

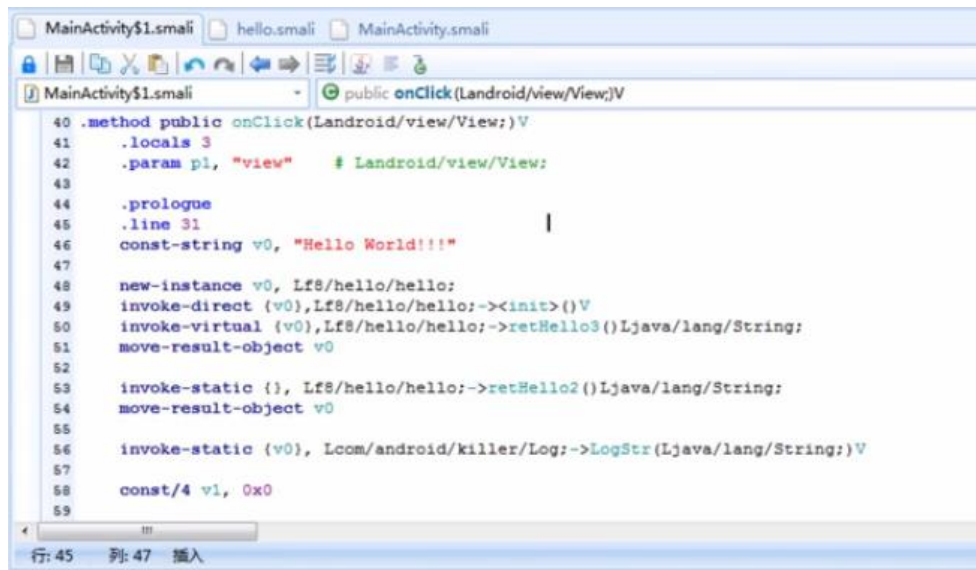
## 1. smali 代码插入

用 smali 代码编写程序与用 java 代码编写程序一样，在原理上是可行的，不过前提是对 smali 语法足够熟悉。以插入打印 log 代码为例，原理就是通过直接插入有效的 smali 代码，从而输出程序的数据内容。更为复杂一点的则可以直接修改整个程序的执行流程。

## 2. smali 代码编写

一个静态返回 HelloWorld 的方法：

- 1) `.class public Lf8/helloworld/helloStr;     #类声明`  
`.super Ljava/lang/Object;                #父类声明`  
  
`.method public static retHello()Ljava/lang/String; #函数声明`  
`.locals 1                                  #寄存器数量`  
`const-string v0, "Hello World from StaticMethod" #新建字符串`  
`return-object v0                            #返回 Object 类型`  
`.end method                                #方法结束`
- 2) 返回静态 field 的方法  
`.field public static final hStr:Ljava/lang/String; = "Hello World from static field"`  
`#field 声明与初始化`  
`.method public static retHello2()Ljava/lang/String;`  
`.local 1`  
`sget-object v0, Lf8/helloworld/helloStr;->hStr:Ljava/lang/String   #获取 filed`  
`return-object v0`  
`.end method`
- 3) 普通的函数  
`.method public constructor <init>()V`  
`.locals 0`  
`invoke-direct {p0}, Ljava/lang/Object; -> <init>()V`  
`return-void`  
`.end method`  
  
`.method public retHello3()Ljava/lang/String;`  
`.locals 1`  
`const-string v0, "Hello World from Method"`  
`return-object v0`  
`.end method`



#### 4) 普通的 field 与函数

.field public hStr2:Ljava/lang/String;

.method public constructor <init>()V

.locals 1

invoke-direct {p0}, Ljava/lang/Object;-><init>()V

const-string v0, "Hello field" #初始化非静态 field

input-object v0, p0, Lf8/helloworld/helloStr;->hStr2:Ljava/lang/String;

return-void

.end method

.method public retHello4()Ljava/lang/String;

.locals 1

iget-object v0, p0, Lf8/helloworld/helloStr;->hStr2:Ljava/lang/String;

return-object v0

.end method

调用的时候需要先初始化一个实例

new-instance v1, Lf8/helloworld/helloStr;

invoke-direct {v1}, Lf8/helloworld/helloStr;-><init>()V

invoke-virtual {v1}, Lf8/helloworld/helloStr;->retHello4()Ljava/lang/String;

move-result-object v1

```

1 .class Lfs/hello/hello;
2 .super Ljava/lang/Object;
3
4 .field public static final hStr:Ljava/lang/String; = "Hello World from static field"
5 .field public hStr2:Ljava/lang/String;
6
7 .method public constructor<init>()V
8     #p0 --> this
9     .locals 1
10    invoke-direct {p0}, Ljava/lang/Object;-><init>()V
11    const-string v0, "HelloWorld from field"
12    iput-object v0, p0, Lfs/hello/hello;->hStr2:Ljava/lang/String;
13    return-void
14 .end method
15
16 .method public static retHello()Ljava/lang/String;
17     .locals 1
18     const-string v0, "HelloWorld from staticMethod"
19     return-object v0
20 .end method

```

```

19 return-object v0
20 .end method
21
22 .method public static retHello2()Ljava/lang/String;
23     .locals 1
24     sget-object v0, Lfs/hello/hello;->hStr:Ljava/lang/String;
25     return-object v0
26 .end method
27
28 .method public static retHello3()Ljava/lang/String;
29     .locals 1
30     const-string v0, "HelloWorld from normal Method"
31     return-object v0
32 .end method
33
34 .method public static retHello4()Ljava/lang/String;
35     .locals 1
36     iget-object v0, p0, Lfs/hello/hello;->hStr2:Ljava/lang/String;
37     return-object v0
38 .end method

```

```

40 .method public onClick(Landroid/view/View;)V
41     .locals 3
42     .param p1, "view"    # Landroid/view/View;
43
44     .prologue
45     .line 31
46     const-string v0, "Hello World!!!"
47
48     new-instance v0, Lfs/hello/hello;
49     invoke-direct {v0}, Lfs/hello/hello;-><init>()V
50     invoke-virtual {v0}, Lfs/hello/hello;->retHello4()Ljava/lang/String;
51     move-result-object v0
52
53     invoke-static {v0}, Lcom/android/killer/Log;->LogStr(Ljava/lang/String;)V
54

```

### 3. Android Log

来自于 android/util/Log 包

方法为 d(String, String)l, e, i, v 等

调用方法为

invoke-static {v0, v1}, Landroid/util/Log;->v(Ljava/lang/String;Ljava/lang/String;)I

Android Killer 中自带

### 4. LoadLibrary

invoke-static {v0}, Ljava/lang/System;->loadLibrary(Ljava/lang/String;)V

### 5. stackTrace

打印当前函数堆栈，方法为 Thread.dumpStack();  
invoke-static {}, Ljava/lang/Thread;-->dumpStack()V

## 6. Method Trace

函数跟踪

invoke-static {}, Landroid/os/Debug;-->startMethodTracing()V #函数头部

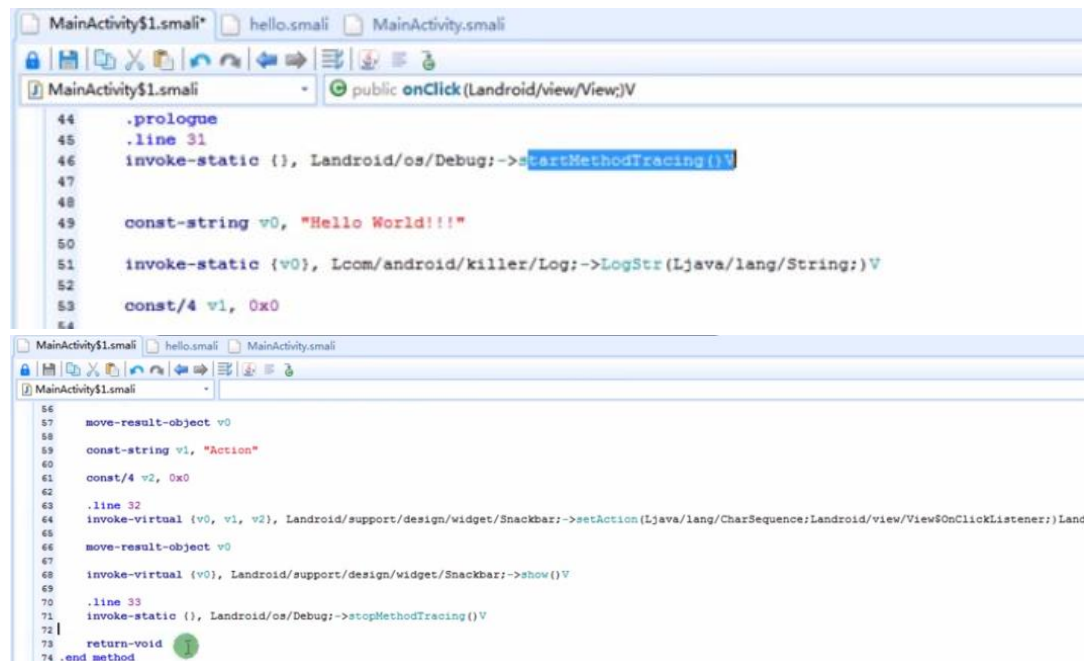
invoke-static {}, Landroid/os/Debug;-->stopMethodTracing()V #函数尾部

添加权限

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

保存的 trace 文件，可以 dump 出来用 monitor 打开

adb pull /storage/sdcard/dmtrace.trace 本地路径 #dump 指令



## 7. 字符串处理

新建字符串:

const-string v1, "%d"

字符串格式化:

const-string v1, "%d" #格式化描述符

const/4 v2, 0x1 #数组长度

new-array v2, v2, [Ljava/lang/Object; #创建 Object 数组

aput-object v3, v2, v4 #填充数组

...

invoke-static {v1, v2}, Ljava/lang/String;-->format(Ljava/lang/String;[Ljava/lang/Object;)

Ljava/lang/String; #格式化字符串

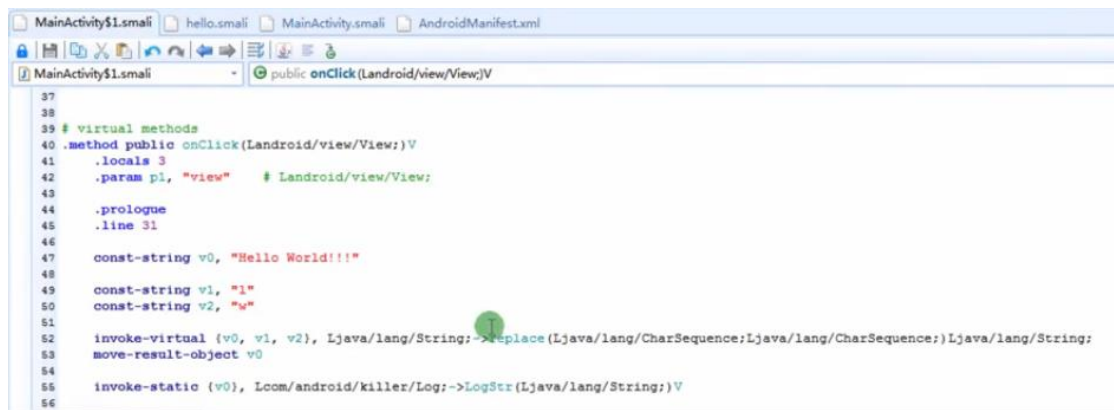
字符串替换:

invoke-virtual {v0, v1, v2},

Ljava/lang/String;-->replace(Ljava/lang/CharSequence;Ljava/lang/CharSequence;)Ljava/lang/String;

invoke-virtual {v0, v1, v2},

Ljava/lang/String;-->replaceAll(Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;



```
37
38
39 # virtual methods
40 .method public onClick(Landroid/view/View;)V
41 .locals 3
42 .param p1, "view" # Landroid/view/View;
43
44 .prologue
45 .line 31
46
47 const-string v0, "Hello World!!!"
48
49 const-string v1, "1"
50 const-string v2, "u"
51
52 invoke-virtual {v0, v1, v2}, Ljava/lang/String;-.>replace(Ljava/lang/CharSequence;Ljava/lang/CharSequence;)Ljava/lang/String;
53 move-result-object v0
54
55 invoke-static {v0}, Lcom/android/killer/Log;->LogStr(Ljava/lang/String;)V
56
```

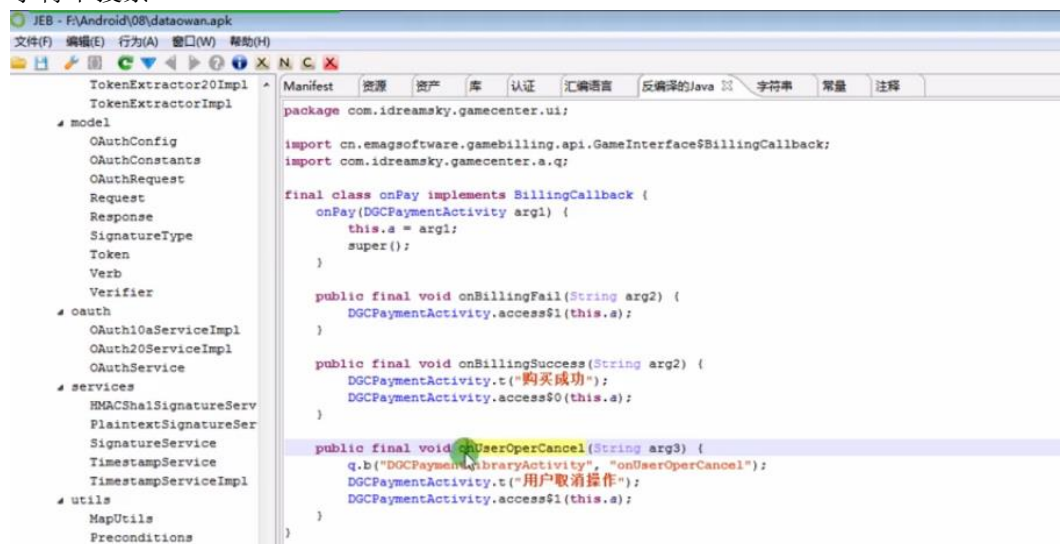
## 8. waitForDebugger

调试

invoke-static {}, Landroid/os/Debug;->waitForDebugger()V

## 9. Demo 实例

### 1) 字符串搜索



```
package com.idreamsky.gamecenter.ui;

import cn.emagsoftware.gamebilling.api.GameInterface$BillingCallback;
import com.idreamsky.gamecenter.a.q;

final class onPay implements BillingCallback {
    onPay(DGCPaymentActivity arg1) {
        this.a = arg1;
        super();
    }

    public final void onBillingFail(String arg2) {
        DGCPaymentActivity.access$1(this.a);
    }

    public final void onBillingSuccess(String arg2) {
        DGCPaymentActivity.t("购买成功");
        DGCPaymentActivity.access$0(this.a);
    }

    public final void onUserOperCancel(String arg3) {
        q.b("DGCPaymentLibraryActivity", "onUserOperCancel");
        DGCPaymentActivity.t("用户取消操作");
        DGCPaymentActivity.access$1(this.a);
    }
}
```

### 2) 插入 dump 堆栈的方法

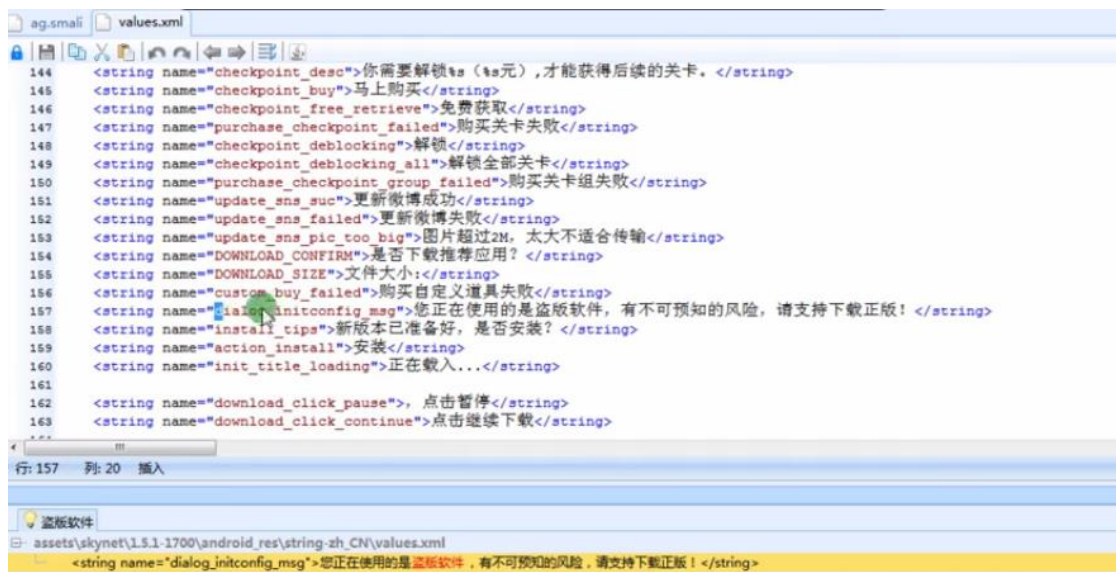


```
54 # invokes: Lcom/idreamsky/gamecenter/ui/DGCPaymentActivity;->notifyPaymentTransactionSuccess()V
55 invoke-static {v0}, Lcom/idreamsky/gamecenter/ui/DGCPaymentActivity;->access$0(Lcom/idreamsky/gamecenter/ui/DG
56
57 .line 667
58 return-void
59 .end method
60
61 .method public final onUserOperCancel(Ljava/lang/String;)V
62 .locals 2
63
64 .prologue
65 .line 654
66 invoke-static {}, Ljava/lang/Thread;->dumpStack()V
67
68 const-string v0, "DGCPaymentLibraryActivity"
69
70 const-string v1, "onUserOperCancel"
71
72 invoke-static {v0, v1}, Lcom/idreamsky/gamecenter/a/q;->b(Ljava/lang/String;Ljava/lang/CharSequence;)V
73
```

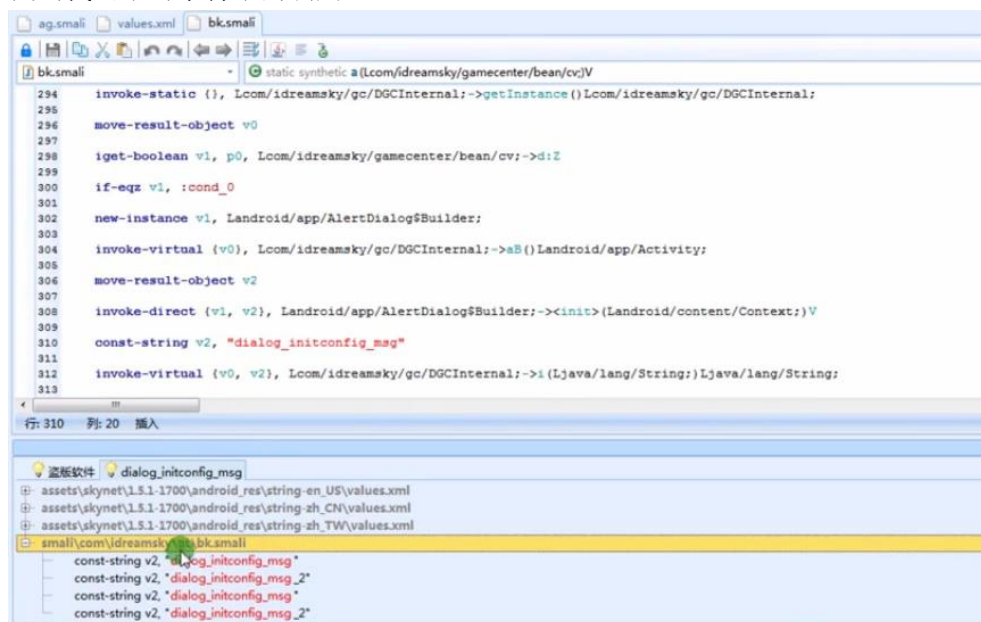
### 3) 对付签名验证

搜索目标字符串

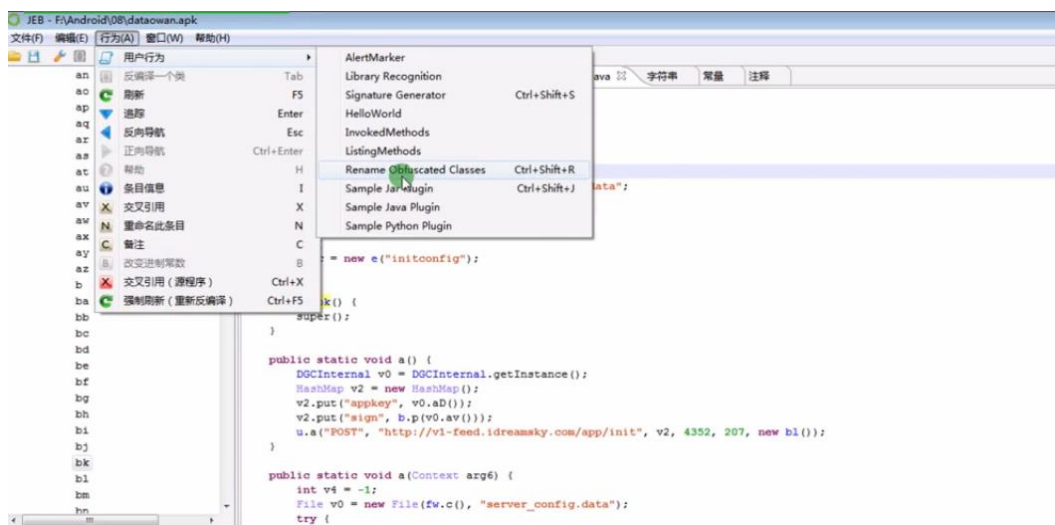




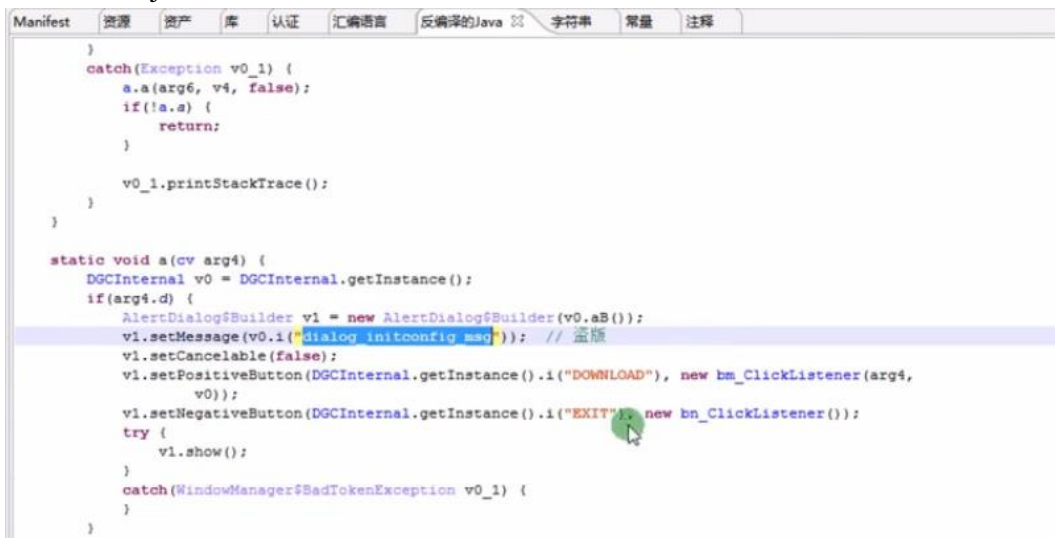
找到代码中对字符串的调用



执行 jeb 中重命名脚本应对混淆



找到对应的 java 代码

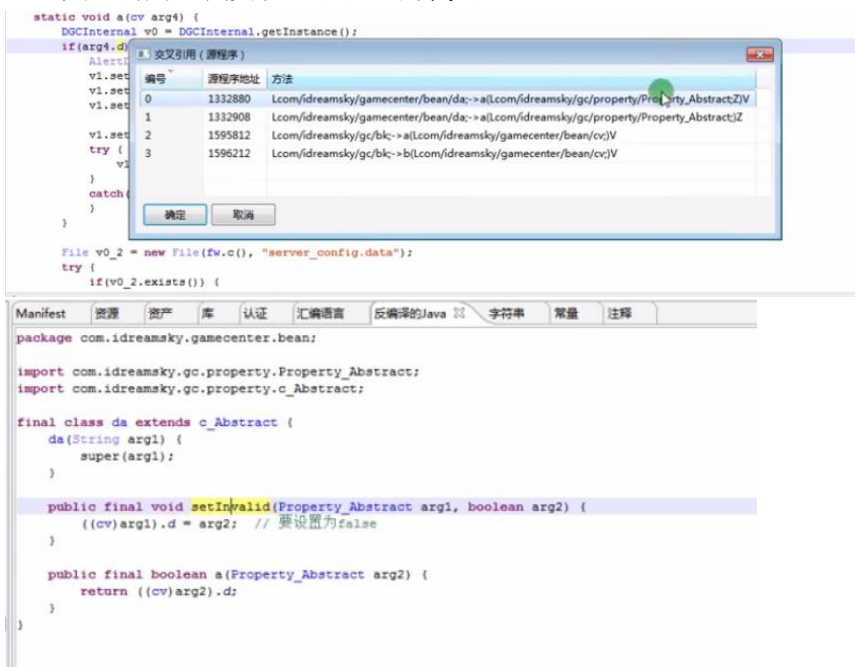


```
    }
    catch(Exception v0_1) {
        a.a(arg6, v4, false);
        if(!a.a) {
            return;
        }
        v0_1.printStackTrace();
    }
}

static void a(cv arg4) {
    DGCInternal v0 = DGCInternal.getInstance();
    if(arg4.d) {
        AlertDialog$Builder v1 = new AlertDialog$Builder(v0.aB());
        v1.setMessage(v0.i("dialog_initconfig_msg")); // 蓝版
        v1.setCancelable(false);
        v1.setPositiveButton(DGCInternal.getInstance().i("DOWNLOAD"), new bm_ClickListener(arg4, v0));
        v1.setNegativeButton(DGCInternal.getInstance().i("EXIT"), new bn_ClickListener());
        try {
            v1.show();
        }
        catch(WindowManager$BadTokenException v0_1) {
        }
    }
}
```

可以直接修改 DOWNLOAD 和 EXIT 相关的代码

通过交叉引用查找实现签名验证的代码



```
static void a(cv arg4) {
    DGCInternal v0 = DGCInternal.getInstance();
    if(arg4.d) {
        AlertDialog$Builder v1 = new AlertDialog$Builder(v0.aB());
        v1.setMessage(v0.i("dialog_initconfig_msg"));
        v1.setCancelable(false);
        v1.setPositiveButton(DGCInternal.getInstance().i("DOWNLOAD"), new bm_ClickListener(arg4, v0));
        v1.setNegativeButton(DGCInternal.getInstance().i("EXIT"), new bn_ClickListener());
        try {
            v1.show();
        }
        catch(WindowManager$BadTokenException v0_1) {
        }
    }
}

File v0_2 = new File(fw.c(), "server_config.data");
try {
    if(v0_2.exists()) {

```

编号	源程序地址	方法
0	1332880	Lcom/idreamsky/gamecenter/bean/da->a(Lcom/idreamsky/gc/property/Property_Abstract;Z)V
1	1332908	Lcom/idreamsky/gamecenter/bean/da->a(Lcom/idreamsky/gc/property/Property_Abstract;Z)V
2	1595812	Lcom/idreamsky/gc/bk->a(Lcom/idreamsky/gamecenter/bean/cv)V
3	1596212	Lcom/idreamsky/gc/bk->b(Lcom/idreamsky/gamecenter/bean/cv)V

```
    }
}

package com.idreamsky.gamecenter.bean;

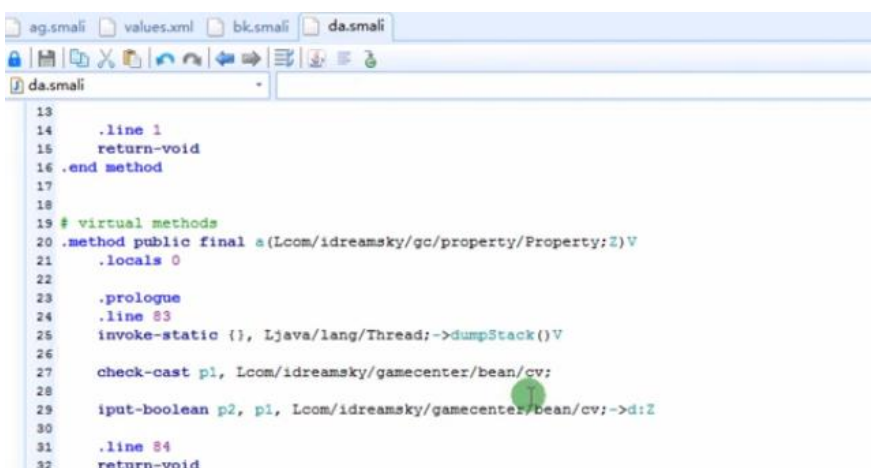
import com.idreamsky.gc.property.Property_Abstract;
import com.idreamsky.gc.property.c_Abstract;

final class da extends c_Abstract {
    da(String arg1) {
        super(arg1);
    }

    public final void setInvalid(Property_Abstract arg1, boolean arg2) {
        ((cv)arg1).d = arg2; // 要设置为false
    }

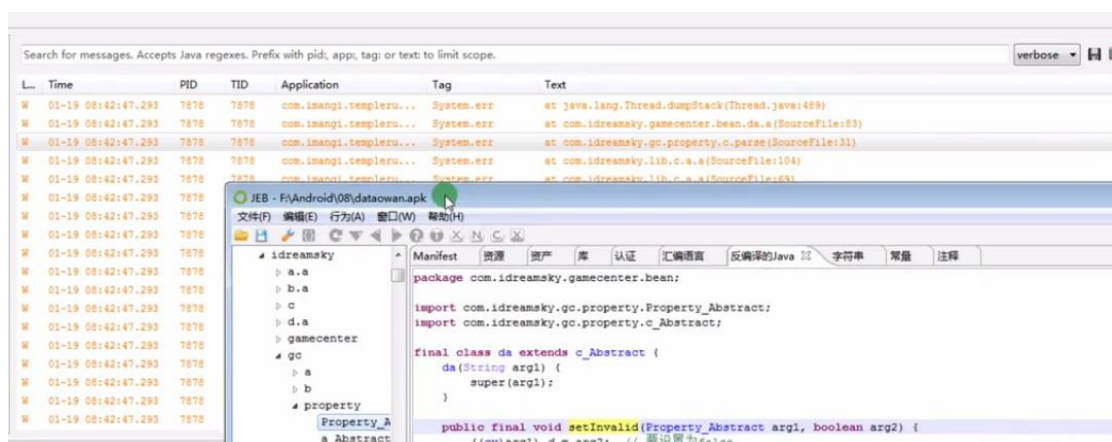
    public final boolean a(Property_Abstract arg2) {
        return ((cv)arg2).d;
    }
}
```

交叉引用无法使用，然后插入堆栈跟踪代码



```
13
14 .line 1
15 return-void
16 .end method
17
18
19 # virtual methods
20 .method public final a(Lcom/idreamsky/gc/property/Property;Z)V
21 .locals 0
22
23 .prologue
24 .line 83
25 invoke-static {}, Ljava/lang/Thread; ->dumpStack()V
26
27 check-cast p1, Lcom/idreamsky/gamecenter/bean/cv;
28
29 iput-boolean p2, p1, Lcom/idreamsky/gamecenter/bean/cv; ->d:Z
30
31 .line 84
32 return-void
```

根据 dump 出的堆栈信息分析函数调用



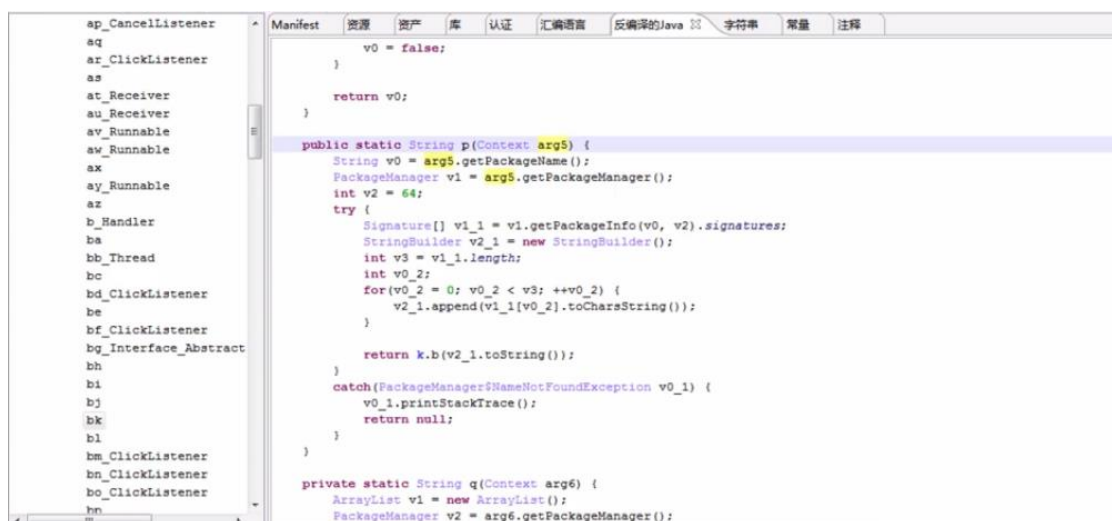
找到签名验证的方法

```
DD_inRead
bc
bd_ClickListener
be
bf_ClickListener
bg_Interface_Abstract
bh
bi
bj
bk

public static void a() {
    DGCInternal v0 = DGCInternal.getInstance();
    HashMap v2 = new HashMap();
    v2.put("appkey", v0.aD());
    v2.put("sign", b.b(v0.av()));
    u.a("POST", "http://v1-feed.idreamsky.com/app/init", v2, 4352, 207, new b1());
}

public static void a(Context arg6) {
    int v4 = -1;
}
```

分析方法 p0



对函数和变量进行重命名

```
as
at_Receiver
au_Receiver
av_Runnable
aw_Runnable
ax
ay_Runnable
az
b_Handler
ba
bb_Thread
bc
bd_ClickListener
be
bf_ClickListener
bg_Interface_Abstract
bh
bi
bj
bk

public static String getSignatures(Context ctx) {
    String packageName = ctx.getPackageName();
    PackageManager packageManager = ctx.getPackageManager();
    int v2 = 64;
    try {
        Signature[] sigList = packageManager.getPackageInfo(packageName, v2).signatures;
        StringBuilder v2_1 = new StringBuilder();
        int v3 = sigList.length;
        int v0_2;
        for(v0_2 = 0; v0_2 < v3; ++v0_2) {
            v2_1.append(sigList[v0_2].toCharsString());
        }
        return k.getMDS(v2_1.toString());
    } catch (PackageManager$NameNotFoundException v0_1) {
        v0_1.printStackTrace();
        return null;
    }
}

private static String q(Context arg6) {
}
```

该 app 是对 packageName 信息进行 MD5 运算后 post 到在线网站中进行验证



Charles 3.10.1 - Session 1 \*  
File Edit View Proxy Tools Window Help

Structure Sequence

- https://23.23.194.123
  - <unknown>
- http://74.125.204.102
- http://config.pinyin.sogou.com
- http://tj5.3lsoft.com
- http://in1.feed.uu.cc
- http://analytics.idreamsky.com
- http://v1-feed.idreamsky.com
- app
  - http://in1.secure.uu.cc
  - http://in1.service.uu.cc:80
  - http://c.uu.cc:90
  - http://123.125.80.78

Overview Request Response Summary Chart Notes

1 %7B%22appkey%22%3A%2275ed8c70047d3d140c7a%22%2C%22sign%22%3A%223876126041f65216720df7bbb14961a3%22%7D

%22%3A%223876126041f65216720df7bbb14961a3%22%7D+

将抓取的 MD5 值写进程序中

The screenshot shows the Dalvik Disassembler interface with the file `b.smali` open. The decompiled code for the `goto` method is displayed as follows:

```

3069     goto :goto_0
3070 .end method
3071
3072 .method public static p(Landroid/content/Context;)Ljava/lang/String;
3073     .locals 5
3074
3075     .prologue
3076     .line 1137
3077     const-string v0, "3876126041f65216720df7bbb14961a3"
3078     return-object v0
3079
3080     invoke-virtual {p0}, Landroid/content/Context;-.>getPackageName()Ljava/lang/
3081
3082     move-result-object v0

```

## 最后 app 运行无签名提示