鲁宏伟/luhw@hust.edu.cn

第六讲 Android逆向实例分析


1:22
自毁程序密码
当Bob带领银河飞行队赶到时，飞碟已坠落在小山谷里，驾驶员在坠落前启动了自毁程序，飞碟中的一切已化为灰烬，唯一幸免的是一部手机，但需要开机密码。
输入密码
阿里钱盾
安全脉搏
SecPulse

用反编译工具

开始 | AliCrackme_2

工程信息 | 工程管理器 | 工程搜索

名称：自毁程序密码
包名：com.yaotong.crackme
入口：com.yaotong.crackme.MainActi...
版本信息：Ver：1.0(1) SDK：8 TargetSDK：19

- **Activity**
  - com.yaotong.crackme.MainActivity
  - com.yaotong.crackme.ResultActivity
- **Receiver**
- **Service**
- **Uses-Permission**

```
当前 Apktool 使用版本: Android Killer Default APKTOOL
正在反编译 APK, 请稍等...
>I: 使用 ShakaApktool 2.0.0-20150914
>I: 正在加载资源列表...
>I: 反编译 AndroidManifest.xml 与资源...
>I: 正在从框架文件加载资源列表: C:\Users\bruce\apktool\framework\1.apk
>I: 常规资源列表...
>I: 反编译资源文件...
>I: 反编译 values */* XMLs...
>I: 反编译 classes.dex...
>I: 复制 assets 和 libs...
>I: 复制未知文件...
>I: 复制原始文件...
APK 反编译完成!
正在反编译 APK 源码, 请稍等...
>dex2jar E:\Tools\AndroidKiller_v1.3.1\projects\AliCrackme_2\ProjectSrc\classes.dex -> .\classes
-dex2jar.jar
APK 源码反编译完成!
正在提取 APK 源码, 请稍等...
APK 源码提取完成!
----------------------------
APK 所有反编译工作全部完成!!!
----------------------------
正在对当前工程进行分析, 这将有助于您更加方便快捷的了解当前工程的信息!
正在分析中, 请稍等...
该 APK 未检测到其他信息
分析完成!
```

用反编译工具

开始 AliCrackme_2

工程信息 工程管理器 工程搜索

- Project
  - lib
    - armeabi
      - libcrackme.so
  - original
    - META-INF
    - AndroidManifest.xml
  - res
  - smali
    - android
    - com
      - yaotong
        - crackme
          - BuildConfig.smali
          - MainActivity$1.smali
          - MainActivity.smali
          - R$attr.smali
          - R$color.smali
          - R$dimen.smali
          - R$drawable.smali
          - R$id.smali
          - R$layout.smali
          - R$string.smali
          - R.smali
          - ResultActivity.smali
  - unknown
  - AndroidManifest.xml
  - apktool.yml

Java Decompiler - MainActivity.class

File Edit Navigate Search Help

smali

- android.annotation
- com.yaotong.crackme
  - BuildConfig
    - BuildConfig
      - DEBUG : boo
  - MainActivity
    - MainActivity
      - btn_submit :
      - inputCode : I
      - onCreate(Bu
      - securityChec
      - {...}
  - R
  - ResultActivity
    - ResultActivity
      - onCreate(Bu

MainActivity.class | ResultActivity.class

```
package com.yaotong.crackme;

import android.app.Activity;

public class MainActivity
  extends Activity
{
  public Button btn_submit;
  public EditText inputCode;

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

  protected void onCreate(Bundle paramBundle)
  {
    super.onCreate(paramBundle);
    setContentView(2130903040);
    getWindow().setBackgroundDrawableResource(2130837504);
    this.inputCode = ((EditText)findViewById(2131099648));
    this.btn_submit = ((Button)findViewById(2131099649));
    this.btn_submit.setOnClickListener(new View.OnClickListener()
    {
      public void onClick(View paramAnonymousView)
      {
        paramAnonymousView = MainActivity.this.inputCode.getText().toString();
        if (MainActivity.this.securityCheck(paramAnonymousView))
        {
          paramAnonymousView = new Intent(MainActivity.this, ResultActivity.class);
          MainActivity.this.startActivity(paramAnonymousView);
          return;
        }
        Toast.makeText(MainActivity.this.getApplicationContext(), "验证码校验失败", 0).show();
      }
    });
  }

  public native boolean securityCheck(String paramString);
}
```

用反编译工具

注意，只有当“MainActivity.this.securityCheck(str)”返回1，会调用以下函数，提示成功！

Java Decompiler - MainActivity.class

File Edit Navigate Search Help

smali

- android.annotation
- com.yaotong.crackme
  - BuildConfig
    - BuildConfig
      - DEBUG : boo
  - MainActivity
    - MainActivity
      - btn_submit :
      - inputCode : I
      - onCreate(Bu
      - securityChec
      - {...}
  - R
  - ResultActivity
    - ResultActivity
      - onCreate(Bu

MainActivity.class | ResultActivity.class

```
package com.yaotong.crackme;

import android.app.Activity;

public class MainActivity
  extends Activity
{
  public Button btn_submit;
  public EditText inputCode;

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

  protected void onCreate(Bundle paramBundle)
  {
    super.onCreate(paramBundle);
    setContentView(2130903040);
    getWindow().setBackgroundDrawableResource(2130837504);
    this.inputCode = ((EditText)findViewById(2131099648));
    this.btn_submit = ((Button)findViewById(2131099649));
    this.btn_submit.setOnClickListener(new View.OnClickListener()
    {
      public void onClick(View paramAnonymousView)
      {
        paramAnonymousView = MainActivity.this.inputCode.getText().toString();
        if (MainActivity.this.securityCheck(paramAnonymousView))
        {
          paramAnonymousView = new Intent(MainActivity.this, ResultActivity.class);
          MainActivity.this.startActivity(paramAnonymousView);
          return;
        }
        Toast.makeText(MainActivity.this.getApplicationContext(), "验证码校验失败", 0).show();
      }
    });
  }

  public native boolean securityCheck(String paramString);
}
```

用反编译工具

注意，只有当“MainActivity.this.securityCheck(str)”返回1，会调用以下函数，提示成功！

smali

- android.annotation
- com.yaotong.crackme
  - BuildConfig
    - BuildConfig
      - DEBUG : boo
  - MainActivity
    - MainActivity
      - btn_submit :
      - inputCode : I
      - onCreate(Bu
      - securityChec
      - {...}
  - R
  - ResultActivity
    - ResultActivity
      - onCreate(Bu

MainActivity.class | ResultActivity.class

```
package com.yaotong.crackme;

import android.app.Activity;

public class ResultActivity
  extends Activity
{
  public void onCreate(Bundle paramBundle)
  {
    super.onCreate(paramBundle);
    paramBundle = new TextView(this);
    paramBundle.setText("Congratulations!!!You Win!!");
    setContentView(paramBundle);
  }
}
```

用反编译工具

这个时候，需要分析函数“securityCheck(String paramString)”，定义在so文件中。

Name

- JNI_OnLoad
- Java_com_yaotong_crackme_MainActivity_securityCheck
- _Unwind_Complete

```
signed int __fastcall Java_com_yaotong_crackme_MainActivity_securityCheck(JNIEnv *a1, int a2, int a3)
{
  JNIEnv *v3; // r5
  int v4; // r4
  unsigned __int8 *v5; // r0
  char *v6; // r2
  int v7; // r3
  signed int v8; // r1

  v3 = a1;
  v4 = a3;
  if ( !byte_6359 )
  {
    sub_2494(&unk_6304, 8, &unk_446B, &unk_4468, 2, 7);
    byte_6359 = 1;
  }
  if ( !byte_635A )
  {
    sub_24F4(&unk_636C, 25, &unk_4530, &unk_4474, 3, 117);
    byte_635A = 1;
  }
  _android_log_print(4, &unk_6304, &unk_636C);
  v5 = (unsigned __int8 *)((int (__fastcall *)(JNIEnv *, int, _DWORD))(*v3
  v6 = off_628C;
  while ( 1 )
  {
    v7 = (unsigned __int8)*v6;
    if ( v7 != *v5 )
      break;
    ++v6;
    ++v5;
    v8 = 1;
    if ( !v7 )
      return v8;
  }
  return 0;
}
```

```
v6 = off_628C;
while ( 1 )
{
  v7 = (unsigned __int8)*v6;
  if ( v7 != *v5 )
    break;
```

xrefs to off_628C

Directi	Ty	Address	Text
	r	Java_com_yaotong_cr···	LDR R2, [R1,R7]; off_628C ...
D···	o	.text:off_1308	DCD off_628C - 0x5FBC

OK Cancel Search Help

Line 1 of 2

```
data:0000628C off_628C        DCD aWojiushidaan      ; DATA XREF: Java_com_yaotong_crackme_Mair
data:0000628C                                        ; .text:off_1308↑o  会是它吗？
data:0000628C ; .data         ends                   ; "wojiushidaan"
```

会是它吗?

```
data:0000628C off_628C        DCD aWojiushidaan        ; DATA XREF: Java_com_yaotong_crackme_Mair
data:0000628C                                          ; .text:off_1308↑o  会是它吗?
data:0000628C ; .data         ends                     ;"wojiushidaan"
```

- 运行程序，试一试。应该是错误的！
- 程序一定是在什么地方修个了这个字符串。但代码中找不到明显的与这个变量相关的代码，怎么办？
- 如果能够进行动态调试，直接定位到比较这个变量的地方，就会知道它本来的面目是什么了！但是，设计者会让我们轻易得逞吗？当然不会了……反调试是必须的！
- 什么时候会加入反调试的代码呢？当然是在程序刚刚开始运行的时候做这件事情了。
- 好了，接下来要搞清楚，Android程序是从哪里开始的。

Android程序运行过程

- 首先是“init_array”，Android系统在加载App时，通过系统的linker程序先加载这个函数，对App进行初始化，
- 然后再调用“JNI_OnLoad”

Android程序-init_array

- shift+F7

IDA Vie··· | Pseudoco··· | Strings win··· | Program Segmenta··· | Hex Vie··· | St

Name	Start	End	R	W	X	D	L	Align	Bas	Type	Class	AD	T	DS
LOAD	00000000	000010A8	R	.	X	.	L	mempage	01	public	CODE	32	00	0C
.plt	000010A8	00001164	R	.	X	.	L	dword	04	public	CODE	32	00	0C
.text	00001164	00004444	R	.	X	.	L	dword	05	public	CODE	32	00	0C
LOAD	00004444	00004450	R	.	X	.	L	mempage	01	public	CODE	32	00	0C
.rodata	00004450	00004562	R	.	.	.	L	para	06	public	CONST	32	00	0C
LOAD	00004562	00004564	R	.	X	.	L	mempage	01	public	CODE	32	00	0C
.ARM.extab	00004564	000045AC	R	.	.	.	L	dword	07	public	CONST	32	00	0C
.ARM.exidx	000045AC	000046BC	R	.	.	.	L	dword	08	public	CONST	32	00	0C
.fini_array	00005E84	00005E8C	R	W	.	.	L	dword	09	public	DATA	32	00	0C
.init_array	00005E8C	00005E94	R	W	.	.	L	dword	0A	public	DATA	32	00	0C
LOAD	00005E94	00005F94	R	W	.	.	L	mempage	02	public	DATA	32	00	0C
.got	00005F94	00006000	R	W	.	.	L	dword	0B	public	DATA	32	00	0C
.data	00006000	00006290	R	W	.	.	L	dword	0C	public	DATA	32	00	0C
.bss	00006290	00006390	R	W	.	.	L	para	0D	public	BSS	32	00	0C
.prgend	00006390	00006391	?	?	?	.	L	byte	0E	public		32	00	0E
extern	00006394	000063D8	?	?	?	.	L	dword	0F	public		32	00	0F
abs	000063D8	000063E4	?	?	?	.	L	dword	10	public		32	00	10

```
.init_array:00005E8C ; Segment type: Pure data
.init_array:00005E8C                 AREA .init_array, DATA
.init_array:00005E8C                 ; ORG 0x5E8C
.init_array:00005E8C                 DCD sub_2378
.init_array:00005E90                 DCB     0
.init_array:00005E91                 DCB     0
.init_array:00005E92                 DCB     0
.init_array:00005E93                 DCB     0
.init_array:00005E93 ; .init_array   ends
```

```
1 int sub_2378()
2 {
3   return sub_22AC((int)sub_1CA8);
4 }
```

Android程序-init_array

```
int sub_2378()
{
  return sub_22AC((int)sub_1CA8);
}
```

```
int __fastcall sub_22AC(int a1)
{
  int v1; // r4
  int *v2; // r5
  int v3; // r1
  int *v4; // r0
  int result; // r0

  v1 = a1;
  v2 = (int *)dword_62C8;
  if ( !dword_62C8 )
  {
    v2 = (int *)malloc(0x30u);
    *v2 = 0;
    v2[11] = 0;
    dword_62C4 = (int)v2;
    dword_62C8 = (int)v2;
  }
  v3 = *v2;
  if ( *v2 < 10 )
  {
    v4 = v2;
  }
  else
  {
    v4 = (int *)malloc(0x30u);
    *v4 = 0;
    v4[11] = 0;
    v2[11] = (int)v4;
    dword_62C8 = (int)v4;
    v3 = *v4;
  }
  v4[v3 + 1] = v1;
  result = dword_62C8;
  ++*(_DWORD *)dword_62C8;
  return result;
}
```

```
int (*sub_1CA8())(void)
{
  int (__fastcall *v0)(signed int, void *); // r4
  int (*result)(void); // r0

  if ( !byte_635F )
  {
    sub_24F4((int)&unk_62D7, 6, (int)&unk_4509, (int)"9HbB", 4u, 197);// //dlsym
    byte_635F = 1;
  }
  v0 = (int (__fastcall *)(signed int, void *))dlsym((void *)0xFFFFFFFF, (const char *)&unk_62D7);
  if ( !byte_6360 )
  {
    sub_24F4((int)&unk_62EF, 7, (int)&unk_448B, (int)&unk_4488, 2u, 213);// getpid
    byte_6360 = 1;
  }
  dword_6294 = (int (*)(void))v0(-1, &unk_62EF);
  if ( !byte_6361 )
  {
    sub_239C(&unk_630C, 8, (char *)&unk_44F3, (int)"LNAt", 4u);// sprintf
    byte_6361 = 1;
  }
  dword_6298 = v0(-1, &unk_630C);
  if ( !byte_6362 )
  {
    sub_239C(&unk_62DD, 6, (char *)&unk_44AC, (int)"cOXt", 4u);// fopen
    byte_6362 = 1;
  }
  dword_629C = v0(-1, &unk_62DD);
  if ( !byte_6363 )
  {
    sub_24F4((int)&unk_62E3, 6, (int)&unk_4481, (int)"BMT", 3u, 1);// fgets
    byte_6363 = 1;
  }
  dword_62A0 = (int (__fastcall *)(_DWORD, _DWORD, _DWORD))v0(-1, &unk_62E3);
  if ( !byte_6364 )
  {
    sub_239C(&unk_62F6, 7, (char *)&unk_44C4, (int)&unk_44C1, 2u);// strstr
    byte_6364 = 1;
  }
  dword_62A4 = v0(-1, &unk_62F6);
  if ( !byte_6365 )
  {
    sub_254C(&unk_62FD, 7, &unk_44CC, &unk_44FC, 0, 1);// sscanf
    byte_6365 = 1;
  }
```

Android程序-init_array

```
int __fastcall sub_24F4(int result, int a2, int a3, int a4, unsigned int a5, int a6)
{
  int v6; // r7
  unsigned int v7; // r4

  v6 = result;
  if ( a2 )
  {
    v7 = 0;
    do
    {
      result = (*(unsigned __int8 *)(a3 + v7) ^ a6) - *(unsigned __int8 *)(a4 + v7 % a5);
      *(_BYTE *)(v6 + v7++) = result;
    }
    while ( a2 != v7 );
  }
  return result;
}
```

```
_BYTE *__fastcall sub_254C(_BYTE *result, int a2, unsigned __int8 *a3, int a4, int a5, char a6)
{
  unsigned int v6; // r3
  unsigned int v7; // t1

  if ( a2 )
  {
    v6 = a3[a2 - 1];
    do
    {
      v7 = *a3++;
      --a2;
      *result++ = ((_BYTE)v7 << a6) | (v6 >> (8 - a6));
      v6 = v7;
    }
    while ( a2 );
  }
  return result;
}
```

Android程序-JNI_OnLoad

```
signed int __fastcall JNI_OnLoad(JNIEnv *a1)
{
  JNIEnv *v1; // r4
  _DWORD *v2; // r5
  int v3; // r6
  _DWORD *v4; // r6
  signed int v5; // r6
  int v7; // [sp-8h] [bp-28h]
  char v8; // [sp+0h] [bp-20h]

  v1 = a1;
  dword_62C8 = 0;
  v2 = (_DWORD *)dword_62C4;
  if ( dword_62C4 )
  {
    do
    {
      if ( *v2 >= 1 )
      {
        v3 = 0;
        do
          ((void (*)(void))v2[v3++ + 1])();
        while ( v3 < *v2 );
      }
      v4 = (_DWORD *)v2[11];
      free(v2);
      v2 = v4;
    }
    while ( v4 );
    dword_62C4 = 0;
  }
  dword_62B4(&v8, 0, sub_16A4, 0, 0);
  sub_17F4();
  v5 = 65540;
  if ( ((int (__fastcall *)(JNIEnv *, int *, signed int))(*v1)->FindClass)(v1, &v7, 65540) )
    v5 = -1;
  return v5;
}
```

```
void __noreturn sub_16A4()
{
  while ( 1 )
  {
    sub_130C();
    dword_62B0(3);
  }
}
```

```
int sub_130C()
{
  int v0; // r8
  void (__fastcall *v1)(char *, void *, int); // r4
  int (__fastcall *v2)(char *, void *); // r5
  unsigned __int8 *v3; // r6
  char *v4; // r4
  int (__fastcall *v5)(char *, void *); // r5
  char *v6; // r11
  unsigned __int8 *v7; // r4
  char *v8; // r6
  void (__fastcall *v9)(char *, void *, char *, int *); // r11
  int v11; // [sp+34h] [bp-314h]
  int v12; // [sp+38h] [bp-310h]
  char v13; // [sp+3Ch] [bp-30Ch]
  char v14; // [sp+BCh] [bp-28Ch]
  char v15; // [sp+2BCh] [bp-8Ch]
  int v16; // [sp+320h] [bp-28h]

  _aeabi_memset(&v15, 100, 0);
  v0 = dword_6294();                                  // getpid
  v1 = (void (__fastcall *)(char *, void *, int))dword_6298;// sprintf
  // pthread_create
```

Android程序-JNI_OnLoad

```
1 int sub_130C()
2 {
3   int v0; // r8
```

```
20   _aeabi_memset(&v15, 100, 0);
21   v0 = dword_6294();                          // getpid
22   v1 = (void (__fastcall *)(char *, void *, int))dword_6298;// sprintf
23   if ( !byte_635B )
24   {
25     sub_239C(&unk_6349, 16, (char *)&unk_4550, (int)"s!#L", 4u);//  /proc/%d/status
26     byte_635B = 1;
27   }
28   v1(&v15, &unk_6349, v0);                    // sprintf(&v15, "/proc/%d/status", getpid());
29   v2 = (int (__fastcall *)(char *, void *))dword_629C;// fopen
30   if ( !byte_635C )
31   {
32     sub_254C(&unk_6290, 2, (unsigned __int8 *)&unk_454A, (int)&unk_44FC, 0, 1);// r
33     byte_635C = 1;
34   }
35   v11 = v2(&v15, &unk_6290);                  // v11 = fopen(&v15, "r");
36   _aeabi_memset(&v14, 512, 0);
37   v3 = &stru_2C8.st_info;                     // unk_6290 - 0x5FBC
```

Android程序-JNI_OnLoad

```
38  if ( dword_62A0(&v14, 512, v11) )              // fgets(&v14, 512, v11);
39  {
40    v4 = &v13;
41    while ( 1 )
42    {
43      v5 = (int (__fastcall *)(char *, void *))dword_62A4;// strstr
44      if ( !byte_635D )
45      {
46        v6 = v4;
47        v7 = v3;
48        v8 = (char *)&GLOBAL_OFFSET_TABLE_ + (_DWORD)v3;// unk_6290
49        sub_24F4((int)(v8 + 0x95), 10, (int)&unk_4496, (int)&unk_4493, 2u, 157);// unk_6325=v8+0x95:TracePid
50        v8[205] = 1;
51        v3 = v7;
52        v4 = v6;
53      }
54      if ( v5(&v14, &unk_6325) )                 // strstr(v14, "TracePid")
55      {
56        _aeabi_memset(v4, 128, 0);
57        v12 = 0;
58        v9 = (void (__fastcall *)(char *, void *, char *, int *))dword_62A8;// sscanf
59        if ( !byte_635E )
60        {                                        // unk_62D1 = (_BYTE *)&GLOBAL_OFFSET_TABLE_ + (_DWORD)v3 + 0x41:
61                                                 // %s %d
62          sub_239C((_BYTE *)&GLOBAL_OFFSET_TABLE_ + (_DWORD)v3 + 0x41, 6, (char *)&unk_4461, (int)"L79", 3u);
63          *((_BYTE *)&GLOBAL_OFFSET_TABLE_ + (_DWORD)v3 + 206) = 1;
64        }
65        v9(&v14, &unk_62D1, v4, &v12);           // sscanf(&v14, "%s %d", v4, &v12);
66        if ( v12 >= 1 )
67          break;
68      }
69      if ( !dword_62A0(&v14, 512, v11) )         // fgets(&v14, 512, v11)
70        return _stack_chk_guard - v16;
71    }
72    (*(void (__fastcall **)(int, signed int))((char *)&GLOBAL_OFFSET_TABLE_ + (_DWORD)v3 + (unsigned int)&dword_1C))
73      v0,
74      9);                                        // dword_62AC:kill
75  }
76  return _stack_chk_guard - v16;
77 }
```

Android程序-sub_17F4

```
1 int sub_17F4()
2 {
3   int v0; // r0
4   int v1; // r0
5   int v2; // r1
6   int v3; // r0
7   int v4; // r0
8   int v5; // r2
9   signed int v6; // r0
10  int v7; // r0
11
12  dword_62B8((unsigned int)&jolin & -_page_size & 0xFFFFFFFE);// mprotect(), &join=0x00001720
13  v0 = dword_62C0();                                  // lrand48()
14  v1 = _floatsidf(8 * (v0 % 100) + 184);
15  v3 = _muldf3(v1, v2, 0x66666666, 0x3FF66666);
16  v4 = _fixdfsi(v3);
17  v5 = ((v4 + 3) * v4 + 2) * v4;
18  v6 = 0;
  switch ( v5 % 6 + 4 )
  {
    case 0:
      do
      {
        *(_BYTE *)(v6 + ((unsigned int)&jolin & 0xFFFFFFFE)) ^= byte_61B4[v6 % 108];
        ++v6;
      }
      while ( (int *)v6 != &dword_D4 );          // &dword_D4=0xD4
      break;
53    case 4:
54      do
55      {
56        *(_BYTE *)(v6 + ((unsigned int)&jolin & 0xFFFFFFFE)) ^= byte_6004[v6 % 108];
57        ++v6;
58      }
59      while ( (int *)v6 != &dword_D4 );
60      break;
69  }
70  v7 = dword_62BC();                             // cacheflush();
71  return ((int (__fastcall *)(int))jolin)(v7);
72 }
```

luhw@hust.edu.cn

Android程序-jolin

```
.text:00001720                 EXPORT jolin
.text:00001720 jolin                                   ; CODE XREF: sub_17F4+330↓p
.text:00001720                                         ; DATA XREF: LOAD:000001C8↑o ...
.text:00001720                 MCRLT   p14, 6, SP,c11,c6, 4
.text:00001724                 MOVLTS  PC, #0xF23FFFFF
.text:00001728                 TEQVS   R9, #0xB80000
.text:0000172C                 STRNE   R4, [R8,#-0x7F6]
.text:00001730                 ANDVS   R5, R9, #0xCE00000
.text:00001734                 STRLS   R2, [R5,R4,ASR#13]
.text:00001738                 MOVLTS  R12, #0xD43FFFFF
.text:0000173C                 CMPHI   R6, R4,LSL R7
.text:00001740                 ANDNES  R8, R8, R10,LSL#15
.text:00001744                 ANDVS   R5, R6, R4,ROR#12
.text:00001748                 STRVC   R7, [R6],#0x6E1
.text:0000174C                 MOVLTS  LR, #0xDE3FFFFF
.text:00001750                 LDRGE   R8, [R7],#0x796
.text:00001754                 STRNE   R10, [R3,#0x736]
.text:00001758                 SSATVS  R10, #0xA, R5,ASR#18
.text:0000175C                 STRPLBT R3, [R6],#0x6CF
.text:00001760                 STRLTB  LR, [R6,R3,LSL#11]
.text:00001760 ; ---------------------------------------------------------------------------
.text:00001764                 DCD 0xC716A792, 0x1043C757, 0x64365607, 0x32A216C6, 0xB5C6F5FE
.text:00001764                 DCD 0xE5568BF9, 0x1066E774, 0x64365609, 0x2B706D7, 0xB5C6E5AB
.text:00001764                 DCD 0xB22286A3, 0xB3F9E5D3, 0x63549792, 0x16277794, 0x64174A13
.text:00001764                 DCD 0x92E466C4, 0xB5C6E5F2, 0x83149797, 0x1043971F, 0x66367662
.text:00001764                 DCD 0x72B946D2, 0xB384E58F, 0xA4729D97, 0x1045A735, 0x62545664
.text:00001764                 DCD 0x56E636A2, 0xB5C6D587, 0xC323B7BA, 0x14A83865, 0x6F2BDE56
.text:00001764                 DCD 0xD7665E4E, 0x5666F757, 0xF929784A, 0xF587E5A2, 0xF2F47A13
.text:00001764                 DCD 0x881973B5
.text:000017F4
```

去除反调试

```
v9(&v14, &unk_62D1, v4, &v12);          // sscanf(&v14, "%s %d", v4, &v12);
if ( v12 >= 1 )
  break;
```

```
text:000015C8 38 30 8D E2                    ADD     R3, SP, #0x348+var_310
text:000015CC 3B FF 2F E1                    BLX     R11
text:000015D0 38 00 9D E5                    LDR     R0, [SP,#0x348+var_310]
text:000015D4 01 00 50 E3                    CMP     R0, #1
text:000015D8 08 00 00 AA                    BGE     loc_1600
```

```
.text:000015C8 38 30 8D E2                    ADD     R3, SP, #0x348+var_310
.text:000015CC 3B FF 2F E1                    BLX     R11
.text:000015D0 38 00 9D E5                    LDR     R0, [SP,#0x348+var_310]
.text:000015D4 00 00 50 E3                    CMP     R0, #0
.text:000015D8 08 00 00 BA                    BLT     loc_1600
```

```
v9(&v14, &unk_62D1, v4, &v12);
if ( v12 < 0 )
  break;
```

还原 jolin

```
12  dword_62B8((unsigned int)&jolin & -_page_size & 0xFFFFFFFE);// mprotect(), &join=0x00001720
13  v0 = dword_62C0();                                          // lrand48()
14  v1 = _floatsidf(8 * (v0 % 100) + 184);
15  v3 = _muldf3(v1, v2, 0x66666666, 0x3FF66666);
16  v4 = _fixdfsi(v3);
17  v5 = ((v4 + 3) * v4 + 2) * v4;
18  v6 = 0;
19  switch ( v5 % 6 + 4 )
      case 4:
        do
        {
          *(_BYTE *)(v6 + ((unsigned int)&jolin & 0xFFFFFFFE)) ^= byte_6004[v6 % 108];
          ++v6;
        }
        while ( (int *)v6 != &dword_D4 );
        break;
```

```
_BYTE byte_6004[108]
te_6004        DCB 0xA6, 0x96, 0xE6, 0x57, 0x87, 0xF5, 0x66, 0x56, 0x96
                                   ; DATA XREF: sub_17F4+2E0↑o
                                   ; .text:off_1B54↑o
               DCB 0x87, 0xB6, 0x86, 0xF6, 0x67, 0x87, 0xF5, 0x66, 0x56
               DCB 0x96, 0x87, 0xC6, 0x76, 6, 0x77, 0x87, 0xF5, 0x66
               DCB 0x56, 0x96, 0x87, 0xD6, 0x66, 0x16, 0x87, 0x87, 0xF5
               DCB 0x66, 0x56, 0x96, 0x87, 0xE6, 0x56, 0x26, 0x97, 0x87
               DCB 0xF5, 0x66, 0x56, 0x96, 0x87, 0xF6, 0x46, 0x36, 0xA7
               DCB 0x87, 0xF5, 0x66, 0x56, 0x96, 0x87, 0xA6, 0x36, 0x46
               DCB 0xB7, 0x87, 0xF5, 0x66, 0x56, 0x96, 0x87, 0xB6, 0x26
               DCB 0x56, 0xC7, 0x87, 0xF5, 0x66, 0x56, 0x96, 0x87, 0xC6
               DCB 0x16, 0x66, 0xD7, 0x87, 0xF5, 0x66, 0x56, 0x96, 0x87
               DCB 0xD6, 6, 0x76, 0xE7, 0x87, 0xF5, 0x66, 0x56, 0x96
               DCB 0x87, 0xD6, 6, 0x76, 0xE7, 0x87, 0xF5, 0x66, 0x56
```

```
00001720  96 DE CB BE 37 F5 F9 B3  2E B7 29 63 F6 47 08 15
00001730  CE 56 09 62 C4 26 85 97  AF C5 F3 B3 94 C7 46 81
00001740  8A 87 18 10 64 56 06 60  E1 76 86 74 87 E5 F6 B3
00001750  96 87 97 A4 36 A7 83 15  55 A9 B9 66 CF 36 E6 54
00001760  83 E5 C6 B7 92 A7 16 C7  57 C7 43 10 07 56 36 64
00001770  C6 16 A2 32 FE F5 C6 B5  F9 8B 56 E5 74 E7 66 10
00001780  09 56 36 64 D7 06 B7 02  AB E5 C6 B5 A3 86 22 B2
00001790  D3 E5 F9 B3 92 97 54 63  94 77 27 16 13 4A 17 64
000017A0  C4 66 E4 92 F2 E5 C6 B5  97 97 14 83 1F 97 43 10
000017B0  62 76 36 66 D2 46 B9 72  8F E5 84 B3 97 9D 72 A4
000017C0  35 A7 45 10 64 56 54 62  A2 36 E6 56 87 D5 C6 B5
000017D0  BA B7 23 C3 65 38 A8 14  56 DE 2B 6F 4E 5E 66 D7
000017E0  57 F7 66 56 4A 78 29 F9  A2 E5 87 F5 13 7A F4 F2
000017F0  B5 73 19 88
```

还原 jolin

```
00001720  30 48 2D E9 B0 00 9F E5  B8 30 9F E5 00 20 8F E0
00001730  A8 00 9F E5 02 50 83 E0  28 30 95 E5 02 40 90 E7
00001740  9C 00 9F E5 02 00 90 E7  07 20 A0 E3 00 10 90 E5
00001750  00 00 61 E2 00 00 04 E0  33 FF 2F E1 69 00 A0 E3
00001760  04 10 A0 E1 04 20 A0 E1  01 00 C4 E5 61 00 A0 E3
00001770  00 00 C4 E5 79 00 A0 E3  6F 0C 80 E3 02 00 E1 E5
00001780  6F 00 A0 E3 01 00 C1 E5  2C 10 A0 E3 05 10 C4 E5
00001790  54 10 9F E5 04 10 E2 E5  62 10 A0 E3 75 1C 81 E3
000017A0  02 10 E2 E5 75 10 A0 E3  01 10 C2 E5 09 10 C4 E5
000017B0  04 20 A0 E1 34 10 9F E5  08 10 E2 E5 01 1A 84 E2
000017C0  03 00 C2 E5 02 00 C2 E5  04 00 A0 E1 00 20 A0 E3
000017D0  2C 30 95 E5 33 FF 2F E1
000017E0  D0 02 00 00 DC FF FF FF
000017F0  63 75 6F 6F
```

```
v6 = off_628C;
while ( 1 )
{
  v7 = (unsigned __int8)*v6;
  if ( v7 != *v5 )
    break;
```

xrefs to off_628C

Directi	Ty	Address	Text
D···	o	.text:off_1308	DCD off_628C - 0x5FBC
D···	r	jolin+1C	LDR R4, [R0,R2]; off_628C ..
D···	o	.text:off_17E0	DCD off_628C - 0x5FBC

```
_0 + 1 + 2) = 'o';
12  byte_5[(_DWORD)v0] = ',';
13  byte_4[(_DWORD)v0] = 'u';
14  byte_4[(_DWORD)v0 + (unsigned int)&dword_0 + 2] = 'b';
15  byte_4[(_DWORD)v0 + (unsigned int)&dword_0 + 2 + (_DWORD)&dword_0 + 1] = 'u';
16  byte_9[(_DWORD)v0] = 'u';
17  byte_8[(_DWORD)v0] = 'c';
18  v1 = &byte_8[(_DWORD)v0];
19  *((_BYTE *)&dword_0 + (_DWORD)v1 + 3) = 'o';
20  *((_BYTE *)&dword_0 + (_DWORD)v1 + 2) = 'o';
21  return dword_62BC(v0, v0 + 4096, 0);
22 }
```

还原 off_628c

```
v6 = off_628C;
while ( 1 )
{
  v7 = (unsigned __int8)*v6;
  if ( v7 != *v5 )
    break;
```

xrefs to off_628C

Directi	Ty	Address	Text
D···	o	.text:off_1308	DCD off_628C - 0x5FBC
D···	r	jolin+1C	LDR R4, [R0,R2]; off_628C ..
D···	o	.text:off_17E0	DCD off_628C - 0x5FBC

```
LOAD:00000000 7F 45 4C 46        dword_0          DCD 0x464C457F
```

```
MOV     R0, #0x69 ; 'i'
MOV     R1, R4
MOV     R2, R4
STRB    R0, [R4,#(aWojiushidaan+1 - 0x4450)]
MOV     R0, #0x61 ; 'a'
STRB    R0, [R4]          ; "wojiushidaan"
MOV     R0, #'oy'
STRB    R0, [R1,#(aWojiushidaan+2 - 0x4450)]!
MOV     R0, #0x6F ; 'o'
STRB    R0, [R1,#(aWojiushidaan+3 - 0x4452)]
MOV     R1, #0x2C ; ','
STRB    R1, [R4,#(aWojiushidaan+5 - 0x4450)]
LDR     R1, ='ub,u'
STRB    R1, [R2,#(aWojiushidaan+4 - 0x4450)]!
MOV     R1, #'ub'
STRB    R1, [R2,#(aWojiushidaan+6 - 0x4454)]!
MOV     R1, #0x75 ; 'u'
STRB    R1, [R2,#(aWojiushidaan+7 - 0x4456)]
STRB    R1, [R4,#(aWojiushidaan+9 - 0x4450)]
MOV     R2, R4
LDR     R1, ='oduc'
STRB    R1, [R2,#(aWojiushidaan+8 - 0x4450)]!
```

```
int jolin()
{
  char *v0; // r4
  char *v1; // r2

  v0 = off_628C;
  dword_62B8((unsigned int)off_628C & -_page_size);// mprotect
  *((_BYTE *)&dword_0 + (_DWORD)v0 + 1) = 'i';
  *v0 = 97;
  *((_BYTE *)&dword_0 + (_DWORD)v0 + 2) = 'y';
  *((_BYTE *)&dword_0 + (_DWORD)v0 + (unsigned int)&dword_0 + 1 + 2) = 'o';
  byte_5[(_DWORD)v0] = ',';
  byte_4[(_DWORD)v0] = 'u';
  byte_4[(_DWORD)v0 + (unsigned int)&dword_0 + 2] = 'b';
  byte_4[(_DWORD)v0 + (unsigned int)&dword_0 + 2 + (_DWORD)&dword_0 + 1] = 'u';
  byte_9[(_DWORD)v0] = 'u';
  byte_8[(_DWORD)v0] = 'c';
  v1 = &byte_8[(_DWORD)v0];
  *((_BYTE *)&dword_0 + (_DWORD)v1 + 3) = 'o';
  *((_BYTE *)&dword_0 + (_DWORD)v1 + 2) = 'o';
  return dword_62BC(v0, v0 + 4096, 0);
}
```

还原 off_628c

```
v6 = off_628C;
while ( 1 )
{
  v7 = (unsigned __int8)*v6;
  if ( v7 != *v5 )
    break;
```

aiyou,bucuoo

xrefs to off_628C

Directi	Ty	Address	Text
D···	o	.text:off_1308	DCD off_628C - 0x5FBC
D···	r	jolin+1C	LDR R4, [R0,R2]; off_628C ..
D···	o	.text:off_17E0	DCD off_628C - 0x5FBC

```
LOAD:00000000 7F 45 4C 46        dword_0          DCD 0x464C457F
```

```
MOV     R0, #0x69 ; 'i'
MOV     R1, R4
MOV     R2, R4
STRB    R0, [R4,#(aWojiushidaan+1 - 0x4450)]
MOV     R0, #0x61 ; 'a'
STRB    R0, [R4]          ; "wojiushidaan"
MOV     R0, #'oy'
STRB    R0, [R1,#(aWojiushidaan+2 - 0x4450)]!
MOV     R0, #0x6F ; 'o'
STRB    R0, [R1,#(aWojiushidaan+3 - 0x4452)]
MOV     R1, #0x2C ; ','
STRB    R1, [R4,#(aWojiushidaan+5 - 0x4450)]
LDR     R1, ='ub,u'
STRB    R1, [R2,#(aWojiushidaan+4 - 0x4450)]!
MOV     R1, #'ub'
STRB    R1, [R2,#(aWojiushidaan+6 - 0x4454)]!
MOV     R1, #0x75 ; 'u'
STRB    R1, [R2,#(aWojiushidaan+7 - 0x4456)]
STRB    R1, [R4,#(aWojiushidaan+9 - 0x4450)]
MOV     R2, R4
LDR     R1, ='oduc'
STRB    R1, [R2,#(aWojiushidaan+8 - 0x4450)]!
```

```
int jolin()
{
  char *v0; // r4
  char *v1; // r2

  v0 = off_628C;
  dword_62B8((unsigned int)off_628C & -_page_size);// mprotect
  *((_BYTE *)&dword_0 + (_DWORD)v0 + 1) = 'i';
  *v0 = 97;
  *((_BYTE *)&dword_0 + (_DWORD)v0 + 2) = 'y';
  *((_BYTE *)&dword_0 + (_DWORD)v0 + (unsigned int)&dword_0 + 1 + 2) = 'o';
  byte_5[(_DWORD)v0] = ',';
  byte_4[(_DWORD)v0] = 'u';
  byte_4[(_DWORD)v0 + (unsigned int)&dword_0 + 2] = 'b';
  byte_4[(_DWORD)v0 + (unsigned int)&dword_0 + 2 + (_DWORD)&dword_0 + 1] = 'u';
  byte_9[(_DWORD)v0] = 'u';
  byte_8[(_DWORD)v0] = 'c';
  v1 = &byte_8[(_DWORD)v0];
  *((_BYTE *)&dword_0 + (_DWORD)v1 + 3) = 'o';
  *((_BYTE *)&dword_0 + (_DWORD)v1 + 2) = 'o';
  return dword_62BC(v0, v0 + 4096, 0);
}
```

无反调试程序的调试步骤

1. adb push d:\android_server（IDA的dbgsrv目录下）/data/local/tmp/android_server
2. adb shell
3. su(一定要有root权限)
4. cd /data/local/tmp
5. chmod 777 android_server(执行权限要给)
6. ./an*
7. 再开一个cmd，adb forward tcp:23946 tcp:23946（端口转发，调试手机上的某个进程要有协议支持通信）
8. 打开待调试的应用程序，就可以调试了

有反调试程序的调试步骤

- 在很多情况下我们遇到的是有反调试并且用上面的步骤，附加进去以后直接就退出了，这样的例子数不胜数，那就是反调试惹的祸。

1. 启动 android_server
2. adb forward tcp:23946 tcp:23946

有反调试程序的调试步骤

3. adb shell am start -D -n 包名/类名
4. 打开IDA，附加上对应的进程之后，设置IDA中的load so的时机，在debug options中设置一下
5. adb forward tcp:8700 jdwp:进程号（jdwp是后面jdb调试器的协议，转换到待调试的指定的应用程序）
6. jdb进行附加” jdb -connect com.sun.jdi.SocketAttach:hostname=localhost,port=8700
7. 可以下断点，开始调试了

IDA动态调试so时的三个层次

- so的加载时的过程：
- .init-->.init array-->JNI_Onload-->java_com_XXX;
- 在脱壳的过程中有时候会在一些系统级的.so中下断点比如：fopen, fget, dvmdexfileopen,等等
- 而.init以及.init_array一般会作为壳的入口地方，称它为外壳级的.so文件
- 这里归纳为三类：

1. 应用级别的：java_com_XXX；
2. 外壳级别的：JNI_Onload, .init, .init_array;
3. 系统级别的：fopen,fget,dvmdexfileopen；

IDA动态调试so时的三个层次

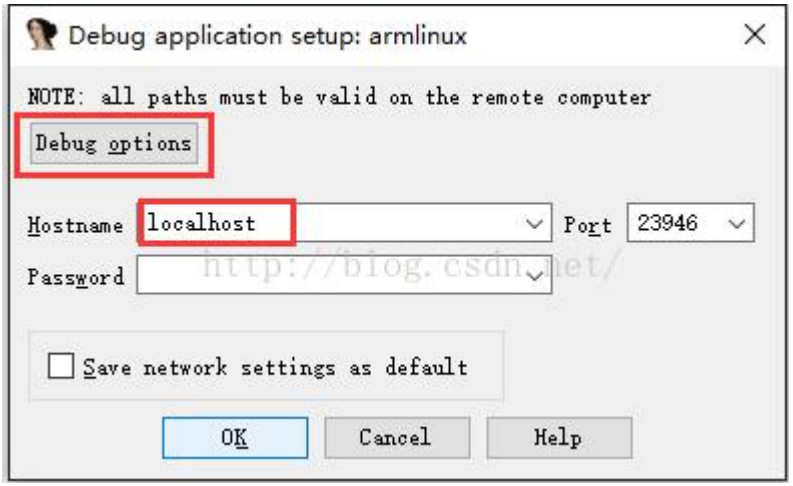
- 对于在应用级别的和系统级别的调试比较简单容易理解
- 从上面的.SO的加载执行过程我们知道如果说反调试放在外壳级别的.SO文件的话，就会遇到程序在应用级核心函数一下断点就退出的尴尬
- 事实上多数的反调试会放在这里，那么绕过反调试就必须要在这些地方下断点
- 下面就重点介绍一下如何在.init_array和JNI_Onload处理时下断点。

在JNI_Onload处下断点方法一

1. 启动android_server;
2. 端口转发以及调试模式启动

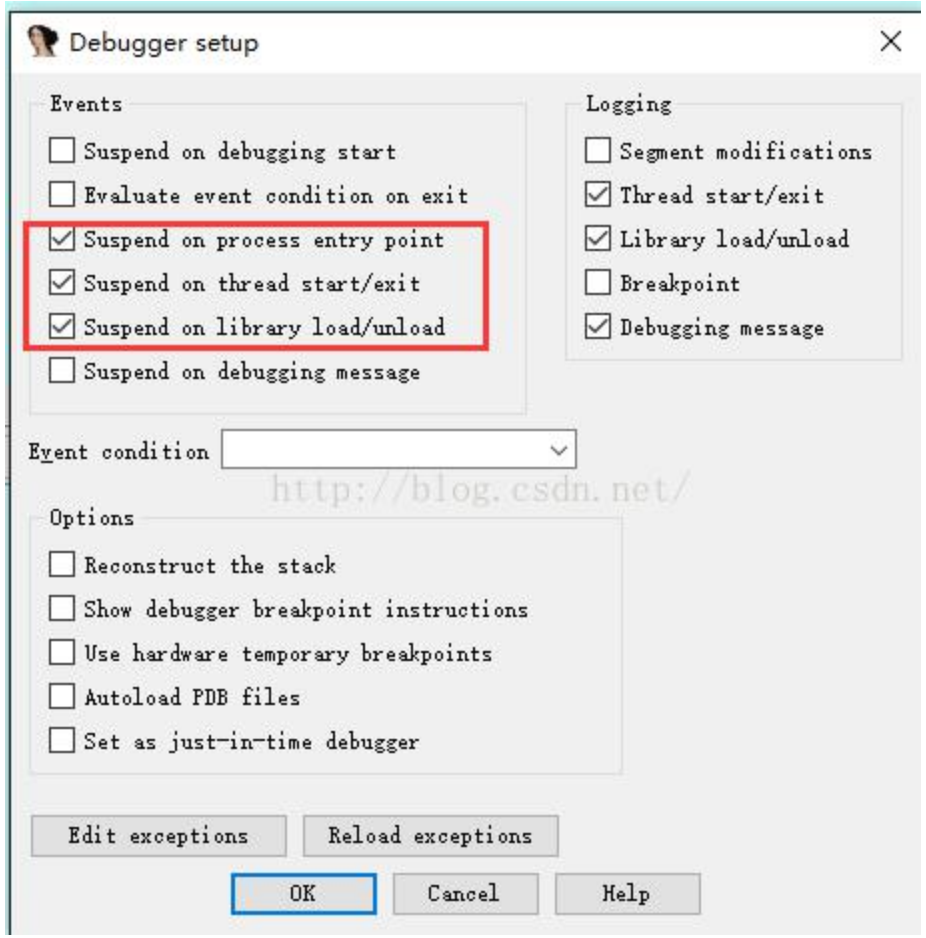
```
C:\Users\ZBB>adb forward tcp:23946 tcp:23946

C:\Users\ZBB>adb shell am start -D -n com.yaotong.crackme/com.yaotong.crackme.MainActivity
Starting: Intent { cmp=com.yaotong.crackme/.MainActivity }
```

3. 打开IDA，设置



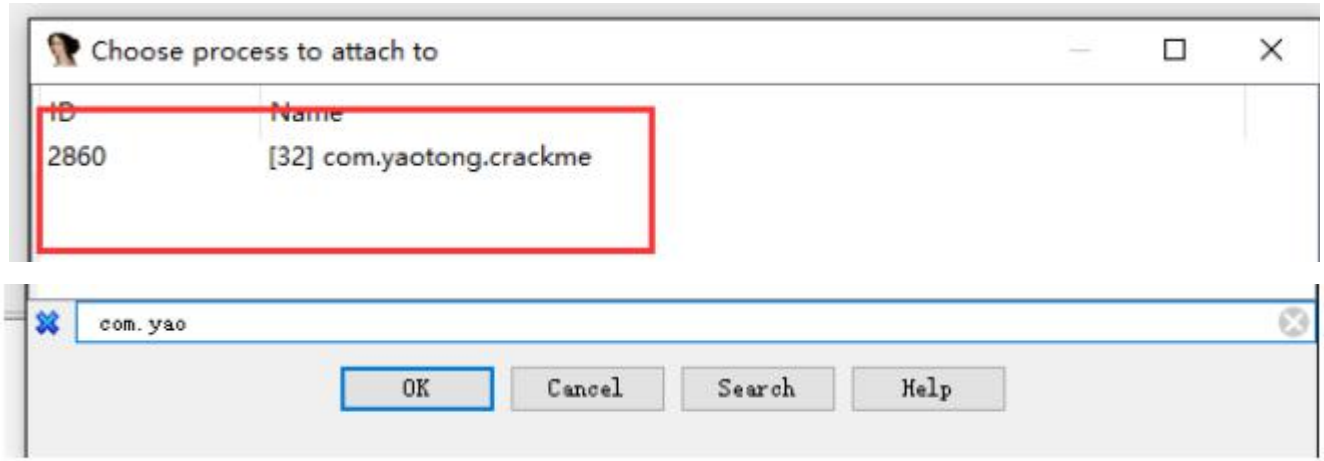
在JNI_Onload处下断点方法一

5. 这一步很重要在Debugger option下面选择这三个选项（让在load so的每个接口处停下来）



在JNI_Onload处下断点方法一

4. 附加上对应的进程进去之后如图



在JNI_Onload处下断点方法一

6. jdwp协议端口转发

```
C:\Users\ZBB>adb forward tcp:8700 jdwp:2860
```


7. .jdb附加

```
C:\Users\ZBB>jdb -connect com.sun.jdi.SocketAttach:hostname=localhost,port=8700
设置未捕获的java.lang.Throwable
设置延迟的未捕获的java.lang.Throwable
正在初始化jdb...
>
```


8. F9执行，忽略提示框；这时候运行到linker处，如图

```
linker:40121E27 DCB 0xF0 ;
linker:40121E28 DCB 0x4D ; M
linker:40121E29 DCB 0xFE ;
linker:40121E2A CODE16
PC linker:40121E2A DCB     8
LR linker:40121E2B DCB 0x48 ; H
```

在JNI_Onload处下断点方法一

IDA View-PC

```
libc.so:4011882D DCB    0
libc.so:4011882E DCB    0
libc.so:4011882F DCB 0xEF
libc.so:40118830 ; ---------------------------------
libc.so:40118830 LDMFD           SP!, {R4,R7}
libc.so:40118834 BX              LR
libc.so:40118834 ; ---------------------------------
libc.so:40118838 __futex_syscall4 DCB 0xF9
libc.so:40118839 DCB 0xFF
libc.so:4011883A DCB 0xFF
libc.so:4011883B DCB 0xEA
libc.so:4011883C __futex_wait DCB 0x90
libc.so:4011883D DCB    0
libc.so:4011883E DCB 0x2D ; -
libc.so:4011883F DCB 0xE9
libc.so:40118840 DCB    2
```

General registers

```
R0 FFFFFFFC
R1 00000080
R2 FFFFFA92

N 0
Z 1
C 1
V 0
Q 0
```

Modules

Path	Ba
/system/bin/app_process	400
/system/lib/libgabi++.so	400
/system/lib/libaudioutils.so	400
/system/lib/libsslengine.so	400
/system/bin/linker	400
/system/lib/libcamera_metadata.so	400
/system/lib/libcorkscrew.so	400

Hex View-1

```
40000000  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
40000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
40000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
40000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
40000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
40000050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
40000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
40000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
40000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
40000090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Stack view

```
BE8E71E8  41953E78  debug059:41953E78
BE8E71EC  BE8E7208  [stack]:BE8E7208
BE8E71F0  00000001
BE8E71F4  41953E68  debug059:41953E68
BE8E71F8  41953E78  debug059:41953E78
BE8E71FC  41953AD8  debug059:41953AD8
BE8E7200  41C61390  debug082:41C61390
BE8E7204  4010A820  libc.so:__pthread_cond_timec
BE8E7208  00000000
```

在JNI_Onload处下断点方法一

9. 这时候找JNI_Onload的绝对地址：基地址+相对地址；
10. 基地址为：ctrl+s显示为：

Choose segment to jump

Name	Start	End	R	W	X	D	L
libcrackme.so	4151E000	41523000	R	.	X	D	.
libcrackme.so	41523000	41524000	R	.	.	D	.
libcrackme.so	41524000	41525000	R	W	.	D	.

- 相对地址，用IDA静态分析libcrack.so可得到相对地址：

```
text:00001B9C JNI_OnLoad                                ; DATA XREF: LOAD:00000
text:00001B9C
text:00001B9C var_20          = -0x20
text:00001B9C
text:00001B9C                 STMFD   SP!, {R4-R9,R11,LR}
text:00001BA0                 ADD     R11, SP, #0x18
text:00001BA4                 SUB     SP, SP, #8
text:00001BA8                 MOV     R4, R0
```

- 绝对地址为：4151E000+1B9C=4151FB9C

在JNI_Onload处下断点方法一

- 按下“G”键输入4151FB9C
- 如图所示：按下F2下好断点，再按F9执行到断点处就可以愉快的调试了

```
libcrackme.so:4151FB9C
libcrackme.so:4151FB9C JNI_OnLoad
libcrackme.so:4151FB9C
libcrackme.so:4151FB9C var_20= -0x20
libcrackme.so:4151FB9C
libcrackme.so:4151FB9C STMFD           SP!, {R4-R9,R11,LR}
libcrackme.so:4151FBA0 ADD             R11, SP, #0x18
libcrackme.so:4151FBA4 SUB             SP, SP, #8
libcrackme.so:4151FBA8 MOV             R4, R0
libcrackme.so:4151FBAC LDR             R0, =(unk_41523FBC - 0x4151FBC0)
```

在JNI_Onload处下断点方法二

1. 首先把要分析的libcrackme.so文件拉进IDA里面在要下断点的JNI_Onload处下好断点如图所示：

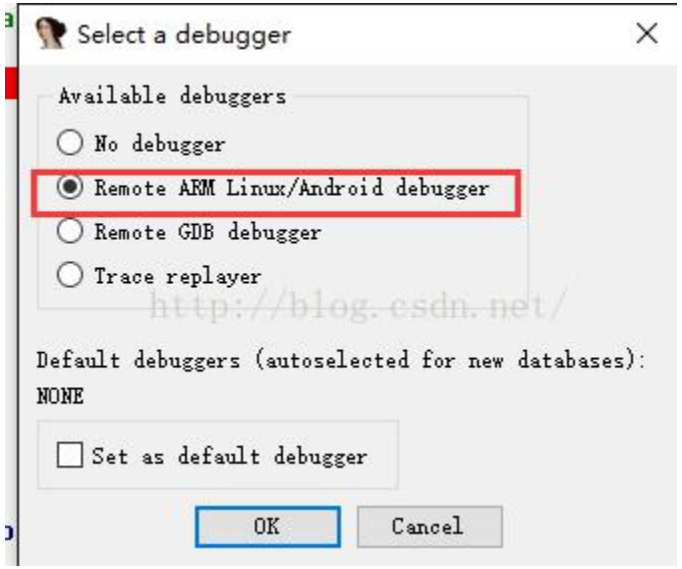
```
.text:00001B9C JNI_OnLoad
.text:00001B9C
.text:00001B9C handle          = -0x20
.text:00001B9C
.text:00001B9C                 STMFD   SP!, {R4-R9,R11,LR}
.text:00001BA0                 ADD     R11, SP, #0x18
.text:00001BA4                 SUB     SP, SP, #8
.text:00001BA8                 MOV     R4, R0
.text:00001BAC                 LDR     R0, =(_GLOBAL_OFFSET_TABLE_ - 0x1BC0)
.text:00001BB0                 LDR     R9, =(unk_6290 - 0x5FBC)
```

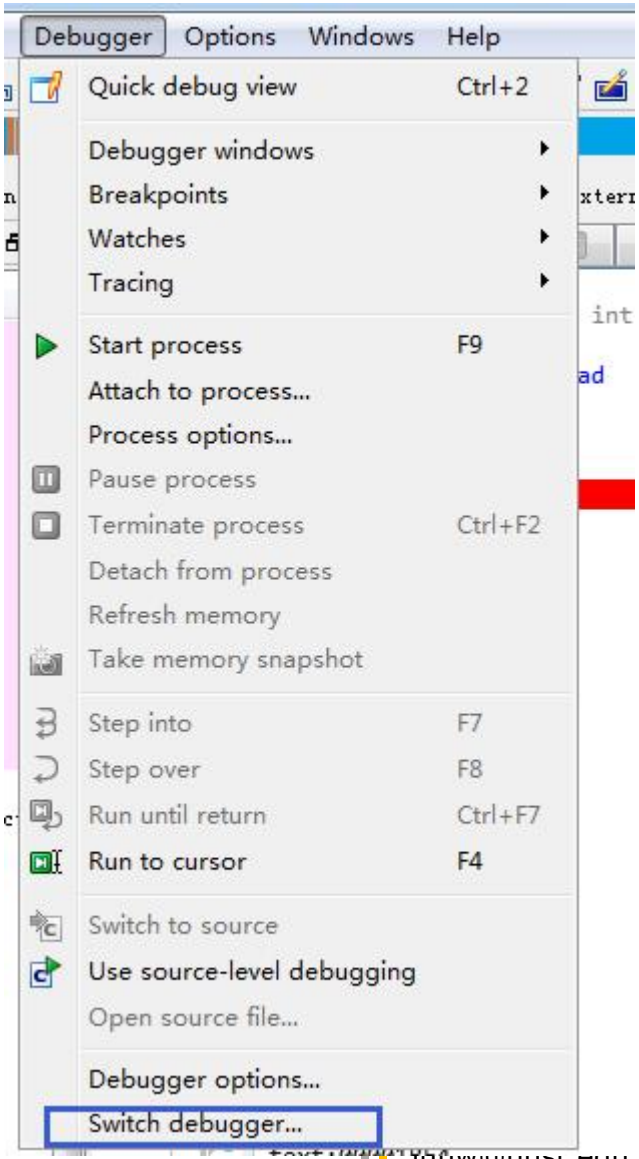
2. 启动android_server与上面一样；
3. 端口转发以及调试模式启动：如图所示

```
C:\Users\ZBB>adb forward tcp:23946 tcp:23946

C:\Users\ZBB>adb shell am start -D -n com.yaotong.crackme/com.yaotong.crackme.MainActivity
Starting: Intent { cmp=com.yaotong.crackme/.MainActivity }
```

在JNI_Onload处下断点方法二

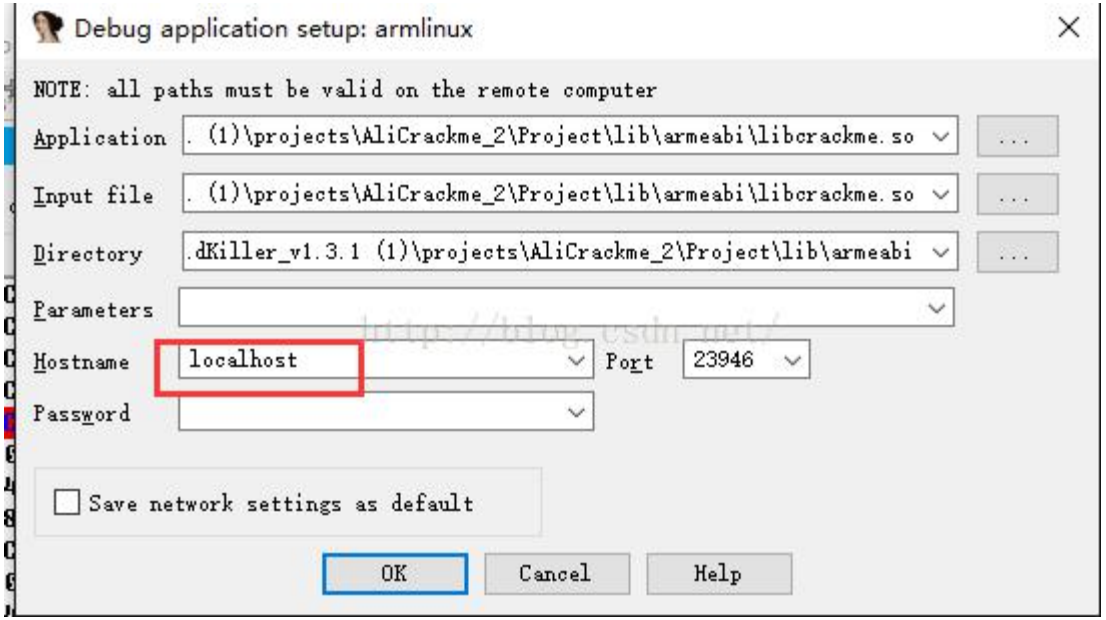
4. 先设置一下Debugger 如图所示：





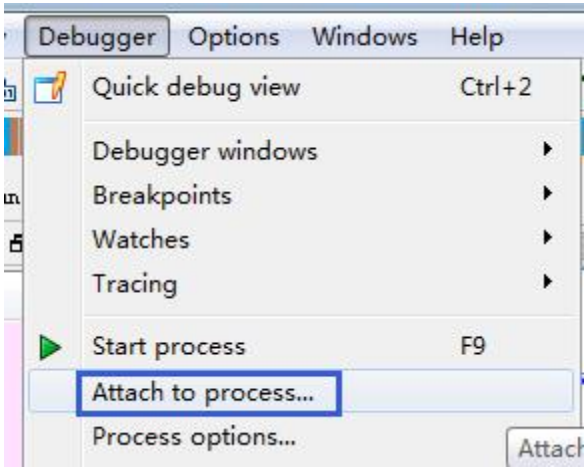
在JNI_Onload处下断点方法二

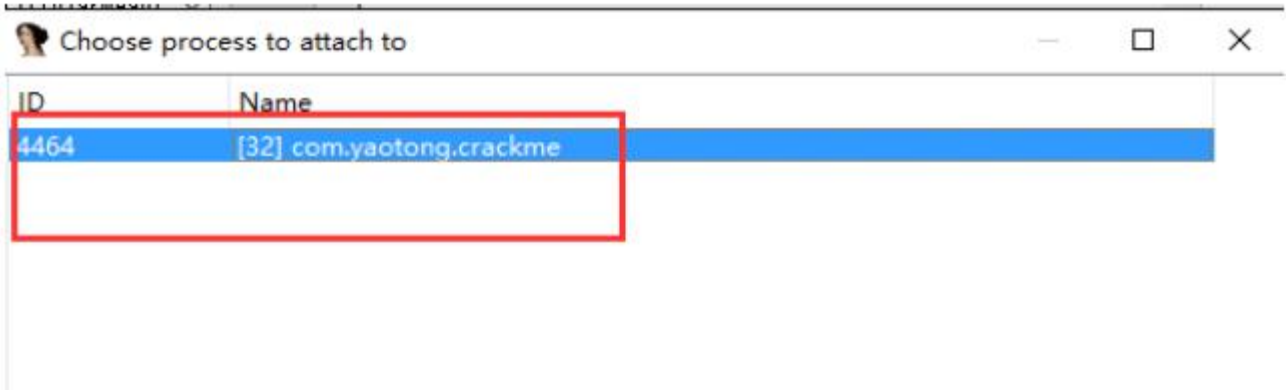
5. IDA进行附加进程回到之前静态分析libcrackme.so的IDA界面单击Debugger -> Process options 配置调试信息，这里只需配置hostname为localhost，其余的保持默认设置即可



在JNI_Onload处下断点方法二

6. 单击Debugger -> Attach to process进行附加进程





在JNI_Onload处下断点方法二

7. jdwp转发jdb附加

```
C:\Users\ZBB>adb forward tcp:8700 jdwp:4464

C:\Users\ZBB>jdb -connect com.sun.jdi.SocketAttach:hostname=localhost,port=8700
```

8. F9执行一路取消就OK，得到如图所示：

```
.text:4151FB9C JNI_OnLoad
.text:4151FB9C
.text:4151FB9C handle= -0x20
.text:4151FB9C
.text:4151FB9C STMFD   SP!, {R4-R9,R11,LR}
.text:4151FBA0 ADD     R11, SP, #0x18
.text:4151FBA4 SUB     SP, SP, #8
.text:4151FBA8 MOV     R4, R0
.text:4151FBAC LDR     R0, =(_GLOBAL_OFFSET_TABLE_ - 0x4151FBC0) ; fscanf
```

在JNI_Onload处下断点方法三

1. 如果JNI_Onload被加密了，就不能直接在里面设置断点，这时需要在libdvm.so里面设置断点。
2. Android虚拟机源码：dalvik2/vm/Native.cpp

```
84    v25 = (int (__fastcall *)(int, _DWORD))dlsym(v12, "JNI_OnLoad");
85    if ( v25 )
86    {
87      v26 = *(_DWORD *)(v14 + 908);
88      *(_DWORD *)(v14 + 908) = v3;
89      v30 = v26;
90      v27 = dvmChangeStatus(v14, 7);
91      if ( byte_B61D1 )
92        __android_log_print(4, "dalvikvm", "[Calling JNI_OnLoad for \"%s\"]", v5);
93      v28 = v25(dword_B8088, 0);
```

调用JNI_Onload函数

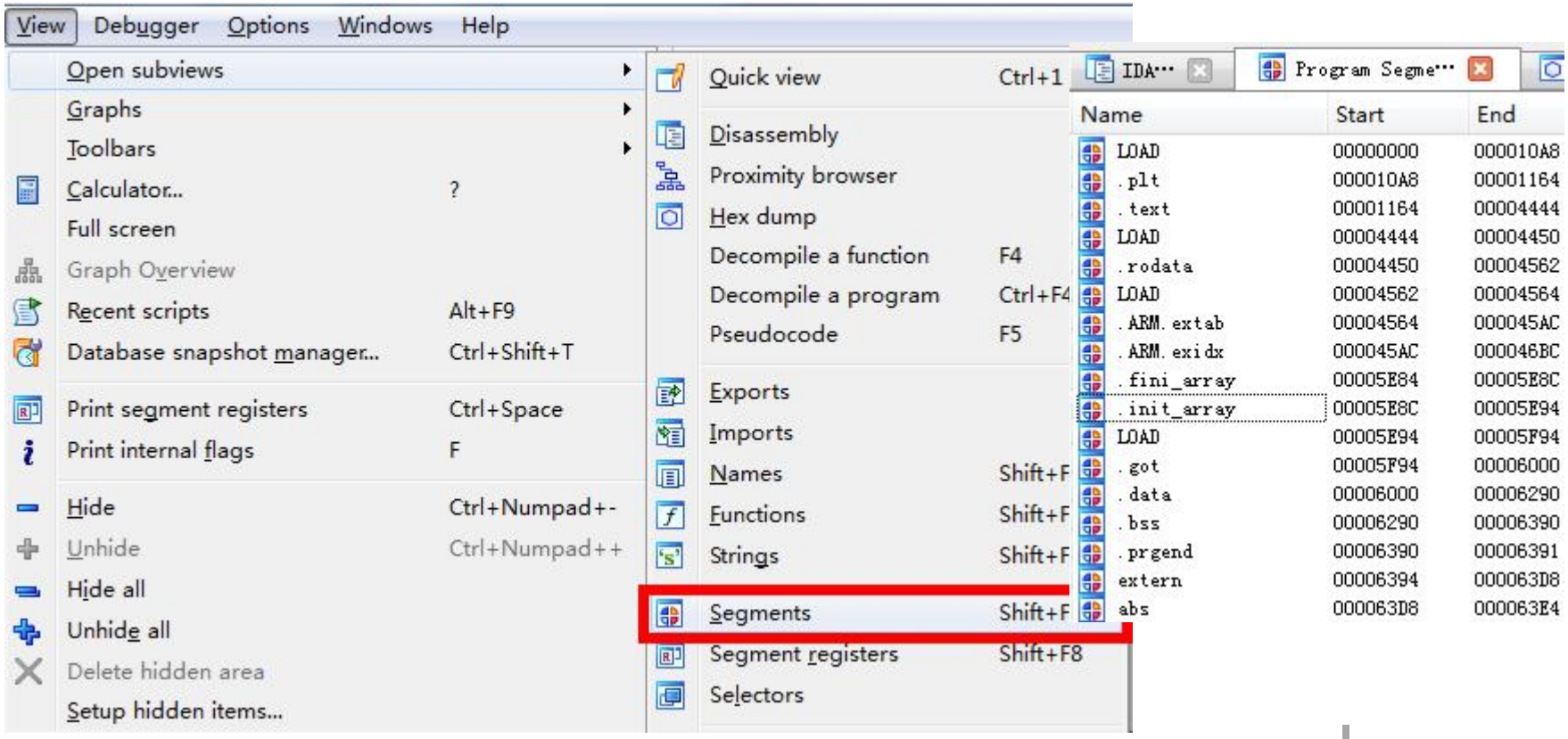
```
.text:0005082A    MOV     R0, R10 ; handle
.text:0005082C    ADD     R1, PC  ; "JNI_OnLoad"
.text:0005082E    BLX     dlsym
.text:00050832    MOV     R8, R0
.text:00050834    CBNZ    R0, loc 5084A
.text:00050874    LDR     R0, =(gDvmJni_ptr - 0xB1C74)
.text:00050876    LDR.W   R1, [R11,R0] ; gDvmJni
.text:0005087A    LDR     R0, [R1,#(dword_B8088 - 0xB8080)]
.text:0005087C    MOVS    R1, #0
.text:0005087E    BLX     R8
```

在.init_array处下断点

1. 我们知道so文件在被加载的时候会首先执行.init_array中的函数，然后再执行JNI_OnLoad()函数。
2. JNI_Onload()函数因为有符号表所以非常容易找到，但是.init_array里的函数需要自己去找一下。
3. 首先打开view ->Open subviews->Segments。然后点击.init.array就可以看到.init_array中的函数了。

在.init_array处下断点

1. 首先打开view ->Open subviews->Segments。然后点击.init.array就可以看到.init_array中的函数了。



luhw@hust.edu.cn