系统权限

本文内容

关键类

- `Manifest.permission`
- `Manifest.permission_group`

另请参阅

- 使用系统权限

设计模式
权限

视频
Google I/O 2015 - Android M 权限：开发者最佳做法

Android 是一个权限分隔的操作系统，其中每个应用都有其独特的系统标识（Linux 用户 ID 和组 ID）。系统各部分也分隔为不同的标识。Linux 据此将不同的应用以及应用与系统分隔开来。

其他更详细的安全功能通过“权限”机制提供，此机制会限制特定进程可以执行的具体操作，并且根据 URI 权限授权临时访问特定的数据段。

本文档介绍应用开发者可以如何使用 Android 提供的安全功能。一般性的 Android 安全性概览 在“Android 开源项目”中提供。

安全架构

Android 安全架构的中心设计点是：在默认情况下任何应用都没有权限执行对其他应用、操作系统或用户有不利影响的任何操作。这包括读取或写入用户的私有数据（例如联系人或电子邮件）、读取或写入其他应用程序的文件、执行网络访问、使设备保持唤醒状态等。

由于每个 Android 应用都是在进程沙盒中运行，因此应用必须显式共享资源和数据。它们的方法是声明需要哪些权限来获取基本沙盒未提供的额外功能。应用以静态方式声明它们需要的权限，然后 Android 系统提示用户同意。

应用沙盒不依赖用于开发应用的技术。特别是，Dalvik VM 不是安全边界，任何应用都可运行原生代码（请参阅 Android NDK）。各类应用 — Java、原生和混合 — 以同样的方式放在沙盒中，彼此采用相同程度的安全防护。

应用签署

所有 APK（.apk 文件）都必须使用证书签署，其私钥由开发者持有。此证书用于识别应用的作者。证书不需要由证书颁发机构签署；Android 应用在理想情况下可以而且通常也是使用自签名证书。证书在 Android 中的作用是识别应用的作者。这允许系统授予或拒绝应用对签名级权限的访问，以及授予或拒绝应用获得与另一应用相同的 Linux 身份的请求。

用户 ID 和文件访问

在安装时，Android 为每个软件包提供唯一的 Linux 用户 ID。此 ID 在软件包在该设备上的使用寿命期间保持不变。在不同设备上，相同软件包可能有不同的 UID；重要的是每个软件包在指定设备上的 UID 是唯一的。

由于在进程级实施安全性，因此任何两个软件包的代码通常都不能在同一进程中运行，因为它们需要作为不同的 Linux 用户运行。您可以在每个软件包的 AndroidManifest.xml 的 manifest 标记中使用 sharedUserId 属性，为它们分配相同的用户 ID。这样做以后，出于安全目的，两个软件包将被视为同一个应用，具有相同的用户 ID 和文件权限。请注意，为保持安全性，只有两个签署了相同签名（并且请求相同的 sharedUserId）的应用才被分配同一用户 ID。

应用存储的任何数据都会被分配该应用的用户 ID，并且其他软件包通常无法访问这些数据。使用 getSharedPreferences(String, int)、openFileOutput(String, int) 或 openOrCreateDatabase(String, int, SQLiteDatabase.CursorFactory) 创建新文件时，可以使用 MODE_WORLD_READABLE 和/或 MODE_WORLD_WRITEABLE 标记允许任何其他软件包读取/写入文件。设置这些标记时，文件仍归您的应用所有，但其全局读取和/或写入权限已适当设置，使任何其他应用都可看见它。

使用权限

基本 Android 应用默认情况下未关联权限，这意味着它无法执行对用户体验或设备上任何数据产生不利影响的任何操作。要利用受保护的设备功能，必须在应用清单中包含一个或多个 <uses-permission> 标记。

例如，需要监控传入的短信的应用要指定：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.app.myapp" >
    <uses-permission android:name="android.permission.RECEIVE_SMS" />
    ...
</manifest>
```

如果您的应用在其清单中列出*正常*权限（即，不会对用户隐私或设备操作造成很大风险的权限），系统会自动授予这些权限。如果您的应用在其清单中列出*危险*权限（即，可能影响用户隐私或设备正常操作的权限），系统会要求用户明确授予这些权限。Android 发出请求的方式取决于系统版本，而系统版本是应用的目标：

> **权限级别**
>
> 如需了解有关不同保护级别权限的详细信息，请参阅正常权限和危险权限。

- 如果设备运行的是 Android 6.0（API 级别 23）或更高版本，并且应用的 targetSdkVersion 是 23 或更高版本，则应用在运行时向用户请求权限。用户可随时调用权限，因此应用在每次运行时均需检查自身是否具备所需的权限。如需了解有关在应用中请求权限的详细信息，请参阅使用系统权限培训指南。
- 如果设备运行的是 Android 5.1（API 级别 22）或更低版本，并且应用的 targetSdkVersion 是 22 或更低版本，则系统会在用户安装应用时要求用户授予权限。如果将新权限添加到更新的应用版本，系统会在用户更新应用时要求授予该权限。用户一旦安装应用，他们撤销权限的唯一方式是卸载应用。

通常，权限失效会导致 SecurityException 被扔回应用。但不能保证每个地方都是这样。例如，sendBroadcast(Intent) 方法在数据传递到每个接收者时会检查权限，在方法调用返回后，即使权限失效，您也不会收到异常。但在几乎所有情况下，权限失效会记入系统日志。

Android 系统提供的权限请参阅 Manifest.permission。此外，任何应用都可定义并实施自己的权限，因此这不是所有可能权限的详尽列表。

可能在程序运行期间的多个位置实施特定权限：

- 在调用系统时，防止应用执行某些功能。
- 在启动 Activity 时，防止应用启动其他应用的 Activity。
- 在发送和接收广播时，控制谁可以接收您的广播，谁可以向您发送广播。

- 在访问和操作内容提供程序时。
- 绑定至服务或启动服务。

自动权限调整

随着时间的推移，平台中可能会加入新的限制，要想使用特定 API，您的应用可能必须请求之前不需要的权限。因为现有应用假设可随意获取这些 API 应用的访问权限，所以 Android 可能会将新的权限请求应用到应用清单，以免在新平台版本上中断应用。Android 将根据为 `targetSdkVersion` 属性提供的值决定应用是否需要权限。如果该值低于在其中添加权限的版本，则 Android 会添加该权限。

例如，API 级别 4 中加入了 `WRITE_EXTERNAL_STORAGE` 权限，用以限制访问共享存储空间。如果您的 `targetSdkVersion` 为 3 或更低版本，则会向更新 Android 版本设备上的应用添加此权限。

> **注意**：如果某权限自动添加到应用，则即使您的应用可能实际并不需要这些附加权限，Google Play 上的应用列表也会列出它们。

为避免这种情况，并且删除您不需要的默认权限，请始终将 `targetSdkVersion` 更新至最高版本。可在 `Build.VERSION_CODES` 文档中查看各版本添加的权限。

正常权限和危险权限

系统权限分为几个保护级别。需要了解的两个最重要保护级别是*正常*权限和*危险*权限：

- *正常*权限涵盖应用需要访问其沙盒外部数据或资源，但对用户隐私或其他应用操作风险很小的区域。例如，设置时区的权限就是正常权限。如果应用声明其需要正常权限，系统会自动向应用授予该权限。如需当前正常权限的完整列表，请参阅正常权限。
- 危险权限涵盖应用需要涉及用户隐私信息的数据或资源，或者可能对用户存储的数据或其他应用的操作产生影响的区域。例如，能够读取用户的联系人属于危险权限。如果应用声明其需要危险权限，则用户必须明确向应用授予该权限。

权限组

所有危险的 Android 系统权限都属于权限组。如果设备运行的是 Android 6.0（API 级别 23），并且应用的 `targetSdkVersion` 是 23 或更高版本，则当用户请求危险权限时系统会发生以下行为：

- 如果应用请求其清单中列出的危险权限，而应用目前在权限组中没有任何权限，则系统会向用户显示一个对话框，描述应用要访问的权限组。对话框不描述该组内的具体权限。例如，如果应用请求 `READ_CONTACTS` 权限，系统对话框只说明该应用需要访问设备的联系信息。如果用户批准，系统将向应用授予其请求的权限。
- 如果应用请求其清单中列出的危险权限，而应用在同一权限组中已有另一项危险权限，则系统会立即授予该权限，而无需与用户进行任何交互。例如，如果某应用已经请求并且被授予了 `READ_CONTACTS` 权限，然后它又请求 `WRITE_CONTACTS`，系统将立即授予该权限。

> **特殊权限**
>
> 有许多权限其行为方式与正常权限及危险权限都不同。`SYSTEM_ALERT_WINDOW` 和 `WRITE_SETTINGS` 特别敏感，因此大多数应用不应该使用它们。如果某应用需要其中一种权限，必须在清单中声明该权限，并且发送请求用户授权的 intent。系统将向用户显示详细管理屏幕，以响应该 intent。
>
> 如需了解有关如何请求这些权限的详情，请参阅 `SYSTEM_ALERT_WINDOW` 和 `WRITE_SETTINGS` 参考条目。

> 任何权限都可属于一个权限组，包括正常权限和应用定义的权限。但权限组仅当权限危险时才影响用户体验。可以忽略正常权限的权限组。

如果设备运行的是 Android 5.1（API 级别 22）或更低版本，并且应用的 `targetSdkVersion` 是 22 或更低版本，则系统会在安装时要求用户授予权限。再次强调，系统只告诉用户应用需要的权限*组*，而不告知具体权限。

表 1. 危险权限和权限组。

权限组	权限
`CALENDAR`	• `READ_CALENDAR` • `WRITE_CALENDAR`
`CAMERA`	• `CAMERA`
`CONTACTS`	• `READ_CONTACTS`

	• WRITE_CONTACTS • GET_ACCOUNTS
LOCATION	• ACCESS_FINE_LOCATION • ACCESS_COARSE_LOCATION
MICROPHONE	• RECORD_AUDIO
PHONE	• READ_PHONE_STATE • CALL_PHONE • READ_CALL_LOG • WRITE_CALL_LOG • ADD_VOICEMAIL • USE_SIP • PROCESS_OUTGOING_CALLS
SENSORS	• BODY_SENSORS
SMS	• SEND_SMS • RECEIVE_SMS • READ_SMS • RECEIVE_WAP_PUSH • RECEIVE_MMS
STORAGE	• READ_EXTERNAL_STORAGE • WRITE_EXTERNAL_STORAGE

定义和实施权限

要实施您自己的权限，必须先使用一个或多个 `<permission>` 元素在 `AndroidManifest.xml` 中声明它们。

例如，想要控制谁可以开始其中一个 Activity 的应用可如下所示声明此操作的权限：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapp" >
    <permission android:name="com.example.myapp.permission.DEADLY_ACTIVITY"
        android:label="@string/permlab_deadlyActivity"
        android:description="@string/permdesc_deadlyActivity"
        android:permissionGroup="android.permission-group.COST_MONEY"
        android:protectionLevel="dangerous" />
    ...
</manifest>
```

> **注**：系统不允许多个软件包使用同一名称声明权限，除非所有软件包都使用同一证书签署。如果软件包声明权限，则系统不允许用户安装具有相同权限名称的其他软件包，除非这些软件包使用与第一个软件包相同的证书签署。为避免命名冲突，建议对自定义权限使用相反域名样式命名，例如 `com.example.myapp.ENGAGE_HYPERSPACE`。

`protectionLevel` 属性是必要属性，用于指示系统如何向用户告知需要权限的应用，或者谁可以拥有该权限，具体如链接的文档中所述。

`android:permissionGroup` 属性是可选属性，只是用于帮助系统向用户显示权限。大多数情况下，您要将此设为标准系统组（列在 `android.Manifest.permission_group` 中），但您也可以自己定义一个组。建议使用现有的组，因为这样可简化向用户显示的权限 UI。

需要为权限提供标签和描述。这些是用户在查看权限列表（`android:label`）或单一权限详细信息（`android:description`）时可以看到的字符串资源。标签应简短；用几个词描述权限保护的功能的关键部分。描述应该用几个句子描述权限允许持有人执行的操作。我们的约定是用两个句子描述：第一句描述权限，第二句向用户提醒为应用授予权限后可能出现的错误类型。

下面是 CALL_PHONE 权限的标签和描述示例：

```
<string name="permlab_callPhone">directly call phone numbers</string>
<string name="permdesc_callPhone">Allows the application to call
    phone numbers without your intervention. Malicious applications may
    cause unexpected calls on your phone bill. Note that this does not
    allow the application to call emergency numbers.</string>
```

您可以使用 Settings 应用和 shell 命令 `adb shell pm list permissions` 查看系统中当前定义的权限。要使用 Settings 应用，请转到 **Settings** > **Applications**。选择一个应用并向下滚动查看该应用使用的权限。对于开发者，adb '-s' 选项以类似于用户将会看到的形式显示权限：

```
$ adb shell pm list permissions -s
All Permissions:

Network communication: view Wi-Fi state, create Bluetooth connections, full
Internet access, view network state

Your location: access extra location provider commands, fine (GPS) location,
mock location sources for testing, coarse (network-based) location

Services that cost you money: send SMS messages, directly call phone numbers

...
```

自定义权限建议

应用可以定义自己的自定义权限，并通过定义 `<uses-permission>` 元素请求其他应用的自定义权限。不过，您应该仔细评估您的应用是否有必要这样做。

- 如果要设计一套向彼此显示功能的应用，请尽可能将应用设计为每个权限只定义一次。如果所有应用并非使用同一证书签署，则必须这样做。即使所有应用使用同一证书签署，最佳做法也是每个权限只定义一次。
- 如果功能仅适用于使用与提供应用相同的签名所签署的应用，您可能可以使用签名检查避免定义自定义权限。当一个应用向另一个应用发出请求时，第二个应用可在遵从该请求之前验证这两个应用是否使用同一证书签署。
- 如果您要开发一套只在您自己的设备上运行的应用，则应开发并安装管理该套件中所有应用权限的软件包。此软件包本身无需提供任何服务。它只是声明所有权限，然后套件中的其他应用通过 `<uses-permission>` 元素请求这些权限。

实施 AndroidManifest.xml 中的权限

您可以通过 `AndroidManifest.xml` 应用高级权限，限制访问系统或应用的全部组件。要执行此操作，在所需的组件上包含 `android:permission` 属性，为用于控制访问它的权限命名。

`Activity` 权限（应用于 `<activity>` 标记）限制谁可以启动相关的 Activity。在 `Context.startActivity()` 和 `Activity.startActivityForResult()` 时会检查权限；如果调用方没有所需的权限，则调用会抛出 `SecurityException`。

`Service` 权限（应用于 `<service>` 标记）限制谁可以启动或绑定到相关的服务。在 `Context.startService()`、`Context.stopService()` 和 `Context.bindService()` 时会检查权限；如果调用方没有所需的权限，则调用会抛出 `SecurityException`。

`BroadcastReceiver` 权限（应用于 `<receiver>` 标记）限制谁可以发送广播给相关的接收方。在 `Context.sendBroadcast()` 返回后检查权限，因为系统会尝试将提交的广播传递到指定的接收方。因此，权限失效不会导致向调用方抛回异常；只是不会传递该 intent。同样，可以向 `Context.registerReceiver()` 提供权限来控制谁可以广播到以编程方式注册的接收方。另一方面，可以在调用 `Context.sendBroadcast()` 时提供权限来限制允许哪些 BroadcastReceiver 对象接收广播（请参阅下文）。

`ContentProvider` 权限（应用于 `<provider>` 标记）限制谁可以访问 `ContentProvider` 中的数据。（内容提供程序有重要的附加安全工具可用，称为 URI 权限，将在后面介绍。）与其他组件不同，您可以设置两个单独的权限属性：`android:readPermission` 限制谁可以读取提供程序，`android:writePermission` 限制谁可以写入提供程序。请注意，如果提供程序有读取和写入权限保护，仅拥有写入权限并不表示您可以读取提供程序。第一次检索提供程序时将会检查权限（如果没有任何权限，将会抛出 SecurityException），对提供程序执行操作时也会检查权限。使用 `ContentResolver.query()` 需要拥有读取权限；使用 `ContentResolver.insert()`、`ContentResolver.update()`、`ContentResolver.delete()` 需要写入权限。在所有这些情况下，没有所需

的权限将导致调用抛出 `SecurityException`。

发送广播时实施权限

除了实施谁可以向注册的 `BroadcastReceiver` 发送 intent 的权限（如上所述），您还可以指定在发送广播时需要的权限。通过使用权限字符串调用 `Context.sendBroadcast()`，您可以要求接收方的应用必须拥有该权限才可接收您的广播。

请注意，接收者和广播者可能需要权限。此时，这两项权限检查都必须通过后方可将 intent 传递到相关的目标。

其他权限实施

可对任何服务调用实施任意细化的权限。这可通过 `Context.checkCallingPermission()` 方法完成。使用所需的权限字符串调用，它将返回一个整数，表示权限是否已授予当前的调用进程。请注意，仅在执行从另一个进程传入的调用（通常是通过从服务发布的 IDL 界面或者指定给另一进程的某种其他方式完成）时才可使用此方法。

检查权限还有许多其他有用的方法。如果您有另一个进程的 pid，可以使用 Context 方法 `Context.checkPermission(String, int, int)` 检查针对该 pid 的权限。如果您有另一个应用的软件包名称，可以使用直接的 PackageManager 方法 `PackageManager.checkPermission(String, String)` 了解是否已为特定软件包授予特定权限。

URI 权限

到目前为止所述的是标准权限系统，内容提供程序仅仅使用此系统通常是不够的。内容提供程序可能需要通过读取和写入权限保护自己，而其直接客户端也需要将特定 URI 传给其他应用以便于它们运行。邮件应用中的附件是一个典型的示例。应通过权限保护对邮件的访问，因为这是敏感的用户数据。但是，如果将图像附件的 URI 提供给图像查看程序，该图像查看程序不会有打开附件的权限，因为它没有理由拥有访问所有电子邮件的权限。

此问题的解决方法是采用 per-URI 权限机制：在启动 Activity 或返回结果给 Activity 时，调用方可以设置 `Intent.FLAG_GRANT_READ_URI_PERMISSION` 和/或 `Intent.FLAG_GRANT_WRITE_URI_PERMISSION`。这将授予接收 Activity 权限访问 intent 中的特定数据 URI，而不管它是否具有访问 intent 对应的内容提供程序中数据的任何权限。

此机制支持常见的能力式模型，其中用户交互（打开附件、从列表中选择联系人等）驱动临时授予细化的权限。这是一项关键功能，可将应用所需的权限缩小至只与其行为直接相关的权限。

但授予细化的 URI 权限需要与拥有这些 URI 的内容提供程序进行一定的合作。强烈建议内容提供程序实施此功能，并且通过 `android:grantUriPermissions` 属性或 `<grant-uri-permissions>` 标记声明支持此功能。

在 `Context.grantUriPermission()`、`Context.revokeUriPermission()` 和 `Context.checkUriPermission()` 方法中可以找到更多信息。

继续阅读以下方面的内容：

隐含功能要求的权限

有关如何请求某些权限的信息会隐式将您的应用限制于包含相应硬件或软件功能的设备。

`<uses-permission>`

声明应用所需系统权限的清单标记的 API 参考。

`Manifest.permission`

所有系统权限的 API 参考。

您可能还对以下内容感兴趣：

设备兼容性

有关 Android 在不同设备类型上工作方式的信息，并介绍了如何针对不同设备优化您的应用，或如何限制您的应用在不同设备上的可

用性。

Android 安全性概览

有关 Android 平台安全模式的详细论述。