应用加固攻防

汪海（逆巴）

个人简介

阿里移动安全
http://weibo.com/alimobilesecurity

syclover安全小组
http://weibo.com/sycloversyc

议题内容

1.加固脱壳意义

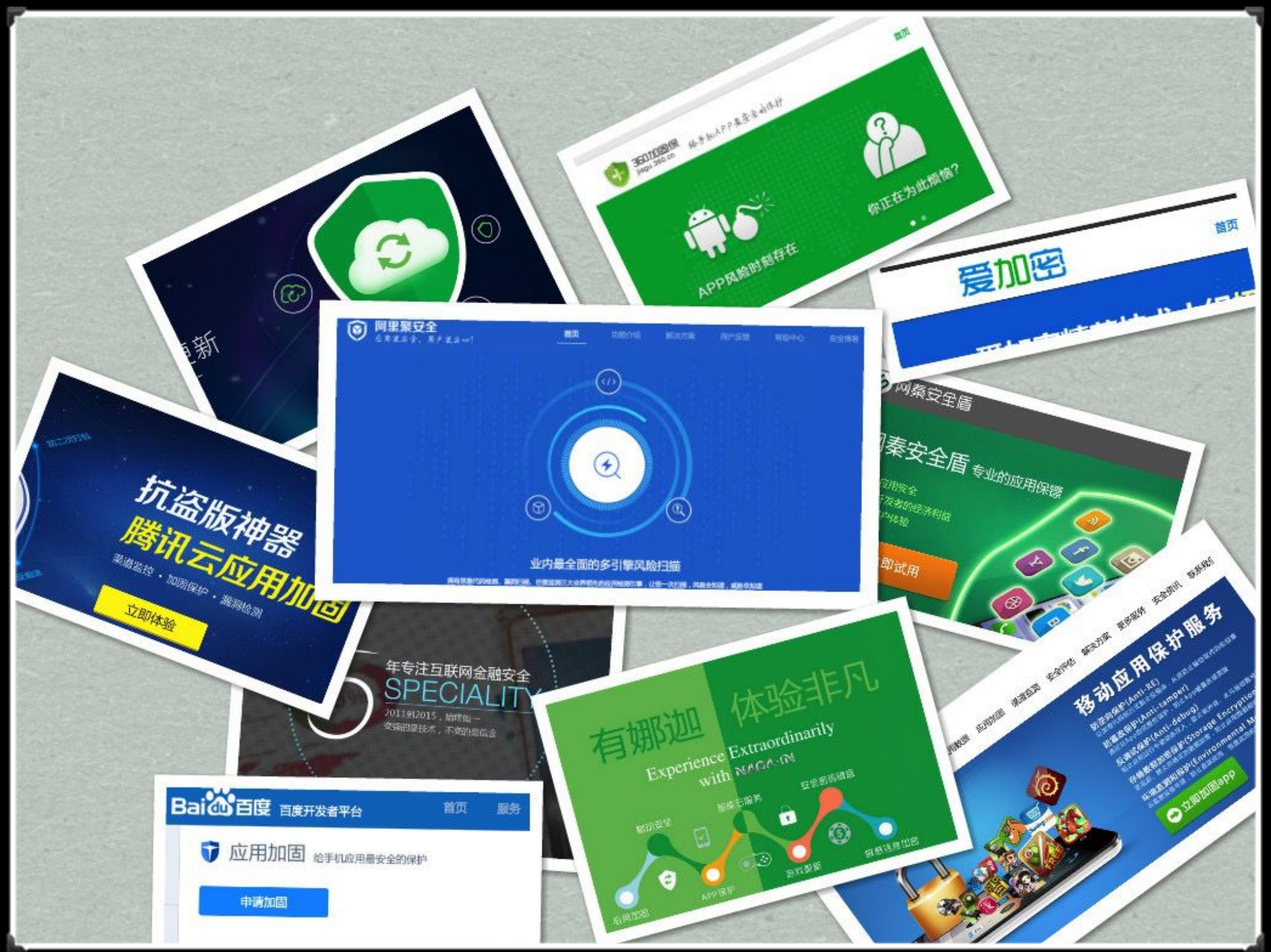
2.目前加固现状

3.某些加固分析

4.如何制作通用脱壳

目前国内加固厂商

Ali,360,tencent,baidu,bangcle,ijiami,naga,通付盾，网秦等



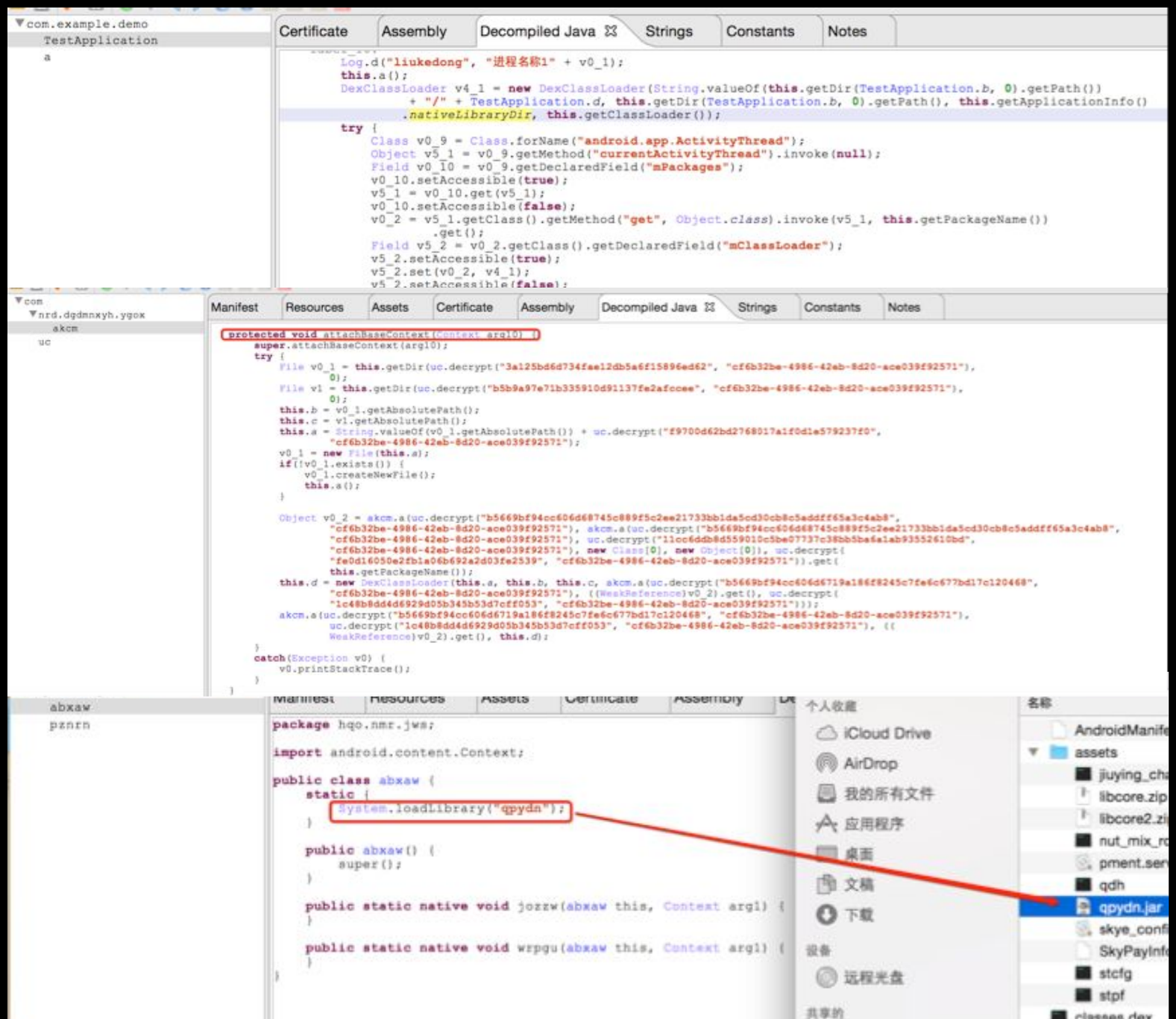
加固意义

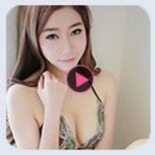
- android java编写，开发门槛低容易被反编译
- android市场混乱，且可自签名，导致大量应用被二次打包，植入广告，木马
- 手机root后，利用hook等技术手段对应用进行动态攻击

脱壳的意义

- 灰色产业大量利用加固
- 给病毒分析人员，以及杀毒引擎带来了挑战
- 漏洞审计，游戏辅助

挖掘到灰色产业大量使用的加固





名称: 魅惑影音

版本: 2.2.1

大小: 1.35 MB

入库: 2015-01-17 15:27:27

加固: others 下载

下载 详情

名称: 情涩视频

版本: 2.2.1

大小: 1.36 MB

入库: 2015-01-24 02:20:43

加固: others 下载

下载 详情

名称: 私密快播

版本: 7.8.0

大小: 2.23 MB

入库: 2015-02-03 21:01:02

加固: others 下载

下载 详情

上午11:49

成人影院
极限快播
捕鱼达人3
成人快播
360手机助手
畅游海贼王..
一键加速
53%
精品推荐
海贼王女帝..
单身交友
搜索
77%
午夜播吧
乐嗨秀场
搞你妹
金蟾千炮捕鱼
妹妹走光了2

资费20元，不含通信费。游戏激活：与美女畅玩可点击进入，轮番上阵任你挑!一次性收取20元（不含通信费）。客服电话4006706603

支付
取消

乎想像。
开 搞

加固技术

第一代加固技术

- 原理：基于android本身提供的类加载技术，源dex被整包加密放到资源目录，壳接管进程启动点

目前某保使用次方案

存在的问题

- 内存中存在连续的解密后dex，可直接dump拿到
- 整体加密对于逆向分析相对简单（存放明显）
- 加固厂商应对内存dump

  dex加载完后抹掉混，淆dex头部等

  内存检查进程是否被注入，以及ZjDroid等脱壳工具

  使用Inotify对/proc/pid/mem和/proc/pid/pagemap进行监视

第二代Dex加密－基于方法保护

- 原理：Java虚拟机在第一次执行某个类的某个方法前，才需要加载这个方法的代码指令

加固方案

- 1.修改 DexCode，access_flag
- 2.修改DexCode,hook DvmReceloveClass

优点

- 此方案dex在内存中不连续，内存dump成本高
- 对于静态分析也是一个挑战
- 将原dex方法指令提取（DexCode），加密存放。存放形式多种相对dex整体加密，更加隐蔽

缺点

与第一代相比性能折损

目前大多加固厂商都使用此方案

某些加固分析

某保-第一代dex加固

某厂商一第二代dex加固

dex加载过程，以及dex结构

java层

BaseDexClassLoader　　DexFile　　mCookie

native层

DexOrJar
isDex
RawDexFile* pRawDexFile;
JarFile*　pJarFile;

JarFile
DvmDex*　pDvmDex;

RawDexFile
DvmDex* pDvmDex

DvmDex
DexFile*　pDexFile
DexHeader* pDexHeader
Stringbject* pResStrings
ClassObject* pResClasses
……

struct DexFile
const DexOptHeader* pOptHeader;
const DexHeader*　pHeader;
const DexStringId*　pStringIds;
const DexTypeId*　pTypeIds;
const DexFieldId*　pFieldIds;const
DexMethodId*　pMethodIds;
const DexProtoId*　pProtoIds;
const DexClassDef*　pClassDefs;

DexHeader
dex_magic
checksum
…

dex结构

dex_header

u1 magic[8]
u4 checksum
u4 file_size
u4 string_ids_size
u4 stringIdsOff
.....
u4 class_defs_size
u4 classDefsOff
.....

DexClassDef_Item

u4 classIdx;
u4 access_flag;
u4 superClassIdx;
u4 interfacesOff;
u4 sourceFileIdx;
u4 annotations;
u4 classDataOff;
u4 statiValusoff;

DexClassData_Item

u4 staticFieldSize;
u4 instanceFieldSize;
u4 directMethodSize;
u4 virtualMethodSize;
DexEncodedFieldList staticFields;
DexEncodedFieldList instanceFields;
DexEncodedMethodList directMethods;
DexEncodedMethodList virtualMethods;

EncodedMethod

uleb128 method_idx_diff;

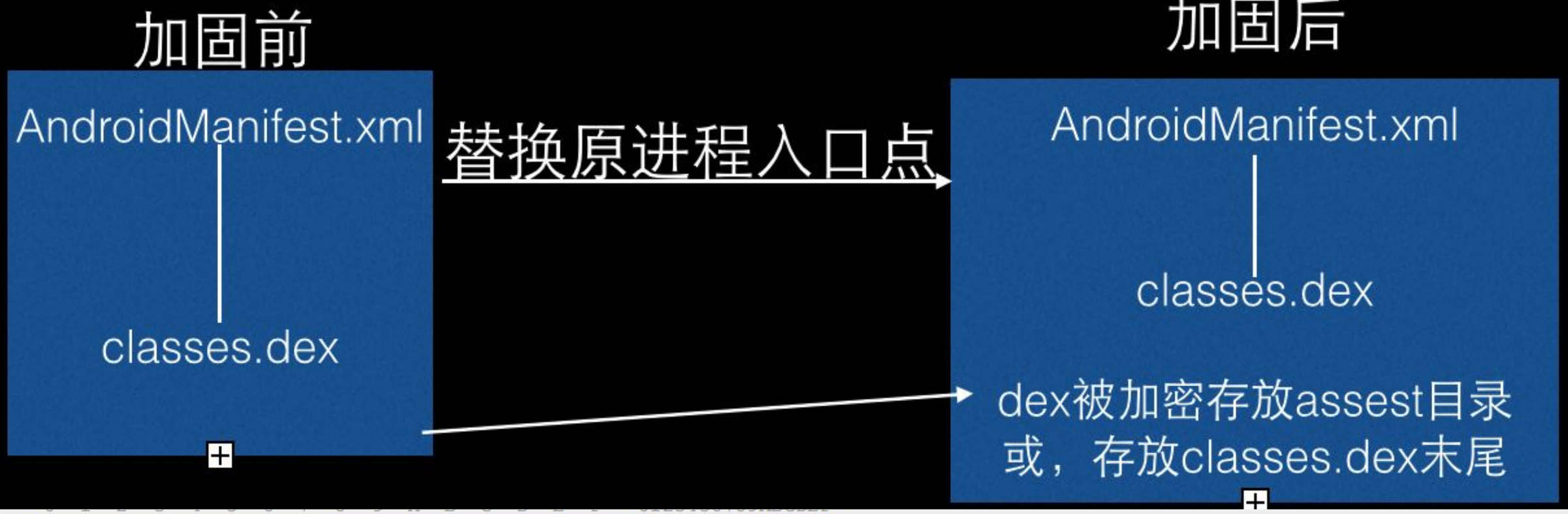
uleb128 access_flags;

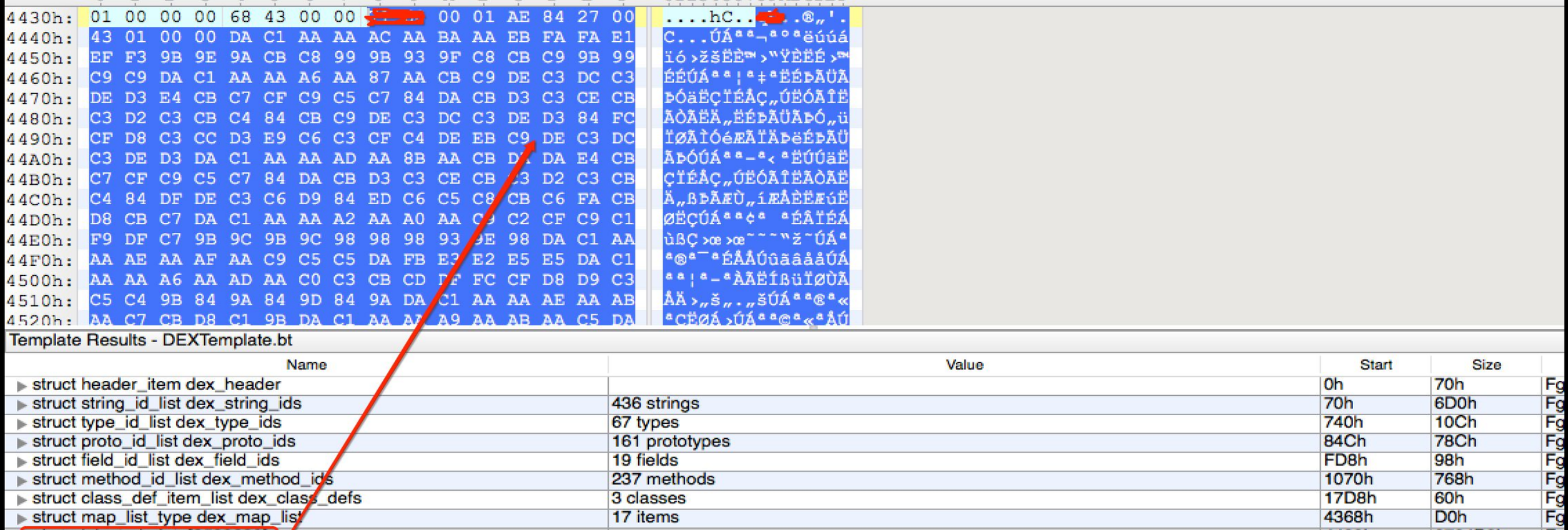
uleb128 code_off;

DexCode

u2 registersSize; u2 insSize;
u2 outsSize;
u2 triesSize;
u4 debugInfoOff; u4 insnsSize;
u2 insns[1];
...

某保


加固前
AndroidManifest.xml
classes.dex
替换原进程入口点
加固后
AndroidManifest.xml
classes.dex
dex被加密存放assest目录
或，存放classes.dex末尾


4430h: 01 00 00 00 68 43 00 00 00 01 AE 84 27 00
4440h: 43 01 00 00 DA C1 AA AA AC AA BA AA EB FA FA E1
4450h: EF F3 9B 9E 9A CB C8 99 9B 93 9F C8 CB C9 9B 99
4460h: C9 C9 DA C1 AA AA A6 AA 87 AA CB C9 DE C3 DC C3
4470h: DE D3 E4 CB C7 CF C9 C5 C7 84 DA CB D3 C3 CE CB
4480h: C3 D2 C3 CB C4 84 CB C9 DE C3 DC C3 DE D3 84 FC
4490h: CF D8 C3 CC D3 E9 C6 C3 CF C4 DE EB C9 DE C3 DC
44A0h: C3 DE D3 DA C1 AA AA AD AA 8B AA CB DA DA E4 CB
44B0h: C7 CF C9 C5 C7 84 DA CB D3 C3 CE CB C3 D2 C3 CB
44C0h: C4 84 DF DE C3 C6 D9 84 ED C6 C5 C8 CB C6 FA CB
44D0h: D8 CB C7 DA C1 AA AA A2 AA A0 AA C9 C2 CF C9 C1
44E0h: F9 DF C7 9B 9C 9B 9C 98 98 98 93 9E 98 DA C1 AA
44F0h: AA AE AA AF AA C9 C5 C5 DA FB E3 E2 E5 E5 DA C1
4500h: AA AA A6 AA AD AA C0 C3 CB CD DF FC CF D8 D9 C3
4510h: C5 C4 9B 84 9A 84 9D 84 9A DA C1 AA AA AE AA AB
4520h: AA C7 CB D8 C1 9B DA C1 AA AA A9 AA AB AA C5 DA
Template Results - DEXTemplate.bt
Name
Value
Start
Size
struct header_item dex_header
0h
70h
struct string_id_list dex_string_ids
436 strings
70h
6D0h
struct type_id_list dex_type_ids
67 types
740h
10Ch
struct proto_id_list dex_proto_ids
161 prototypes
84Ch
78Ch
struct field_id_list dex_field_ids
19 fields
FD8h
98h
struct method_id_list dex_method_ids
237 methods
1070h
768h
struct class_def_item_list dex_class_defs
3 classes
17D8h
60h
struct map_list_type dex_map_list
17 items
4368h
D0h
ubyte injected_data[2589880]
4438h
2784B8h

```
if ( !findModuleAddr(
        "apk@classes.dex",
        (bool (__cdecl *)(const unsigned __int8 *, unsigned int, void *))saveBaseAndSize,
        &odexInfo,
        (char *)&packageName) )
{
  odexInfo_ = odexInfo;
  addr = operator new(0x18u);
  DexUtil::DexUtil(addr, odexInfo, *((_DWORD *)odexInfo_ + 8));
  dexUtil = addr;
  if ( DexUtil::loadPadding((DexUtil *)addr) )// find encrypt_dex,
  {
    sigKey = (const char *)(*(_DWORD *)(dexUtil + 8) + 8);
    key = strchr(sigKey, 0x3A);
    sigStr_off = key;
    if ( key < sigKey )
    {
      ret = 0;
      goto LABEL_6;
    }
    sigData = atoi(key + 1);
    resSum = sigData;
    ret = nativeSigCheck(env_, (char *)&packageName, sigData);
    if ( !ret )
LABEL_6:
      exit(ret);
    ((void (__fastcall *)(_JNIEnv *, signed int))env_->functions->PushLocalFrame)(env_, 128);
    if ( androidVersion(env_) == 5 )
    {
      k = getDexOrJarFor4x(env_, 1, 0);        // 2 true dexFile,cookie
      if ( k )
      {
        mCookie = getDexOrJarFor4x(env_, 0, 0);// 参数3 false 构造DexorJar
        ret_ = updateDexOrJar4X(mCookie, dexUtil);// 重写函数填充DexorJar结构，这里面完成dex解密
LABEL_12:
        k = 0;
        if ( !ret_ )
        {
          NopActivity(env_, mCookie);          // superClass is Activity nop
          k = 1;
          len_ = sigStr_off - sigKey;
          if ( (signed int)len_ > 0 )
          {
            appEnter = malloc(len_ + 1);
            memcpy(appEnter, sigKey, len_);
            *((_BYTE *)appEnter + len_) = 0;
            if ( len_ - 1 <= 0xFE )
              updateApplication(env_, (const char *)appEnter);// 更新application
```

```
signed int __fastcall updateDexOrJar4X(signed int mCookie, int encrypt_dexInfo)
{
  signed int DexOrJar; // r4@1
  struct_pDvmDex *pDvmDex; // r5@1
  void *pRawDexFile; // r0@3
  signed int result; // r0@2

  DexOrJar = mCookie;
  pDvmDex = getDvmDex((struct_a1 *)encrypt_dexInfo);
  if ( pDvmDex )
  {
    result = DexOrJar;
    if ( DexOrJar )
    {
      *(_BYTE *)(DexOrJar + 4) = 1;
      pRawDexFile = calloc(1u, 8u);
      *(_DWORD *)(DexOrJar + 8) = pRawDexFile;
      *((_DWORD *)pRawDexFile + 1) = pDvmDex;
      result = 0;
      *(_BYTE *)(DexOrJar + 5) = 0;
      *(_DWORD *)(DexOrJar + 16) = pDvmDex->dword28;
    }
  }
  else
  {
    result = -1;
  }
  return result;
}
```

内存解密

```
if ( !mappingForLzma((int)&dexBase, DexInfo->dexData, DexInfo->dexSize) )
{
  pDvmDex = 0;
  if ( !dex_recover_from_mem(dexBase, key) )
  {
    dexBase_ = dexBase;
    string_ids_size = *(_DWORD *)(dexBase + 0x38);
    method_ids_size = *(_DWORD *)(dexBase + 0x58);
    type_ids_size = *(_DWORD *)(dexBase + 0x40);
    field_ids_size = *(_DWORD *)(dexBase + 0x50);
    v20 = androidVersion(0);
    if ( v20 > 4 )
    {
      str_size = 4 * string_ids_size;
      type_size = 4 * type_ids_size;
      method_size = 4 * method_ids_size;
      fd = open("/dev/zero", 2);
      pDvmDex = 0;
      if ( fd != -1 )
      {
        lena = (str_size + type_size + 4167 + method_size + 4 * field_ids_size) >> 12 << 12;
        pDvmDex = (struct_pDvmDex *)mmap(0, lena, 3, 2, fd, 0);
        v7 = close(fd);
        if ( pDvmDex == (struct_pDvmDex *)-1 )
        {
LABEL_8:
          pDvmDex = 0;
          goto LABEL_9;
        }
        if ( v7 == -1 )
        {
          munmap(pDvmDex, lena);
          goto LABEL_8;
        }
      }
LABEL_9:
      pDvmDex->pDexFile = DexUtil::dexFileSetupBasicPointers(DexFile, dexBase, 0);
      pDvmDex->pResStrings = (char *)pDvmDex + 72;
      pClassObject = &pDvmDex->char48 + str_size;
      pDvmDex->pResClasses = pClassObject;
      pResMethods = (int)&pClassObject[type_size];
      pDvmDex->pResMethods = pResMethods;
      pDvmDex->pHeader = dexBase_;
      pDvmDex->pResFields = pResMethods + method_size;
      pDvmDex->pInterfaceCache = sub_130D0(); // pInterfaceCache
      pthread_mutex_init((pthread_mutex_t *)((char *)&pDvmDex->pthread_mutex30 + 4), 0);
      pDvmDex->isMappedReadOnly = 0;
      dexBase_2 = dexBase;
      pDvmDex->dword20 = dexBase;
      pDvmDex->dword28 = dexBase_2;
      v11 = v28;
      pDvmDex->dword24 = v28;
      pDvmDex->dword2C = v11;
      return pDvmDex;
```

重写函数构建dvmDex

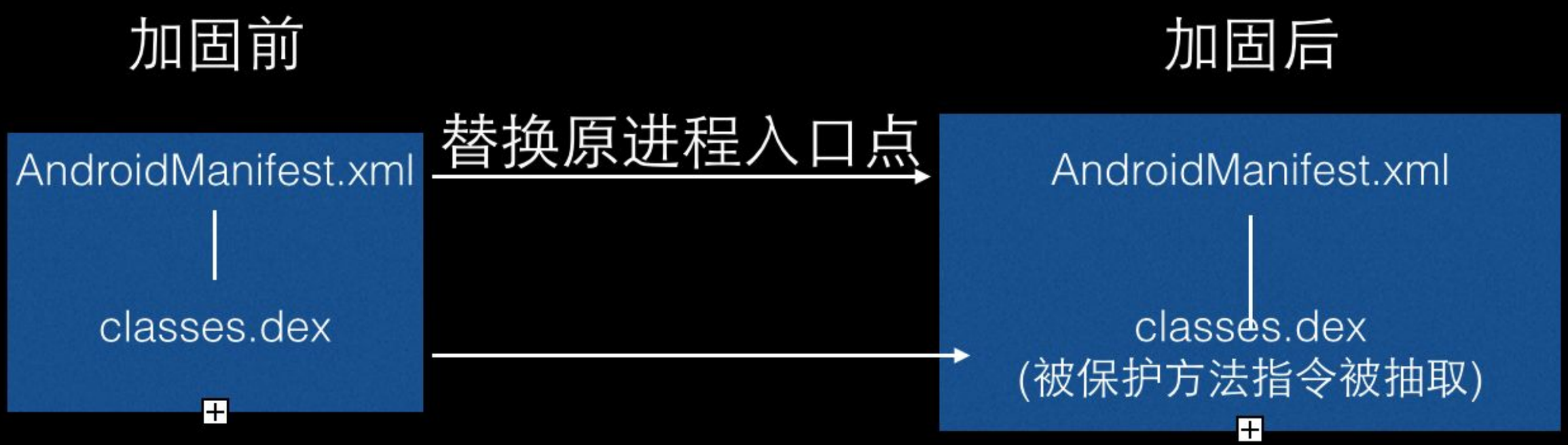
内存解密还原 dex,自己重写函数构造 DexorJar结构，最后修改 mCookie
此方案步骤：

1.由dex字节流获构建 dexFile 结构，重写 dexFileSetupBasicPointers 函数。

2.由 dexFile 构建 DvmDex 结构，重写 allocateAuxStructures 函数。

3.DvmDex → RawDexFile → DexOrJar
save DexOrJar to ClassClader’s mCookie

第二代加固


加固前
AndroidManifest.xml
classes.dex
替换原进程入口点
加固后
AndroidManifest.xml
classes.dex
(被保护方法指令被抽取)

method code 隐藏

```
public main() {
    super();
    this.m_text = new TTextEdit();
    this.m_data = new TDataStruct();
}

private void MyInit() {
}

private void PaiData(TDataStruct arg9) {
}

private void UiSetTextSize() {
}

protected void onActivityResult(int arg8, int arg9, Intent arg10) {
}

public void onClick(View arg4) {
}

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    main.layoutMain = this.getLayoutInflater().inflate(2130903047, null);
    this.setContentView(main.layoutMain);
    this.m_btoPan = this.findViewById(2131165275);
```

启动加载edog,并会执行native方法

method code 隐藏,access_flag被修改

```
protected native void onCreate(Bundle arg1) {
}

protected void onDestroy() {
    if(com.boco.nfc.c.a.b.v != null) {
        com.boco.nfc.c.a.b.v.dismiss();
    }

    this.b();
    super.onDestroy();
}

protected void onPause() {
    super.onPause();
    if(BaseActivity.q != null) {
        BaseActivity.q.disableForegroundDispatch(((Activity)this));
    }
}
```

类实例化前执行静态代码

```
static {
    ShellHelper.StartShell("com.boco.nfc.activity", 44);
}

public BaseActivity() {
    super();
    this.n = "";
    this.o = "";
    this.p = "";
}

public void a(String arg2) {
    Toast.makeText(((Context)this), ((CharSequence)arg2), 0).show();
}
```

```
LOAD:00005C18                 EXPORT Java_com_edog_ELibrary_d1
LOAD:00005C18 Java_com_edog_ELibrary_d1
LOAD:00005C18
LOAD:00005C18 arg_7           =  7
LOAD:00005C18 arg_3C          =  0x3C
LOAD:00005C18 arg_314         =  0x314
LOAD:00005C18
LOAD:00005C18                 LDR             R3, =0x877D0FDE
LOAD:00005C1A                 ADD             R7, SP, #0x350
LOAD:00005C1C                 LDR             R7, =0x5E6F97F6
LOAD:00005C1E                 VLDMIA          PC!, {S23-<bad register>}
LOAD:00005C22                 LDRH            R3, [R0,#6]
LOAD:00005C24                 STR             R2, [R4,#0x6C]
LOAD:00005C26                 STR             R3, [R7,#arg_3C]
LOAD:00005C28                 STR             R6, [SP,#arg_314]
LOAD:00005C2A                 STRB            R5, [R7,#arg_7]
LOAD:00005C2C                 UND             #0x77 ; 'w'
LOAD:00005C2C ; End of function Java_com_edog_ELibrary_d1
LOAD:00005C2C
LOAD:00005C2C ; ---------------------------------------------------------------------------
LOAD:00005C2E                 DCW 0xECCA
LOAD:00005C30                 DCD 0xE07596FD, 0x27A4B2FC, 0xC2FCE2EF, 0x3B82B846, 0xAEDC96D3
LOAD:00005C30                 DCD 0x63DF9FF6, 0x18A00EFE, 0x17C9BBA0, 0xFCDC9ECE
LOAD:00005C54 dword_5C54      DCD 0x877D0FDE          ; DATA XREF: Java_com_edog_ELibrary_d1+r
LOAD:00005C58 ; ---------------------------------------------------------------------------
LOAD:00005C58                 B               _Z13GetMethodSizeP11ClassObjecti ; GetMethodSize(ClassObject *,int)
LOAD:00005C5A ; ---------------------------------------------------------------------------
LOAD:00005C5A                 ASRS            R4, R7, #3
LOAD:00005C5C                 LSLS            R7, R6, #0x1D
LOAD:00005C5E                 STR             R7, [R3,#0x3C]
LOAD:00005C60                 LSRS            R6, R7, #0x1B
LOAD:00005C62                 ADDS            R4, R5, R0
LOAD:00005C64                 B               loc_6464
LOAD:00005C64 ; ---------------------------------------------------------------------------
LOAD:00005C66                 DCW 0xE6FE
LOAD:00005C68                 DCD 0x1C760EFE, 0x17D5B6C8, 0xA48556FD, 0x914F0821, 0xFCFD56FD
LOAD:00005C68                 DCD 0x82FD0FDE, 0x19C20EFE, 0x5D5697FD, 0xB867FE6C, 0x63DF66DE
LOAD:00005C68                 DCD 0xA764DCFE, 0x3B86B6D3, 0xE580901, 0xAEFDD7DE, 0xCBE896CD
LOAD:00005C68                 DCD 0x5FFE22DE, 0x1B140EFE, 0x17C465EB, 0x30FDBC64, 0x91630821
LOAD:00005C68                 DCD 0x5D0C4EC9, 0x7FFECDE2, 0xE0FDCCDC, 0x7FFECDC6, 0xE0FC0384
LOAD:00005C68                 DCD 0x7FFF02AD
LOAD:00005CD0
LOAD:00005CD0 ; =============== S U B R O U T I N E =======================================
LOAD:00005CD0
```

so加固　针对无自定so，笨办法内存dump,修复

获取系统信息，签名校验，反调试，内存加载class数据

```
__int64 __fastcall Java_com_edog_ELibrary_d1(int env, int a2, int pkgName, int envir, unsigned int context)
{
  int env_; // r4@1
  const char *properties; // r0@1
  __int64 mode; // r0@1
  __int64 result; // r0@4
  char s; // [sp+Ch] [bp-E4h]@1
  int v10; // [sp+D4h] [bp-1Ch]@1

  env_ = env;
  v10 = __stack_chk_guard;
  v75205624 = (*(int (__cdecl **)(int))(*(_DWORD *)env + 0x2A4))(env);// pkgName
  v75205620 = ANDROID_API_LEVEL(env_);                  // sdk version (19)
  ANDROID_PLATFORM_VERSION(env_);                       // 系统版本 (4.4)
  ANDROID_PLATFORM_MODEL(env_);                         // AOSP
  ANDROID_PLATFORM_BRAND(env_);                         // Android
  verify(__PAIR__(context, env_));                      // 签名校验
  sprintf(&s, "/data/data/%s/lib/libfdog.so", v75205624);
  anti(&s, &s);                                         // fdog 反调试
  openMemory();                                         // 加载data到内存，为恢复类准备
  properties = (const char *)(*(int (__fastcall **)(int))(*(_DWORD *)env_ + 0x2A4))(env_);
  LODWORD(mode) = strstr(properties, "art");
  if ( (_DWORD)mode || v75205620 > 20 )
    LODWORD(mode) = 1;
  result = restore(mode);                               // inline hook dvmResolveClass
  if ( v10 != __stack_chk_guard )
    __stack_chk_fail(result);
  return result;
}
```

```
int __fastcall replaceFun(int ClassObject, int classId, int a3)
{
  int classObject; // r4@1
  int j; // r6@1
  int size; // r7@1
  int direct_method; // r5@1
  int virtual_method; // r5@4
  int i; // r6@4

  classObject = ClassObject;
  j = 0;
  size = GetMethodSize(ClassObject, 4 * ((unsigned int)(*(_DWORD *)(ClassObject + 0x48) + 1) <= 0));
  direct_method = *(_DWORD *)(classObject + 0x64);
  while ( j < *(_DWORD *)(classObject + 0x60) )
  {
    restoreMethod((void *)classObject, direct_method);// 对direct_method 内的method修复
    direct_method += size;
    ++j;
  }
  virtual_method = *(_DWORD *)(classObject + 0x6C);
  for ( i = 0; i < *(_DWORD *)(classObject + 0x68); ++i )
  {
    restoreMethod((void *)classObject, virtual_method);// 对virtual_method 内的method修复
    virtual_method += size;
  }
  return v75205650(classObject);
}
```

修改入口,壳对DvmResolveClass hook

执行到某个类方法时，首先会调用初始化类，流经DvmResolveClass方法

DvmResolveClass会首先调用replaceFun,对保护method还原，
最后交给虚拟机执行

```
void *__fastcall restoreMethod(void *result, int method)
{
  int method_; // r7@1
  const char *className; // r4@2
  _BYTE *dex_code; // r6@4
  const void *insns; // r5@5
  int v6; // ST14_4@8
  size_t size; // r4@8
  void *addr; // r6@8
  const void *v9; // r0@8
  unsigned int debugInfoOff; // [sp+8h] [bp-30h]@5
  int v11; // [sp+Ch] [bp-2Ch]@5
  int v12; // [sp+10h] [bp-28h]@5
  int dest; // [sp+1Ch] [bp-1Ch]@8

  method_ = method;
  if ( method )
  {
    className = (const char *)*((_DWORD *)result + 6);
    if ( className )
    {
      result = strchr(*((const char **)result + 6), 0x4C);
      if ( result )
      {
        dex_code = *(_BYTE **)(method_ + 32);    // dexCode  结构体大小18
        if ( dex_code )
        {
          insns = dex_code - 16;
          debugInfoOff = *((_DWORD *)dex_code - 2);
          v11 = *((_WORD *)dex_code - 5);
          v12 = *((_DWORD *)dex_code - 1);
          if ( debugInfoOff > 0x1FFFFFFF )
          {
            result = strstr(className, "Landroid/");
            if ( !result && !*dex_code )
            {
              dest = 0;
              v6 = v75205628;
              memcpy(&dest, (const void *)((4 * debugInfoOff & 0x3FFFFFF) + v75205628), 4u);
              size = 2 * (v12 + 8 + 4 * v11 + 4 * (v11 + 1));
              addr = malloc(2 * (v12 + 8 + 4 * v11 + 4 * (v11 + 1)));
              memset(addr, 0, size);
              memcpy(addr, insns, size);
              v9 = (const void *)dbone_crypt_ins(debugInfoOff, v6 + dest, 2 * v12, 1);
              *((_DWORD *)addr + 2) = 0;
              result = memcpy((char *)addr + 16, v9, 2 * v12);
              *(_DWORD *)(method_ + 32) = (char *)addr + 16;
            }
          }
        }
      }
    }
  }
  return result;
```

修复被保护

通用脱壳制作

1.dex加载过程，以及dex结构

2.对关键点拦截，以及内存重建

改rom or hook

```
 */
public class BaseDexClassLoader extends ClassLoader {
    private final DexPathList pathList;

    /**
     * Constructs an instance.
     *
     * @param dexPath the list of jar/apk files containing classes and
     * resources, delimited by {@code File.pathSeparator}, which
     * defaults to {@code ":"} on Android
     * @param optimizedDirectory directory where optimized dex files
     * should be written; may be {@code null}
     * @param libraryPath the list of directories containing native
     * libraries, delimited by {@code File.pathSeparator}; may be
     * {@code null}
     * @param parent the parent class loader
     */
    public BaseDexClassLoader(String dexPath, File optimizedDirectory,
            String libraryPath, ClassLoader parent) {
        super(parent);
        this.pathList = new DexPathList(this, dexPath, libraryPath, optimizedDirectory);
    }
```

dex二进制流拦截

```
int dvmDexFileOpenPartial(const void* addr, int len, DvmDex** ppDvmDex)
{
    DvmDex* pDvmDex;
    DexFile* pDexFile;
    int parseFlags = kDexParseDefault;
    int result = -1;

    /* -- file is incomplete, new checksum has not yet been calculated
    if (gDvm.verifyDexChecksum)
        parseFlags |= kDexParseVerifyChecksum;
    */

    pDexFile = dexFileParse((u1*)addr, len, parseFlags);
    if (pDexFile == NULL) {
        ALOGE("DEX parse failed");
        goto bail;
    }
    pDvmDex = allocateAuxStructures(pDexFile);
    if (pDvmDex == NULL) {
        dexFileFree(pDexFile);
        goto bail;
    }

    pDvmDex->isMappedReadOnly = false;
    *ppDvmDex = pDvmDex;
    result = 0;

bail:
    return result;
}
```

挑战

- dex被破坏
- 字节码抽离

重建支离破碎的dex

- rebuild DexHeader
- fix CodeOff

fix CodeOff

1.通过DexClassDef获取 classname，找到对应的 ClassObject （clazz）对象;

2遍历 directMethods 和virtualMethods，获取 MethodIdx,以及methodName

3通过ClassObject （clazz）对象，methodIdx.protoIdx, MethodName对象通过对比DexCode,DexMethodId 等差异，判断是否存在代码隐藏。并恢复

一些发现

- 反注入, 反模拟器等

成果

- 灰色产业壳，以及各加固厂商壳都可以脱掉，提高引擎查杀能力。