_4110BE()函数。一串长算法，发现首先将输入的flag每3位变成4位。然后有64位密码表。其实就是个base64加密（记下来,base64加密算法的特征）。

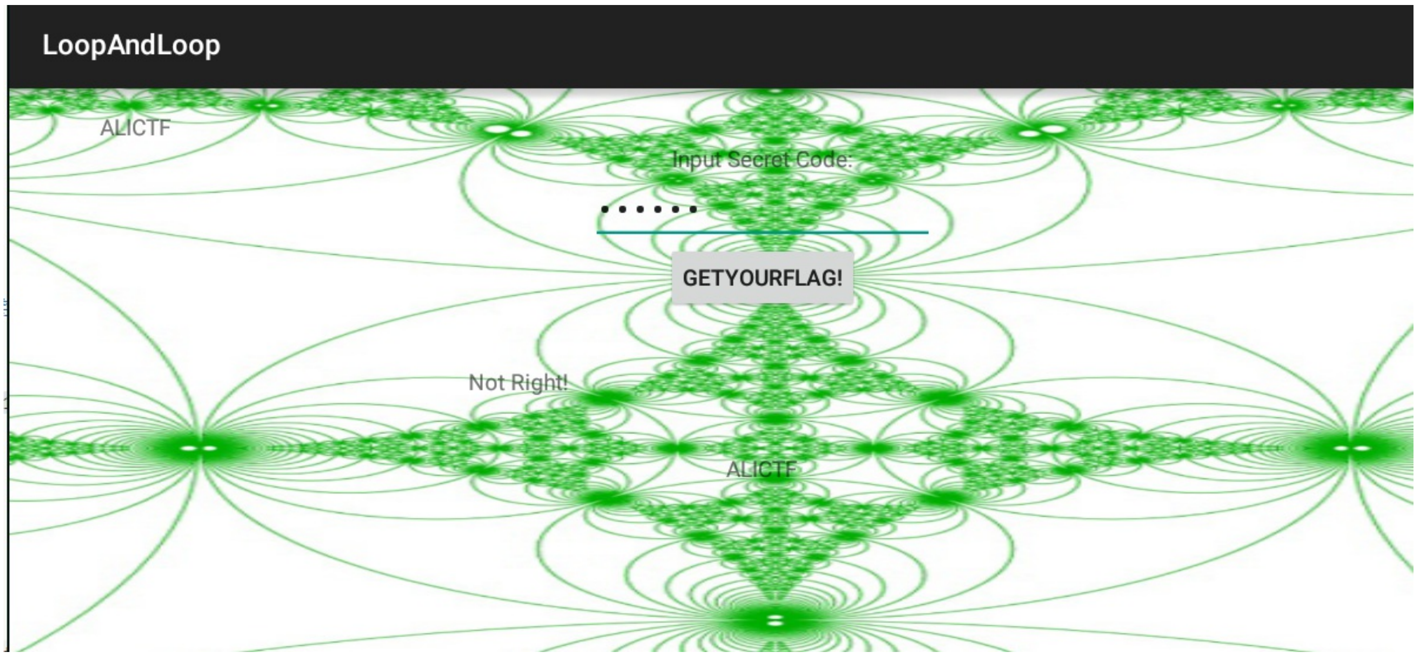
"ABC"的转化为base64字符的逻辑如下：

	A		B	C
ASCII十进制	65		66	67
8bit/byte	01000001		01000010	01000011
6bit/byte	010000	010100	001001	000011
base64十进制	16	20	9	3
base64字符	Q	U	J	D

也就是将刚刚得到的值base64解密就是flag。

LoopAndLoop(阿里CTF)

下载文件发现是apk，先安装运行下发现有一个输入框，随便输入点`getyourflag` 跳出Not Right



用`jadx-gui`反编译，双击看`MainActivity`查看

```
package net.bluelotus.tomorrow.easyandroid;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    public native int chec(int i, int i2);
    public native String stringFromJNI2(int i);

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView((int) R.layout.activity_main);
        final TextView tv1 = (TextView) findViewById(R.id.textView2);
        final TextView tv2 = (TextView) findViewById(R.id.textView3);
        final EditText ed = (EditText) findViewById(R.id.editText);
        ((Button) findViewById(R.id.button)).setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                try {
                    int in_int = Integer.parseInt(ed.getText().toString());
                    if (MainActivity.this.check(in_int, 99) == 1835996258) {
                        tv1.setText("The flag is:");
                        tv2.setText("alictf{" + MainActivity.this.stringFromJNI2(in_int) + "}");
                        return;
                    }
                    tv1.setText("Not Right!");
                } catch (NumberFormatException e) {
                    tv1.setText("Not a Valid Integer number");
                }
            }
        });
    }

    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

    public String messageMe(String text) {
        return "LoopOk" + text;
    }

    public int check(int input, int s) {
        return chec(input, s);
    }

    public int check1(int input, int s) {
        int t = input;
        for (int i = 1; i < 100; i++) {
            t += i;
        }
        return chec(t, s);
    }

    public int check2(int input, int s) {
        int t = input;
        int i;
        if (s % 2 == 0) {
            for (i = 1; i < 1000; i++) {
                t += i;
            }
            return chec(t, s);
        }
        for (i = 1; i < 1000; i++) {
            t -= i;
        }
        return chec(t, s);
    }

    public int check3(int input, int s) {
        int t = input;
        for (int i = 1; i < 10000; i++) {
            t += i;
        }
        return chec(t, s);
    }

    static {
        System.loadLibrary("lhm");
    }
}
```

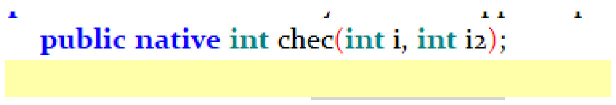
看到关键代码：

```
public void onClick(View v) {
                try {
                    int in_int = Integer.parseInt(ed.getText().toString());
                    if (MainActivity.this.check(in_int, 99) == 1835996258) {
                        tv1.setText("The flag is:");
                        tv2.setText("alictf{" + MainActivity.this.stringFromJNI2(in_int) + "}");
                        return;
                    }
                    tv1.setText("Not Right!");
                } catch (NumberFormatException e) {
                    tv1.setText("Not a Valid Integer number");
                }
            }
```

流程为：将用户输入作为参数`1(one)`，99作为参数`2(two)`传入`check`函数里，如果返回的值为`1835996258`，则将用户输入作为参数传入`stringFromJNI2`函数计算，返回值与`alictf{`和`}`拼接组成flag 。

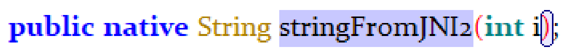
于是我们只要逆向出check函数，将1835996258带入得到的值，拿到apk里边运行即可得到flag。

追过去发现`check`函数调用了`chec`函数 为Native层的函数



```
public native int chec(int i, int i2);
```

`stringFromJNI2`函数也为Native层的函数

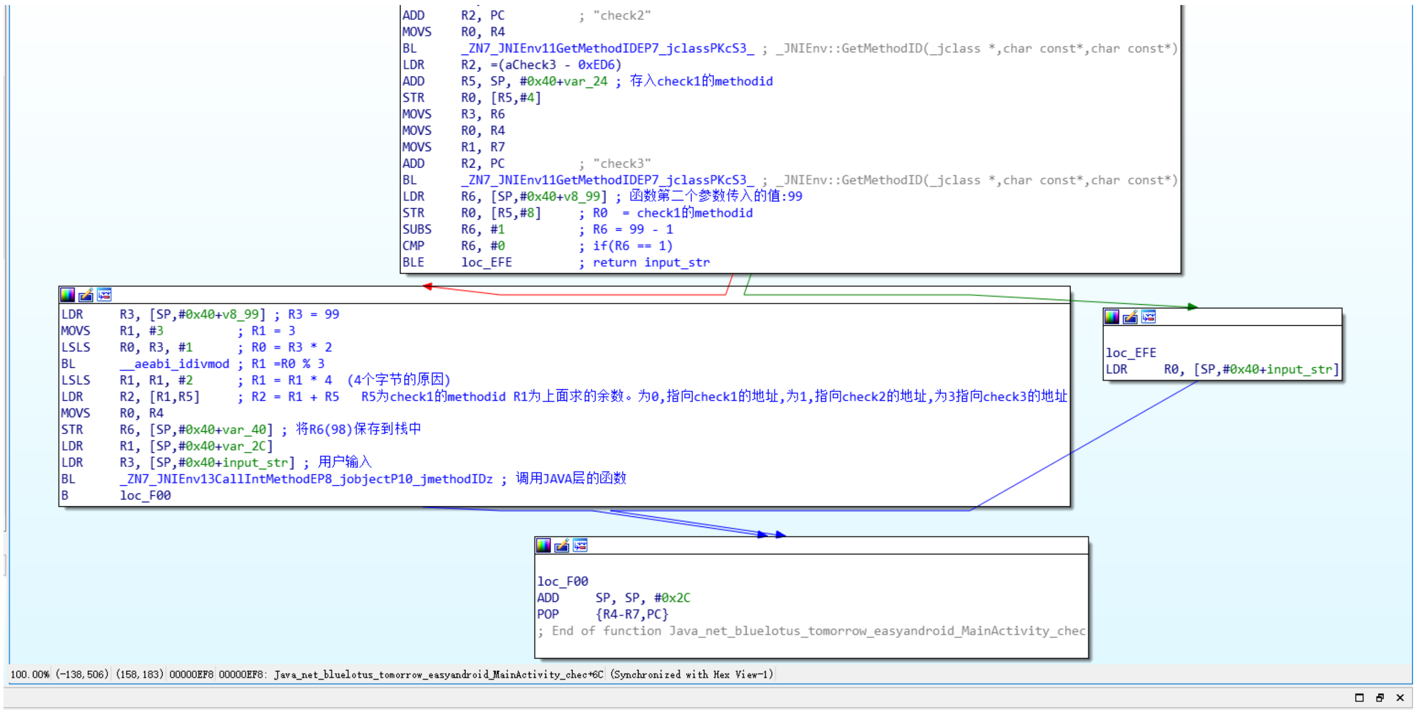


```
public native String stringFromJNI2(int i);
```

加载了`System.loadLibrary("lhm");`，所以逆向liblhm.so文件。

用IDA打开，还是上面的办法，找到了chec函数
这部分看得比较混乱，查了比较多的资料，所以有不对之处请指出来。

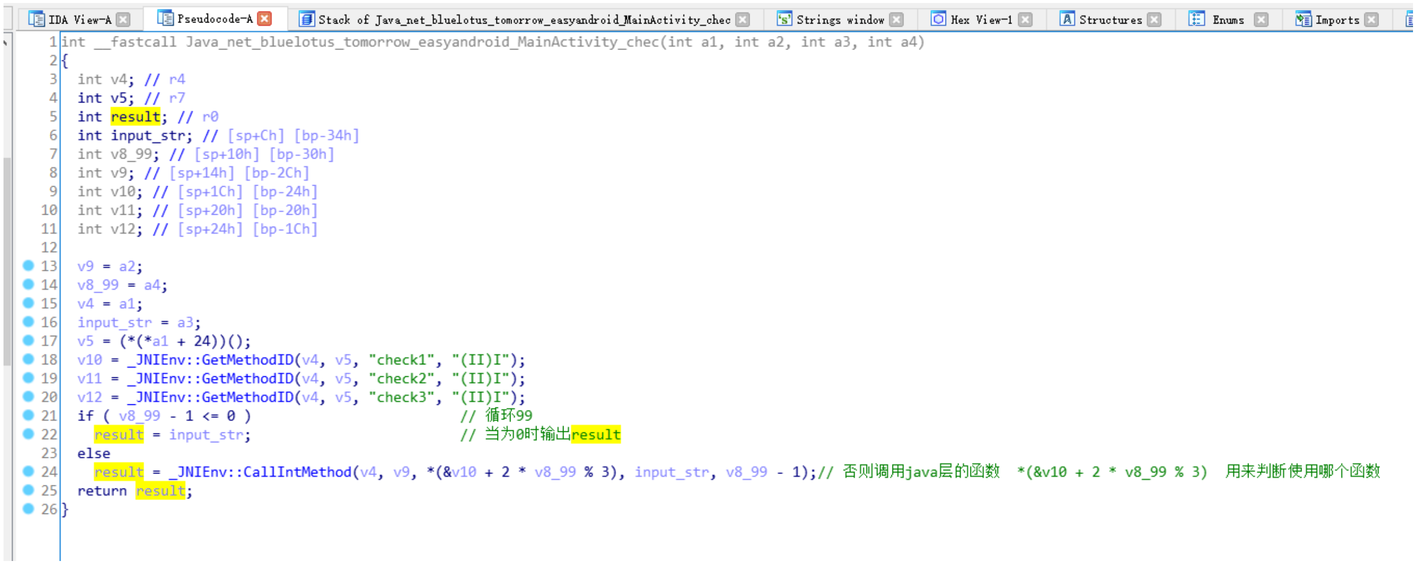
汇编



```
ADD     R2, PC          ; "check2"
MOVS    R0, R4
BL      _ZN7_JNIEnv11GetMethodIDEP7_jclassPKcS3_ ; _JNIEnv::GetMethodID(_jclass *,char const*,char const*)
LDR     R2, =(aCheck3 - 0xED6)
ADD     R5, SP, #0x40+var_24 ; 存入check1的methodid
STR     R0, [R5,#4]
MOVS    R3, R6
MOVS    R0, R4
MOVS    R1, R7
ADD     R2, PC          ; "check3"
BL      _ZN7_JNIEnv11GetMethodIDEP7_jclassPKcS3_ ; _JNIEnv::GetMethodID(_jclass *,char const*,char const*)
LDR     R6, [SP,#0x40+v8_99] ; 函数第二个参数传入的值:99
STR     R0, [R5,#8]     ; R0  = check1的methodid
SUBS    R6, #1          ; R6 = 99 - 1
CMP     R6, #0          ; if(R6 == 1)
BLE     loc_EFE         ; return input_str

LDR     R3, [SP,#0x40+v8_99] ; R3 = 99
MOVS    R1, #3          ; R1 = 3
LSLS    R0, R3, #1      ; R0 = R3 * 2
BL      __aeabi_idivmod ; R1 =R0 % 3
LSLS    R1, R1, #2      ; R1 = R1 * 4  (4个字节的原因)
LDR     R2, [R1,R5]     ; R2 = R1 + R5   R5为check1的methodid R1为上面求的余数。为0,指向check1的地址,为1,指向check2的地址,为3指向check3的地址
MOVS    R0, R4
STR     R6, [SP,#0x40+var_40] ; 将R6(98)保存到栈中
LDR     R1, [SP,#0x40+var_2C]
LDR     R3, [SP,#0x40+input_str] ; 用户输入
BL      _ZN7_JNIEnv13CallIntMethodEP8_jobjectP10_jmethodIDz ; 调用JAVA层的函数
B       loc_F00

loc_EFE
LDR     R0, [SP,#0x40+input_str]

loc_F00
ADD     SP, SP, #0x2C
POP     {R4-R7,PC}
; End of function Java_net_bluelotus_tomorrow_easyandroid_MainActivity_chec
```

```
100.00% (-138,506) (158,183) 00000EF8 00000EF8: Java_net_bluelotus_tomorrow_easyandroid_MainActivity_chec+6C (Synchronized with Hex View-1)
```

伪代码



```
int __fastcall Java_net_bluelotus_tomorrow_easyandroid_MainActivity_chec(int a1, int a2, int a3, int a4)
{
  int v4; // r4
  int v5; // r7
  int result; // r0
  int input_str; // [sp+Ch] [bp-34h]
  int v8_99; // [sp+10h] [bp-30h]
  int v9; // [sp+14h] [bp-2Ch]
  int v10; // [sp+1Ch] [bp-24h]
  int v11; // [sp+20h] [bp-20h]
  int v12; // [sp+24h] [bp-1Ch]

  v9 = a2;
  v8_99 = a4;
  v4 = a1;
  input_str = a3;
  v5 = (*(*a1 + 24))();
  v10 = _JNIEnv::GetMethodID(v4, v5, "check1", "(II)I");
  v11 = _JNIEnv::GetMethodID(v4, v5, "check2", "(II)I");
  v12 = _JNIEnv::GetMethodID(v4, v5, "check3", "(II)I");
  if ( v8_99 - 1 <= 0 )                        // 循环99
    result = input_str;                        // 当为0时输出result
  else
    result = _JNIEnv::CallIntMethod(v4, v9, *(&v10 + 2 * v8_99 % 3), input_str, v8_99 - 1);// 否则调用java层的函数  *(&v10 + 2 * v8_99 % 3)  用来判断使用哪个函数
  return result;
}
```

上面是自己加了注释，然后通过看汇编与伪代码分析得出流程，即将传入的99进行2 * i % 3运算，判断得到的余数。

1. 如果等于0，将one与two-1传到JAVA层的check1进行计算
2. 如果等于1，将one与two-1传到JAVA层的check2进行计算
3. 如果等于2，将one与two-1传到JAVA层的check3进行计算

去查看下check123函数

```
public int check1(int input, int s) {
        int t = input;
        for (int i = 1; i < 100; i++) {
            t += i;
        }
        return chec(t, s);
    }

    public int check2(int input, int s) {
        int t = input;
        int i;
        if (s % 2 == 0) {
            for (i = 1; i < 1000; i++) {
                t += i;
            }
            return chec(t, s);
        }
        for (i = 1; i < 1000; i++) {
            t -= i;
        }
        return chec(t, s);
    }

    public int check3(int input, int s) {
        int t = input;
        for (int i = 1; i < 10000; i++) {
            t += i;
        }
        return chec(t, s);
    }
```

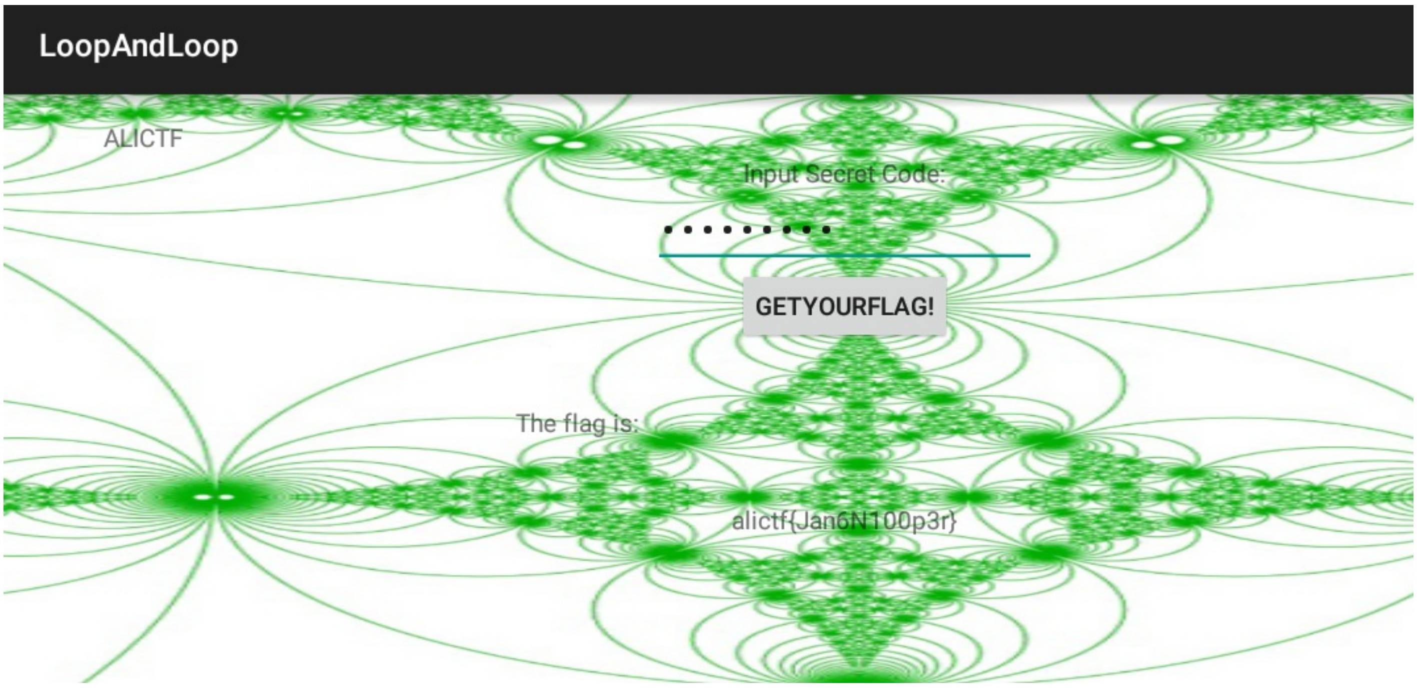
发现只是简单的遍历然后加减运算，计算完又返回`chec`函数

只到two小于等于1，输出结果。

于是写逆函数就不难了,check123 加变减，减变加就可以了。本来从99到2(因为two小于等于1)，变成从2到99。

```
#!usr/bin/env python
#!coding=utf-8

__author__ = 'zhengjim'

def check1(input,s):
    t = input
    for i in range(1,100):
        t -= i
    return t

def check2(input,s):
    t = input
    if(s % 2 == 0):
        for i in range(1,1000):
            t -= i
        return t
    for i in range(1,1000):
        t += i
    return t

def check3(input,s):
    t = input
    for i in range(1,10000):
        t -= i
    return t

output = 1835996258
for i in range(2,100):
    flag =  2 * i % 3
    if flag == 0 :
        output = check1(output, i-1)
    elif flag == 1 :
        output = check2(output, i-1)
    elif flag == 2 :
        output = check3(output, i-1)
print output
```

得到`236492408`，带入apk运行出现flag。



easy-100(LCTF)

下载文件发现是apk，先安装运行下（我的逍遥安卓运行失败，不懂为啥）。
用`jeb2`反编译(用`jadx-gui`反编译出了问题，a方法重载反编译出了问题)，双击看MainActivity查看

```
package com.example.ring.myapplication;

import android.content.pm.ApplicationInfo;
import android.os.Bundle;
import android.support.v7.a.q;
import java.io.InputStream;

public class MainActivity extends q {
    private String v;

    public MainActivity() {
        super();
    }

    static String a(MainActivity arg1) {
        return arg1.v;
    }

    static boolean a(MainActivity arg1, String arg2, String arg3) {
        return arg1.a(arg2, arg3);
    }

    private boolean a(String arg4, String arg5) {
        return new c().a(arg4, arg5).equals(new String(new byte[]{21, -93, -68, -94, 86, 117, -19, -68, -92
    }

    protected void onCreate(Bundle arg3) {
        super.onCreate(arg3);
        this.setContentView(2130968602);
        ApplicationInfo v0 = this.getApplicationInfo();
        v0.flags &= 2;
        this.p();
        this.findViewById(2131427413).setOnClickListener(new d(this));
    }

    private void p() {
        try {
            InputStream v0_1 = this.getResources().getAssets().open("url.png");
            int v1 = v0_1.available();
            byte[] v2 = new byte[v1];
            v0_1.read(v2, 0, v1);
            byte[] v0_2 = new byte[16];
            System.arraycopy(v2, 144, v0_2, 0, 16);
            this.v = new String(v0_2, "utf-8");
        }
        catch(Exception v0) {
            v0.printStackTrace();
        }
    }
}
```

首先看`onCreate()`方法

```
protected void onCreate(Bundle arg3) {
        super.onCreate(arg3);
        this.setContentView(2130968602);
        ApplicationInfo v0 = this.getApplicationInfo();
        v0.flags &= 2;
        this.p();
        this.findViewById(2131427413).setOnClickListener(new d(this));
    }
```

执行了`p()`方法，然后创建了一个按钮监听事件在`classs d`。

跟过去看下`class d`

```
package com.example.ring.myapplication;

import android.view.View$OnClickListener;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

class d implements View$OnClickListener {
    d(MainActivity arg1) {
        this.a = arg1;
        super();
    }

    public void onClick(View arg5) {
        if(MainActivity.a(this.a, MainActivity.a(this.a), this.a.findViewById(2131427414).getText().toStrin
            View v0 = this.a.findViewById(2131427412);
            Toast.makeText(this.a.getApplicationContext(), "Congratulations!", 1).show();
            ((TextView)v0).setText(2131099682);
        }
        else {
            Toast.makeText(this.a.getApplicationContext(), "Oh no.", 1).show();
        }
    }
}
```

如果`a()`方法返回真，则输出flag。第一个参数为句柄，第二个参数调用了另外一个a方法返回一个字符串，第三个参数是我们输入的字符串。
跟过去看下`a()`方法,发现为重载（JAVA重载概念）

```
    static String a(MainActivity arg1) {
        return arg1.v;
    }

    static boolean a(MainActivity arg1, String arg2, String arg3) {
        return arg1.a(arg2, arg3);
    }

    private boolean a(String arg4, String arg5) {
        return new c().a(arg4, arg5).equals(new String(new byte[]{21, -93, -68, -94, 86, 117, -19, -68, -92
    }
```

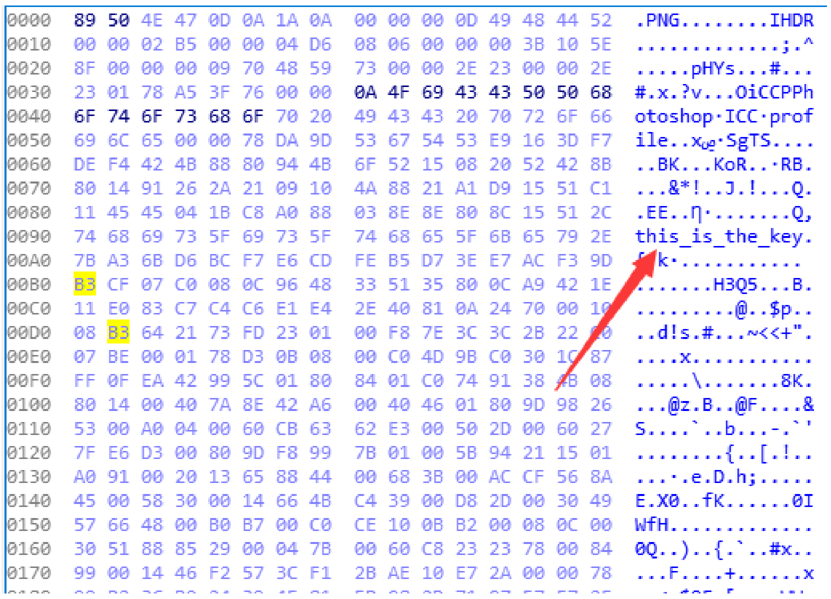
1. `static String a(MainActivity arg1)`方法直接返回了字符串，返回的是arg1.v
2. `private boolean a(String arg4, String arg5)`方法中调用了equals方法进行比较返回布尔值。

由于`arg4`是类d中传入的`MainActivity.a(this.a)`,所以得先看返回了什么字符串`v`,
而`v`是`MainActivity`的String类型的数据成员以及有相应的方法进行赋值`p`方法。

```
private void p() {
        try {
            InputStream v0_1 = this.getResources().getAssets().open("url.png");
            int v1 = v0_1.available();
            byte[] v2 = new byte[v1];
            v0_1.read(v2, 0, v1);
            byte[] v0_2 = new byte[16];
            System.arraycopy(v2, 144, v0_2, 0, 16);
            this.v = new String(v0_2, "utf-8");
        }
        catch(Exception v0) {
            v0.printStackTrace();
        }
    }
```

首先读取`url.png`文件以二进制数据取出来。从文件的144位置开始，读取16字符保存为`v`。

用`winhex`打开这张文件，找到144位置，往后的16位为`this_is_the_key.`



```
0000   89 50 4E 47 0D 0A 1A 0A  00 00 00 0D 49 48 44 52   .PNG........IHDR
0010   00 00 02 B5 00 00 04 D6  08 06 00 00 00 3B 10 5E   .............;.^
0020   8F 00 00 00 09 70 48 59  73 00 00 2E 23 00 00 2E   .....pHYs...#...
0030   23 01 78 A5 3F 76 00 00  0A 4F 69 43 43 50 50 68   #.x.?v...OiCCPPh
0040   6F 74 6F 73 68 6F 70 20  49 43 43 20 70 72 6F 66   otoshop·ICC·prof
0050   69 6C 65 00 00 78 DA 9D  53 67 54 53 E9 16 3D F7   ile..xڝ·SgTS....
0060   DE F4 42 4B 88 80 94 4B  6F 52 15 08 20 52 42 8B   ..BK...KoR..·RB.
0070   80 14 91 26 2A 21 09 10  4A 88 21 A1 D9 15 51 C1   ...&*!..J.!...Q.
0080   11 45 45 04 1B C8 A0 88  03 8E 8E 80 8C 15 51 2C   .EE..ῃ·.......Q,
0090   74 68 69 73 5F 69 73 5F  74 68 65 5F 6B 65 79 2E   this_is_the_key.
00A0   7B A3 6B D6 BC F7 E6 CD  FE B5 D7 3E E7 AC F3 9D   {.k·...........
00B0   B3 CF 07 C0 08 0C 96 48  33 51 35 80 0C A9 42 1E   .......H3Q5...B.
00C0   11 E0 83 C7 C4 C6 E1 E4  2E 40 81 0A 24 70 00 10   .........@..$p..
00D0   08 B3 64 21 73 FD 23 01  00 F8 7E 3C 3C 2B 22 00   ..d!s.#...~<<+".
00E0   07 BE 00 01 78 D3 0B 08  00 C0 4D 9B C0 30 1C 87   ....x...........
00F0   FF 0F EA 42 99 5C 01 80  84 01 C0 74 91 38 4B 08   .....\.......8K.
0100   80 14 00 40 7A 8E 42 A6  00 40 46 01 80 9D 98 26   ...@z.B..@F....&
0110   53 00 A0 04 00 60 CB 63  62 E3 00 50 2D 00 60 27   S....`..b...-.`'
0120   7F E6 D3 00 80 9D F8 99  7B 01 00 5B 94 21 15 01   ........{..[.!..
0130   A0 91 00 20 13 65 88 44  00 68 3B 00 AC CF 56 8A   ...·.e.D.h;.....
0140   45 00 58 30 00 14 66 4B  C4 39 00 D8 2D 00 30 49   E.X0..fK......0I
0150   57 66 48 00 B0 B7 00 C0  CE 10 0B B2 00 08 0C 00   WfH.............
0160   30 51 88 85 29 00 04 7B  00 60 C8 23 23 78 00 84   0Q..)..{.`..#x..
0170   99 00 14 46 F2 57 3C F1  2B AE 10 E7 2A 00 00 78   ...F....+......x
```

得到了key后，要看回三参数的a方法

```
static boolean a(MainActivity arg1, String arg2, String arg3) {
        return arg1.a(arg2, arg3);
    }

private boolean a(String arg4, String arg5) {
        return new c().a(arg4, arg5).equals(new String(new byte[]{21, -93, -68, -94, 86, 117, -19, -68, -92
    }
```

发现三参数a方法调用了两参数a方法，将`this_is_the_key.`与用户输入作为参数传了进去。.而两参数a方法是调用类c的两参数a方法.计算完后和后面的字节比较。相等返回真。跟过去看下类c的两参数a方法

```
public String a(String arg5, String arg6) {
        String v0 = this.a(arg5);
        String v1 = "";
        a v2 = new a();
        v2.a(v0.getBytes());
        try {
            v0 = new String(v2.b(arg6.getBytes()), "utf-8");
        }
        catch(Exception v0_1) {
            v0_1.printStackTrace();
            v0 = v1;
        }

        return v0;
    }
```

首先将`this_is_the_key.`传入一个参数a方法，然后将返回值赋值给v0。看下一个参数a方法。

```
 private String a(String arg4) {
        String v0_2;
        try {
            arg4.getBytes("utf-8");
            StringBuilder v1 = new StringBuilder();
            int v0_1;
            for(v0_1 = 0; v0_1 < arg4.length(); v0_1 += 2) {
                v1.append(arg4.charAt(v0_1 + 1));
                v1.append(arg4.charAt(v0_1));
            }

            v0_2 = v1.toString();
        }
        catch(UnsupportedEncodingException v0) {
            v0.printStackTrace();
            v0_2 = null;
        }

        return v0_2;
    }
```

将传入的字符串每两个字符为一组然后交换这两个字符的位置最后返回改变后的字符串。就是变成`htsii__sht_eek.y`,可以手动也可以写脚本。

```
#!usr/bin/env python
#!coding=utf-8

__author__ = 'zhengjim'

key = 'this_is_the_key.'
ckey =""
for i in range(0,len(key),2):
    ckey += key[i+1]
    ckey += key[i]
print(ckey)
```

回到类c的两参数a方法，实例化的了类a，然后将用户输入作为参数带入。跟过去看看。

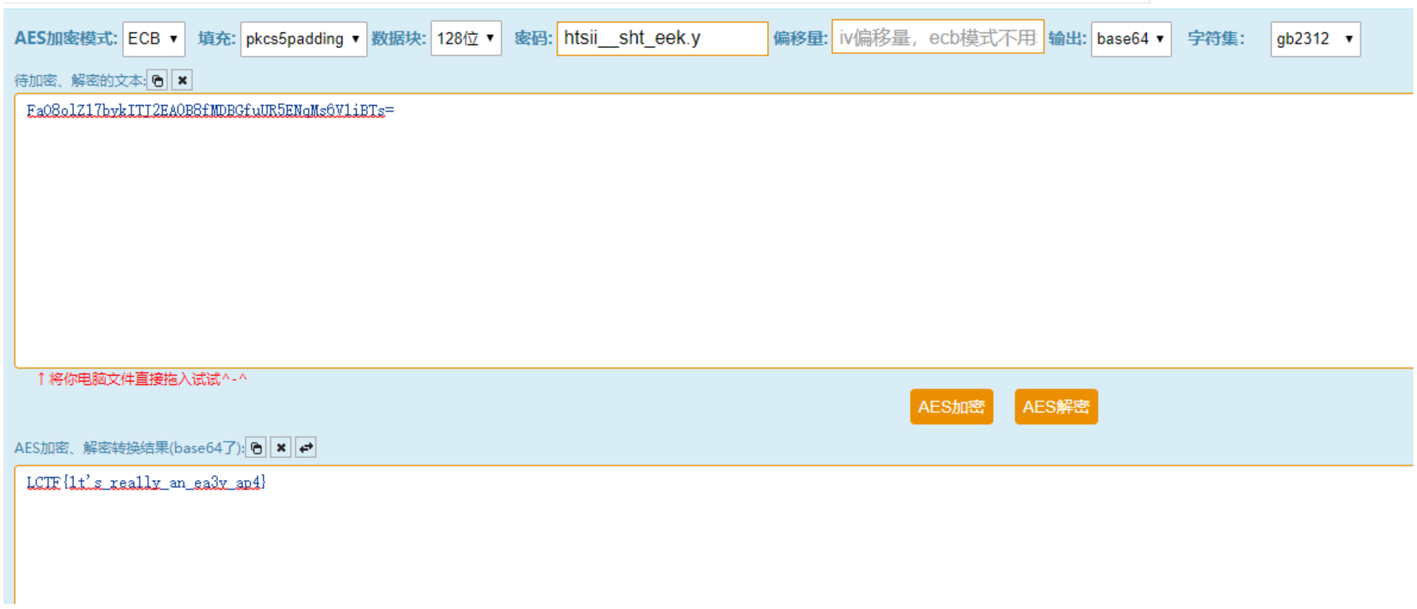
```
package com.example.ring.myapplication;

import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import javax.crypto.Cipher;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.SecretKeySpec;

public class a {
    private SecretKeySpec a;
    private Cipher b;

    public a() {
        super();
    }

    protected void a(byte[] arg4) {
        if(arg4 != null) {
            goto label_15;
        }

        try {
            this.a = new SecretKeySpec(MessageDigest.getInstance("MD5").digest("".getBytes("utf-8")), "AES"
            this.b = Cipher.getInstance("AES/ECB/PKCS5Padding");
            return;
        label_15:
            this.a = new SecretKeySpec(arg4, "AES");
            this.b = Cipher.getInstance("AES/ECB/PKCS5Padding");
        }
        catch(UnsupportedEncodingException v0) {
            v0.printStackTrace();
        }
        catch(NoSuchAlgorithmException v0_1) {
            v0_1.printStackTrace();
        }
        catch(NoSuchPaddingException v0_2) {
            v0_2.printStackTrace();
        }
    }

    protected byte[] b(byte[] arg4) {
        this.b.init(1, this.a);
        return this.b.doFinal(arg4);
    }
}
```

发现是用AES加密，ECB模式，PKCS5Padding填充。然后要找下密钥。`this.a = new SecretKeySpec(arg4, "AES");`是将arg4作为密钥。而arg4则是在类c中传入的`v0.getBytes()`，也就是密钥为`htsii__sht_eek.y`。

回到类c的两参数a方法，将返回的字符串赋值给`v0`然后再返回。到了`MainActivity`的两参数a方法，与那串字符串比较。正确则返回真，就出现`Congratulations!`。
所以百度了一个AES解密网站。由于网站输出的是base64。所以讲密文转为base64格式。

```
#!usr/bin/env python
#!coding=utf-8

__author__ = 'zhengjim'

import base64

str1 = [21, -93, -68, -94, 86, 117, -19, -68, -92, 33,50, 118, 16, 13, 1, -15, -13, 3, 4, 103, -18, 81, 30,
ctext = ''
for i in str1:
    ctext += chr((i+256)%256)
a = base64.b64encode(ctext)
print(a)
```

得到了密文为`Fa08olZ17bykITJ2EA0B8fMDBGfuUR5ENqMs6V1iBTs=`,密钥为`htsii__sht_eek.y`,AES解密后出现flag。


AES加密模式: ECB 填充: pkcs5padding 数据块: 128位 密码: htsii__sht_eek.y 偏移量: iv偏移量，ecb模式不用 输出: base64 字符集: gb2312

待加密、解密的文本:

```
Fa08olZ17bykITJ2EA0B8fMDBGfuUR5ENqMs6V1iBTs=
```

AES加密、解密转换结果(base64了):

```
LCTF{1t's_really_an_ea3y_ap4}
```

SafeBox(NJCTF)

首先下载发现是apk，安装运行下。就一个输入框，其他的按不了。
用`jadx-gui`反编译下,双击`MainActivity`查看。

```java
package com.geekerchina.hi;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView((int) R.layout.activity_main);
        final EditText Et1 = (EditText) findViewById(R.id.editText);
        ((Button) findViewById(R.id.button)).setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                String strTmp = "NJCTF{";
                int i = Integer.parseInt(Et1.getText().toString());
                if (i > 10000000 && i < 99999999) {
                    int t = 1;
                    int t1 = 10000000;
                    int flag = 1;
                    if (Math.abs(((i / 1000) % 100) - 36) == 3 && (i % 1000) % 584 == 0) {
                        for (int j = 0; j < 4; j++) {
                            if ((i / t) % 10 != (i / t1) % 10) {
                                flag = 0;
                                break;
                            }
                            t *= 10;
                            t1 /= 10;
                        }
                        if (flag == 1) {
                            char c2 = (char) ((i / 10000) % 100);
                            char c3 = (char) ((i / 100) % 100);
                            Et1.setText(strTmp + ((char) (i / 1000000)) + c2 + c3 + "f4n}");
                        }
                    }
                }
            }
        });
    }

    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

看到onCreate方法关键位置18行-37行，输入一个8位数满足条件后，将其变换后与NJCTF{和f4n}拼接。

用python脚本来爆破

```
#!usr/bin/env python
#!coding=utf-8

__author__ = 'zhengjim'

import math

for i in range(10000000, 99999999):
    t = 1
    t1 =10000000
    flag = 1
    if (abs(((i / 1000) % 100) - 36) == 3 and (i % 1000) % 584 == 0):
        for j in range(4):
            if ((i / t) % 10 != (i / t1) % 10):
                flag = 0
                break
            t *= 10
            t1 /= 10
        if(flag ==1):
            print i
            c2 = chr((i / 10000) % 100)
            c3 = chr((i / 100) % 100)
            print('NJCTF{'+chr(i / 1000000)+c2+c3+'f4n}')
```

得到i应该为48533584，flag为NJCTF{05#f4n}，但提交却发现错误了。看了好几遍发现没错。再看目录发现了类androidTest。

```
package com.geekerchina.hi;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class androidTest extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView((int) R.layout.build);
        final EditText Et1 = (EditText) findViewById(R.id.editText);
        ((Button) findViewById(R.id.button)).setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                String strTmp = "NJCTF{have";
                int i = Integer.parseInt(Et1.getText().toString());
                if (i > 10000000 && i < 99999999) {
                    int t = 1;
                    int t1 = 10000000;
                    int flag = 1;
                    if (Math.abs(((i / 1000) % 100) - 36) == 3 && (i % 1000) % 584 == 0) {
                        for (int j = 0; j < 3; j++) {
                            if ((i / t) % 10 != (i / t1) % 10) {
                                flag = 0;
                                break;
                            }
                            t *= 10;
                            t1 /= 10;
                        }
                        if (flag == 1) {
                            char c2 = (char) ((i / 10000) % 100);
                            char c3 = (char) (((i / 100) % 100) + 10);
                            Et1.setText(strTmp + ((char) (i / 1000000)) + c2 + c3 + "f4n}");
                        }
                    }
                }
            }
        });
    }

    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

和`MainActivity`很像，但有细微不同：

1. 第27行的`String strTmp = "NJCTF{have";`
2. 第27行的`for (int j = 0; j < 3; j++) {`
3. 第39行的`char c3 = (char) (((i / 100) % 100) + 10);`

python脚本爆破

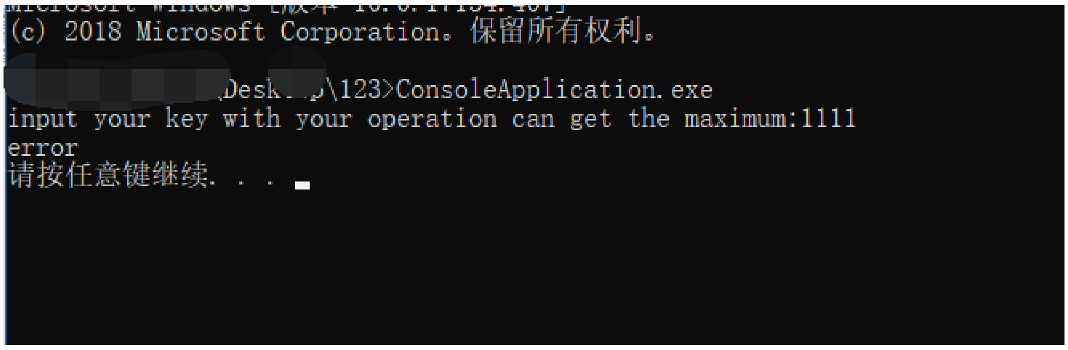
```
#!usr/bin/env python
#!coding=utf-8

__author__ = 'zhengjim'

import math

for i in range(10000000, 99999999):
    t = 1
    t1 = 10000000
    flag = 1
    if (abs(((i / 1000) % 100) - 36) == 3 and (i % 1000) % 584 == 0):
        for j in range(3):
            if ((i / t) % 10 != (i / t1) % 10):
                flag = 0
                break
            t *= 10
            t1 /= 10
        if (flag == 1):
            print i
            c2 = chr((i / 10000) % 100)
            c3 = chr((i / 100) % 100 + 10)
            print('NJCTF{have' + chr(i / 1000000) + c2 + c3 + 'f4n}')
```

得到两组答案。

1. i为`48533584`，flag为`NJCTF{have05-f4n}`
2. i为`48539584`，flag为`NJCTF{have05if4n}`
   均提交试试发现第二组为正确。

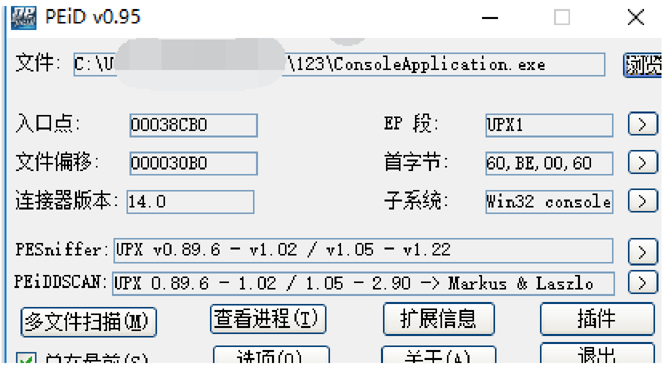
Mountain climbing

下载后运行,发现要输入最大数字，乱输后跳出error。



```
Microsoft Windows [版本 10.0.17134.407]
(c) 2018 Microsoft Corporation。保留所有权利。

[redacted]Desk[redacted]p\123>ConsoleApplication.exe
input your key with your operation can get the maximum:1111
error
请按任意键继续. . .
```

用`PEID` 查看下 发现有 UPX的壳。



PEiD v0.95

文件: C:\U[redacted]\123\ConsoleApplication.exe

入口点: 00038CB0　　EP 段: UPX1

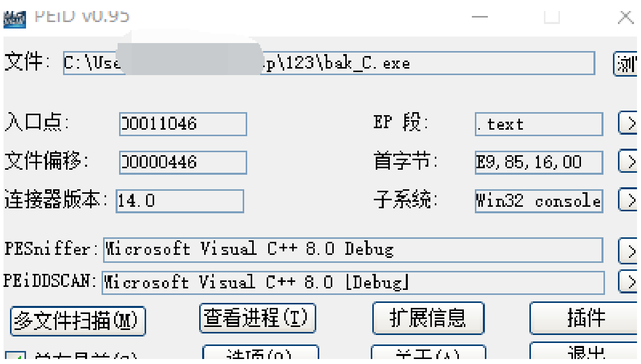
文件偏移: 000030B0　　首字节: 60,BE,00,60

连接器版本: 14.0　　子系统: Win32 console

PESniffer: UPX v0.89.6 - v1.02 / v1.05 - v1.22

PEiDDSCAN: UPX 0.89.6 - 1.02 / 1.05 - 2.90 -> Markus & Laszlo

直接用52pojie的脱UPX工具进行脱壳。成功



PEiD v0.95

文件: C:\Use[redacted]p\123\bak_C.exe

入口点: 00011046　　EP 段: .text

文件偏移: 00000446　　首字节: E9,85,16,00

连接器版本: 14.0　　子系统: Win32 console

PESniffer: Microsoft Visual C++ 8.0 Debug

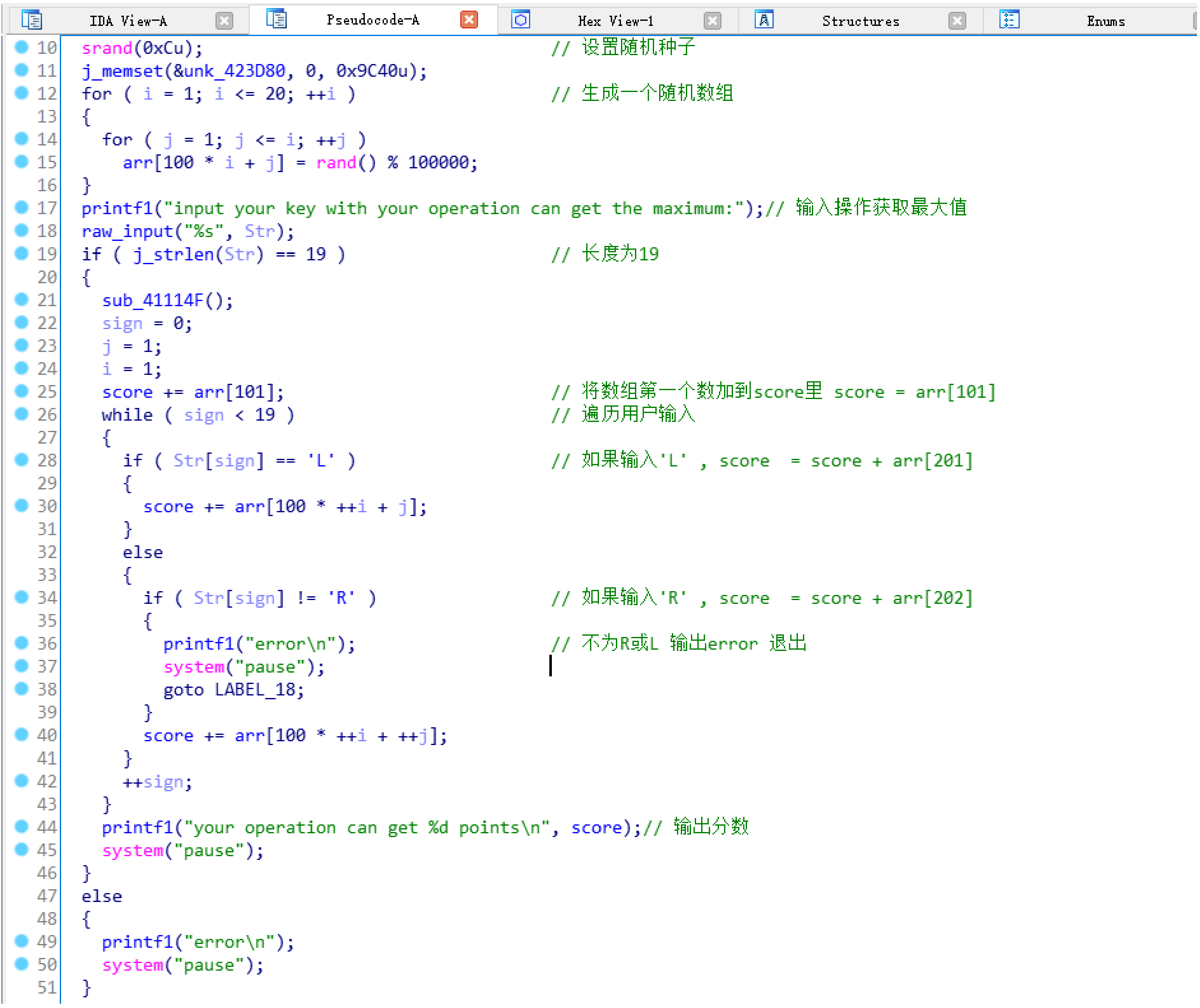
PEiDDSCAN: Microsoft Visual C++ 8.0 [Debug]

载入IDA

```
__int64 main_0()
{
  int v0; // edx
  __int64 v1; // ST04_8
  char v3; // [esp+0h] [ebp-160h]
  int v4; // [esp+D0h] [ebp-90h]
  int j; // [esp+DCh] [ebp-84h]
  int i; // [esp+E8h] [ebp-78h]
  char Str[104]; // [esp+F4h] [ebp-6Ch]

  srand(0xCu);
  j_memset(&unk_423D80, 0, 0x9C40u);
  for ( i = 1; i <= 20; ++i )
  {
    for ( j = 1; j <= i; ++j )
      dword_41A138[100 * i + j] = rand() % 100000;
  }
  ((void (__cdecl *)(const char *, char))sub_41134D)("input your key with your operation can get the maximu
  sub_411249("%s", (unsigned int)Str);
  if ( j_strlen(Str) == 19 )
  {
    sub_41114F(Str);
    v4 = 0;
    j = 1;
    i = 1;
    dword_423D78 += dword_41A138[101];
    while ( v4 < 19 )
    {
      if ( Str[v4] == 76 )
      {
        dword_423D78 += dword_41A138[100 * ++i + j];
      }
      else
      {
        if ( Str[v4] != 82 )
        {
          ((void (__cdecl *)(const char *, char))sub_41134D)("error\n", v3);
          system("pause");
          goto LABEL_18;
        }
        dword_423D78 += dword_41A138[100 * ++i + ++j];
      }
      ++v4;
    }
    sub_41134D("your operation can get %d points\n", dword_423D78);
    system("pause");
  }
  else
  {
    ((void (__cdecl *)(const char *, char))sub_41134D)("error\n", v3);
    system("pause");
  }
LABEL_18:
  HIDWORD(v1) = v0;
  LODWORD(v1) = 0;
  return v1;
}
```

首先生成一个数组存在。这个数组由伪随机数生成。`srand(0xCu)`随机数种子一定，那么rand出来的数也是一定的。

然后往下看，下面是自己加入了注释。



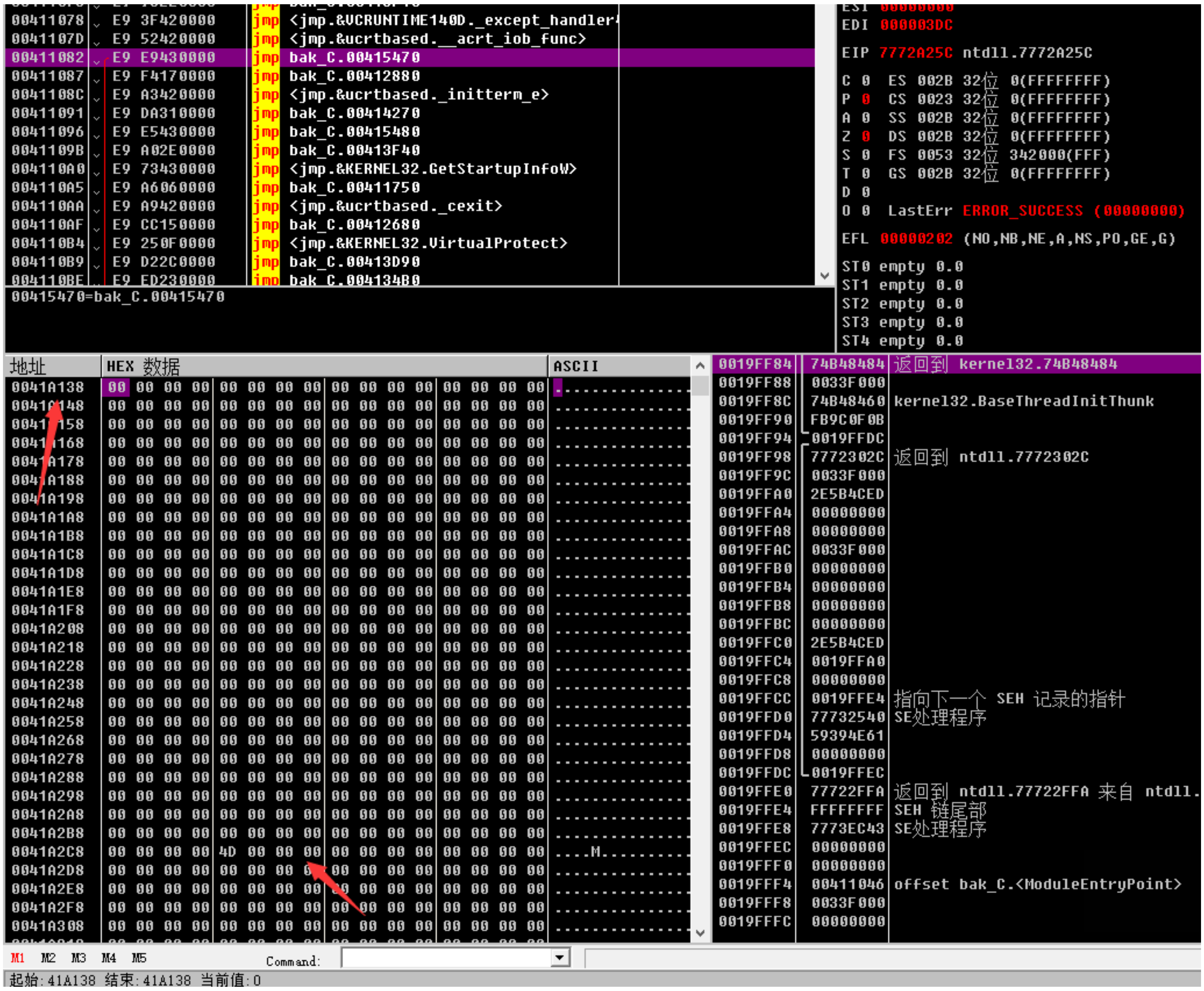
```
srand(0xCu);                                   // 设置随机种子
j_memset(&unk_423D80, 0, 0x9C40u);
for ( i = 1; i <= 20; ++i )                    // 生成一个随机数组
{
  for ( j = 1; j <= i; ++j )
    arr[100 * i + j] = rand() % 100000;
}
printf1("input your key with your operation can get the maximum:");// 输入操作获取最大值
raw_input("%s", Str);
if ( j_strlen(Str) == 19 )                     // 长度为19
{
  sub_41114F();
  sign = 0;
  j = 1;
  i = 1;
  score += arr[101];                           // 将数组第一个数加到score里 score = arr[101]
  while ( sign < 19 )                          // 遍历用户输入
  {
    if ( Str[sign] == 'L' )                    // 如果输入'L' , score  = score + arr[201]
    {
      score += arr[100 * ++i + j];
    }
    else
    {
      if ( Str[sign] != 'R' )                  // 如果输入'R' , score  = score + arr[202]
      {
        printf1("error\n");                    // 不为R或L 输出error 退出
        system("pause");
        goto LABEL_18;
      }
      score += arr[100 * ++i + ++j];
    }
    ++sign;
  }
  printf1("your operation can get %d points\n", score);// 输出分数
  system("pause");
}
else
{
  printf1("error\n");
  system("pause");
}
```

总得来看就是先将从`arr[101]` 相加，往下的循环为：

1. 第一次循环：经过用户按"L"或"R"来控制加`arr[201]`还是`arr[202]`
2. 第二次循环：**(情况1)**第一次选择了`arr[201]`:经过用户按"L"或"R"来控制加`arr[301]`还是`arr[302]` **(情况2)**第二次选择了`arr[202]`:经过用户按"L"或"R"来控制加`arr[302]`还是`arr[303]`

跟两次循环帮组理解，这样子就很清楚了。整个题可以理解成站在山顶往下走，每一行走到的数字累加，每次只能走一格（左或右），走到最后一行。然后最后的数要最大。与题目Mountain climbing呼应

那首先要先找到这座山，看ida反编译后是`dword_41A138`存在`0041A138`,我们用`OD`载入后运后，跟过去看看。



```
00411078  E9 3F420000   jmp <jmp.&VCRUNTIME140D._except_handler
0041107D  E9 52420000   jmp <jmp.&ucrtbased.__acrt_iob_func>
00411082  E9 E9430000   jmp bak_C.00415470
00411087  E9 F4170000   jmp bak_C.00412880
0041108C  E9 A3420000   jmp <jmp.&ucrtbased._initterm_e>
00411091  E9 DA310000   jmp bak_C.00414270
00411096  E9 E5430000   jmp bak_C.00415480
0041109B  E9 A02E0000   jmp bak_C.00413F40
004110A0  E9 73430000   jmp <jmp.&KERNEL32.GetStartupInfoW>
004110A5  E9 A6060000   jmp bak_C.00411750
004110AA  E9 A9420000   jmp <jmp.&ucrtbased._cexit>
004110AF  E9 CC150000   jmp bak_C.00412680
004110B4  E9 250F0000   jmp <jmp.&KERNEL32.VirtualProtect>
004110B9  E9 D22C0000   jmp bak_C.00413D90
004110BE  E9 ED230000   jmp bak_C.004134B0
00415470=bak_C.00415470
```

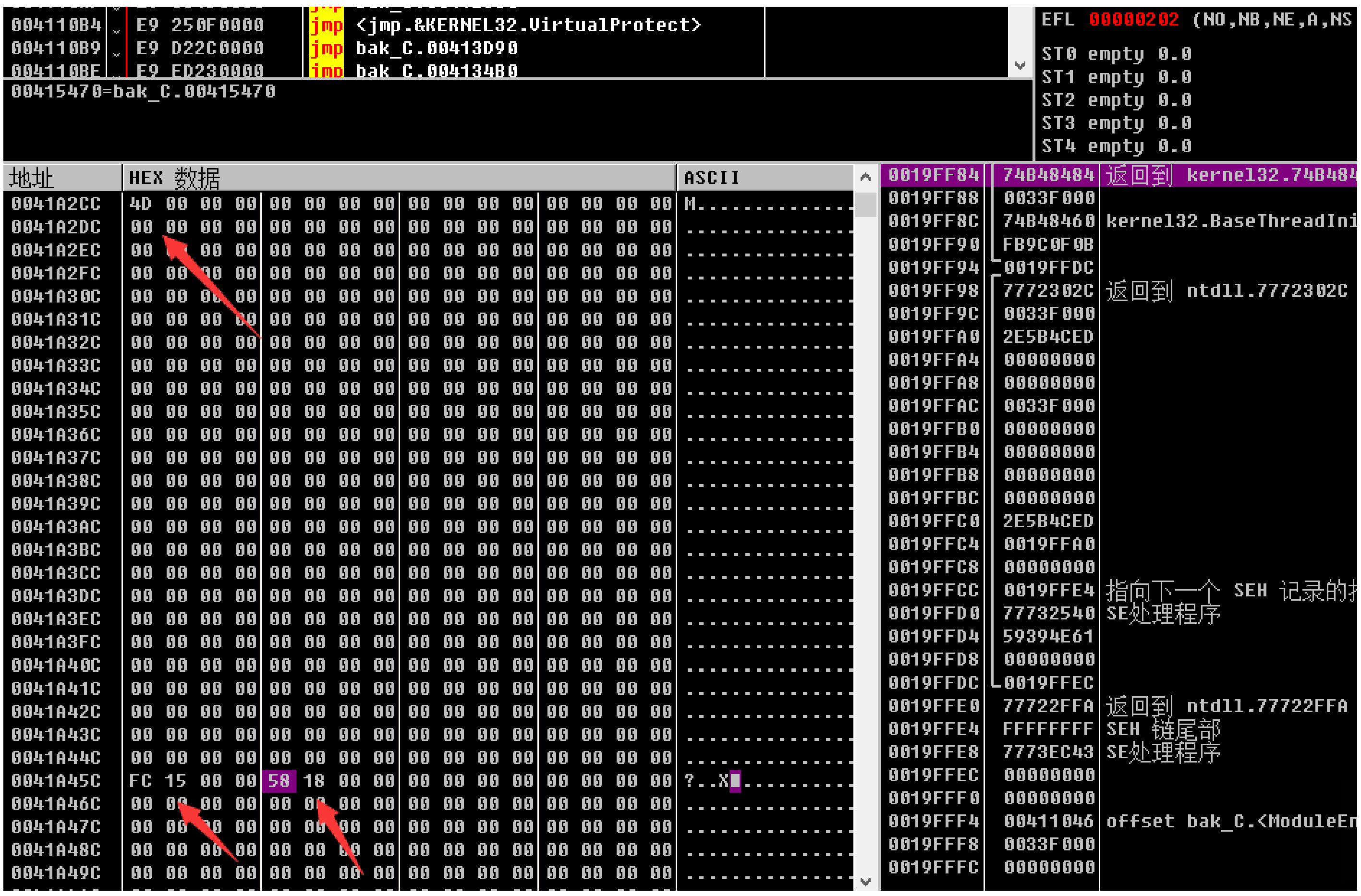
```
EDI 000003DC

EIP 7772A25C ntdll.7772A25C

C 0  ES 002B 32位 0(FFFFFFFF)
P 0  CS 0023 32位 0(FFFFFFFF)
A 0  SS 002B 32位 0(FFFFFFFF)
Z 0  DS 002B 32位 0(FFFFFFFF)
S 0  FS 0053 32位 342000(FFF)
T 0  GS 002B 32位 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00000202 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
```

```
地址      HEX 数据                                         ASCII
0041A138  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A148  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A158  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A168  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A178  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A188  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A198  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A1A8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A1B8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A1C8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A1D8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A1E8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A1F8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A208  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A218  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A228  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A238  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A248  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A258  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A268  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A278  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A288  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A298  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A2A8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A2B8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A2C8  00 00 00 00 4D 00 00 00 00 00 00 00 00 00 00 00  ....M...........
0041A2D8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A2E8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A2F8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A308  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
0019FF84  74B48484  返回到 kernel32.74B48484
0019FF88  0033F000
0019FF8C  74B48460  kernel32.BaseThreadInitThunk
0019FF90  FB9C0F0B
0019FF94  0019FFDC
0019FF98  7772302C  返回到 ntdll.7772302C
0019FF9C  0033F000
0019FFA0  2E5B4CED
0019FFA4  00000000
0019FFA8  00000000
0019FFAC  0033F000
0019FFB0  00000000
0019FFB4  00000000
0019FFB8  00000000
0019FFBC  00000000
0019FFC0  2E5B4CED
0019FFC4  0019FFA0
0019FFC8  00000000
0019FFCC  0019FFE4  指向下一个 SEH 记录的指针
0019FFD0  77732540  SE处理程序
0019FFD4  59394E61
0019FFD8  00000000
0019FFDC  0019FFEC
0019FFE0  77722FFA  返回到 ntdll.77722FFA 来自 ntdll.
0019FFE4  FFFFFFFF  SEH 链尾部
0019FFE8  7773EC43  SE处理程序
0019FFEC  00000000
0019FFF0  00000000
0019FFF4  00411046  offset bak_C.<ModuleEntryPoint>
0019FFF8  0033F000
0019FFFC  00000000
```

```
M1  M2  M3  M4  M5        Command:
起始:41A138 结束:41A138 当前值:0
```

dword是4字节的，而且第一个数是存在在[101]位置的。所以首位置存在41A2CC位置。往下的都是往后400节。


```
004110B4  E9 250F0000   jmp <jmp.&KERNEL32.VirtualProtect>
004110B9  E9 D22C0000   jmp bak_C.00413D90
004110BE  E9 ED230000   jmp bak_C.004134B0
00415470=bak_C.00415470

EFL 00000202 (NO,NB,NE,A,NS
ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
```

```
地址      HEX 数据                                          ASCII
0041A2CC  4D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  M...............
0041A2DC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A2EC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A2FC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A30C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A31C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A32C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A33C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A34C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A35C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A36C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A37C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A38C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A39C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A3AC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A3BC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A3CC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A3DC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A3EC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A3FC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A40C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A41C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A42C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A43C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A44C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A45C  FC 15 00 00 58 18 00 00 00 00 00 00 00 00 00 00  ?...X...........
0041A46C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A47C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A48C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0041A49C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

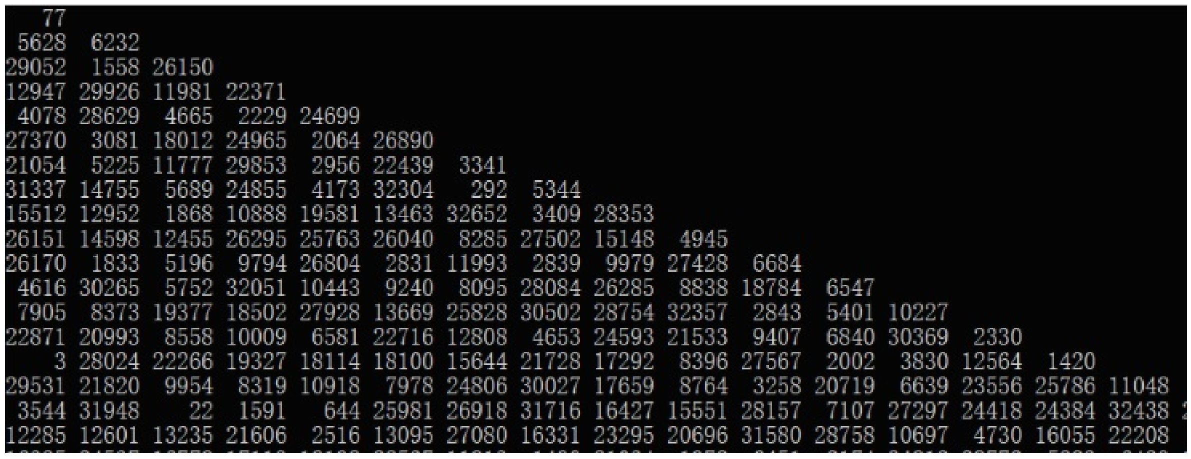
```
0019FF84  74B48484  返回到 kernel32.74B484
0019FF88  0033F000
0019FF8C  74B48460  kernel32.BaseThreadIni
0019FF90  FB9C0F0B
0019FF94  0019FFDC
0019FF98  7772302C  返回到 ntdll.7772302C
0019FF9C  0033F000
0019FFA0  2E5B4CED
0019FFA4  00000000
0019FFA8  00000000
0019FFAC  0033F000
0019FFB0  00000000
0019FFB4  00000000
0019FFB8  00000000
0019FFBC  00000000
0019FFC0  2E5B4CED
0019FFC4  0019FFA0
0019FFC8  00000000
0019FFCC  0019FFE4  指向下一个 SEH 记录的指
0019FFD0  77732540  SE处理程序
0019FFD4  59394E61
0019FFD8  00000000
0019FFDC  0019FFEC
0019FFE0  77722FFA  返回到 ntdll.77722FFA
0019FFE4  FFFFFFFF  SEH 链尾部
0019FFE8  7773EC43  SE处理程序
0019FFEC  00000000
0019FFF0  00000000
0019FFF4  00411046  offset bak_C.<ModuleEn
0019FFF8  0033F000
0019FFFC  00000000
```

还是要注意小端存储问题。所以可以得到arr[101] = 4D(16进制) =77 , arr[201] = 15FC(16进制) = 5628(10进制) , arr[202] = 1858(16进制) = 6232(10进制)

有耐心的话可以一行行扣出来，没有的话，直接还原c代码生成"一座山"。
代码如下：

```
srand(0xCu);
for (int i = 1; i <= 20; ++i)
{
    for (int j = 1; j <= i; ++j)
        arr[100 * i + j] = rand() % 100000;
}


for (int i = 1; i <= 20; ++i)
{
    for (int j = 1; j <= i; ++j)
        printf("%5d ", arr[100 * i + j]);
    cout << endl;
}
```

得到完整的，与OD获取的值是一致。

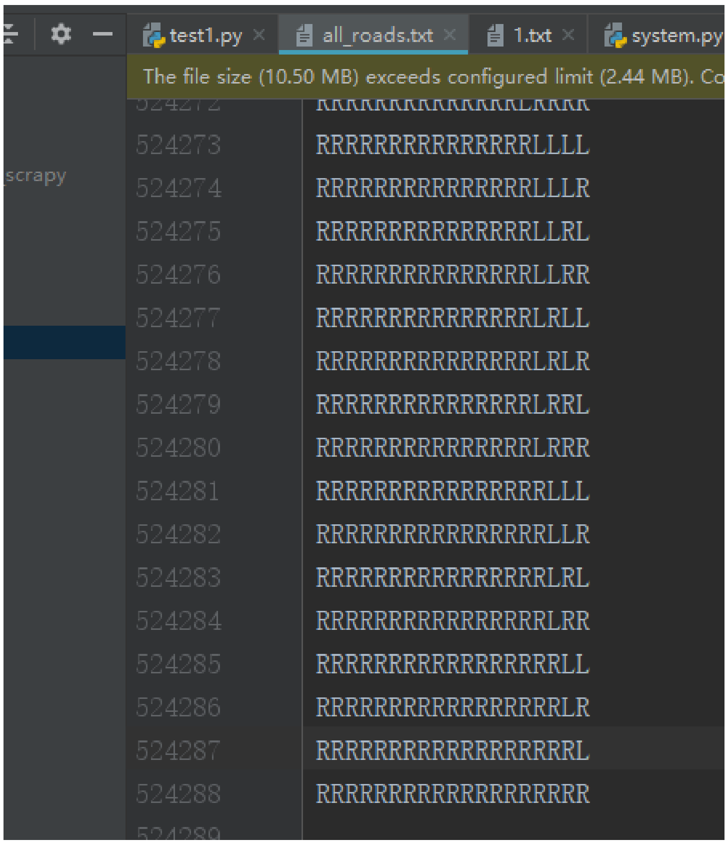


```
   77
 5628  6232
29052  1558 26150
12947 29926 11981 22371
 4078 28629  4665  2229 24699
27370  3081 18012 24965  2064 26890
21054  5225 11777 29853  2956 22439  3341
31337 14755  5689 24855  4173 32304   292  5344
15512 12952  1868 10888 19581 13463 32652  3409 28353
26151 14598 12455 26295 25763 26040  8285 27502 15148  4945
26170  1833  5196  9794 26804  2831 11993  2839  9979 27428  6684
 4616 30265  5752 32051 10443  9240  8095 28084 26285  8838 18784  6547
 7905  8373 19377 18502 27928 13669 25828 30502 28754 32357  2843  5401 10227
22871 20993  8558 10009  6581 22716 12808  4653 24593 21533  9407  6840 30369  2330
    3 28024 22266 19327 18114 18100 15644 21728 17292  8396 27567  2002  3830 12564  1420
29531 21820  9954  8319 10918  7978 24806 30027 17659  8764  3258 20719  6639 23556 25786 11048
 3544 31948    22  1591   644 25981 26918 31716 16427 15551 28157  7107 27297 24418 24384 32438
12285 12601 13235 21606  2516 13095 27080 16331 23295 20696 31580 28758 10697  4730 16055 22208
```

接下来就是找到最大解的路线。直接遍历所有路线，然后比较最大数。

首先生成所有路径

```
#!usr/bin/env python
#!coding=utf-8

__author__ = 'zhengjim'

import itertools

words = "LR"
r = itertools.product(words,repeat=19)
f = open("all_roads.txt",'a')
for i in r:
    f.write("".join(i)+"\n")
f.close()
```



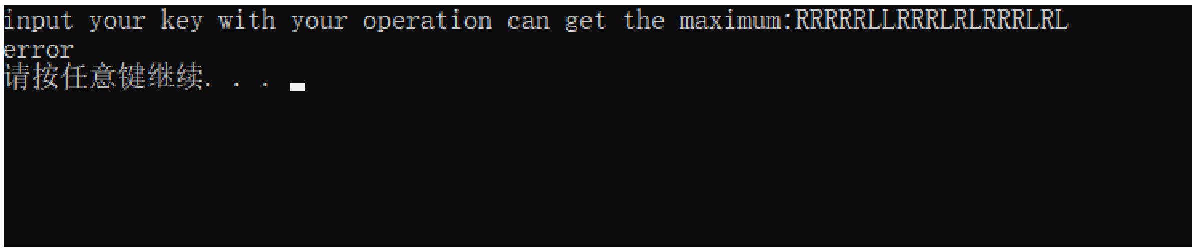
```
The file size (10.50 MB) exceeds configured limit (2.44 MB). Co
RRRRRRRRRRRRRRRLLLL
RRRRRRRRRRRRRRRLLLR
RRRRRRRRRRRRRRRLLRL
RRRRRRRRRRRRRRRLLRR
RRRRRRRRRRRRRRRLRLL
RRRRRRRRRRRRRRRLRLR
RRRRRRRRRRRRRRRLRRL
RRRRRRRRRRRRRRRLRRR
RRRRRRRRRRRRRRRRLLL
RRRRRRRRRRRRRRRRLLR
RRRRRRRRRRRRRRRRLRL
RRRRRRRRRRRRRRRRLRR
RRRRRRRRRRRRRRRRRLL
RRRRRRRRRRRRRRRRRLR
RRRRRRRRRRRRRRRRRRL
RRRRRRRRRRRRRRRRRRR
```

然后将每个走法得到的值进行判断大小，最大的值就是我们要的答案。

```
#!usr/bin/env python
#!coding=utf-8

__author__ = 'zhengjim'


s = [
    [77],
    [5628, 6232],
    [29052, 1558, 26150],
    [12947, 29926, 11981, 22371],
    [4078, 28629, 4665, 2229, 24699],
    [27370, 3081, 18012, 24965, 2064, 26890],
    [21054, 5225, 11777, 29853, 2956, 22439, 3341],
    [31337, 14755, 5689, 24855, 4173, 32304, 292, 5344],
    [15512, 12952, 1868, 10888, 19581, 13463, 32652, 3409, 28353],
    [26151, 14598, 12455, 26295, 25763, 26040, 8285, 27502, 15148, 4945],
    [26170, 1833, 5196, 9794, 26804, 2831, 11993, 2839, 9979, 27428, 6684],
    [4616, 30265, 5752, 32051, 10443, 9240, 8095, 28084, 26285, 8838, 18784, 6547],
    [7905, 8373, 19377, 18502, 27928, 13669, 25828, 30502, 28754, 32357, 2843, 5401, 10227],
    [22871, 20993, 8558, 10009, 6581, 22716, 12808, 4653, 24593, 21533, 9407, 6840, 30369, 2330],
    [3, 28024, 22266, 19327, 18114, 18100, 15644, 21728, 17292, 8396, 27567, 2002, 3830, 12564, 1420],
    [29531, 21820, 9954, 8319, 10918, 7978, 24806, 30027, 17659, 8764, 3258, 20719, 6639, 23556, 25786, 110
    [3544, 31948, 22, 1591, 644, 25981, 26918, 31716, 16427, 15551, 28157, 7107, 27297, 24418, 24384, 32438
    [12285, 12601, 13235, 21606, 2516, 13095, 27080, 16331, 23295, 20696, 31580, 28758, 10697, 4730, 16055,
    [16325, 24537, 16778, 17119, 18198, 28537, 11813, 1490, 21034, 1978, 6451, 2174, 24812, 28772, 5283, 64
    [7299, 6961, 32019, 24731, 29103, 17887, 17338, 26840, 13216, 8789, 12474, 24299, 19818, 18218, 14564,

all_score = {}
with open('all_roads.txt', 'r')as f:
    for line in f.readlines():
        row = 0
        go = 0
        score = s[row][go]
        for i in line:
            if i == 'L':
                row += 1
                score += s[row][go]
            elif i == 'R':
                row += 1
                go += 1
                score += s[row][go]
        all_score[line] = score

max_road = max(all_score, key=all_score.get)
print(max_road, all_score[max_road])
```

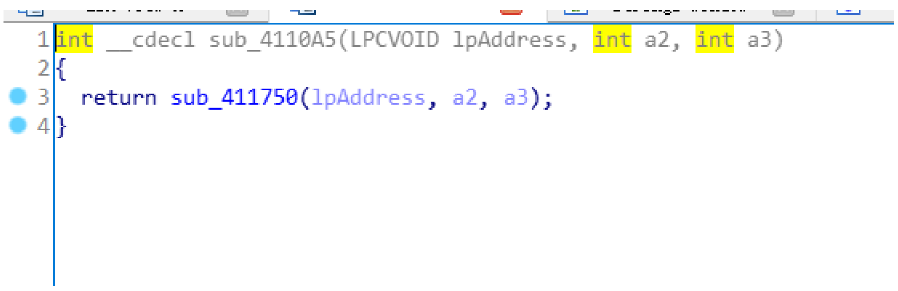
得到了最大路径:RRRRRLLRRRLRLRRRLRL和最大值444740。

在程序输入却还是error



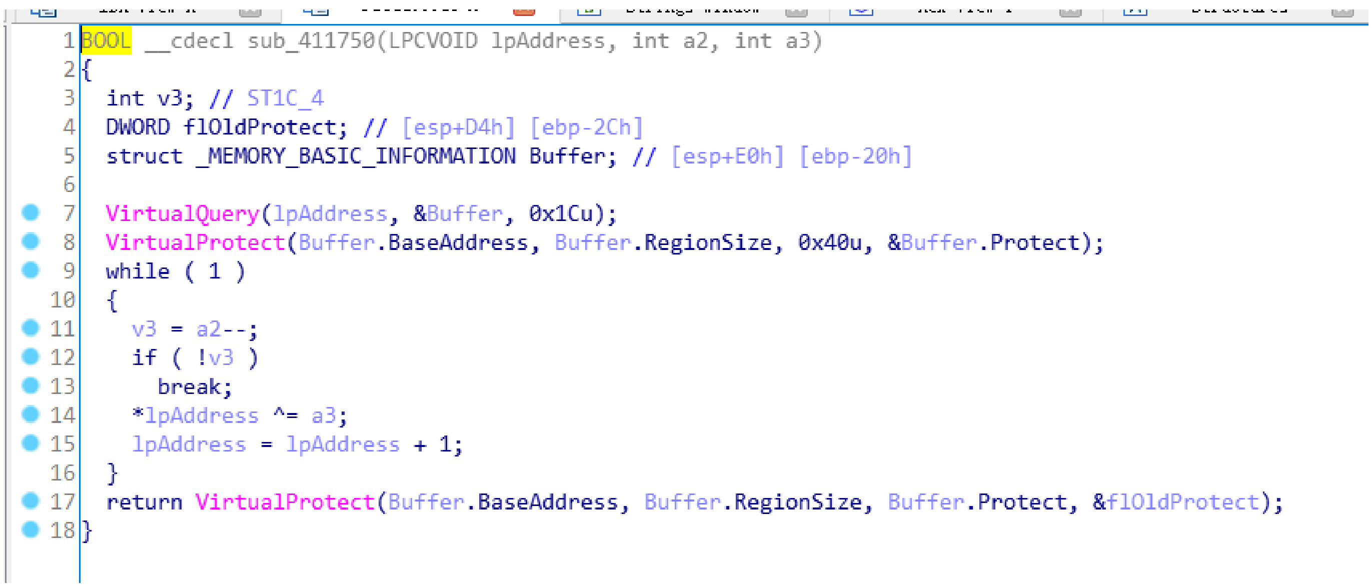
```
input your key with your operation can get the maximum:RRRRRLLRRRLRLRRRLRL
error
请按任意键继续. . .
```

不知道是哪错了，又看了一遍程序，发现第22行的sub_41114F(Str)对我们输入的数据进行了处理。跟进去,发现又调用了sub_411900(Str)在跟进去，发现调用了sub_4110A5(nullsub_1, sub_411994 - nullsub_1, 4)。再跟进去。



```
int __cdecl sub_4110A5(LPCVOID lpAddress, int a2, int a3)
{
  return sub_411750(lpAddress, a2, a3);
}
```

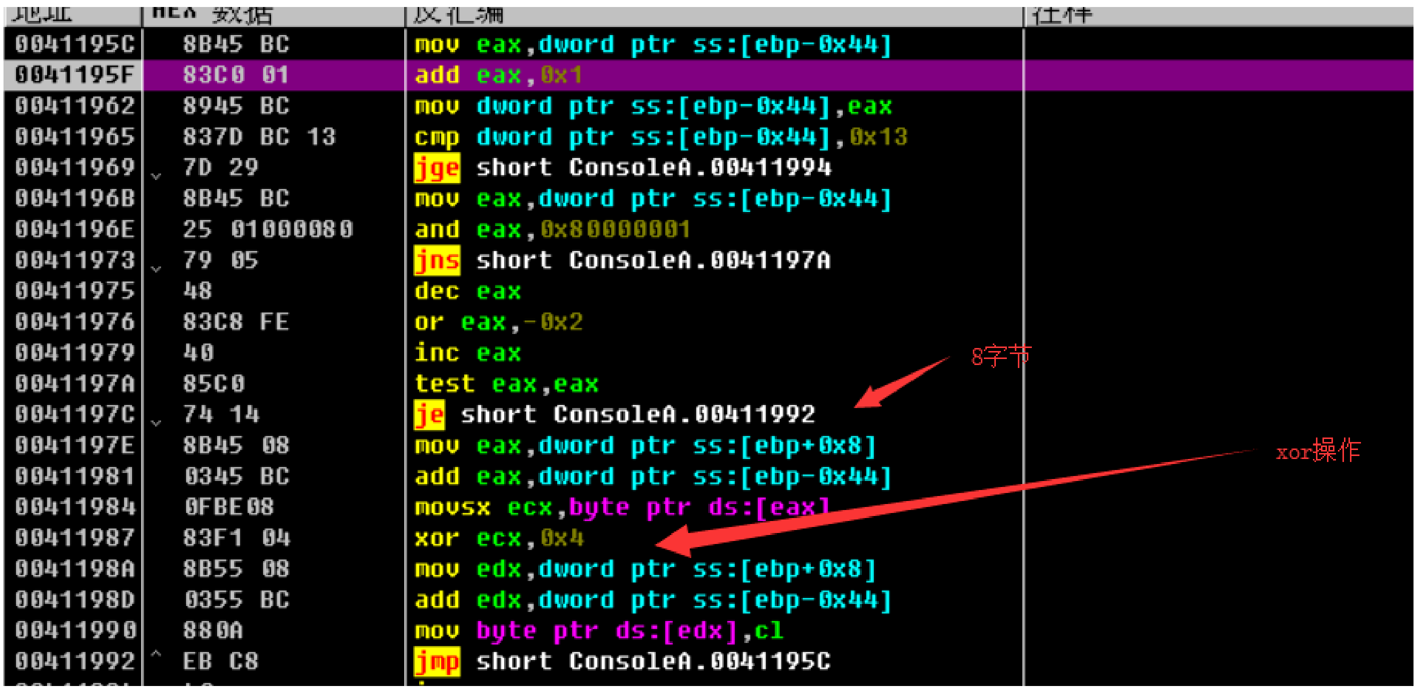
再往里跟。sub_411750(lpAddress, a2, a3 = 4);



```
BOOL __cdecl sub_411750(LPCVOID lpAddress, int a2, int a3)
{
  int v3; // ST1C_4
  DWORD flOldProtect; // [esp+D4h] [ebp-2Ch]
  struct _MEMORY_BASIC_INFORMATION Buffer; // [esp+E0h] [ebp-20h]

  VirtualQuery(lpAddress, &Buffer, 0x1Cu);
  VirtualProtect(Buffer.BaseAddress, Buffer.RegionSize, 0x40u, &Buffer.Protect);
  while ( 1 )
  {
    v3 = a2--;
    if ( !v3 )
      break;
    *lpAddress ^= a3;
    lpAddress = lpAddress + 1;
  }
  return VirtualProtect(Buffer.BaseAddress, Buffer.RegionSize, Buffer.Protect, &flOldProtect);
}
```

结合OD来查看。

前面几行是往内存获取值理解成获取用户输入。因为调用到了内存，所以结合OD来查看。

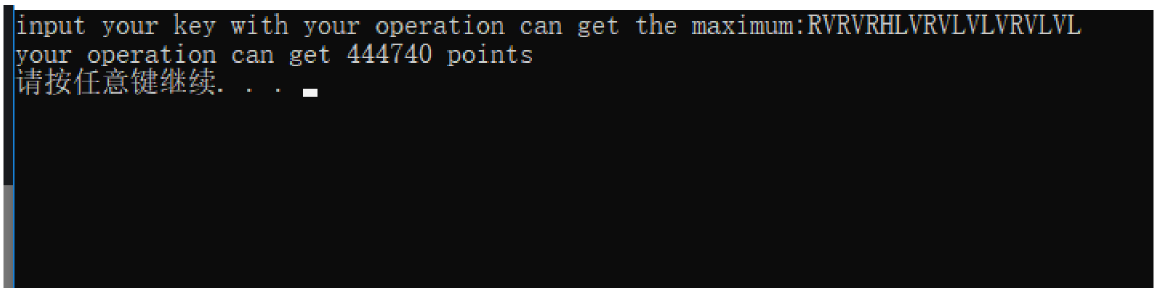


```
0041195C   8B45 BC      mov eax,dword ptr ss:[ebp-0x44]
0041195F   83C0 01      add eax,0x1
00411962   8945 BC      mov dword ptr ss:[ebp-0x44],eax
00411965   837D BC 13   cmp dword ptr ss:[ebp-0x44],0x13
00411969 ⌄ 7D 29        jge short ConsoleA.00411994
0041196B   8B45 BC      mov eax,dword ptr ss:[ebp-0x44]
0041196E   25 01000080  and eax,0x80000001
00411973 ⌄ 79 05        jns short ConsoleA.0041197A
00411975   48           dec eax
00411976   83C8 FE      or eax,-0x2
00411979   40           inc eax
0041197A   85C0         test eax,eax
0041197C ⌄ 74 14        je short ConsoleA.00411992
0041197E   8B45 08      mov eax,dword ptr ss:[ebp+0x8]
00411981   0345 BC      add eax,dword ptr ss:[ebp-0x44]
00411984   0FBE08       movsx ecx,byte ptr ds:[eax]
00411987   83F1 04      xor ecx,0x4
0041198A   8B55 08      mov edx,dword ptr ss:[ebp+0x8]
0041198D   0355 BC      add edx,dword ptr ss:[ebp-0x44]
00411990   880A         mov byte ptr ds:[edx],cl
00411992 ^ EB C8        jmp short ConsoleA.0041195C
```

因为一个dword占了4字节，所以8字节为第二字符。所以就是偶数位的字符与传入的4进行xor运算。

```
#!usr/bin/env python
#!coding=utf-8

__author__ = 'zhengjim'

max_road = 'RRRRRLLRRRLRLRRRLRL'
flag = ''
for i, s in enumerate(max_road):
    if (i - 1) % 2 == 0:
        flag += chr(ord(s) ^ 4)
    else:
        flag += s
print(flag)
```

得到flag：RVRVRHLVRVLVLVRVLVL



```
input your key with your operation can get the maximum:RVRVRHLVRVLVLVRVLVL
your operation can get 444740 points
请按任意键继续. . .
```