Android8.1 Framework--PackageManagerServer--签名部分

如果一个菲系统签名的APK，安装系统中需要platform的权限，如何做？思路有以下几个：

- 修改APK的签名，也就是伪装签名，（常用在游戏破解）
- 增加用户分组，需要Linux系统底层支持，改起来比较烦。类似与一个文件系统chmod 777这可以让所有用户去读写的方式。
- 伪装APP，需要依附系统APP，取得对应的context。
- ## 较为温柔的方法：

改签名，同时替换pkg中applicationinfo的数据。在鉴权方法中加入相应的代码。

- ## packagemanager简单解析

packageManagerServer中值得注意的是一个handler，具体代码如下：

```
处理枚举：

    static final int SEND_PENDING_BROADCAST = 1;
    static final int MCS_BOUND = 3;
    static final int END_COPY = 4;
    static final int INIT_COPY = 5;
    static final int MCS_UNBIND = 6;
    static final int START_CLEANING_PACKAGE = 7;
    static final int FIND_INSTALL_LOC = 8;
    static final int POST_INSTALL = 9;
    static final int MCS_RECONNECT = 10;
    static final int MCS_GIVE_UP = 11;
    static final int UPDATED_MEDIA_STATUS = 12;
    static final int WRITE_SETTINGS = 13;
    static final int WRITE_PACKAGE_RESTRICTIONS = 14;
    static final int PACKAGE_VERIFIED = 15;
    static final int CHECK_PENDING_VERIFICATION = 16;
    static final int START_INTENT_FILTER_VERIFICATIONS = 17;
    static final int INTENT_FILTER_VERIFIED = 18;
    static final int WRITE_PACKAGE_LIST = 19;
    static final int INSTANT_APP_RESOLUTION_PHASE_TWO = 20;


   class PackageHandler extends Handler {
        private boolean mBound = false;
        final ArrayList<HandlerParams> mPendingInstalls =
            new ArrayList<HandlerParams>();

        private boolean connectToService() {
            if (DEBUG_SD_INSTALL) Log.i(TAG, "Trying to bind to" +
                    " DefaultContainerService");
            Intent service = new
Intent().setComponent(DEFAULT_CONTAINER_COMPONENT);
            Process.setThreadPriority(Process.THREAD_PRIORITY_DEFAULT);
            if (mContext.bindServiceAsUser(service, mDefContainerConn,
                    Context.BIND_AUTO_CREATE, UserHandle.SYSTEM)) {
                Process.setThreadPriority(Process.THREAD_PRIORITY_BACKGROUN
D);
                mBound = true;
                return true;
            }
            Process.setThreadPriority(Process.THREAD_PRIORITY_BACKGROUND);
            return false;
        }
    。。。。


        void doHandleMessage(Message msg) {
            switch (msg.what) {
                case INIT_COPY: {

    ….
```

在内部处理机制中，主要的业务调用都在这个handler中处理，比如初始化copy(INIT_COPY),清理包(START_CLEANING_PACKAGE)等.我们主要看安装部分代码逻辑.具体代码如下:

```
case POST_INSTALL: {
                    if (DEBUG_INSTALL) Log.v(TAG, "Handling post-install
for " + msg.arg1);

                    PostInstallData data = mRunningInstalls.get(msg.arg1);
                    final boolean didRestore = (msg.arg2 != 0);
                    mRunningInstalls.delete(msg.arg1);

                    if (data != null) {
                        InstallArgs args = data.args;
                        PackageInstalledInfo parentRes = data.res;

                        final boolean grantPermissions = (args.installFlags
                                &
PackageManager.INSTALL_GRANT_RUNTIME_PERMISSIONS) != 0;
                        final boolean killApp = (args.installFlags
                                & PackageManager.INSTALL_DONT_KILL_APP) ==
0;
                        final boolean virtualPreload = ((args.installFlags
                                & PackageManager.INSTALL_VIRTUAL_PRELOAD)
!= 0);
                        final String[] grantedPermissions =
args.installGrantPermissions;
```

```
                        // Handle the parent package
                        handlePackagePostInstall(parentRes,
grantPermissions, killApp,
                                virtualPreload, grantedPermissions,
didRestore,
                                args.installerPackageName, args.observer);

                        // Handle the child packages
                        final int childCount =
(parentRes.addedChildPackages != null)
                                ? parentRes.addedChildPackages.size() : 0;
                        for (int i = 0; i < childCount; i++) {
                            PackageInstalledInfo childRes =
parentRes.addedChildPackages.valueAt(i);
                            handlePackagePostInstall(childRes,
grantPermissions, killApp,
                                    virtualPreload, grantedPermissions,
false /*didRestore*/,
                                    args.installerPackageName,
args.observer);
                        }

                        // Log tracing if needed
                        if (args.traceMethod != null) {
                            Trace.asyncTraceEnd(TRACE_TAG_PACKAGE_MANAGER,
args.traceMethod,
                                    args.traceCookie);
                        }
                    } else {
                        Slog.e(TAG, "Bogus post-install token " +
msg.arg1);
                    }

                    Trace.asyncTraceEnd(TRACE_TAG_PACKAGE_MANAGER,
"postInstall", msg.arg1);
                } break;
```

做一个实验，将system.uid属性加入到测试的demo工程，然后编译安装。如下的manifest。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.ecarx.genesis.fastadb"
          android:sharedUserId="android.uid.system"
      >
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
```

安装后，看到logcat当中关于packageManagerServer的打印：

```
2019-02-13 17:04:01.863 1478-1504/? I/PackageManager: Verification timed
out for file:///data/app/vmdl1517622653.tmp
2019-02-13 17:04:01.865 1478-1504/? I/PackageManager: Continuing with
installation of file:///data/app/vmdl1517622653.tmp
2019-02-13 17:18:25.023 1478-1504/? I/PackageManager: Verification timed
out for file:///data/app/vmdl799982913.tmp
2019-02-13 17:18:25.023 1478-1504/? I/PackageManager: Continuing with
installation of file:///data/app/vmdl799982913.tmp
2019-02-13 17:18:26.939 1478-1504/? I/PackageManager: Package
com.glearnlite.glearnslite codePath changed from
/data/app/com.glearnlite.glearnslite-vgazT9dw8ofDXzWwSi9i3A== to
/data/app/com.glearnlite.glearnslite-XIym3-QRJ07wwUoF29Lzyg==; Retaining
data and using new
2019-02-13 17:18:26.940 1478-1504/? W/PackageManager: Code path for
com.glearnlite.glearnslite changing from
/data/app/com.glearnlite.glearnslite-vgazT9dw8ofDXzWwSi9i3A== to
/data/app/com.glearnlite.glearnslite-XIym3-QRJ07wwUoF29Lzyg==
2019-02-13 17:18:26.940 1478-1504/? W/PackageManager: Resource path for
com.glearnlite.glearnslite changing from
/data/app/com.glearnlite.glearnslite-vgazT9dw8ofDXzWwSi9i3A== to
/data/app/com.glearnlite.glearnslite-XIym3-QRJ07wwUoF29Lzyg==
2019-02-13 17:23:46.198 1478-1504/? I/PackageManager: Verification timed
out for file:///data/app/vmdl1435423611.tmp
2019-02-13 17:23:46.199 1478-1504/? I/PackageManager: Continuing with
installation of file:///data/app/vmdl1435423611.tmp
2019-02-13 17:23:47.567 1478-1504/? I/PackageManager: Package
com.glearnlite.glearnslite codePath changed from
/data/app/com.glearnlite.glearnslite-XIym3-QRJ07wwUoF29Lzyg== to
/data/app/com.glearnlite.glearnslite-JmuO1tKN1J8P2v4CnAVqPg==; Retaining
data and using new
2019-02-13 17:23:47.568 1478-1504/? W/PackageManager: Code path for
com.glearnlite.glearnslite changing from
/data/app/com.glearnlite.glearnslite-XIym3-QRJ07wwUoF29Lzyg== to
/data/app/com.glearnlite.glearnslite-JmuO1tKN1J8P2v4CnAVqPg==
2019-02-13 17:23:47.578 1478-1504/? W/PackageManager: Resource path for
com.glearnlite.glearnslite changing from
/data/app/com.glearnlite.glearnslite-XIym3-QRJ07wwUoF29Lzyg== to
/data/app/com.glearnlite.glearnslite-JmuO1tKN1J8P2v4CnAVqPg==
2019-02-28 17:07:20.514 1478-1504/? I/PackageManager: Verification timed
out for file:///data/app/vmdl1633228157.tmp
2019-02-28 17:07:20.515 1478-1504/? I/PackageManager: Continuing with
installation of file:///data/app/vmdl1633228157.tmp
2019-02-28 17:07:22.851 1478-1504/? I/PackageManager: Package
com.glearnlite.glearnslite codePath changed from
/data/app/com.glearnlite.glearnslite-JmuO1tKN1J8P2v4CnAVqPg== to
/data/app/com.glearnlite.glearnslite-OjYkLAV8BNY73a8fAqZG6A==; Retaining
data and using new
2019-02-28 17:07:22.852 1478-1504/? W/PackageManager: Code path for
com.glearnlite.glearnslite changing from
```

```
/data/app/com.glearnlite.glearnslite-JmuO1tKN1J8P2v4CnAVqPg== to
/data/app/com.glearnlite.glearnslite-OjYkLAV8BNY73a8fAqZG6A==
2019-02-28 17:07:22.852 1478-1504/? W/PackageManager: Resource path for
com.glearnlite.glearnslite changing from
/data/app/com.glearnlite.glearnslite-JmuO1tKN1J8P2v4CnAVqPg== to
/data/app/com.glearnlite.glearnslite-OjYkLAV8BNY73a8fAqZG6A==
2019-03-07 15:36:03.624 1478-1504/? I/PackageManager.DexOptimizer: Running
dexopt (dexoptNeeded=1) on: /data/app/vmdl124466309.tmp/base.apk
pkg=com.ecarx.genesis.fastadb isa=x86
dexoptFlags=boot_complete,debuggable,public target-filter=quicken
oatDir=/data/app/vmdl124466309.tmp/oat sharedLibraries=null
2019-03-07 15:36:07.071 1478-1504/? E/PackageManager: Adding duplicate
shared id: 1000 name=com.ecarx.genesis.fastadb
2019-03-07 15:36:07.095 1478-1504/? W/PackageManager:
com.android.server.pm.Installer$InstallerException:
android.os.ServiceSpecificException: Failed to delete
/data/user_de/0/com.ecarx.genesis.fastadb (code 2)
2019-03-07 15:36:07.104 1478-1504/? W/PackageManager: Package couldn't be
installed in /data/app/com.ecarx.genesis.fastadb-56lTioF07krIj4Qtvk1KfA==
    com.android.server.pm.PackageManagerException: Package
com.ecarx.genesis.fastadb has no signatures that match those in shared user
android.uid.system; ignoring!
        at
com.android.server.pm.PackageManagerService.verifySignaturesLP(PackageManag
erService.java:9147)
        at
com.android.server.pm.PackageManagerService.scanPackageDirtyLI(PackageManag
erService.java:10373)
        at
com.android.server.pm.PackageManagerService.scanPackageLI(PackageManagerSer
vice.java:10058)
        at
com.android.server.pm.PackageManagerService.scanPackageTracedLI(PackageMana
gerService.java:10034)
        at
com.android.server.pm.PackageManagerService.installNewPackageLIF(PackageMan
agerService.java:16935)
        at
com.android.server.pm.PackageManagerService.installPackageLI(PackageManager
Service.java:18176)
        at
com.android.server.pm.PackageManagerService.installPackageTracedLI(PackageM
anagerService.java:17731)
        at com.android.server.pm.PackageManagerService.-wrap33(Unknown
Source:0)
        at
com.android.server.pm.PackageManagerService$6.run(PackageManagerService.jav
a:15202)
        at android.os.Handler.handleCallback(Handler.java:789)
        at android.os.Handler.dispatchMessage(Handler.java:98)
        at android.os.Looper.loop(Looper.java:164)
        at android.os.HandlerThread.run(HandlerThread.java:65)
        at com.android.server.ServiceThread.run(ServiceThread.java:46)
```

查阅系统源代码，发现，安装的入口有一个：

```
  private void installPackageLI(InstallArgs args, PackageInstalledInfo res)
{
        final int installFlags = args.installFlags;
        final String installerPackageName = args.installerPackageName;
        final String volumeUuid = args.volumeUuid;


        ` ` ` ` `

        try (PackageFreezer freezer = freezePackageForInstall(pkgName,
installFlags,
                "installPackageLI")) {
            if (replace) {
                if (pkg.applicationInfo.isStaticSharedLibrary()) {
                    // Static libs have a synthetic package name containing
the version
                    // and cannot be updated as an update would get a new
package name,
                    // unless this is the exact same version code which is
useful for
                    // development.
                    PackageParser.Package existingPkg =
mPackages.get(pkg.packageName);
                    if (existingPkg != null && existingPkg.mVersionCode !=
pkg.mVersionCode) {
                        res.setError(INSTALL_FAILED_DUPLICATE_PACKAGE,
"Packages declaring "
                                + "static-shared libs cannot be updated");
                        return;
                    }
                }
                replacePackageLIF(pkg, parseFlags, scanFlags |
SCAN_REPLACING, args.user,
                        installerPackageName, res, args.installReason);
            } else {
                installNewPackageLIF(pkg, parseFlags, scanFlags |
SCAN_DELETE_DATA_ON_FAILURES,
                        args.user, installerPackageName, volumeUuid, res,
args.installReason);
            }
        }
        。。。。。
```

可以看到对应的出错位置，上面的dexOptimizer吧APK中的信息解析完毕，但是在Add的时候报错，首先看到源码段：

```
private void replaceSystemPackageLIF(PackageParser.Package deletedPackage,
            PackageParser.Package pkg, final int policyFlags, int
scanFlags, UserHandle user,
            int[] allUsers, String installerPackageName,
PackageInstalledInfo res,
            int installReason) {
        if (DEBUG_INSTALL) Slog.d(TAG, "replaceSystemPackageLI: new=" + pkg
                + ", old=" + deletedPackage);

        final boolean disabledSystem;

        // Remove existing system package
        removePackageLI(deletedPackage, true);
```

这一段是替换系统应用的代码段，异常爆出的代码段。

```

    /*
     * Install a non-existing package.
     */
    private void installNewPackageLIF(PackageParser.Package pkg, final int
policyFlags,
            int scanFlags, UserHandle user, String installerPackageName,
String volumeUuid,
            PackageInstalledInfo res, int installReason) {
        Trace.traceBegin(TRACE_TAG_PACKAGE_MANAGER, "installNewPackage");

        // Remember this for later, in case we need to rollback this
install
        String pkgName = pkg.packageName;

        if (DEBUG_INSTALL) Slog.d(TAG, "installNewPackageLI: " + pkg);

        synchronized(mPackages) {

```

这一段是安装一个新application所在的代码入口。以新安装为例，我们找到newPacakageLIF的定义方法，找到以下出错代码段：

```
    try {
            //异常代码段
            PackageParser.Package newPackage = scanPackageTracedLI(pkg,
policyFlags, scanFlags,
                    System.currentTimeMillis(), user);

            updateSettingsLI(newPackage, installerPackageName, null, res,
user, installReason);

            if (res.returnCode == PackageManager.INSTALL_SUCCEEDED) {
                prepareAppDataAfterInstallLIF(newPackage);

            } else {
                // Remove package from internal structures, but keep around
any
                // data that might have already existed
                deletePackageLIF(pkgName, UserHandle.ALL, false, null,
                        PackageManager.DELETE_KEEP_DATA, res.removedInfo,
true, null);
            }
        } catch (PackageManagerException e) {
            res.setError("Package couldn't be installed in " +
pkg.codePath, e);
        }
```

跟踪scanPackageTracedLI这个方法到scanPackageDirtyLI方法，具体如下定义：

```
    private PackageParser.Package scanPackageDirtyLI(PackageParser.Package
pkg,
            final int policyFlags, final int scanFlags, long currentTime,
@Nullable UserHandle user)
                    throws PackageManagerException {
        if (DEBUG_PACKAGE_SCANNING) {
            if ((policyFlags & PackageParser.PARSE_CHATTY) != 0)
                Log.d(TAG, "Scanning package " + pkg.packageName);
        }

        applyPolicy(pkg, policyFlags);

        assertPackageIsValid(pkg, policyFlags, scanFlags);

        // Initialize package source and resource directories
        final File scanFile = new File(pkg.codePath);
        final File destCodeFile = new
File(pkg.applicationInfo.getCodePath());
        final File destResourceFile = new
File(pkg.applicationInfo.getResourcePath());

        SharedUserSetting suid = null;
        PackageSetting pkgSetting = null;

        // Getting the package setting may have a side-effect, so if we
        // are only checking if scan would succeed, stash a copy of the
        // old setting to restore at the end.
        PackageSetting nonMutatedPs = null;
        。。。。。
}
```

一步一步排查，我们找到了系统签名认证的代码段，也就是问题出错的代码段：

```
   } else {
                try {
                    // SIDE EFFECTS; compareSignaturesCompat() changes
KeysetManagerService
```

```
                    // 这里对package的签名做了限制，如果签名认证有问题，会报异常。
                    verifySignaturesLP(signatureCheckPs, pkg);
                    // We just determined the app is signed correctly, so
bring
                    // over the latest parsed certs.
                    pkgSetting.signatures.mSignatures = pkg.mSignatures;
                } catch (PackageManagerException e) {
                    if ((policyFlags & PackageParser.PARSE_IS_SYSTEM_DIR)
== 0) {
                        throw e;
                    }
                    // The signature has changed, but this package is in
the system
                    // image...  let's recover!
                    pkgSetting.signatures.mSignatures = pkg.mSignatures;
                    // However...  if this package is part of a shared
user, but it
                    // doesn't match the signature of the shared user,
let's fail.
                    // What this means is that you can't change the
signatures
                    // associated with an overall shared user, which
doesn't seem all
                    // that unreasonable.
                    if (signatureCheckPs.sharedUser != null) {
                        if
(compareSignatures(signatureCheckPs.sharedUser.signatures.mSignatures,
                                pkg.mSignatures) !=
PackageManager.SIGNATURE_MATCH) {
                            throw new PackageManagerException(
                                    INSTALL_PARSE_FAILED_INCONSISTENT_CERTI
FICATES,
                                    "Signature mismatch for shared user: "
                                            + pkgSetting.sharedUser);
                        }
                    }
                    // File a report about this.
                    String msg = "System package " + pkg.packageName
                            + " signature changed; retaining data.";
                    reportSettingsProblem(Log.WARN, msg);
                }
```

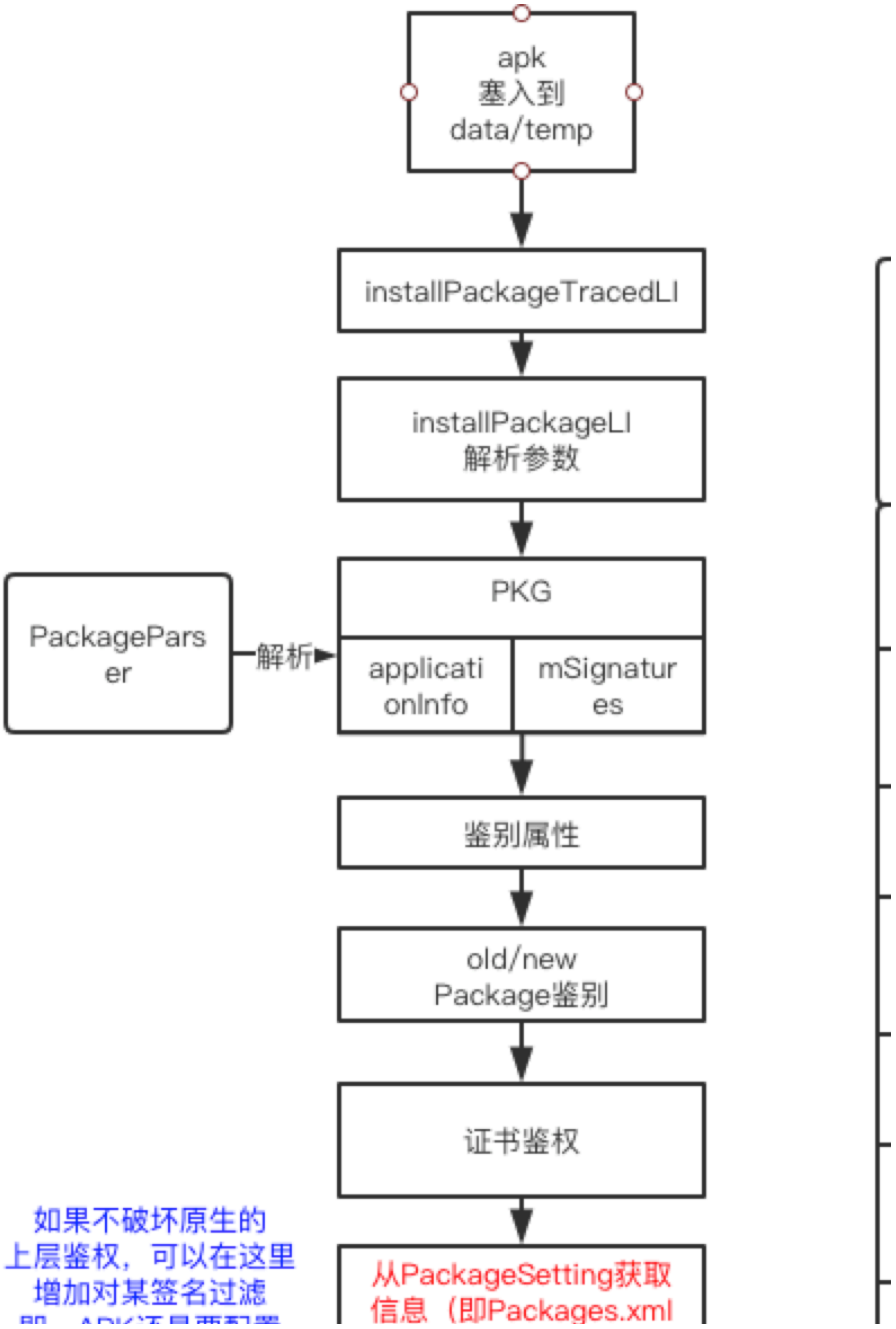
接下来分析一下鉴权签名的那段函数方法：

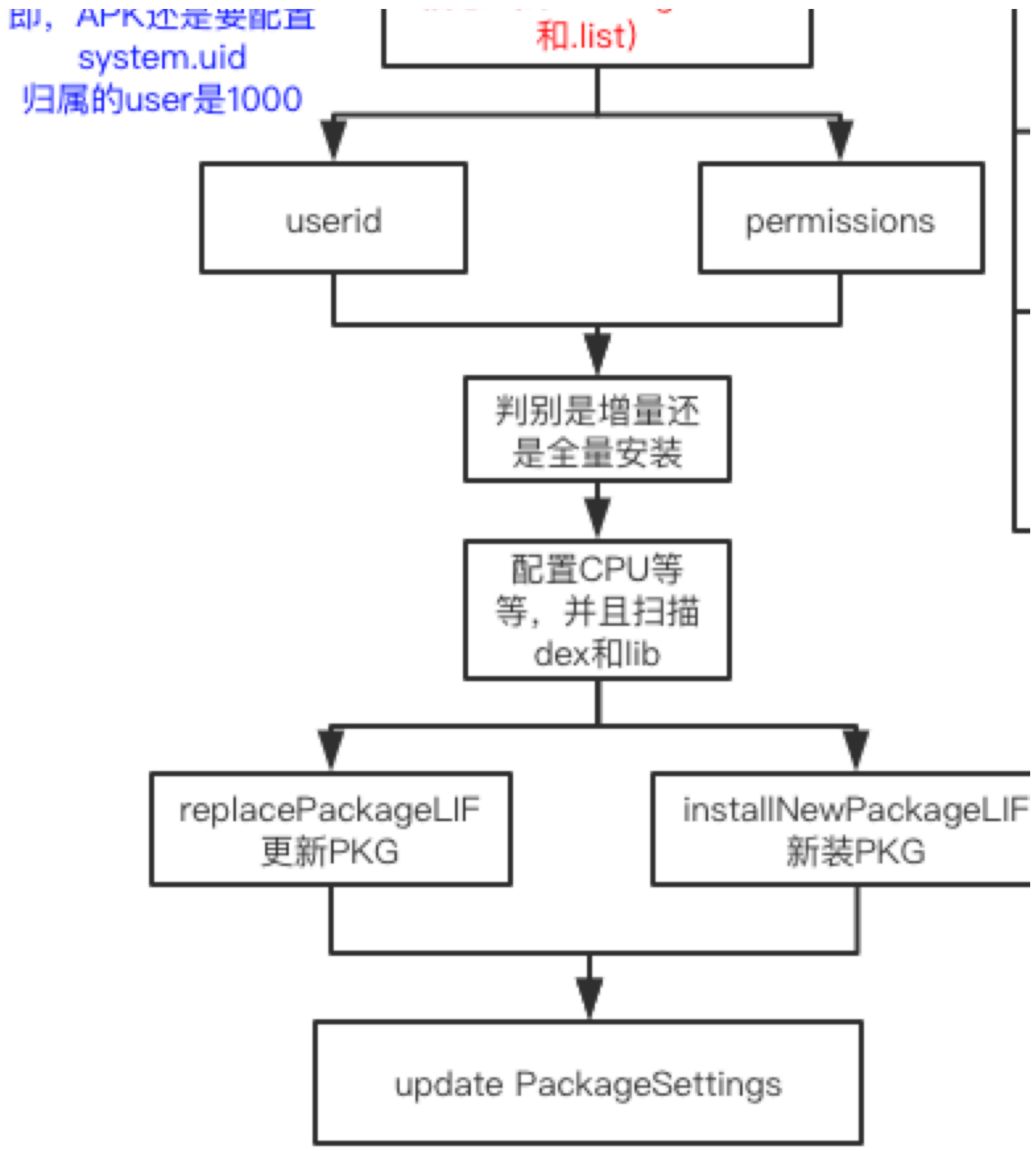
```
private void verifySignaturesLP(PackageSetting pkgSetting,
PackageParser.Package pkg)
            throws PackageManagerException {
        if (pkgSetting.signatures.mSignatures != null) {
            //已安装的APP，签名认证
            // Already existing package. Make sure signatures match
            boolean match =
compareSignatures(pkgSetting.signatures.mSignatures, pkg.mSignatures)
```

```
                    == PackageManager.SIGNATURE_MATCH;
            if (!match) {
                match = compareSignaturesCompat(pkgSetting.signatures, pkg)
                        == PackageManager.SIGNATURE_MATCH;
            }
            if (!match) {
                match = compareSignaturesRecover(pkgSetting.signatures,
pkg)
                        == PackageManager.SIGNATURE_MATCH;
            }
            if (!match) {
                throw new
PackageManagerException(INSTALL_FAILED_UPDATE_INCOMPATIBLE, "Package "
                        + pkg.packageName + " signatures do not match the "
                        + "previously installed version; ignoring!");
            }
        }
        //认证用户签名
        // Check for shared user signatures
        if (pkgSetting.sharedUser != null &&
pkgSetting.sharedUser.signatures.mSignatures != null) {
            // Already existing package. Make sure signatures match
            boolean match =
compareSignatures(pkgSetting.sharedUser.signatures.mSignatures,
                    pkg.mSignatures) == PackageManager.SIGNATURE_MATCH;
            if (!match) {
                match =
compareSignaturesCompat(pkgSetting.sharedUser.signatures, pkg)
                        == PackageManager.SIGNATURE_MATCH;
            }
            if (!match) {
                match =
compareSignaturesRecover(pkgSetting.sharedUser.signatures, pkg)
                        == PackageManager.SIGNATURE_MATCH;
            }
            if (!match) {
                throw new
PackageManagerException(INSTALL_FAILED_SHARED_USER_INCOMPATIBLE,
                        "Package " + pkg.packageName
                        + " has no signatures that match those in shared
user "
                        + pkgSetting.sharedUser.name + "; ignoring!");
            }
        }
    }
```

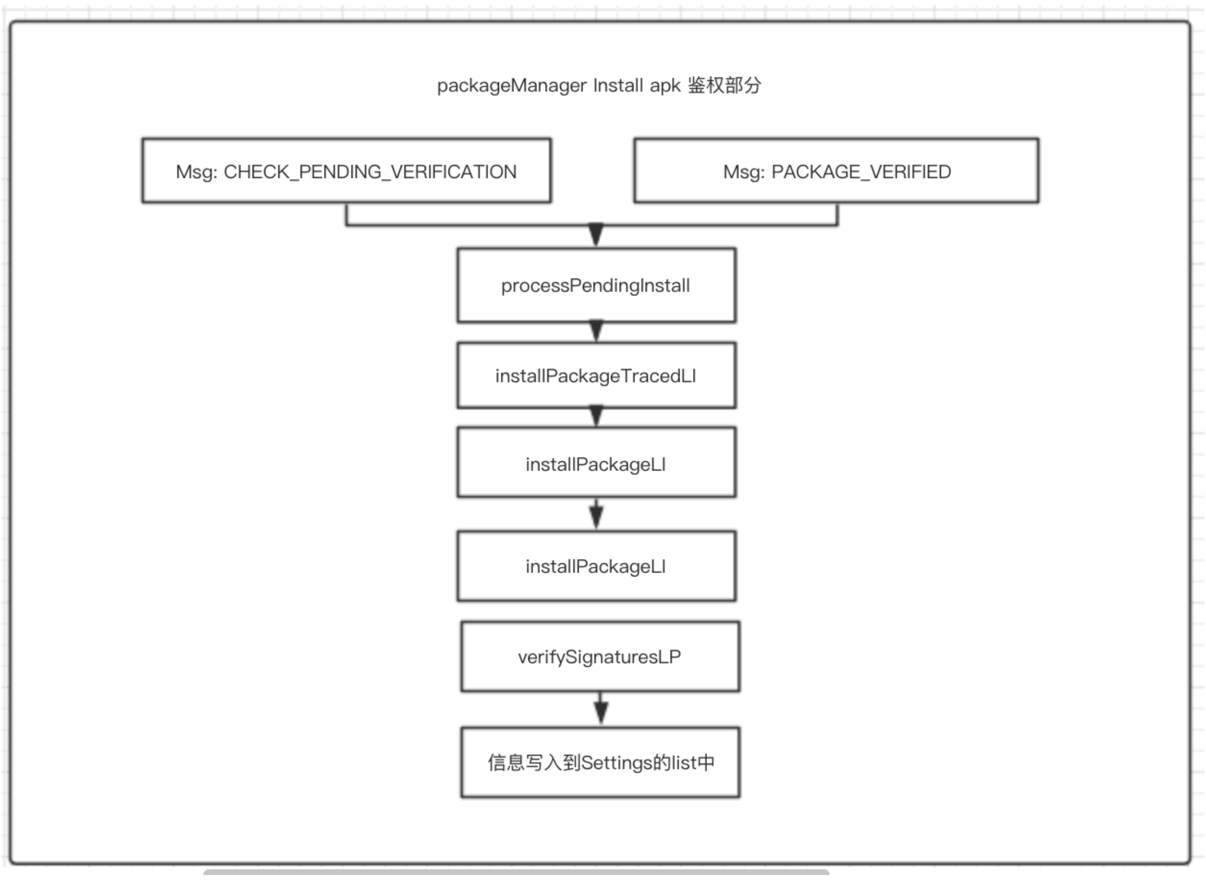
成员变量注意：
Bool mPromoteSystemApps
PackageParser.Package mPlatformPackage;
verifySignaturesLP方法内，对比了pkgsetting的签名和对应安装APK的签名是否一致，

首先判断是签名,然后判断shareduserid系统签名.这里可以了解到的是,以下的一个流程:





即，APK还是要配置
system.uid
归属的user是1000
和.list)
userid
permissions
判别是增量还是全量安装
配置CPU等等，并且扫描dex和lib
replacePackageLIF 更新PKG
installNewPackageLIF 新装PKG
update PackageSettings



在最后的一个地方,写入到settings的列表中的包括应用的签名信息,在APP启动的时候,不会去拿去APK本身的签名,而是用setting中记录的签名信息、userID等。

签名信息的具体内容如下:

```
    private final byte[] mSignature;
    private int mHashCode;
    private boolean mHaveHashCode;
    private SoftReference<String> mStringRef;
    private Certificate[] mCertificateChain;
```

对以上参数加log打印后会发现:

```
2019-03-15 16:08:28.717 1554-1581/system_process I/sephyioths: signature
:android.content.pm.Signature@6ec3a8c7size 1
2019-03-15 16:08:28.717 1554-1581/system_process I/sephyioths:
mCertificates
:OpenSSLRSAPublicKey{modulus=d2d2c486d96c2f18b30d16f7901f29d2ba6ec30430745b
3dda69361f4eb41de7ce82f64502d9a156a8bf5a1fb9d90255f5a61c3a93ed9f279221d1bdb
e8e278e9f84b254da635283cd514bf740170c137709e10a4e087d9c3622a00bafd3f108ff68
d5d377dc65546d97f15141843558bec132d149c1b99e69871bae4fa1cecdafb0710bed060a1
98c30db7a86223ae1cfddbc63452aabcc1015af3eec0ae3804a95858b1cb281ffaaee37dce1
c7aac90476fceacb0ad2a4b0db5e2c2ca982194e02afe2cae2fec6694baeb12a6ca452963ca
7824dae2c6c6ecd43e6a7ba525a2ac457259fe7ae77eca393d40f8874f3f423ae435da73d0a
d04754bb564dad0f,publicExponent=10001}
2019-03-15 16:08:28.717 1554-1581/system_process I/sephyioths: mSigningKeys
```

```
:
{OpenSSLRSAPublicKey{modulus=d2d2c486d96c2f18b30d16f7901f29d2ba6ec30430745b
3dda69361f4eb41de7ce82f64502d9a156a8bf5a1fb9d90255f5a61c3a93ed9f279221d1bdb
e8e278e9f84b254da635283cd514bf740170c137709e10a4e087d9c3622a00bafd3f108ff68
d5d377dc65546d97f15141843558bec132d149c1b99e69871bae4fa1cecdafb0710bed060a1
98c30db7a86223ae1cfddbc63452aabcc1015af3eec0ae3804a95858b1cb281ffaaee37dce1
c7aac90476fceacb0ad2a4b0db5e2c2ca982194e02afe2cae2fec6694baeb12a6ca452963ca
7824dae2c6c6ecd43e6a7ba525a2ac457259fe7ae77eca393d40f8874f3f423ae435da73d0a
d04754bb564dad0f,publicExponent=10001}}
```

参加对比的是publicKey.通过上述的方法，可以伪装签名安装。package的对应类

```
/**
     * Representation of a full package parsed from APK files on disk. A package
     * consists of a single base APK, and zero or more split APKs.
     */
    public final static class Package implements Parcelable {

        public String packageName;

        // The package name declared in the manifest as the package can be
        // renamed, for example static shared libs use synthetic package names.
        public String manifestPackageName;

        /** Names of any split APKs, ordered by parsed splitName */
        public String[] splitNames;

        // TODO: work towards making these paths invariant

        public String volumeUuid;

        /**
         * Path where this package was found on disk. For monolithic packages
         * this is path to single base APK file; for cluster packages this is
         * path to the cluster directory.
         */
        public String codePath;

        /** Path of base APK */
        public String baseCodePath;
        /** Paths of any split APKs, ordered by parsed splitName */
        public String[] splitCodePaths;

        /** Revision code of base APK */
        public int baseRevisionCode;
        /** Revision codes of any split APKs, ordered by parsed splitName */
        public int[] splitRevisionCodes;

        /** Flags of any split APKs; ordered by parsed splitName */
        public int[] splitFlags;

        /**
         * Private flags of any split APKs; ordered by parsed splitName.
         *
         * {@hide}
         */
        public int[] splitPrivateFlags;

        public boolean baseHardwareAccelerated;

        // For now we only support one application per package.
        public ApplicationInfo applicationInfo = new ApplicationInfo();

        public final ArrayList<Permission> permissions = new ArrayList<Permission>(0);
        public final ArrayList<PermissionGroup> permissionGroups = new ArrayList<PermissionGroup>(0);
        public final ArrayList<Activity> activities = new ArrayList<Activity>(0);
        public final ArrayList<Activity> receivers = new ArrayList<Activity>(0);
        public final ArrayList<Provider> providers = new ArrayList<Provider>(0);
        public final ArrayList<Service> services = new ArrayList<Service>(0);
        public final ArrayList<Instrumentation> instrumentation = new ArrayList<Instrumentation>(0);

        public final ArrayList<String> requestedPermissions = new ArrayList<String>();

        public ArrayList<String> protectedBroadcasts;

        public Package parentPackage;
        public ArrayList<Package> childPackages;

        public String staticSharedLibName = null;
        public int staticSharedLibVersion = 0;
        public ArrayList<String> libraryNames = null;
        public ArrayList<String> usesLibraries = null;
        public ArrayList<String> usesStaticLibraries = null;
        public int[] usesStaticLibrariesVersions = null;
        public String[][] usesStaticLibrariesCertDigests = null;
        public ArrayList<String> usesOptionalLibraries = null;
        public String[] usesLibraryFiles = null;

        public ArrayList<ActivityIntentInfo> preferredActivityFilters = null;

        public ArrayList<String> mOriginalPackages = null;
        public String mRealPackage = null;
        public ArrayList<String> mAdoptPermissions = null;

        // We store the application meta-data independently to avoid multiple unwanted references
        public Bundle mAppMetaData = null;

        // The version code declared for this package.
        public int mVersionCode;

        // The version name declared for this package.
        public String mVersionName;

        // The shared user id that this package wants to use.
        public String mSharedUserId;

        // The shared user label that this package wants to use.
        public int mSharedUserLabel;

        // Signatures that were read from the package.
        public Signature[] mSignatures;
        public Certificate[][] mCertificates;

        // For use by package manager service for quick lookup of
        // preferred up order.
        public int mPreferredOrder = 0;

        // For use by package manager to keep track of when a package was last used.
        public long[] mLastPackageUsageTimeInMills =
                new long[PackageManager.NOTIFY_PACKAGE_USE_REASONS_COUNT];

        // // User set enabled state.
        // public int mSetEnabled = PackageManager.COMPONENT_ENABLED_STATE_DEFAULT;
        //
        // // Whether the package has been stopped.
        // public boolean mSetStopped = false;

        // Additional data supplied by callers.
        public Object mExtras;

        // Applications hardware preferences
        public ArrayList<ConfigurationInfo> configPreferences = null;

        // Applications requested features
        public ArrayList<FeatureInfo> reqFeatures = null;

        // Applications requested feature groups
        public ArrayList<FeatureGroupInfo> featureGroups = null;

        public int installLocation;

        public boolean coreApp;

        /* An app that's required for all users and cannot be uninstalled for a user */
        public boolean mRequiredForAllUsers;

        /* The restricted account authenticator type that is used by this application */
        public String mRestrictedAccountType;

        /* The required account type without which this application will not function */
        public String mRequiredAccountType;

        public String mOverlayTarget;
        public int mOverlayPriority;
        public boolean mIsStaticOverlay;
        public boolean mTrustedOverlay;

        /**
         * Data used to feed the KeySetManagerService
         */
        public ArraySet<PublicKey> mSigningKeys;
        public ArraySet<String> mUpgradeKeySets;
        public ArrayMap<String, ArraySet<PublicKey>> mKeySetMapping;

        /**
         * The install time abi override for this package, if any.
         *
         * TODO: This seems like a horrible place to put the abiOverride because
         * this isn't something the packageParser parsers. However, this fits in with
         * the rest of the PackageManager where package scanning randomly pushes
         * and prods fields out of {@code this.applicationInfo}.
         */
        public String cpuAbiOverride;
        /**
         * The install time abi override to choose 32bit abi's when multiple abi's
         * are present. This is only meaningfull for multiarch
```

```
applications.
         * The use32bitAbi attribute is ignored if cpuAbiOverride is also set.
         */
        public boolean use32bitAbi;

        public byte[] restrictUpdateHash;

        /** Set if the app or any of its components are visible to instant applications. */
        public boolean visibleToInstantApps;
        /** Whether or not the package is a stub and must be replaced by the full version. */
        public boolean isStub;

        public Package(String packageName) {
            this.packageName = packageName;
            this.manifestPackageName = packageName;
            applicationInfo.packageName = packageName;
            applicationInfo.uid = -1;
        }


```

applicationinfo对应的class：

```
/*
 * Copyright (C) 2007 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package android.content.pm;

import static android.os.Build.VERSION_CODES.DONUT;

import android.annotation.IntDef;
import android.annotation.SystemApi;
import android.annotation.TestApi;
import android.content.Context;
import android.content.pm.PackageManager.NameNotFoundException;
```

```java
import android.content.res.Resources;
import android.graphics.drawable.Drawable;
import android.os.Environment;
import android.os.Parcel;
import android.os.Parcelable;
import android.os.UserHandle;
import android.os.storage.StorageManager;
import android.text.TextUtils;
import android.util.Printer;
import android.util.SparseArray;

import com.android.internal.util.ArrayUtils;

import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.text.Collator;
import java.util.Arrays;
import java.util.Comparator;
import java.util.Objects;
import java.util.UUID;

/**
 * Information you can retrieve about a particular application.  This
 * corresponds to information collected from the AndroidManifest.xml's
 * &lt;application&gt; tag.
 */
public class ApplicationInfo extends PackageItemInfo implements Parcelable {

    /**
     * Default task affinity of all activities in this application. See
     * {@link ActivityInfo#taskAffinity} for more information.  This comes
     * from the "taskAffinity" attribute.
     */
    public String taskAffinity;

    /**
     * Optional name of a permission required to be able to access this
     * application's components.  From the "permission" attribute.
     */
    public String permission;

    /**
     * The name of the process this application should run in.  From the
     * "process" attribute or, if not set, the same as
     * <var>packageName</var>.
     */
    public String processName;

    /**
```

```
     * Class implementing the Application object.  From the "class"
     * attribute.
     */
    public String className;

    /**
     * A style resource identifier (in the package's resources) of the
     * description of an application.  From the "description" attribute
     * or, if not set, 0.
     */
    public int descriptionRes;

    /**
     * A style resource identifier (in the package's resources) of the
     * default visual theme of the application.  From the "theme" attribute
     * or, if not set, 0.
     */
    public int theme;

    /**
     * Class implementing the Application's manage space
     * functionality.  From the "manageSpaceActivity"
     * attribute. This is an optional attribute and will be null if
     * applications don't specify it in their manifest
     */
    public String manageSpaceActivityName;

    /**
     * Class implementing the Application's backup functionality.  From
     * the "backupAgent" attribute.  This is an optional attribute and
     * will be null if the application does not specify it in its manifest.
     *
     * <p>If android:allowBackup is set to false, this attribute is ignored.
     */
    public String backupAgentName;

    /**
     * An optional attribute that indicates the app supports automatic backup of app data.
     * <p>0 is the default and means the app's entire data folder + managed external storage will
     * be backed up;
     * Any negative value indicates the app does not support full-data backup, though it may still
     * want to participate via the traditional key/value backup API;
     * A positive number specifies an xml resource in which the application has defined its backup
     * include/exclude criteria.
     * <p>If android:allowBackup is set to false, this attribute is ignored.
     *
     * @see android.content.Context#getNoBackupFilesDir()
     * @see #FLAG_ALLOW_BACKUP
     *
     * @hide
     */
    public int fullBackupContent = 0;

    /**
     * The default extra UI options for activities in this application.
     * Set from the {@link android.R.attr#uiOptions} attribute in the
     * activity's manifest.
     */
    public int uiOptions = 0;

    /**
     * Value for {@link #flags}: if set, this application is installed in the
     * device's system image.
     */
    public static final int FLAG_SYSTEM = 1<<0;

    /**
     * Value for {@link #flags}: set to true if this application would like to
     * allow debugging of its
     * code, even when installed on a non-development system.  Comes
     * from {@link android.R.styleable#AndroidManifestApplication_debuggable
     * android:debuggable} of the &lt;application&gt; tag.
     */
    public static final int FLAG_DEBUGGABLE = 1<<1;

    /**
     * Value for {@link #flags}: set to true if this application has code
     * associated with it.  Comes
     * from {@link android.R.styleable#AndroidManifestApplication_hasCode
     * android:hasCode} of the &lt;application&gt; tag.
     */
    public static final int FLAG_HAS_CODE = 1<<2;

    /**
     * Value for {@link #flags}: set to true if this application is persistent.
     * Comes from {@link android.R.styleable#AndroidManifestApplication_persistent
     * android:persistent} of the &lt;application&gt; tag.
```

```java
     */
    public static final int FLAG_PERSISTENT = 1<<3;

    /**
     * Value for {@link #flags}: set to true if this application holds the
     * {@link android.Manifest.permission#FACTORY_TEST} permission and the
     * device is running in factory test mode.
     */
    public static final int FLAG_FACTORY_TEST = 1<<4;

    /**
     * Value for {@link #flags}: default value for the corresponding ActivityInfo flag.
     * Comes from {@link android.R.styleable#AndroidManifestApplication_allowTaskReparenting
     * android:allowTaskReparenting} of the &lt;application&gt; tag.
     */
    public static final int FLAG_ALLOW_TASK_REPARENTING = 1<<5;

    /**
     * Value for {@link #flags}: default value for the corresponding ActivityInfo flag.
     * Comes from {@link android.R.styleable#AndroidManifestApplication_allowClearUserData
     * android:allowClearUserData} of the &lt;application&gt; tag.
     */
    public static final int FLAG_ALLOW_CLEAR_USER_DATA = 1<<6;

    /**
     * Value for {@link #flags}: this is set if this application has been
     * installed as an update to a built-in system application.
     */
    public static final int FLAG_UPDATED_SYSTEM_APP = 1<<7;

    /**
     * Value for {@link #flags}: this is set if the application has specified
     * {@link android.R.styleable#AndroidManifestApplication_testOnly
     * android:testOnly} to be true.
     */
    public static final int FLAG_TEST_ONLY = 1<<8;

    /**
     * Value for {@link #flags}: true when the application's window can be
     * reduced in size for smaller screens.  Corresponds to
     * {@link android.R.styleable#AndroidManifestSupportsScreens_smallScreens
     * android:smallScreens}.
     */
    public static final int FLAG_SUPPORTS_SMALL_SCREENS = 1<<9;

    /**
     * Value for {@link #flags}: true when the application's window can be
     * displayed on normal screens.  Corresponds to
     * {@link android.R.styleable#AndroidManifestSupportsScreens_normalScreens
     * android:normalScreens}.
     */
    public static final int FLAG_SUPPORTS_NORMAL_SCREENS = 1<<10;

    /**
     * Value for {@link #flags}: true when the application's window can be
     * increased in size for larger screens.  Corresponds to
     * {@link android.R.styleable#AndroidManifestSupportsScreens_largeScreens
     * android:largeScreens}.
     */
    public static final int FLAG_SUPPORTS_LARGE_SCREENS = 1<<11;

    /**
     * Value for {@link #flags}: true when the application knows how to adjust
     * its UI for different screen sizes.  Corresponds to
     * {@link android.R.styleable#AndroidManifestSupportsScreens_resizeable
     * android:resizeable}.
     */
    public static final int FLAG_RESIZEABLE_FOR_SCREENS = 1<<12;

    /**
     * Value for {@link #flags}: true when the application knows how to
     * accomodate different screen densities.  Corresponds to
     * {@link android.R.styleable#AndroidManifestSupportsScreens_anyDensity
     * android:anyDensity}.
     */
    public static final int FLAG_SUPPORTS_SCREEN_DENSITIES = 1<<13;

    /**
     * Value for {@link #flags}: set to true if this application would like to
     * request the VM to operate under the safe mode. Comes from
     * {@link android.R.styleable#AndroidManifestApplication_vmSafeMode
     * android:vmSafeMode} of the &lt;application&gt; tag.
     */
    public static final int FLAG_VM_SAFE_MODE = 1<<14;

    /**
     * Value for {@link #flags}: set to <code>false</code> if the application does not wish
     * to permit any OS-driven backups of its data; <code>true</code> otherwise.
     *
     * <p>Comes from the
     * {@link android.R.styleable#AndroidManifestApplication_allowBackup android:allowBackup}
     * attribute of the &lt;application&gt; tag.
     */
    public static final int FLAG_ALLOW_BACKUP = 1<<15;

    /**
     * Value for {@link #flags}: set to <code>false</code> if the application must be kept
     * in memory following a full-system restore operation; <code>true</code> otherwise.
     * Ordinarily, during a full system restore operation each application is shut down
     * following execution of its agent's onRestore() method.  Setting this attribute to
     * <code>false</code> prevents this.  Most applications will not need to set this attribute.
     *
     * <p>If
     * {@link android.R.styleable#AndroidManifestApplication_allowBackup android:allowBackup}
     * is set to <code>false</code> or no
     * {@link android.R.styleable#AndroidManifestApplication_backupAgent android:backupAgent}
     * is specified, this flag will be ignored.
     *
     * <p>Comes from the
     * {@link android.R.styleable#AndroidManifestApplication_killAfterRestore android:killAfterRestore}
     * attribute of the &lt;application&gt; tag.
     */
    public static final int FLAG_KILL_AFTER_RESTORE = 1<<16;

    /**
     * Value for {@link #flags}: Set to <code>true</code> if the application's backup
     * agent claims to be able to handle restore data even "from the future,"
     * i.e. from versions of the application with a versionCode greater than
     * the one currently installed on the device.  <i>Use with caution!</i>  By default
     * this attribute is <code>false</code> and the Backup Manager will ensure that data
     * from "future" versions of the application are never supplied during a restore operation.
     *
     * <p>If
     * {@link android.R.styleable#AndroidManifestApplication_allowBackup android:allowBackup}
     * is set to <code>false</code> or no
     * {@link android.R.styleable#AndroidManifestApplication_backupAgent android:backupAgent}
     * is specified, this flag will be ignored.
     *
     * <p>Comes from the
     * {@link android.R.styleable#AndroidManifestApplication_restoreAnyVersion android:restoreAnyVersion}
     * attribute of the &lt;application&gt; tag.
     */
    public static final int FLAG_RESTORE_ANY_VERSION = 1<<17;

    /**
     * Value for {@link #flags}: Set to true if the application is
     * currently installed on external/removable/unprotected storage.  Such
     * applications may not be available if their storage is not currently
     * mounted.  When the storage it is on is not available, it will look like
     * the application has been uninstalled (its .apk is no longer available)
     * but its persistent data is not removed.
     */
    public static final int FLAG_EXTERNAL_STORAGE = 1<<18;

    /**
     * Value for {@link #flags}: true when the application's window can be
     * increased in size for extra large screens.  Corresponds to
     * {@link android.R.styleable#AndroidManifestSupportsScreens_xlargeScreens
     * android:xlargeScreens}.
     */
    public static final int FLAG_SUPPORTS_XLARGE_SCREENS = 1<<19;

    /**
     * Value for {@link #flags}: true when the application has requested a
     * large heap for its processes.  Corresponds to
     * {@link android.R.styleable#AndroidManifestApplication_largeHeap
     * android:largeHeap}.
     */
    public static final int FLAG_LARGE_HEAP = 1<<20;

    /**
     * Value for {@link #flags}: true if this application's package is
in
     * the stopped state.
     */
    public static final int FLAG_STOPPED = 1<<21;

    /**
     * Value for {@link #flags}: true  when the application is willing
to support
     * RTL (right to left). All activities will inherit this value.
     *
     * Set from the {@link android.R.attr#supportsRtl} attribute in the
     * activity's manifest.
     *
     * Default value is false (no support for RTL).
     */
    public static final int FLAG_SUPPORTS_RTL = 1<<22;

    /**
     * Value for {@link #flags}: true if the application is currently
     * installed for the calling user.
     */
    public static final int FLAG_INSTALLED = 1<<23;

    /**
     * Value for {@link #flags}: true if the application only has its
     * data installed; the application package itself does not currently
     * exist on the device.
     */
    public static final int FLAG_IS_DATA_ONLY = 1<<24;

    /**
     * Value for {@link #flags}: true if the application was declared to
be a
     * game, or false if it is a non-game application.
     *
     * @deprecated use {@link #CATEGORY_GAME} instead.
     */
    @Deprecated
    public static final int FLAG_IS_GAME = 1<<25;

    /**
     * Value for {@link #flags}: {@code true} if the application asks
that only
     * full-data streaming backups of its data be performed even though
it defines
     * a {@link android.app.backup.BackupAgent BackupAgent}, which
```

```
normally
     * indicates that the app will manage its backed-up data via incremental
     * key/value updates.
     */
    public static final int FLAG_FULL_BACKUP_ONLY = 1<<26;

    /**
     * Value for {@link #flags}: {@code true} if the application may use cleartext network traffic
     * (e.g., HTTP rather than HTTPS; WebSockets rather than WebSockets Secure; XMPP, IMAP, STMP
     * without STARTTLS or TLS). If {@code false}, the app declares that it does not intend to use
     * cleartext network traffic, in which case platform components (e.g., HTTP stacks,
     * {@code DownloadManager}, {@code MediaPlayer}) will refuse app's requests to use cleartext
     * traffic. Third-party libraries are encouraged to honor this flag as well.
     *
     * <p>NOTE: {@code WebView} does not honor this flag.
     *
     * <p>This flag is ignored on Android N and above if an Android Network Security Config is
     * present.
     *
     * <p>This flag comes from
     * {@link android.R.styleable#AndroidManifestApplication_usesCleartextTraffic
     * android:usesCleartextTraffic} of the &lt;application&gt; tag.
     */
    public static final int FLAG_USES_CLEARTEXT_TRAFFIC = 1<<27;

    /**
     * When set installer extracts native libs from .apk files.
     */
    public static final int FLAG_EXTRACT_NATIVE_LIBS = 1<<28;

    /**
     * Value for {@link #flags}: {@code true} when the application's rendering
     * should be hardware accelerated.
     */
    public static final int FLAG_HARDWARE_ACCELERATED = 1<<29;

    /**
     * Value for {@link #flags}: true if this application's package is in
     * the suspended state.
     */
```

```java
    public static final int FLAG_SUSPENDED = 1<<30;

    /**
     * Value for {@link #flags}: true if code from this application will need to be
     * loaded into other applications' processes. On devices that support multiple
     * instruction sets, this implies the code might be loaded into a process that's
     * using any of the devices supported instruction sets.
     *
     * <p> The system might treat such applications specially, for eg., by
     * extracting the application's native libraries for all supported instruction
     * sets or by compiling the application's dex code for all supported instruction
     * sets.
     */
    public static final int FLAG_MULTIARCH  = 1 << 31;

    /**
     * Flags associated with the application.  Any combination of
     * {@link #FLAG_SYSTEM}, {@link #FLAG_DEBUGGABLE}, {@link #FLAG_HAS_CODE},
     * {@link #FLAG_PERSISTENT}, {@link #FLAG_FACTORY_TEST}, and
     * {@link #FLAG_ALLOW_TASK_REPARENTING}
     * {@link #FLAG_ALLOW_CLEAR_USER_DATA}, {@link #FLAG_UPDATED_SYSTEM_APP},
     * {@link #FLAG_TEST_ONLY}, {@link #FLAG_SUPPORTS_SMALL_SCREENS},
     * {@link #FLAG_SUPPORTS_NORMAL_SCREENS},
     * {@link #FLAG_SUPPORTS_LARGE_SCREENS}, {@link #FLAG_SUPPORTS_XLARGE_SCREENS},
     * {@link #FLAG_RESIZEABLE_FOR_SCREENS},
     * {@link #FLAG_SUPPORTS_SCREEN_DENSITIES}, {@link #FLAG_VM_SAFE_MODE},
     * {@link #FLAG_ALLOW_BACKUP}, {@link #FLAG_KILL_AFTER_RESTORE},
     * {@link #FLAG_RESTORE_ANY_VERSION}, {@link #FLAG_EXTERNAL_STORAGE},
     * {@link #FLAG_LARGE_HEAP}, {@link #FLAG_STOPPED},
     * {@link #FLAG_SUPPORTS_RTL}, {@link #FLAG_INSTALLED},
     * {@link #FLAG_IS_DATA_ONLY}, {@link #FLAG_IS_GAME},
     * {@link #FLAG_FULL_BACKUP_ONLY}, {@link #FLAG_USES_CLEARTEXT_TRAFFIC},
     * {@link #FLAG_MULTIARCH}.
     */
    public int flags = 0;

    /**
     * Value for {@link #privateFlags}: true if the application is hidden via restrictions and for
     * most purposes is considered as not installed.
     * {@hide}
     */
    public static final int PRIVATE_FLAG_HIDDEN = 1<<0;

    /**
     * Value for {@link #privateFlags}: set to <code>true</code> if the application
     * has reported that it is heavy-weight, and thus can not participate in
     * the normal application lifecycle.
     *
     * <p>Comes from the
     * android.R.styleable#AndroidManifestApplication_cantSaveState
     * attribute of the &lt;application&gt; tag.
     *
     * {@hide}
     */
    public static final int PRIVATE_FLAG_CANT_SAVE_STATE = 1<<1;

    /**
     * Value for {@link #privateFlags}: Set to true if the application has been
     * installed using the forward lock option.
     *
     * NOTE: DO NOT CHANGE THIS VALUE!  It is saved in packages.xml.
     *
     * {@hide}
     */
    public static final int PRIVATE_FLAG_FORWARD_LOCK = 1<<2;

    /**
     * Value for {@link #privateFlags}: set to {@code true} if the application
     * is permitted to hold privileged permissions.
     *
     * {@hide}
     */
    public static final int PRIVATE_FLAG_PRIVILEGED = 1<<3;

    /**
     * Value for {@link #privateFlags}: {@code true} if the application has any IntentFiler
     * with some data URI using HTTP or HTTPS with an associated VIEW action.
     *
     * {@hide}
     */
    public static final int PRIVATE_FLAG_HAS_DOMAIN_URLS = 1<<4;

    /**
```

```
     * When set, the default data storage directory for this app is pointed at
     * the device-protected location.
     *
     * @hide
     */
    public static final int PRIVATE_FLAG_DEFAULT_TO_DEVICE_PROTECTED_STORAGE = 1 << 5;

    /**
     * When set, assume that all components under the given app are direct boot
     * aware, unless otherwise specified.
     *
     * @hide
     */
    public static final int PRIVATE_FLAG_DIRECT_BOOT_AWARE = 1 << 6;

    /**
     * Value for {@link #privateFlags}: {@code true} if the application is installed
     * as instant app.
     *
     * @hide
     */
    public static final int PRIVATE_FLAG_INSTANT = 1 << 7;

    /**
     * When set, at least one component inside this application is direct boot
     * aware.
     *
     * @hide
     */
    public static final int PRIVATE_FLAG_PARTIALLY_DIRECT_BOOT_AWARE = 1 << 8;


    /**
     * When set, signals that the application is required for the system user and should not be
     * uninstalled.
     *
     * @hide
     */
    public static final int PRIVATE_FLAG_REQUIRED_FOR_SYSTEM_USER = 1 << 9;

    /**
     * When set, the application explicitly requested that its activities be resizeable by default.
     * @see android.R.styleable#AndroidManifestActivity_resizeableActivity
     *
     * @hide
     */
    public static final int PRIVATE_FLAG_ACTIVITIES_RESIZE_MODE_RESIZEABLE = 1 << 10;

    /**
     * When set, the application explicitly requested that its activities *not* be resizeable by
     * default.
     * @see android.R.styleable#AndroidManifestActivity_resizeableActivity
     *
     * @hide
     */
    public static final int PRIVATE_FLAG_ACTIVITIES_RESIZE_MODE_UNRESIZEABLE = 1 << 11;

    /**
     * The application isn't requesting explicitly requesting for its activities to be resizeable or
     * non-resizeable by default. So, we are making it activities resizeable by default based on the
     * target SDK version of the app.
     * @see android.R.styleable#AndroidManifestActivity_resizeableActivity
     *
     * NOTE: This only affects apps with target SDK >= N where the resizeableActivity attribute was
     * introduced. It shouldn't be confused with {@link ActivityInfo#RESIZE_MODE_FORCE_RESIZEABLE}
     * where certain pre-N apps are forced to the resizeable.
     *
     * @hide
     */
    public static final int PRIVATE_FLAG_ACTIVITIES_RESIZE_MODE_RESIZEABLE_VIA_SDK_VERSION =
            1 << 12;

    /**
     * Value for {@link #privateFlags}: {@code true} means the OS should go ahead and
     * run full-data backup operations for the app even when it is in a
     * foreground-equivalent run state.  Defaults to {@code false} if unspecified.
     * @hide
     */
    public static final int PRIVATE_FLAG_BACKUP_IN_FOREGROUND = 1 << 13;

```

```java
    /**
     * Value for {@link #privateFlags}: {@code true} means this application
     * contains a static shared library. Defaults to {@code false} if unspecified.
     * @hide
     */
    public static final int PRIVATE_FLAG_STATIC_SHARED_LIBRARY = 1 << 14;

    /**
     * Value for {@link #privateFlags}: When set, the application will only have its splits loaded
     * if they are required to load a component. Splits can be loaded on demand using the
     * {@link Context#createContextForSplit(String)} API.
     * @hide
     */
    public static final int PRIVATE_FLAG_ISOLATED_SPLIT_LOADING = 1 << 15;

    /**
     * Value for {@link #privateFlags}: When set, the application was installed as
     * a virtual preload.
     * @hide
     */
    public static final int PRIVATE_FLAG_VIRTUAL_PRELOAD = 1 << 16;

    /** @hide */
    @IntDef(flag = true, prefix = { "PRIVATE_FLAG_" }, value = {
            PRIVATE_FLAG_HIDDEN,
            PRIVATE_FLAG_CANT_SAVE_STATE,
            PRIVATE_FLAG_FORWARD_LOCK,
            PRIVATE_FLAG_PRIVILEGED,
            PRIVATE_FLAG_HAS_DOMAIN_URLS,
            PRIVATE_FLAG_DEFAULT_TO_DEVICE_PROTECTED_STORAGE,
            PRIVATE_FLAG_DIRECT_BOOT_AWARE,
            PRIVATE_FLAG_INSTANT,
            PRIVATE_FLAG_PARTIALLY_DIRECT_BOOT_AWARE,
            PRIVATE_FLAG_REQUIRED_FOR_SYSTEM_USER,
            PRIVATE_FLAG_ACTIVITIES_RESIZE_MODE_RESIZEABLE,
            PRIVATE_FLAG_ACTIVITIES_RESIZE_MODE_UNRESIZEABLE,
            PRIVATE_FLAG_ACTIVITIES_RESIZE_MODE_RESIZEABLE_VIA_SDK_VERSION,
            PRIVATE_FLAG_BACKUP_IN_FOREGROUND,
            PRIVATE_FLAG_STATIC_SHARED_LIBRARY,
            PRIVATE_FLAG_ISOLATED_SPLIT_LOADING,
            PRIVATE_FLAG_VIRTUAL_PRELOAD,
    })
    @Retention(RetentionPolicy.SOURCE)
    public @interface ApplicationInfoPrivateFlags {}

    /**
     * Private/hidden flags. See {@code PRIVATE_FLAG_...} constants.
     * @hide
     */
    public @ApplicationInfoPrivateFlags int privateFlags;

    /**
     * @hide
     */
    public static final String METADATA_PRELOADED_FONTS = "preloaded_fonts";

    /**
     * The required smallest screen width the application can run on.  If 0,
     * nothing has been specified.  Comes from
     * {@link android.R.styleable#AndroidManifestSupportsScreens_requiresSmallestWidthDp
     * android:requiresSmallestWidthDp} attribute of the &lt;supports-screens&gt; tag.
     */
    public int requiresSmallestWidthDp = 0;

    /**
     * The maximum smallest screen width the application is designed for.  If 0,
     * nothing has been specified.  Comes from
     * {@link android.R.styleable#AndroidManifestSupportsScreens_compatibleWidthLimitDp
     * android:compatibleWidthLimitDp} attribute of the &lt;supports-screens&gt; tag.
     */
    public int compatibleWidthLimitDp = 0;

    /**
     * The maximum smallest screen width the application will work on.  If 0,
     * nothing has been specified.  Comes from
     * {@link android.R.styleable#AndroidManifestSupportsScreens_largestWidthLimitDp
     * android:largestWidthLimitDp} attribute of the &lt;supports-screens&gt; tag.
     */
    public int largestWidthLimitDp = 0;

    /**
     * Value indicating the maximum aspect ratio the application supports.
     * <p>
     * 0 means unset.
     * @See {@link android.R.attr#maxAspectRatio}.
     * @hide
     */
    public float maxAspectRatio;

    /** @removed */
    @Deprecated
    public String volumeUuid;

    /**
     * UUID of the storage volume on which this application is being hosted. For
     * apps hosted on the default internal storage at
     * {@link Environment#getDataDirectory()}, the UUID value is
     * {@link StorageManager#UUID_DEFAULT}.
     */
    public UUID storageUuid;

    /** {@hide} */
    public String scanSourceDir;
    /** {@hide} */
    public String scanPublicSourceDir;

    /**
     * Full path to the base APK for this application.
     */
    public String sourceDir;

    /**
     * Full path to the publicly available parts of {@link #sourceDir},
     * including resources and manifest. This may be different from
     * {@link #sourceDir} if an application is forward locked.
     */
    public String publicSourceDir;

    /**
     * The names of all installed split APKs, ordered lexicographically.
     */
    public String[] splitNames;

    /**
     * Full paths to zero or more split APKs, indexed by the same order as {@link #splitNames}.
     */
    public String[] splitSourceDirs;

    /**
     * Full path to the publicly available parts of {@link #splitSourceDirs},
     * including resources and manifest. This may be different from
     * {@link #splitSourceDirs} if an application is forward locked.
     *
     * @see #splitSourceDirs
     */
    public String[] splitPublicSourceDirs;

    /**
     * Maps the dependencies between split APKs. All splits implicitly depend on the base APK.
     *
     * Available since platform version O.
     *
     * Only populated if the application opts in to isolated split loading via the
     * {@link android.R.attr.isolatedSplits} attribute in the &lt;manifest&gt; tag of the app's
     * AndroidManifest.xml.
     *
     * The keys and values are all indices into the {@link #splitNames}, {@link #splitSourceDirs},
     * and {@link #splitPublicSourceDirs} arrays.
     * Each key represents a split and its value is an array of splits. The first element of this
     * array is the parent split, and the rest are configuration splits. These configuration splits
     * have no dependencies themselves.
     * Cycles do not exist because they are illegal and screened for during installation.
     *
     * May be null if no splits are installed, or if no dependencies exist between them.
     *
     * NOTE: Any change to the way split dependencies are stored must update the logic that
     *       creates the class loader context for dexopt (DexoptUtils#getClassLoaderContexts).
     *
     * @hide
     */
    public SparseArray<int[]> splitDependencies;

    /**
     * Full paths to the locations of extra resource packages (runtime overlays)
     * this application uses. This field is only used if there are extra resource
     * packages, otherwise it is null.
     *
     * {@hide}
     */
    public String[] resourceDirs;
```

```java

    /**
     * String retrieved from the seinfo tag found in selinux policy. This value
     * can be overridden with a value set through the mac_permissions.xml policy
     * construct. This value is useful in setting an SELinux security context on
     * the process as well as its data directory. The String default is being used
     * here to represent a catchall label when no policy matches.
     *
     * {@hide}
     */
    public String seInfo = "default";

    /**
     * The seinfo tag generated per-user. This value may change based upon the
     * user's configuration. For example, when an instant app is installed for
     * a user. It is an error if this field is ever {@code null} when trying to
     * start a new process.
     * <p>NOTE: We need to separate this out because we modify per-user values
     * multiple times. This needs to be refactored since we're performing more
     * work than necessary and these values should only be set once. When that
     * happens, we can merge the per-user value with the seInfo state above.
     *
     * {@hide}
     */
    public String seInfoUser;

    /**
     * Paths to all shared libraries this application is linked against.  This
     * field is only set if the {@link PackageManager#GET_SHARED_LIBRARY_FILES
     * PackageManager.GET_SHARED_LIBRARY_FILES} flag was used when retrieving
     * the structure.
     */
    public String[] sharedLibraryFiles;

    /**
     * Full path to the default directory assigned to the package for its
     * persistent data.
     */
    public String dataDir;

    /**
     * Full path to the device-protected directory assigned to the package for
     * its persistent data.
     *
     * @see Context#createDeviceProtectedStorageContext()
     */
    public String deviceProtectedDataDir;

    /**
     * Full path to the credential-protected directory assigned to the package
     * for its persistent data.
     *
     * @hide
     */
    @SystemApi
    public String credentialProtectedDataDir;

    /**
     * Full path to the directory where native JNI libraries are stored.
     */
    public String nativeLibraryDir;

    /**
     * Full path where unpacked native libraries for {@link #secondaryCpuAbi}
     * are stored, if present.
     *
     * The main reason this exists is for bundled multi-arch apps, where
     * it's not trivial to calculate the location of libs for the secondary abi
     * given the location of the primary.
     *
     * TODO: Change the layout of bundled installs so that we can use
     * nativeLibraryRootDir & nativeLibraryRootRequiresIsa there as well.
     * (e.g {@code [ "/system/app-lib/Foo/arm", "/system/app-lib/Foo/arm64" ]}
     * instead of {@code [ "/system/lib/Foo", "/system/lib64/Foo" ]}.
     *
     * @hide
     */
    public String secondaryNativeLibraryDir;

    /**
     * The root path where unpacked native libraries are stored.
     * <p>
     * When {@link #nativeLibraryRootRequiresIsa} is set, the libraries are
     * placed in ISA-specific subdirectories under this path, otherwise the
     * libraries are placed directly at this path.
     *
     * @hide
     */
    public String nativeLibraryRootDir;

    /**
     * Flag indicating that ISA must be appended to
     * {@link #nativeLibraryRootDir} to be useful.
     *
     * @hide
     */
    public boolean nativeLibraryRootRequiresIsa;

    /**
     * The primary ABI that this application requires, This is inferred from the ABIs
     * of the native JNI libraries the application bundles. Will be {@code null}
     * if this application does not require any particular ABI.
     *
     * If non-null, the application will always be launched with this ABI.
     *
     * {@hide}
     */
    public String primaryCpuAbi;

    /**
     * The secondary ABI for this application. Might be non-null for multi-arch
     * installs. The application itself never uses this ABI, but other applications that
     * use its code might.
     *
     * {@hide}
     */
    public String secondaryCpuAbi;

    /**
     * The kernel user-ID that has been assigned to this application;
     * currently this is not a unique ID (multiple applications can have
     * the same uid).
     */
    public int uid;

    /**
     * The minimum SDK version this application can run on. It will not run
     * on earlier versions.
     */
    public int minSdkVersion;

    /**
     * The minimum SDK version this application targets.  It may run on earlier
     * versions, but it knows how to work with any new behavior added at this
     * version.  Will be {@link android.os.Build.VERSION_CODES#CUR_DEVELOPMENT}
     * if this is a development build and the app is targeting that.  You should
     * compare that this number is >= the SDK version number at which your
     * behavior was introduced.
     */
    public int targetSdkVersion;

    /**
     * The app's declared version code.
     * @hide
     */
    public int versionCode;

    /**
     * When false, indicates that all components within this application are
     * considered disabled, regardless of their individually set enabled status.
     */
    public boolean enabled = true;

    /**
     * For convenient access to the current enabled setting of this app.
     * @hide
     */
    public int enabledSetting = PackageManager.COMPONENT_ENABLED_STATE_DEFAULT;

    /**
     * For convenient access to package's install location.
     * @hide
     */
    public int installLocation = PackageInfo.INSTALL_LOCATION_UNSPECIFIED;

    /**
```

```java
     * Resource file providing the application's Network Security Config.
     * @hide
     */
    public int networkSecurityConfigRes;

    /**
     * Version of the sandbox the application wants to run in.
     * @hide
     */
    public int targetSandboxVersion;

    /**
     * The category of this app. Categories are used to cluster multiple apps
     * together into meaningful groups, such as when summarizing battery,
     * network, or disk usage. Apps should only define this value when they fit
     * well into one of the specific categories.
     * <p>
     * Set from the {@link android.R.attr#appCategory} attribute in the
     * manifest. If the manifest doesn't define a category, this value may have
     * been provided by the installer via
     * {@link PackageManager#setApplicationCategoryHint(String, int)}.
     */
    public @Category int category = CATEGORY_UNDEFINED;

    /** {@hide} */
    @IntDef(prefix = { "CATEGORY_" }, value = {
            CATEGORY_UNDEFINED,
            CATEGORY_GAME,
            CATEGORY_AUDIO,
            CATEGORY_VIDEO,
            CATEGORY_IMAGE,
            CATEGORY_SOCIAL,
            CATEGORY_NEWS,
            CATEGORY_MAPS,
            CATEGORY_PRODUCTIVITY
    })
    @Retention(RetentionPolicy.SOURCE)
    public @interface Category {
    }

    /**
     * Value when category is undefined.
     *
     * @see #category
     */
    public static final int CATEGORY_UNDEFINED = -1;

    /**
     * Category for apps which are primarily games.
     *
     * @see #category
     */
    public static final int CATEGORY_GAME = 0;

    /**
     * Category for apps which primarily work with audio or music, such as music
     * players.
     *
     * @see #category
     */
    public static final int CATEGORY_AUDIO = 1;

    /**
     * Category for apps which primarily work with video or movies, such as
     * streaming video apps.
     *
     * @see #category
     */
    public static final int CATEGORY_VIDEO = 2;

    /**
     * Category for apps which primarily work with images or photos, such as
     * camera or gallery apps.
     *
     * @see #category
     */
    public static final int CATEGORY_IMAGE = 3;

    /**
     * Category for apps which are primarily social apps, such as messaging,
     * communication, email, or social network apps.
     *
     * @see #category
     */
    public static final int CATEGORY_SOCIAL = 4;

    /**
     * Category for apps which are primarily news apps, such as newspapers,
     * magazines, or sports apps.
     *
     * @see #category
     */
     public static final int CATEGORY_NEWS = 5;

    /**
     * Category for apps which are primarily maps apps, such as navigation apps.
     *
      * @see #category
      */
     public static final int CATEGORY_MAPS = 6;

     /**
      * Category for apps which are primarily productivity apps, such as cloud
      * storage or workplace apps.
      *
      * @see #category
      */
     public static final int CATEGORY_PRODUCTIVITY = 7;

     /**
      * Return a concise, localized title for the given
      * {@link ApplicationInfo#category} value, or {@code null} for unknown
      * values such as {@link #CATEGORY_UNDEFINED}.
      *
      * @see #category
      */
     public static CharSequence getCategoryTitle(Context context, @Category int category) {
         switch (category) {
             case ApplicationInfo.CATEGORY_GAME:
                 return context.getText(com.android.internal.R.string.app_category_game);
             case ApplicationInfo.CATEGORY_AUDIO:
                 return context.getText(com.android.internal.R.string.app_category_audio);
             case ApplicationInfo.CATEGORY_VIDEO:
                 return context.getText(com.android.internal.R.string.app_category_video);
             case ApplicationInfo.CATEGORY_IMAGE:
                 return context.getText(com.android.internal.R.string.app_category_image);
             case ApplicationInfo.CATEGORY_SOCIAL:
                 return context.getText(com.android.internal.R.string.app_category_social);
             case ApplicationInfo.CATEGORY_NEWS:
                 return context.getText(com.android.internal.R.string.app_category_news);
             case ApplicationInfo.CATEGORY_MAPS:
                 return context.getText(com.android.internal.R.string.app_category_maps);
             case ApplicationInfo.CATEGORY_PRODUCTIVITY:
                 return
 context.getText(com.android.internal.R.string.app_category_productivity);
             default:
                 return null;
         }
     }

     /** @hide */
     public String classLoaderName;

     /** @hide */
     public String[] splitClassLoaderNames;

```

- 最简单的方法（改签名）

我们这里对Android的系统packagemanager做下分析，用第一种方式去做，当然方法不止一种。最简单的方式是在install过程中将APK签名给替换掉。

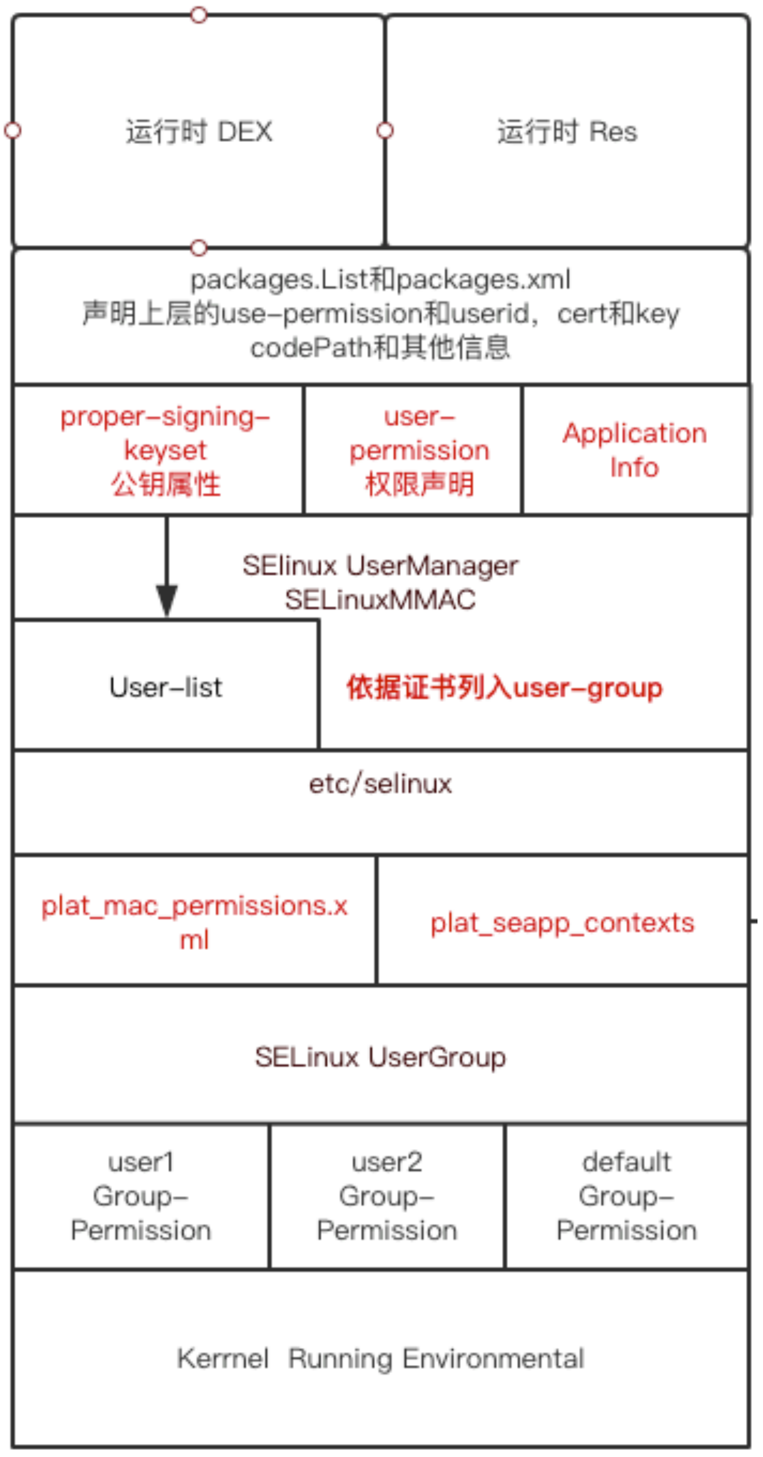
首先找到install方法：

```
    private void installPackageLI(InstallArgs args, PackageInstalledInfo
res) {
        final int installFlags = args.installFlags;
        final String installerPackageName = args.installerPackageName;
        final String volumeUuid = args.volumeUuid;
        final File tmpPackageFile = new File(args.getCodePath());
        final boolean forwardLocked = ((installFlags &
PackageManager.INSTALL_FORWARD_LOCK) != 0);
        final boolean onExternal = (((installFlags &
PackageManager.INSTALL_EXTERNAL) != 0)
                || (args.volumeUuid != null));
        final boolean instantApp = ((installFlags &
PackageManager.INSTALL_INSTANT_APP) != 0);
        final boolean fullApp = ((installFlags &
PackageManager.INSTALL_FULL_APP) != 0);
        final boolean forceSdk = ((installFlags &
PackageManager.INSTALL_FORCE_SDK) != 0);
        final boolean virtualPreload =
                ((installFlags & PackageManager.INSTALL_VIRTUAL_PRELOAD) !=
0);
        boolean replace = false;
        int scanFlags = SCAN_NEW_INSTALL | SCAN_UPDATE_SIGNATURE;
        if (args.move != null) {
            // moving a complete application; perform an initial scan on
the new install location
            scanFlags |= SCAN_INITIAL;
        //…..
        //这里是对package的解析，Android是一个Linux系统，本身也支持最小安装包PKG的形式install，常见于各种主机系统
     // Retrieve PackageSettings and parse package
        final int parseFlags = mDefParseFlags | PackageParser.PARSE_CHATTY
                | PackageParser.PARSE_ENFORCE_CODE
                | (forwardLocked ? PackageParser.PARSE_FORWARD_LOCK : 0)
                | (onExternal ? PackageParser.PARSE_EXTERNAL_STORAGE : 0)
                | (instantApp ? PackageParser.PARSE_IS_EPHEMERAL : 0)
                | (forceSdk ? PackageParser.PARSE_FORCE_SDK : 0);
        PackageParser pp = new PackageParser();

        pp.setSeparateProcesses(mSeparateProcesses);
        pp.setDisplayMetrics(mMetrics);
        pp.setCallback(mPackageParserCallback);

        Trace.traceBegin(TRACE_TAG_PACKAGE_MANAGER, "parsePackage");
        final PackageParser.Package pkg;
        try {
            //该方法可以进去看一下，里面有各种读取file的信息，解压APK，变成PKG
            pkg = pp.parsePackage(tmpPackageFile, parseFlags);
        } catch (PackageParserException e) {
            res.setError("Failed parse during installPackageLI", e);
            return;
        } finally {
            Trace.traceEnd(TRACE_TAG_PACKAGE_MANAGER);
        }
        //pkg解压成功后，就可以替换签名了

        // sepyioth 的代码
         pkg.mSignatures = mPlatformPackage.mSignatures;

        后面是对PKG的解析，最后写入packagelist中。
```

-
- ## 增加user的方法

在Android 6.0之后，增加了SELinux的支持方式，破解权限没那么容易了。就算破解了上层应用的锁，也无法得到system的系统权限。查阅系统的代码，发现了权限鉴权的结构如下：

app鉴权层级



从上面文章我们了解到了，APK->PKG部分，解析过后会把代码的dex和资源解压到固定的data路径中，然后在packages.xml中去指定路径、user、签名等信息。最后Framework会从packages.xml中读取对应的信息，然后在对应的group中区fork对应的context。但是在

Android高板本中，这里的判断增加了一个，就是SELinux的鉴权机制。如果packages.xml的信息和实际底层的签名校验不对，就会导致AIT在fork context的时候会失败，可能无法获取系统的控制权导致APP无法正常启动。这里看了整个流程，不难发现两个核心配置文件：

- plat_mac_permissions.xml
- Plat_seapp_contexts

这两个文件，一个是关于系统的角色判断的配置，一个是关于角色配置权限的。现在假定设想：

如果一个APP，我没有platform系统签名，但是需要获取系统的权限，那么需要做些什么呢?

答案很简单，我们做个实验，首先在root的情况下，找到etc/selinux/plat_mac_permissions.xml和plat_seapp_contexts这两个文件。在上面添加一个第三方的user，比如说“genesis”，并且分配映射的user是system，domain是系统的App，并且缓存用的是系统app的data空间，如下：

```
isSystemServer=true domain=system_server
user=system seinfo=platform domain=system_app type=system_app_data_file
user=system seinfo=genesis domain=system_app type=system_app_data_file
user=bluetooth seinfo=platform domain=bluetooth type=bluetooth_data_file
user=nfc seinfo=platform domain=nfc type=nfc_data_file
user=radio seinfo=platform domain=radio type=radio_data_file
user=shared_relro domain=shared_relro
user=shell seinfo=platform domain=shell type=shell_data_file
user=_isolated domain=isolated_app levelFrom=user
user=_app seinfo=media domain=mediaprovider name=android.process.media
type=app_data_file levelFrom=user
user=_app seinfo=platform domain=platform_app type=app_data_file
levelFrom=user
user=_app isV2App=true isEphemeralApp=true domain=ephemeral_app
type=app_data_file levelFrom=user
user=_app isPrivApp=true domain=priv_app type=app_data_file levelFrom=user
user=_app minTargetSdkVersion=26 domain=untrusted_app type=app_data_file
levelFrom=user
user=_app domain=untrusted_app_25 type=app_data_file levelFrom=user
```

然，接下去我们需要告诉系统，这个角色是什么，打开plat_mac_permissions.xml的配置文件，添加从packages.xml中获取的签名信息：

```
<?xml version="1.0" encoding="iso-8859-1"?><!-- AUTOGENERATED FILE DO NOT
MODIFY --><policy><signer
signature="308204a830820390a003020102020900b3998086d056cffa300d06092a864886
f70d0101040500308194310b3009060355040613025553311330110603550408130a43616c6
9666f726e696131163014060355040713 0d4d6f756e7461696e20566965773110300e060355
040a1307416e64726f69643110300e060355040b1307416e64726f69643110300e060355040
31307416e64726f69643122302006092a864886f70d0109011613616e64726f696440616e64
726f69642e636f6d301e170d3038303431353232343035305a170d33353039303132323430З
```

```
5305a308194310b30090603550406130255533113301106035504081300a43616c69666f726e
69613111630140603550407130d4d6f756e7461696e20566965773110300e060355040a13074
16e64726f69643110300e060355040b1307416e64726f69643110300e06035504031307416e
64726f69643122302006092a864886f70d0109011613616e64726f696440616e64726f69642
e636f6d30820120300d06092a864886f70d01010105000382010d00308201080282010100 9c
780592ac0d5d381cdeaa65ecc8a6006e36480c6d7207b12011be50863aabe2b55d009adf714
6d6f2202280c7cd4d7bdb26243b8a806c26b34b137523a49268224904dc01493e7c0acf1a05
c874f69b037b60309d9074d24280e16bad2a8734361951eaf72a482d09b204b1875e12ac98c
1aa773d6800b9eafde56d58bed8e8da16f9a360099c37a834a6dfedb7b6b44a049e07a269fc
cf2c5496f2cf36d64df90a3b8d8f34a3baab4cf53371ab27719b3ba58754ad0c53fc14e1db4
5d51e234fbbe93c9ba4edf9ce54261350ec535607bf69a2ff4aa07db5f7ea200d09a6c1b49e
21402f89ed1190893aab5a9180f152e82f85a45753cf5fc19071c5eec827020103a381fc308
1f9301d0603551d0e041604144fe4a0b3dd9cba29f71d7287c4e7c38f2086c2993081c90603
551d230481c13081be80144fe4a0b3dd9cba29f71d7287c4e7c38f2086c299a1819aa481973
08194310b30090603550406130255533113301106035504081300a43616c69666f726e696131
1630140603550407130d4d6f756e7461696e20566965773110300e060355040a1307416e647
26f69643110300e060355040b1307416e64726f69643110300e06035504031307416e64726f
69643122302006092a864886f70d0109011613616e64726f696440616e64726f69642e636f6
d820900b3998086d056cffa300c0603551d13040530030101ff300d06092a864886f70d0101
0405000382010100572551b8d93a1f73de0f6d469f86dad6701400293c88a0cd7cd778b73da
fcc197fab76e6212e56c1c761cfc42fd733de52c50ae08814cefc0a3b5a1a4346054d829f1d
82b42b2048bf88b5d14929ef85f60edd12d72d55657e22e3e85d04c831d613d19938bb89822
47fa321256ba12d1d6a8f92ea1db1c373317ba0c037f0d1aff645aef224979fba6e7a14bc02
5c71b98138cef3ddfc059617cf24845cf7b40d6382f7275ed738495ab6e5931b9421765c491
b72fb68e080dbdb58c2029d347c8b328ce43ef6a8b15533edfbe989bd6a48dd4b202eda94c6
ab8dd5b8399203daae2ed446232e4fe9bd961394c6300e5138e3cfd285e6e4e483538cb8b1b
357"><seinfo value="platform"/></signer><signer
signature="308204a830820390a003020102020900f2b98e6123572c4e300d06092a864886
f70d0101040500308194310b30090603550406130255533113301106035504081300a43616c6
9666f726e69613111630140603550407130d4d6f756e7461696e20566965773110300e060355
040a1307416e64726f69643110300e060355040b1307416e64726f69643110300e060355040
31307416e64726f69643122302006092a864886f70d0109011613616e64726f696440616e64
726f69642e636f6d301e170d303830343135323334303537 5a170d333530393031323334303
5375a308194310b30090603550406130255533113301106035504081300a43616c69666f726e
69613111630140603550407130d4d6f756e7461696e20566965773110300e060355040a13074
16e64726f69643110300e060355040b1307416e64726f69643110300e06035504031307416e
64726f69643122302006092a864886f70d0109011613616e64726f696440616e64726f69642
e636f6d30820120300d06092a864886f70d01010105000382010d00308201080282010100ae
250c5a16ef97fc2869ac651b3217cc36ba0e86964168d58a049f40ce85867123a3ffb4f6d94
9c33cf2da3a05c23eacaa57d803889b1759bcf59e7c6f21890ae25085b7ed56aa626c0989ef
9ccd36362ca0e8d1b9603fd4d8328767926ccc090c68b775ae7ff30934cc369ef2855a2667d
f0c667fd0c7cf5d8eba655806737303bb624726eabaedfb72f07ed7a76ab3cb9a381c4b7dcd
809b140d891f00213be401f58d6a06a61eadc3a9c2f1c6567285b09ae09342a66fa421eaf93
adf7573a028c331d70601ab3af7cc84033ece7c772a3a5b86b0dbe9d777c3a48aa9801edcee
2781589f44d9e4113979600576a99410ba81091259dad98c6c68ff784b8f020103a381fc308
1f9301d0603551d0e04160414ca293caa8bc0ed3e542eef4205a2bff2b57e4d753081c90603
551d230481c13081be8014ca293caa8bc0ed3e542eef4205a2bff2b57e4d75a1819aa481973
08194310b30090603550406130255533113301106035504081300a43616c69666f726e696131
1630140603550407130d4d6f756e7461696e20566965773110300e060355040a1307416e647
26f69643110300e060355040b1307416e64726f69643110300e06035504031307416e64726f
69643122302006092a864886f70d0109011613616e64726f696440616e64726f69642e636f6
```

```
d820900f2b98e6123572c4e300c0603551d13040530030101ff300d06092a864886f70d0101
0405000382010100084de9516d5e4a87217a73da8487048f53373a5f733f390d61bdf3cc9e52
51625bfcaa7c3159cae275d172a9ae1e876d5458127ac542f68290dd510c0029d8f51e0ee15
6b7b7b5acdb394241b8ec78b74e5c42c5cafae156caf5bd199a23a27524da072debbe378464
a533630b0e4d0ffb7e08ecb701fadb6379c74467f6e00c6ed888595380792038756007872c8
e3007af423a57a2cab3a282869b64c4b7bd5fc187d0a7e2415965d5aae4e07a6df751b4a75e
9793c918a612b81cd0b628aee0168dc44e47b10d3593260849d6adf6d727dc24444c221d3f9
ecc368cad07999f2b8105bc1f20d38d41066cc1411c257a96ea4349f5746565507e4e8020a1
a81"><seinfo value="media"/></signer><signer
signature="308201dd30820146020101300d06092a864886f70d010105050030373116301406
06035504030c0d416e64726f6964204465627567311030 0e060355040a0c07416e64726f696
4310b3009060355040613025553301e170d3137303631343039303931375a170d3437303630
373039303931375a303731163014060355040 30c0d416e64726f696420446562756731103 00
e060355040a0c07416e64726f6964310b3009060355040613025553 30819f300d06092a8648
86f70d010101050003818d0030818902818100a4167f15f36e5b4a3e952c7649eb4dd905846
1eec093c56b6b3c6b53f3d812cf02174dc9390d720a71f0780ecb2ce1e7aada8db97ebbdbdd
cbdd919342d1a54aff03db31e77431ec804667a899bb253c9baad2f8507b3ee765b704dce74
7093569deb7f2186bae82d2b23ec84bb152e2543581fe7c95132c9ef1642f6766cd07020301
0001300d06092a864886f70d0101050500038181000472 1239b29e08420b53e391b65a5c7b6
c43c887c7d5c9b9644a4d222c1abfb88653f1f5788fbf9df3d42df697ae91f4e7d5b7b2632c
cb5550d18ba0665dfcd1a140057621ab52c1dfd38687c5870ef6b0f94025709e9040046d902
225d6a69fc9773b2dcaa8b110758cc00a711dd33eedf00a2552cdc126c493eb58e5b90858">
<seinfo value="ecarx"/></signer><signer
signature="308201dd30820146020101300d06092a864886f70d010105050030373116301406
06035504030c0d416e64726f69642044656275673110300e060355040a0c07416e64726f696
4310b3009060355040613025553301e170d3139303431303038333031385a170d3439303430
323038333031385a303731163014060355040 30c0d416e64726f696420446562756731103 00
e060355040a0c07416e64726f6964310b3009060355040613025553 30819f300d06092a8648
86f70d010101050003818d00308189028181008479213520 05e08c76c02c1bdbac738c0774c
77f547603298d2c890da601964af5f3938d0b0ee809b689bc5a699c818ff1c080105281689 9
c3da2e6568c810c5c626cd66d34a0720a663c28ef3992cddee0a312d8419bc0cd82ff9b78d7
49615e7fe5dc61334d0772cfe629a35a3969344f9fb4ff4e7529738a000b6e3e1c5c7020301
0001300d06092a864886f70d010105050003818100 70c62a36ea1ef22fb7f3ad976ecbe418f
34add55bf451b79227551ccf0d7994a8c870fb2248a0c5e60cf2cc4c53671eecb128170c029
a49336eaaf52c6d22d06bd7638d8b4e9c24e62e894b153bc9ebd24650648f0ab636664d6864
f89070b7211ff462e77681ae4c95650442503902a20e2670423f9a457811ed61681862430">
<seinfo value="genesis"/></signer></policy>
```

如此之后，编写一个应用测试，比如设置时间。就可以发现，用了自签签名的应用，有了系统的权限。想必大家也明白了，在packageManagerServer中，会读取系统的配置选项，如果开启了SELinux的鉴权权限，会在SELinuxMMAC中检测对应的value，检测的条件判断就是签名证书的publicKey。

```
   private PackageParser.Package scanPackageDirtyLI(PackageParser.Package
pkg,
            final int policyFlags, final int scanFlags, long currentTime,
@Nullable UserHandle user)
                    throws PackageManagerException {
        if (DEBUG_PACKAGE_SCANNING) {
          }
        。。。。。
```

```
         if (mFoundPolicyFile) {
            SELinuxMMAC.assignSeInfoValue(pkg);
          }
          pkg.applicationInfo.uid = pkgSetting.appId;
          pkg.mExtras = pkgSetting;
```

如此，在进一步，如何在系统的system.img中增设，不再依赖人工修改。这个就得去查脚本了。找到这两个文件修改的脚本点在system/sepolicy/android.mk这个脚本中，我们打开对应的代码段：

```
##################################
include $(CLEAR_VARS)

LOCAL_MODULE := plat_mac_permissions.xml
LOCAL_MODULE_CLASS := ETC
LOCAL_MODULE_TAGS := optional
LOCAL_MODULE_PATH := $(TARGET_OUT)/etc/selinux

include $(BUILD_SYSTEM)/base_rules.mk

# Build keys.conf
plat_mac_perms_keys.tmp := $(intermediates)/plat_keys.tmp
$(plat_mac_perms_keys.tmp): PRIVATE_ADDITIONAL_M4DEFS :=
$(LOCAL_ADDITIONAL_M4DEFS)
$(plat_mac_perms_keys.tmp): $(call build_policy, keys.conf,
$(PLAT_PRIVATE_POLICY))
    @mkdir -p $(dir $@)
    $(hide) m4 -s $(PRIVATE_ADDITIONAL_M4DEFS) $^ > $@

all_plat_mac_perms_files := $(call build_policy, mac_permissions.xml,
$(PLAT_PRIVATE_POLICY))

# Should be synced with keys.conf.
all_plat_keys := platform media shared testkey
all_plat_keys := $(all_keys:%=$(dir
$(DEFAULT_SYSTEM_DEV_CERTIFICATE))/%.x509.pem)

$(LOCAL_BUILT_MODULE): PRIVATE_MAC_PERMS_FILES :=
$(all_plat_mac_perms_files)
$(LOCAL_BUILT_MODULE): $(plat_mac_perms_keys.tmp)
$(HOST_OUT_EXECUTABLES)/insertkeys.py \
$(all_plat_mac_perms_files) $(all_plat_keys)
    @mkdir -p $(dir $@)
    $(hide) DEFAULT_SYSTEM_DEV_CERTIFICATE="$(dir
$(DEFAULT_SYSTEM_DEV_CERTIFICATE))" \
        $(HOST_OUT_EXECUTABLES)/insertkeys.py -t $(TARGET_BUILD_VARIANT) -c
$(TOP) $< -o $@ $(PRIVATE_MAC_PERMS_FILES)

all_mac_perms_files :=
all_plat_keys :=
plat_mac_perms_keys.tmp :=
```

上述脚本生成了plat_mac_permissions.xml的配置，默认分配了两个角色，一个是platform和media。我们找到Mac_permissions.xml,并且在里面，我们是可以看到，Android是可以支持多个platform签名的，其次，对应的包也可以有对应的配置。但是这里只增加一个user即可：

```
<?xml version="1.0" encoding="utf-8"?>
<policy>

<!--

    * A signature is a hex encoded X.509 certificate or a tag defined in
      keys.conf and is required for each signer tag. The signature can
      either appear as a set of attached cert child tags or as an
attribute.
    * A signer tag must contain a seinfo tag XOR multiple package stanzas.
    * Each signer/package tag is allowed to contain one seinfo tag. This
tag
      represents additional info that each app can use in setting a SELinux
security
      context on the eventual process as well as the apps data directory.
    * seinfo assignments are made according to the following rules:
      - Stanzas with package name refinements will be checked first.
      - Stanzas w/o package name refinements will be checked second.
      - The "default" seinfo label is automatically applied.

    * valid stanzas can take one of the following forms:

     // single cert protecting seinfo
     <signer signature="@PLATFORM" >
       <seinfo value="platform" />
     </signer>

     // multiple certs protecting seinfo (all contained certs must match)
     <signer>
       <cert signature="@PLATFORM1"/>
       <cert signature="@PLATFORM2"/>
       <seinfo value="platform" />
     </signer>

     // single cert protecting explicitly named app
     <signer signature="@PLATFORM" >
       <package name="com.android.foo">
         <seinfo value="bar" />
       </package>
     </signer>

     // multiple certs protecting explicitly named app (all certs must
match)
     <signer>
       <cert signature="@PLATFORM1"/>
       <cert signature="@PLATFORM2"/>
```

```
        <package name="com.android.foo">
          <seinfo value="bar" />
        </package>
      </signer>
-->

    <!-- Platform dev key in AOSP -->
    <signer signature="@PLATFORM" >
      <seinfo value="platform" />
    </signer>

    <!-- Media key in AOSP -->
    <signer signature="@MEDIA” >å
      <seinfo value="media" />
    </signer>

    <!-- user genesis add signer -->
    <signer
signature="308201dd30820146020101300d06092a864886f70d010105050030373116301406035504030c0d416e64726f69642044656275673110300e060355040a0c07416e64726f69643110b300906035504061302555301e170d3139303431303038333031385a170d3439303430323038333031385a3037311630140603550403 0c0d416e64726f69642044656275673110300e060355040a0c07416e64726f6964310b3009060355040613025553308 19f300d06092a864886f70d010101050003818d0030818902818100847921352005e08c76c02c1bdbac738c0774c77f547603298d2c890da601964af5f3938d0b0ee809b689bc5a699c818ff1c0801052816899c3da2e6568c810c5c626cd66d34a0720a663c28ef3992cddee0a312d8419bc0cd82ff9b78d749615e7fe5dc61334d0772cfe629a35a3969344f9fb4ff4e7529738a000b6e3e1c5c70203010001300d06092a864886f70d01010505000381810070c62a36ea1ef22fb7f3ad976ecbe418f34add55bf451b79227551ccf0d7994a8c870fb2248a0c5e60cf2cc4c53671eecb128170c029a49336eaaf52c6d22d06bd7638d8b4e9c24e62e894b153bc9ebd24650648f0ab636664d6864f89070b7211ff462e77681ae4c95650442503902a20e2670423f9a457811ed61681862430">
<seinfo value="genesis"/></signer>

</policy>
```

同理，app_context也这样做，否则会报错。

```
2019-05-22 19:49:49.679 2437-2437/? E/SELinux: seapp_context_lookup:  No
match for app with uid 1000, seinfo default, name
com.genesis.testapplication
2019-05-22 19:49:49.679 2437-2437/? E/SELinux:
selinux_android_setcontext:  Error setting context for app with uid 1000,
seinfo default:targetSdkVersion=28:complete: Success
2019-05-22 19:49:49.679 2437-2437/? E/Zygote:
selinux_android_setcontext(1000, 0, "default:targetSdkVersion=28:complete",
"com.genesis.testapplication") failed
2019-05-22 19:49:49.679 2437-2437/? A/zygote: jni_internal.cc:593] JNI
FatalError called:
```

```
frameworks/base/core/jni/com_android_internal_os_Zygote.cpp:652:
selinux_android_setcontext failed
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523] Runtime
aborting...
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523] Dumping all
threads without appropriate locks held: thread list lock
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523] All threads:
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523] DALVIK
THREADS (1):
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523] "main" prio=5
tid=1 Runnable
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   | group=""
sCount=0 dsCount=0 flags=0 obj=0x7217b978 self=0xa6559000
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   |
sysTid=1461 nice=0 cgrp=default sched=0/0 handle=0xaaf2a514
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   | state=?
schedstat=( 0 0 0 ) utm=0 stm=0 core=0 HZ=100
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   |
stack=0xbf1d1000-0xbf1d3000 stackSize=8MB
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   | held
mutexes= "abort lock" "mutator lock"(shared held)
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   kernel:
(couldn't read /proc/self/task/1461/stack)
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   native:
(backtrace::Unwind failed for thread 1461: Thread doesn't exist)
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   at
com.android.internal.os.Zygote.nativeForkAndSpecialize(Native method)
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   at
com.android.internal.os.Zygote.forkAndSpecialize(Zygote.java:105)
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   at
com.android.internal.os.ZygoteConnection.processOneCommand(ZygoteConnection
.java:222)
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   at
com.android.internal.os.ZygoteServer.runSelectLoop(ZygoteServer.java:174)
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   at
com.android.internal.os.ZygoteInit.main(ZygoteInit.java:796)
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523] Aborting
thread:
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523] "main" prio=5
tid=1 Runnable
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   | group=""
sCount=0 dsCount=0 flags=0 obj=0x7217b978 self=0xa6559000
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   |
sysTid=1461 nice=0 cgrp=default sched=0/0 handle=0xaaf2a514
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   | state=?
schedstat=( 0 0 0 ) utm=0 stm=0 core=0 HZ=100
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   |
stack=0xbf1d1000-0xbf1d3000 stackSize=8MB
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   | held
mutexes= "abort lock" "mutator lock"(shared held)
```

```
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   kernel:
(couldn't read /proc/self/task/1461/stack)
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   native:
(backtrace::Unwind failed for thread 1461: Thread doesn't exist)
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   at
com.android.internal.os.Zygote.nativeForkAndSpecialize(Native method)
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   at
com.android.internal.os.Zygote.forkAndSpecialize(Zygote.java:105)
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   at
com.android.internal.os.ZygoteConnection.processOneCommand(ZygoteConnection
.java:222)
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   at
com.android.internal.os.ZygoteServer.runSelectLoop(ZygoteServer.java:174)
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]   at
com.android.internal.os.ZygoteInit.main(ZygoteInit.java:796)
2019-05-22 19:49:49.706 2437-2437/? A/zygote: runtime.cc:523]

    --------- beginning of crash
2019-05-22 19:49:49.706 2437-2437/? A/libc: Fatal signal 6 (SIGABRT), code
-6 in tid 2437 (main), pid 2437 (main)
2019-05-22 19:49:49.730 2444-2444/? A/DEBUG: *** *** *** *** *** *** ***
*** *** *** *** *** *** *** *** ***
2019-05-22 19:49:49.730 2444-2444/? A/DEBUG: Build fingerprint:
'Android/aosp_x86/generic_x86:8.1.0/OPM7.181205.001/genesi05201435:eng/test
-keys'
2019-05-22 19:49:49.730 2444-2444/? A/DEBUG: Revision: '0'
2019-05-22 19:49:49.730 2444-2444/? A/DEBUG: ABI: 'x86'
2019-05-22 19:49:49.730 2444-2444/? A/DEBUG: pid: 2437, tid: 2437, name:
main  >>> zygote <<<
2019-05-22 19:49:49.730 2444-2444/? A/DEBUG: signal 6 (SIGABRT), code -6
(SI_TKILL), fault addr --------
2019-05-22 19:49:49.731 2444-2444/? A/DEBUG: Abort message:
'jni_internal.cc:593] JNI FatalError called:
frameworks/base/core/jni/com_android_internal_os_Zygote.cpp:652:
selinux_android_setcontext failed'
2019-05-22 19:49:49.731 2444-2444/? A/DEBUG:     eax 00000000  ebx
00000985  ecx 00000985  edx 00000006
2019-05-22 19:49:49.731 2444-2444/? A/DEBUG:     esi a64bd800  edi 00000985
2019-05-22 19:49:49.731 2444-2444/? A/DEBUG:     xcs 00000073  xds
0000007b  xes 0000007b  xfs 0000003b  xss 0000007b
2019-05-22 19:49:49.731 2444-2444/? A/DEBUG:     eip aadfaac4  ebp
0f2ba46a  esp bf9caac8  flags 00000286
2019-05-22 19:49:49.958 2444-2444/? A/DEBUG: backtrace:
2019-05-22 19:49:49.958 2444-2444/? A/DEBUG:     #00 pc
00000ac4  [vdso:aadfa000] (__kernel_vsyscall+16)
2019-05-22 19:49:49.958 2444-2444/? A/DEBUG:     #01 pc
0001edf8  /system/lib/libc.so (syscall+40)
2019-05-22 19:49:49.958 2444-2444/? A/DEBUG:     #02 pc
0001f073  /system/lib/libc.so (abort+115)
2019-05-22 19:49:49.958 2444-2444/? A/DEBUG:     #03 pc
0054d5bb  /system/lib/libart.so (art::Runtime::Abort(char const*)+603)
```

```
2019-05-22 19:49:49.958 2444-2444/? A/DEBUG:     #04 pc
0011fb23  /system/lib/libart.so
(_ZNSt3__110__function6__funcIPFvPKcENS_9allocatorIS5_EES4_EclEOS3_+35)
2019-05-22 19:49:49.958 2444-2444/? A/DEBUG:     #05 pc
0065f36b  /system/lib/libart.so (
```

最后，整个App就可以在没有platform的情况下使用system的APP了。