

Android 应用程序 通用自动脱壳方法研究

杨文博

- GoSSIP 软件安全小组
 - 微博 @GoSSIP_SJTU
 - www.securitygossip.com
- 上海交通大学网络信息安全协会 (0ops)

>> 简介 Profile



杨文博

上海交通大学计算机系在读博士

Android 加壳保护

Android 加壳保护

移动应用保护服务

防逆向保护(Anti-RE)

以加密代码的方式阻止反编译，从而防止被窃取代码和创意

防篡改保护(Anti-tamper)

通过对App的完整性保护，防止App被篡改或盗版

反调试保护(Anti-debug)

阻止应用运行中被动态注入，防止被外挂、木马偷窃账号密码，修改交易金额等

存储数据加密保护(Storage Encryption)

更底层、跨文件格式的数据加密，防止应用数据被窃取

环境监测和保护(Environmental Monitoring)

云监测设备环境，防止盗版应用、恶意应用的钓鱼攻击

Android 加壳保护

Android 加壳保护



Android 加壳保护

Android 加壳保护

梆梆安全
BANGCLE

爱加密

Android 加壳保护

梆梆安全
BANGCLE

爱加密



应用加固

反静态分析，反动态调试，
反内存窃取，反恶意篡改...



阿里聚安全

应用更安全，用户更安心！



360加固保
jiagu.360.cn

Android 加壳保护

可保护

- 程序逻辑
 - 算法、协议
- 完整性
 - 盗版
 - 外挂

不可保护

- 安全问题
 - 数据存储传输
- 程序漏洞
 - 权限泄露
 - 不安全的**API**

为什么要脱壳

为什么要脱壳

2014年，更多恶意软件采用加壳技术躲避安全软件的查杀。2013年到2014年仅一年时间，加壳恶意软件数量就增长了约18倍（如图8所示）。

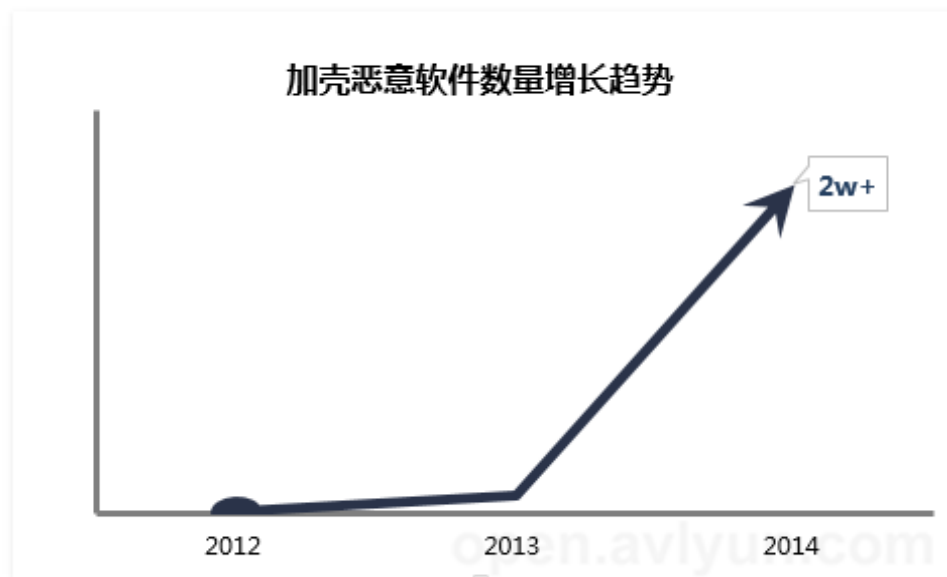
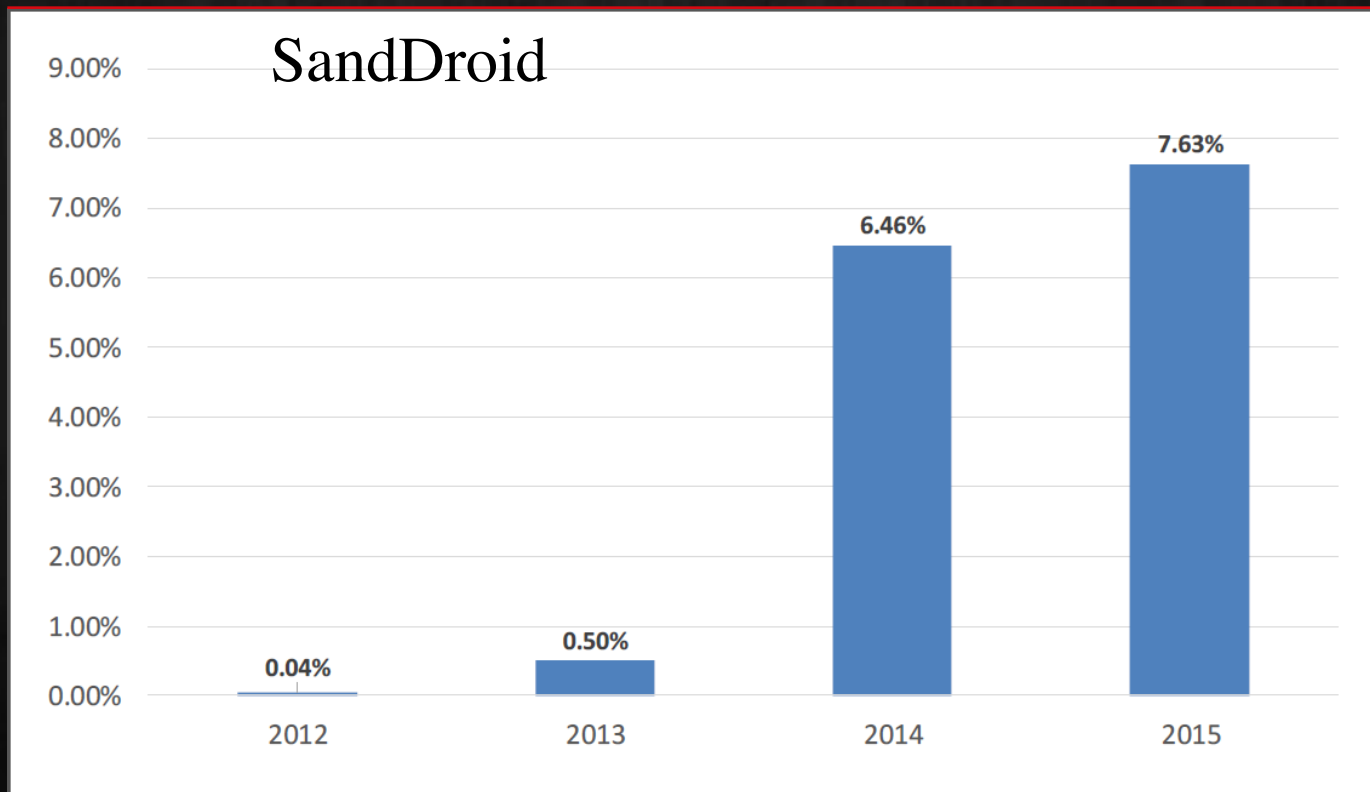


图8 加壳恶意软件数量增长趋势

AVL Team

为什么要脱壳

为什么要脱壳



为什么要脱壳

为什么要脱壳



脱壳的影响

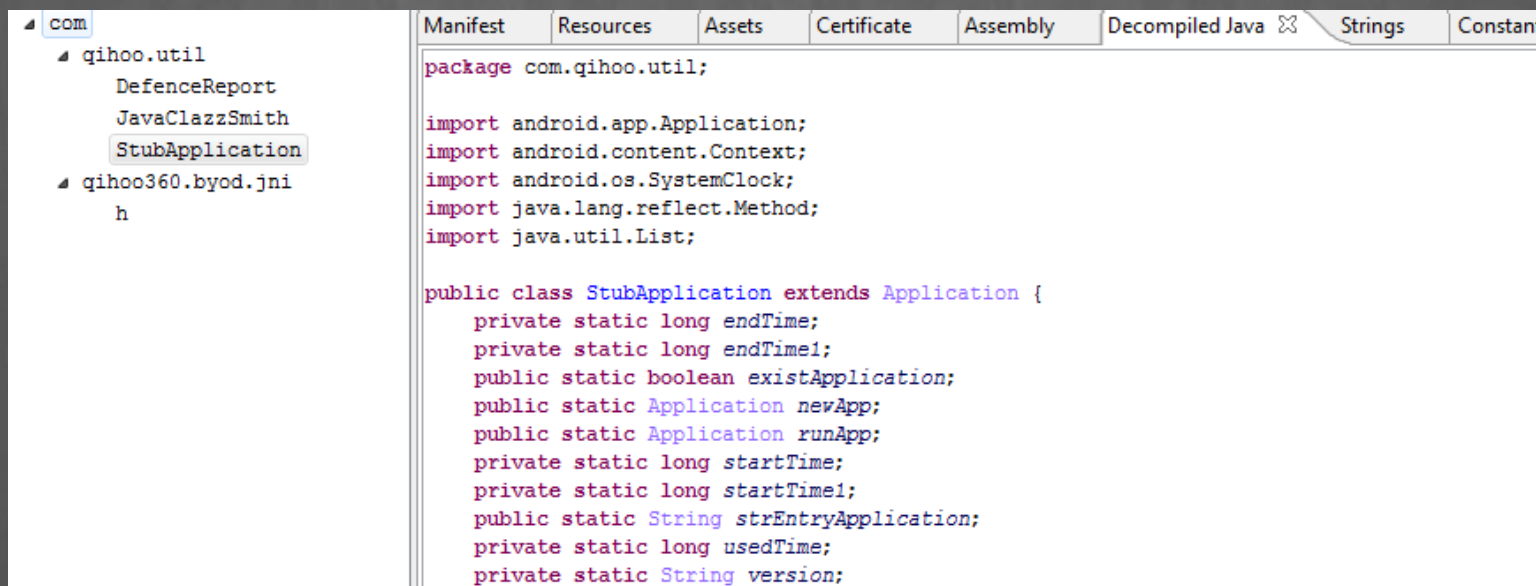
- 程序真实逻辑暴露
- 降低分析门槛
- 篡改APP
- 静态代码审查
- 更容易挖掘漏洞

加固程序特点

· Manifest保留

· 增加入口点类

· Native执行



The screenshot shows a decompiler interface with a package explorer on the left and a code editor on the right. The package explorer shows the package structure: com, qihoo.util (containing DefenceReport, JavaClazzSmith, and StubApplication), and qihoo360.byod.jni (containing h). The code editor displays the decompiled Java code for StubApplication, which is a public class extending Application. The code includes various static fields and methods for application management, such as startTime, endTime, and version.

```
package com.qihoo.util;

import android.app.Application;
import android.content.Context;
import android.os.SystemClock;
import java.lang.reflect.Method;
import java.util.List;

public class StubApplication extends Application {
    private static long endTime;
    private static long endTime1;
    public static boolean existApplication;
    public static Application newApp;
    public static Application runApp;
    private static long startTime;
    private static long startTime1;
    public static String strEntryApplication;
    private static long usedTime;
    private static String version;
```

加固程序特点

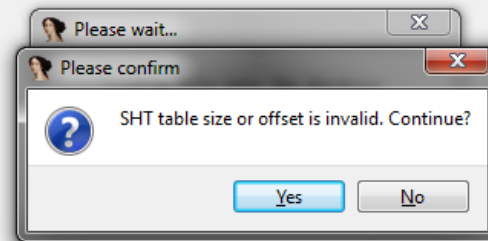
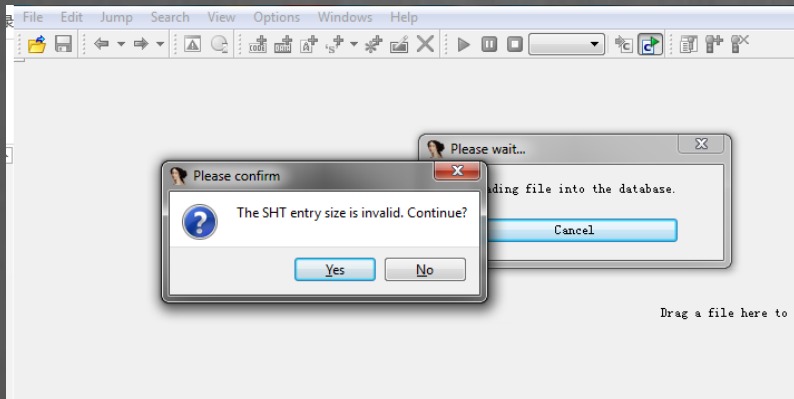
- ARM ELF 头部破坏

- .init_array 段花指令

加固程序特点

- ARM ELF 头部破坏

- .init_array 段花指令



加固程序特点

- ARM ELF 头部破坏

- .init_array 段花指令

加固程序特点

· ARM ELF 头部破坏

· .init_array 段花指令

```
ELF Header:
Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
Class:                               ELF32
Data:                               2's complement, little endian
Version:                             1 (current)
OS/ABI:                             UNIX - System V
ABI Version:                         0
Type:                               DYN (Shared object file)
Machine:                             ARM
Version:                             0x1
Entry point address:                 0x0
Start of program headers:            52 (bytes into file)
Start of section headers:            141416 (bytes into file)
Flags:                               0x50000000, Version5 EABI
Size of this header:                  52 (bytes)
Size of program headers:              32 (bytes)
Number of program headers:            7
Size of section headers:              108 (bytes)
Number of section headers:            102
Section header string table index:    120 <corrupt: out of range>
readelf: Error: Unable to read in 0x2008 bytes of section headers
```

加固程序特点

- ARM ELF 头部破坏

- .init_array 段花指令

加固程序特点

· ARM ELF 头部破坏

· .init_array 段花指令

```
STMFD    SP!, {R0}
ADRL     R0, sub_3484
SUB      R0, R0, #4
BX       R0 ; loc_3480
...
loc_3480:
LDMFD    SP!, {R0}
STMFD    SP!, {R3-R5, LR}  #真正有用的指令
                        又是一个模式
-----
STMFD    SP!, {R0}
ADRL     R0, loc_3304
SUB      R0, R0, #4
BX       R0 ; loc_3300
...
loc_3300:
LDMFD    SP!, {R0}
MOV      R0, #1            #真正有用的指令
                        又是一个模式
-----
STMFD    SP!, {R0}
...
```


加固程序特点

- 解密 JNI_OnLoad 函数

- 解密出另一个ELF文件

- ELF中解压 .text 段

- .text段中提取key再解密.rotext

- 真正的壳程序

加固程序特点

· 解密 JNI

· ELF中解

· .text段中

· 真正的壳

```
1 int __fastcall processDvmLoader(int a1)
2 {
3     int v1; // r5@1
4     int v2; // r6@1
5     int v3; // r4@2
6     int v4; // r0@2
7     int v5; // r2@2
8     DexUtil *u6; // ST00_4@2
9     char *u7; // r0@3
10    char *u8; // r0@3
11    int v9; // r0@4
12    int v10; // r0@5
13    int v11; // r0@9
14    int v12; // r0@9
15    int v13; // r0@11
16    size_t v14; // r4@13
17    void *v15; // r6@14
18    const char *src; // [sp+0h] [bp-30h]@1
19    int v18; // [sp+Ch] [bp-24h]@1
20    int v19; // [sp+10h] [bp-20h]@1
21    DexUtil *u20; // [sp+14h] [bp-1Ch]@1
22
23    v1 = a1;
24    v2 = 0;
25    v18 = 0;
26    v19 = 0;
27    v20 = 0;
28    if ( !findModuleAddr(
29        "apk@Classes.dex",
30        (bool (__cdecl *) (const unsigned __int8 *, unsigned int, void *))sub_56C4,
31        &v18,
32        packageName ) )
33    {
34        v3 = v18;
35        v4 = operator new(2h);
36        v5 = *(DWORD *) (v3 + 32);
37        v6 = (DexUtil *)v4;
38        DexUtil::DexUtil(v4);
39        v20 = v6;
40        if ( DexUtil::loadPadding(v6) )
41        {
42            src = (const char *) ( * ( (DWORD *) v20 + 2 ) + 8 );
43            v7 = strchr( src, 58 );
44            v8 = v7;
45            if ( v7 < src )
46            {
47                v9 = 0;
48                goto LABEL_6;
49            }
50            v10 = atoi(v7 + 1);
51            resSum = v10;
52            v9 = nativeSigCheck(v1, packageName, v10);
```

一个ELF文件

加固程序特点

- 取原 DEX 中的padding数据

- 取一些解密解压参数

- 解密解压padding数据

- 得到 DEX 文件

- 反调试，反分析

加固程序特点

- 隐藏DEX

- 静态逆向难

- 变化快

- 反分析手段
 - 反调试
 - 反内存dump
 - 反静态工具

制作通用脱壳机

通用自动化脱壳

- Dalvik 源码插桩
 - Portable 解释器
 - 绕过反调试
 - 运行时数据
 - 任意脱壳点
 - 真机部署

通用自动化脱壳

- 内存中的 Dalvik 数据结构

通用自动化脱壳

·内存中的 Dalvik 数据结构

```
struct Method {  
    /* the class we are a part of */  
    ClassObject*   clazz;  
  
    /* access flags; low 16 bits are defined by spec (could be u2?) */  
    u4              accessFlags;  
  
    /*  
     * For concrete virtual methods, this is the offset of the method  
     * in "vtable".  
     *  
     * For abstract methods in an interface class, this is the offset  
     * of the method in "iftable[n]->methodIndexArray".  
     */  
    u2              methodIndex;  
  
    /*  
     * Method bounds; not needed for an abstract method.  
     *  
     * For a native method, we compute the size of the argument list, and  
     * set "insSize" and "registerSize" equal to it.  
     */  
    u2              registersSize; /* ins + locals */  
    u2              outsSize;  
    u2              insSize;  
  
    /* method name e.g. "<init>" or "eatLunch" */  
};
```


通用自动化脱壳

- 内存中的 Dalvik 数据结构

通用自动化脱壳

·内存中的 Dalvik 数据结构

```
struct ClassObject : Object {
    /* leave space for instance data; we could access fields directly if we
    | freeze the definition of java/lang/Class */
    u4          instanceData[CLASS_FIELD_SLOTS];

    /* UTF-8 descriptor for the class; from constant pool, or on heap
    | if generated ("[C") */
    const char* descriptor;
    char*       descriptorAlloc;

    /* access flags; low 16 bits are defined by VM spec */
    u4          accessFlags;

    /* VM-unique class serial number, nonzero, set very early */
    u4          serialNumber;

    /* DexFile from which we came; needed to resolve constant pool entries */
    /* (will be NULL for VM-generated, e.g. arrays and primitive classes) */
    DvmDex*     pDvmDex;

    /* state of class initialization */
    ClassStatus status;

    /* if class verify fails, we must return same error on subsequent tries */
    ClassObject* verifyErrorClass;

    /* threadId, used to check for recursive <clinit> invocation */
```

通用自动化脱壳

- 内存中的 Dalvik 数据结构

通用自动化脱壳

·内存中的 Dalvik 数据结构

```
struct DvmDex {  
    /* pointer to the DexFile we're associated with */  
    DexFile*          pDexFile;  
  
    /* clone of pDexFile->pHeader (it's used frequently enough) */  
    const DexHeader*   pHeader;  
  
    /* interned strings; parallel to "stringIds" */  
    struct StringObject** pResStrings;  
  
    /* resolved classes; parallel to "typeIds" */  
    struct ClassObject** pResClasses;  
  
    /* resolved methods; parallel to "methodIds" */  
    struct Method**      pResMethods;  
  
    /* resolved instance fields; parallel to "fieldIds" */  
    /* (this holds both InstField and StaticField) */  
    struct Field**       pResFields;  
  
    /* interface method lookup cache */  
    struct AtomicCache*  pInterfaceCache;  
  
    /* shared memory region with file contents */  
    bool                 isMappedReadOnly;  
    MemMapping           memMap;  
  
    /* lock ensuring mutual exclusion during updates */  
    pthread_mutex_t      modLock;  
};
```

通用自动化脱壳

- 内存中的 Dalvik 数据结构

通用自动化脱壳

·内存中的 Dalvik 数据结构

```
/*
 * Structure representing a DEX file.
 *
 * Code should regard DexFile as opaque, using the API calls provided here
 * to access specific structures.
 */
struct DexFile {
    /* directly-mapped "opt" header */
    const DexOptHeader* pOptHeader;

    /* pointers to directly-mapped structs and arrays in base DEX */
    const DexHeader*    pHeader;
    const DexStringId*  pStringIds;
    const DexTypeId*    pTypeIds;
    const DexFieldId*   pFieldIds;
    const DexMethodId*  pMethodIds;
    const DexProtoId*   pProtoIds;
    const DexClassDef*  pClassDefs;
    const DexLink*      pLinkData;

    /*
     * These are mapped out of the "auxillary" section, and may not be
     * included in the file.
     */
    const DexClassLookup* pClassLookup;
    const void*           pRegisterMapPool;    // RegisterMapClassPool

    /* points to start of DEX file data */
    const u1*             baseAddr;

    /* track memory overhead for auxillary structures */
    int                   overhead;

    /* additional app-specific data structures associated with the DEX */
    //void*                auxData;
};
```

一个Naïve的实现

- 读取 DexFile 结构体
 - 基于源码: C/C++ 实现
 - 以DexFile为输入
- 他山之石
 - dalvik/dexdump/
DexDump.cpp

```

/*
 * Dump the requested sections of the file.
 */
void processDexFile(const char* fileName, DexFile* pDexFile)
{
    char* package = NULL;
    int i;

    if (gOptions.verbose) {
        printf("Opened '%s', DEX version '%.3s'\n", fileName,
            pDexFile->pHeader->magic + 4);
    }

    if (gOptions.dumpRegisterMaps) {
        dumpRegisterMaps(pDexFile);
        return;
    }

    if (gOptions.showFileHeaders) {
        dumpFileHeader(pDexFile);
        dumpOptDirectory(pDexFile);
    }

    if (gOptions.outputFormat == OUTPUT_XML)
        printf("<api>\n");

    for (i = 0; i < (int) pDexFile->pHeader->classDefsSize; i++) {
        if (gOptions.showSectionHeaders)
            dumpClassDef(pDexFile, i);

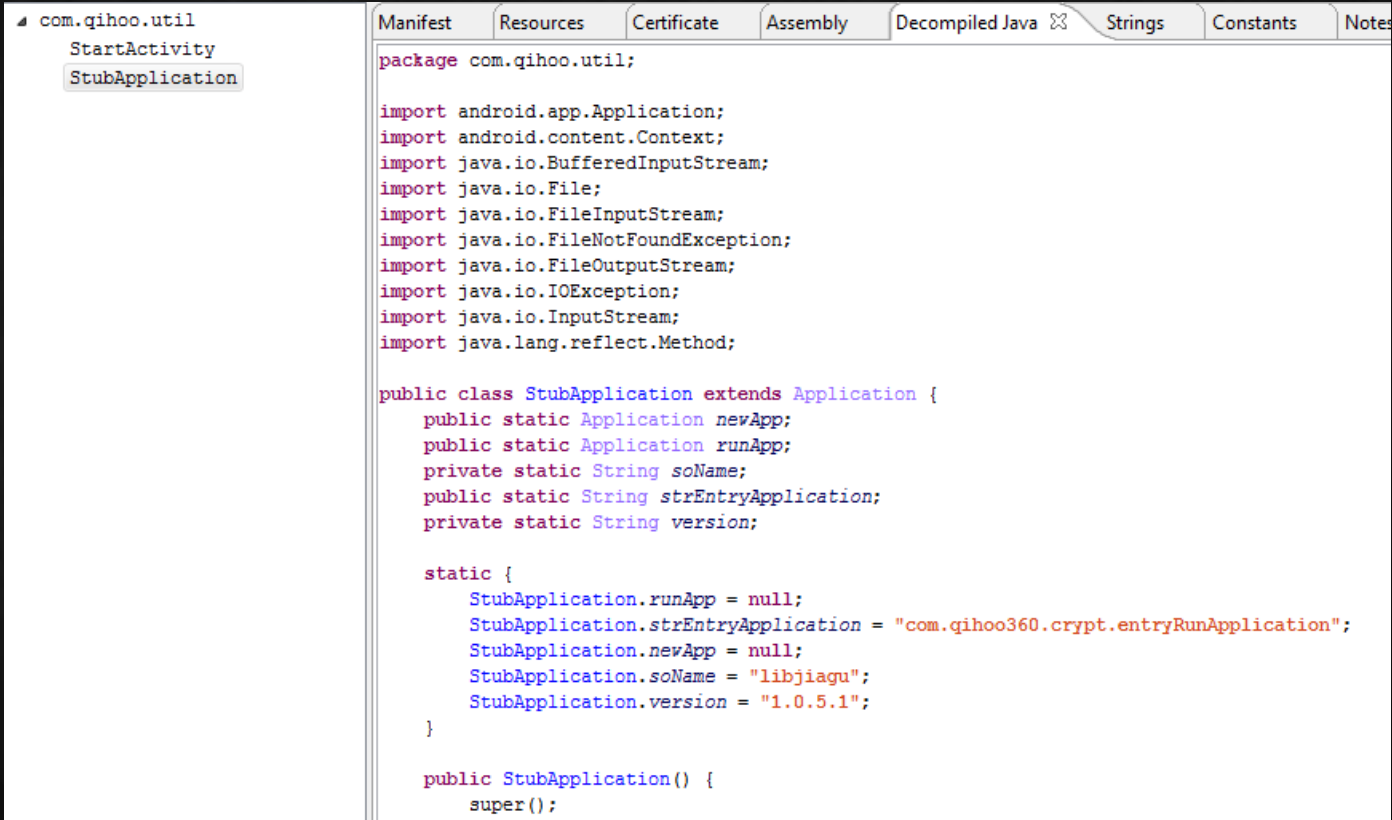
        dumpClass(pDexFile, i, &package);
    }
}

```


效果

- 脱壳点
 - MainActivity.onCreate() in Manifest
- 几乎所有的壳
- 输出
 - 文本
 - Dalvik bytecode

效果



The screenshot displays the Android Studio IDE with the 'Decompiled Java' tab selected. The package explorer on the left shows the project structure: `com.qihoo.util` containing `StartActivity` and `StubApplication`. The main editor area shows the decompiled Java code for `StubApplication`.

```
package com.qihoo.util;

import android.app.Application;
import android.content.Context;
import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.lang.reflect.Method;

public class StubApplication extends Application {
    public static Application newApp;
    public static Application runApp;
    private static String soName;
    public static String strEntryApplication;
    private static String version;

    static {
        StubApplication.runApp = null;
        StubApplication.strEntryApplication = "com.qihoo360.crypt.entryRunApplication";
        StubApplication.newApp = null;
        StubApplication.soName = "libjiagu";
        StubApplication.version = "1.0.5.1";
    }

    public StubApplication() {
        super();
    }
}
```

效果

- 脱壳点
 - MainActivity.onCreate() in Manifest
- 几乎所有的壳
- 输出
 - 文本
 - Dalvik bytecode

效果



效果

- 脱壳点
 - MainActivity.onCreate() in Manifest
- 几乎所有的壳
- 输出
 - 文本
 - Dalvik bytecode

效

```

Virtual methods -
#0 : (in Lcom/example/simple/MyActivity;)
  name : 'onCreate'
  type : '(Landroid/os/Bundle;)V'
  access : 0x0004 (PROTECTED)
  code -
  codeOff : 0x53cc8
  registers : 12
  ins : 2
  outs : 6
  insns size : 78 16-bit code units

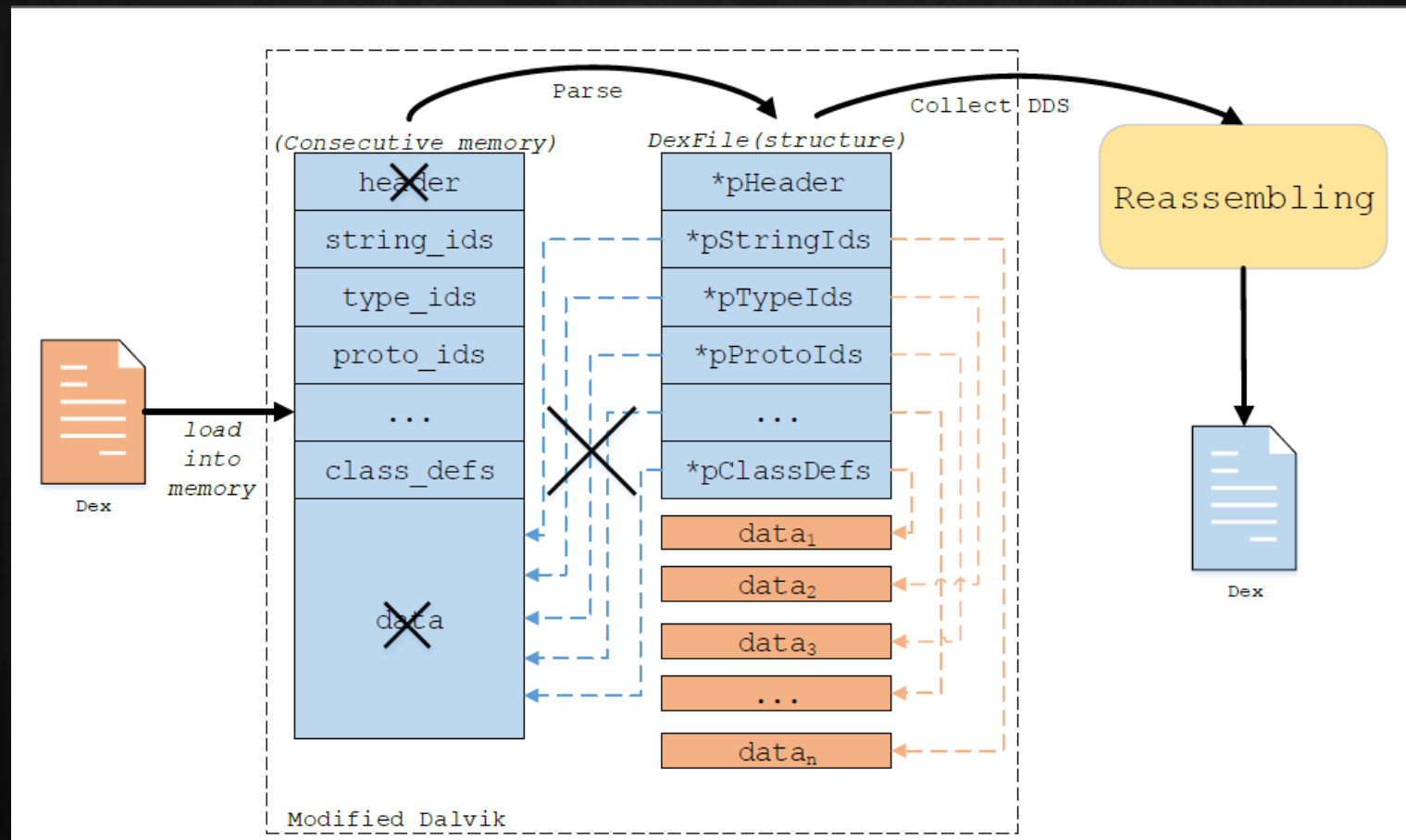
053cc8: |[053cc8] com.example.simple.MyActivity.onCreate:(Landroid/os/Bundle;)V
053cd8: 1202 |0000: const/4 v2, #int 0 // #0
053cda: 6f20 2100 ba00 |0001: invoke-super {v10, v11}, Landroid/app/Activity;.onCreate:(Landroid/os/Bundle;)V // method@0021
053ce0: 1503 037f |0004: const/high16 v3, #int 2130903040 // #7f03
053ce4: 6e20 cf17 3a00 |0006: invoke-virtual {v10, v3}, Lcom/example/simple/MyActivity;.setContentView:(I)V // method@17cf
053cea: 1a03 510a |0009: const-string v3, "TTT" // string@0a51
053cee: 1a04 e608 |000b: const-string v4, "MyActivity.onCreate" // string@08e6
053cf2: 7120 4415 4300 |000d: invoke-static {v3, v4}, Landroid/util/Log;.i:(Ljava/lang/String;Ljava/lang/String;)I // method@1544
053cf8: 2206 2c00 |0010: new-instance v6, Landroid/content/Intent; // type@002c
053cfc: 1c03 6303 |0012: const-class v3, Lcom/example/simple/MyBroadcastReceiver; // type@0363
053d00: 7030 cc00 a603 |0014: invoke-direct {v6, v10, v3}, Landroid/content/Intent;.<init>:(Landroid/content/Context;Ljava/lang/Class;)
V // method@00cc
053d06: 6e20 ce17 6a00 |0017: invoke-virtual {v10, v6}, Lcom/example/simple/MyActivity;.sendBroadcast:(Landroid/content/Intent;)V //
method@17ce
053d0c: 2207 2c00 |001a: new-instance v7, Landroid/content/Intent; // type@002c
053d10: 1c03 6503 |001c: const-class v3, Lcom/example/simple/MyService; // type@0365
053d14: 7030 cc00 a703 |001e: invoke-direct {v7, v10, v3}, Landroid/content/Intent;.<init>:(Landroid/content/Context;Ljava/lang/Class;)
V // method@00cc
053d1a: 6e20 d017 7a00 |0021: invoke-virtual {v10, v7}, Lcom/example/simple/MyActivity;.startService:(Landroid/content/Intent;)
Landroid/content/ComponentName; // method@17d0
053d20: 6e10 c917 0a00 |0024: invoke-virtual {v10}, Lcom/example/simple/MyActivity;.getContentResolver:()
Landroid/content/ContentResolver; // method@17c9
053d26: 0c00 |0027: move-result-object v0
053d28: 1a03 900d |0028: const-string v3, "content://com.example.simple.MyContentProvider/" // string@0d90
053d2c: 7110 8f01 0300 |002a: invoke-static {v3}, Landroid/net/Uri;.parse:(Ljava/lang/String;)Landroid/net/Uri; // method@018f
053d32: 0c01 |002d: move-result-object v1
053d34: 0722 |002e: move-object v3, v2

```

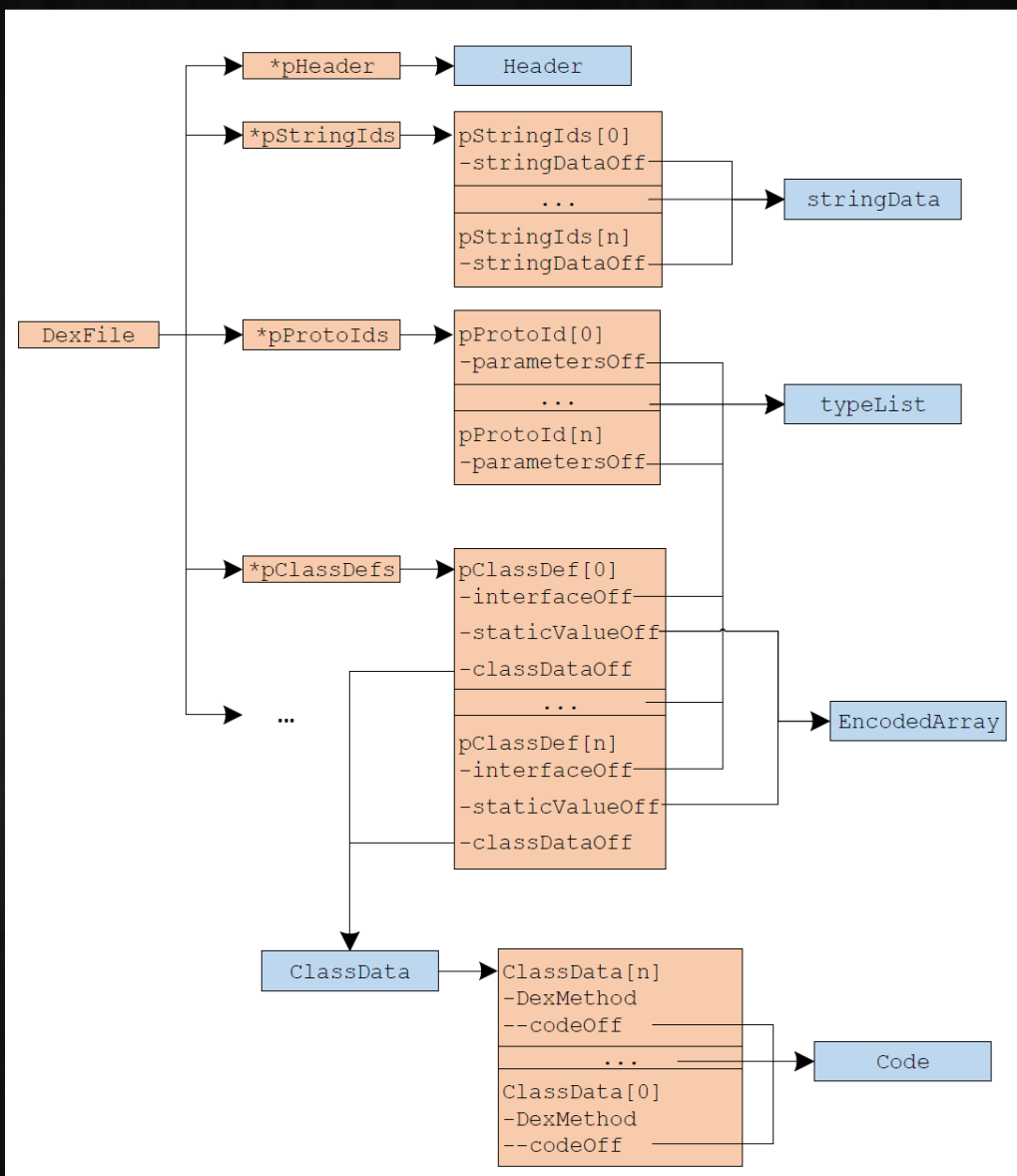
More sophisticated 实现

- DEX 文件重组
 - 获取内存中的Dalvik数据结构
 - DEX 文件重写

More sophisticated 实现



获取Dalvik 数据结构



DEX 文件重写

- 排列顺序
 - dalvik/libdex/DexFile.h
- 调整偏移
 - stringDataOff, parametersOff, interfacesOff, classDataOff, codeOff, ...
- 重新计算
 - DexHeader, MapList
- 差异
 - Uleb128, 4 字节对齐, field/method_idx_diff

实验

实验机

- Nexus 4: Android 4.4.2
- Galaxy Nexus: Android 4.3

10种壳

- /system/lib/libdvm.so
- 梆梆，爱加密，360，
百度，阿里，腾讯
APKProtect，
网秦，LIAPP，
DexProtector

实验

实验机

- Nexus 4: Android 4.4.2
- Galaxy Nexus: Android 4.3

10种壳

- /system/lib/libdvm.so

- 梆梆，爱加密，360，
百度，阿里，腾讯

APKProtect，
网秦，LIAPP，
DexProtector

ALL-KILL!!!

● 一些发现

一些发现

DEX file header:

magic	: ';..\0?..\0'
checksum	: 000518b2
signature	: 7d16...0500
file_size	: 333798
header_size	: 334841
link_size	: 0
link_off	: 0 (0x000000)
string_ids_size	: 6991
string_ids_off	: 112 (0x000070)
type_ids_size	: 1020
type_ids_off	: 28076 (0x006dac)
field_ids_size	: 1662
field_ids_off	: 47648 (0x00ba20)
method_ids_size	: 6368
method_ids_off	: 60944 (0x00ee10)
class_defs_size	: 643
class_defs_off	: 111888 (0x01b510)
data_size	: 686112
data_off	: 133296 (0x0208b0)

● 一些发现

一些发现

DEX file header:

```
magic          : '\0\0\0\0\0\0\0\0'
checksum       : 00000000
signature      : 0000...0000
file_size      : 667868
header_size    : 112
link_size      : 0
link_off       : 0 (0x000000)
string_ids_size : 7000
string_ids_off  : 0 (0x000000)
type_ids_size  : 1020
type_ids_off   : 0 (0x000000)
field_ids_size : 1664
field_ids_off  : 0 (0x000000)
method_ids_size : 6378
method_ids_off : 0 (0x000000)
class_defs_size : 645
class_defs_off  : 0 (0x000000)
data_size      : 533396
data_off       : 132672 (0x020640)
```


● 一些发现

一些发现

```
#1      : (in Lcom/example/simple/MyActivity;)
name    : 'onCreateOptionsMenu'
type    : '(Landroid/view/Menu;)Z'
access  : 0x0001 (PUBLIC)
code    : -
codeOff : 0xffdef5d4
registers : 5
ins     : 2
outs    : 3
insns size : 18 16-bit code units

ffdef5d4:                                [[ffdef5d4] com.example.simple.MyActivity.onCreateOptionsMenu:(Landroid/view/Menu;)Z
ffdef5e4: 7100 1319 0000                |0000: invoke-static {}, Lpnf/this/object/does/not/Exist;.a:()Z // method@1913
ffdef5ea: 0a02                        |0003: move-result v2
ffdef5ec: 7110 1419 0200                |0004: invoke-static {v2}, Lpnf/this/object/does/not/Exist;.b:(I)V // method@1914
ffdef5f2: 6e10 0018 0300                |0007: invoke-virtual {v3}, Lcom/example/simple/MyActivity;.getMenuInflater:()
Landroid/view/MenuInflater; // method@1800
ffdef5f8: 0c00                        |000a: move-result-object v0
ffdef5fa: 1501 077f                    |000b: const/high16 v1, #int 2131165184 // #7f07
ffdef5fe: 6e30 ae15 1004                |000d: invoke-virtual {v0, v1, v4}, Landroid/view/MenuInflater;.inflate:(ILandroid/view/Menu;
)V // method@15ae
ffdef604: 1210                        |0010: const/4 v0, #int 1 // #1
ffdef606: 0f00                        |0011: return v0

virtual methods -
#0      : (in Lpnf/this/object/does/not/Exist;)
name    : 'testdex2jarcrash'
type    : '(Ljava/lang/String;Ljava/lang/String;)V'
access  : 0x0001 (PUBLIC)
code    : -
codeOff : 0xffdef92c
registers : 3
ins     : 3
outs    : 0
insns size : 1 16-bit code units

ffdef92c:                                [[ffdef92c] pnf.this.object.does.not.Exist.testdex2jarcrash:(
Ljava/lang/String;Ljava/lang/String;)V
ffdef93c: 0e00                        |0000: return-void

catches : (none)
positions :
locals :
    0x0000 - 0x0001 reg=0 this Lpnf/this/object/does/not/Exist;
    0x0000 - 0x0001 reg=1 test1 Ljava/lang/String;
    0x0000 - 0x0001 reg=2 test2 Ljava/lang/String;

source_file_idx : -1 (unknown)
```

● 一些发现

一些发现

```
Direct methods  -
#0              : (in Lcom/example/simple/MyApplication;)
  name          : '<init>'
  type          : '()V'
  access        : 0x10001 (PUBLIC CONSTRUCTOR)
  code          : -
  codeOff       : 0xffdef608
  registers     : 1
  ins           : 1
  outs          : 1
  insns size    : 4 16-bit code units
ffdef608:        |[[ffdef608] com.example.simple.MyApplication.<init>:()V
ffdef618: 7010 3e00 0000 |0000: invoke-direct {v0}, Landroid/app/Application;.<init>:()V // method@003e
ffdef61e: 0e00        |0003: return-void
  catches       : (none)
  positions     :
  locals       :
```

● 一些发现

一些发现

```
public class MyActivity extends Activity {
    static {
        System.loadLibrary("simple");
    }

    public MyActivity() {
        super();
    }

    private native int getInt() {
    }

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this setContentView(2130903040);
        Log.i("TTT", "MyActivity.onCreate");
        this.sendBroadcast(new Intent(((Context) this), MyBroadcastReceiver.class));
        this.startService(new Intent(((Context) this), MyService.class));
        this.getContentResolver().query(Uri.parse("content://com.example.simple.MyContentProvider/"),
            null, null, null, null);
        Log.i("TTT", "getInt = " + this.getInt());
    }

    public boolean onCreateOptionsMenu(Menu menu) {
        this.getMenuInflater().inflate(2131165184, menu);
        return 1;
    }
}
```

● 一些发现

一些发现

```
public class MyActivity extends Activity {
    static {
        System.loadLibrary("simple");
    }

    public MyActivity() {
        super();
    }

    private native int getInt() {
    }

    protected void onCreate(Bundle arg2) {
        A.d("Lcom/example/simple/MyActivity;->onCreate001(Landroid/os/Bundle;)V");
        this.onCreate001(arg2);
        A.e("Lcom/example/simple/MyActivity;->onCreate001(Landroid/os/Bundle;)V");
    }

    private void onCreate001(Bundle savedInstanceState) {
    }

    public boolean onCreateOptionsMenu(Menu menu) {
        this.getMenuInflater().inflate(2131165184, menu);
        return 1;
    }
}
```


● 一些发现

一些发现

```
public class MyActivity extends Activity {
    static {
        System.loadLibrary("simple");
    }

    public MyActivity() {
        super();
    }

    private native int getInt() {
    }

    protected void onCreate(Bundle arg2) {
        A.d("Lcom/example/simple/MyActivity;->onCreate001(Landroid/os/Bundle;)V");
        this.onCreate001(arg2);
        A.e("Lcom/example/simple/MyActivity;->onCreate001(Landroid/os/Bundle;)V");
    }

    private void onCreate001(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.setContentView(2130903040);
        Log.i("TTT", "MyActivity.onCreate");
        this.sendBroadcast(new Intent(((Context)this), MyBroadcastReceiver.class));
        this.startService(new Intent(((Context)this), MyService.class));
        this.getContentResolver().query(Uri.parse("content://com.example.simple.MyContentProvider/"),
            null, null, null);
        Log.i("TTT", "getInt = " + this.getInt());
    }

    public boolean onCreateOptionsMenu(Menu menu) {
        this.getMenuInflater().inflate(2131165184, menu);
        return 1;
    }
}
```

● 一些发现

● 一些发现

- hook write()
 - 内容: 0x64 0x65 0x78 ('dex')
 - 内存范围: mapped DEX
- 分离出多个DEX
- debug_info_off
- 混淆

讨论



- 反混淆

- 动态分析的局限性

- 特征

+ 一些想法

- 更好的加固方案
 - 性能 VS 安全性 VS 兼容性
 - 混淆 及 加壳
 - 部分加固
 - after/during development
 - Java -> Native
 - less tricks

Thank you !

@GoSSIP_SJTU @乌云峰会 @各大加固厂商
@slipper @MindMac @lcweik