**Seguridad y permisos**

**En este documento**

1. [Arquitectura de seguridad](http://developer.android.com/guide/topics/security/security.html#arch)
2. [Que firma la solicitud](http://developer.android.com/guide/topics/security/security.html#signing)
3. [ID de usuario y acceso a archivos](http://developer.android.com/guide/topics/security/security.html#userid)
4. [Uso de los permisos](http://developer.android.com/guide/topics/security/security.html#permissions)
5. [La declaración de Permisos y Cumplimiento](http://developer.android.com/guide/topics/security/security.html#declaring)
	1. [... En AndroidManifest.xml](http://developer.android.com/guide/topics/security/security.html#manifest)
	2. [... Al enviar transmisiones](http://developer.android.com/guide/topics/security/security.html#broadcasts)
	3. [Aplicación de otro permiso](http://developer.android.com/guide/topics/security/security.html#enforcement)
6. [URI Permisos](http://developer.android.com/guide/topics/security/security.html#uri)

Este documento describe cómo los desarrolladores de aplicaciones pueden utilizar las características de seguridad proporcionadas por Android. Una más general [para Android Introducción a la seguridad](http://source.android.com/tech/security/index.html) se presenta en el Proyecto de Android Open Source.

Android es un sistema operativo de privilegio, separados, en el que cada aplicación se ejecuta con una identidad distinta del sistema (Linux ID de usuario y la ID de grupo). Las partes del sistema también están separados en identidades distintas. Linux lo aísla las aplicaciones entre sí y desde el sistema.

Adicionales de grano más fino características de seguridad se proporcionan a través de un "permiso" mecanismo que impone restricciones a las operaciones específicas que un proceso en particular puede llevar a cabo y los permisos por URI para la concesión ad-hoc el acceso a elementos específicos de datos.

**Arquitectura de seguridad**

Un punto central del diseño de la arquitectura de seguridad de Android es que no hay ninguna aplicación, por defecto, tiene permiso para realizar cualquier operación que tendrá impactos negativos en otras aplicaciones, el sistema operativo, o el usuario. Esto incluye leer o escribir los datos privados del usuario (por ejemplo, los contactos o correos electrónicos), leer o escribir archivos de otra aplicación, la realización de acceso a la red, manteniendo el dispositivo despierto, etc

Debido a que cajas de arena para Android aplicaciones entre sí, las aplicaciones de manera explícita deben compartir recursos y datos. Ellos hacen esto al declarar que los *permisos* que necesitan para capacidades adicionales no previstos por el recinto de seguridad básica. Aplicaciones estáticamente declarar los permisos que necesitan, y el sistema Android le pide al usuario su consentimiento a la vez que se instala la aplicación. Android no tiene ningún mecanismo para la concesión de permisos de forma dinámica (en tiempo de ejecución), ya que complica la experiencia del usuario en detrimento de la seguridad.

El recinto de seguridad de aplicación no depende de la tecnología utilizada para crear una aplicación. En particular, la máquina virtual Dalvik no es un límite de seguridad, y cualquier aplicación puede ejecutar código nativo (véase [el NDK Android](http://developer.android.com/sdk/ndk/index.html) ). Todos los tipos de aplicaciones - Java, nativa, y híbrido - un recinto de seguridad están en la misma forma y tienen el mismo grado de seguridad de la otra.

**Que firma la solicitud**

Todas las aplicaciones de Android (. Apk archivos) debe estar firmado con un certificado cuya clave privada se lleva a cabo por su desarrollador. Este certificado identifica al autor de la solicitud. El certificado *no* tiene que ser firmado por una autoridad de certificación: es perfectamente admisible, y típico, para aplicaciones de Android para usar certificados firmados. El propósito de los certificados en Android es para distinguir los autores de aplicaciones. Esto permite que el sistema de conceder o denegar el acceso a las aplicaciones de [permisos a nivel de firmas](http://developer.android.com/guide/topics/manifest/permission-element.html#plevel) y para conceder o denegar una solicitud de [petición de ser dada la identidad misma de Linux](http://developer.android.com/guide/topics/manifest/manifest-element.html#uid) como otra aplicación.

**ID de usuario y acceso a archivos**

Al momento de la instalación, Android le da a cada paquete de un ID de usuario de Linux distinta. La identidad se mantiene constante durante la duración de vida del paquete en ese dispositivo. En un dispositivo diferente, el mismo paquete puede tener un UID diferente, lo que importa es que cada paquete tiene un UID distinto en un determinado dispositivo.

Debido a la aplicación de seguridad que sucede en el nivel de proceso, el código de cualquiera de los dos paquetes que normalmente no se puede ejecutar en el mismo proceso, ya que necesitan para funcionar como diferentes usuarios de Linux. Usted puede utilizar el [sharedUserId](http://developer.android.com/reference/android/R.attr.html#sharedUserId) atributo en el AndroidManifest.xml 's [manifest](http://developer.android.com/reference/android/R.styleable.html#AndroidManifest) la etiqueta de cada paquete para hacer que le asigna el mismo ID de usuario. De esta manera, a los fines de la seguridad de los dos paquetes se tratan como siendo la misma aplicación, con el mismo ID de usuario y permisos de los archivos. Obsérvese que con el fin de conservar la seguridad, sólo dos aplicaciones firmado con la misma firma (y solicitando la sharedUserId mismo) se dará el mismo identificador de usuario.

Los datos almacenados por una aplicación que se le asignará ID de la aplicación de usuario, y normalmente no accesibles para otros paquetes. Cuando se crea un nuevo archivo con [getSharedPreferences(String, int)](http://developer.android.com/reference/android/content/Context.html#getSharedPreferences%28java.lang.String,%20int%29) , los [openFileOutput(String, int)](http://developer.android.com/reference/android/content/Context.html#openFileOutput%28java.lang.String,%20int%29) , o [openOrCreateDatabase(String, int, SQLiteDatabase.CursorFactory)](http://developer.android.com/reference/android/content/Context.html#openOrCreateDatabase%28java.lang.String,%20int,%20android.database.sqlite.SQLiteDatabase.CursorFactory%29) , puede utilizar los [MODE\_WORLD\_READABLE](http://developer.android.com/reference/android/content/Context.html#MODE_WORLD_READABLE) y / o [MODE\_WORLD\_WRITEABLE](http://developer.android.com/reference/android/content/Context.html#MODE_WORLD_WRITEABLE) banderas para permitir que cualquier otro paquete para leer / escribir el archivo. Al establecer estos indicadores, el archivo sigue siendo propiedad de su aplicación, pero su lectura global y / o permisos de escritura se han establecido de manera adecuada para cualquier otra aplicación se puede ver.

**Uso de los permisos**

Una aplicación básica de Android no tiene los permisos asociados con ella, lo que significa que no puede hacer nada de lo que tendría efectos adversos sobre la experiencia del usuario o cualquier otro dato en el dispositivo. Para hacer uso de las características protegidas del dispositivo, se debe incluir en sus AndroidManifest.xml uno o más [<uses-permission>](http://developer.android.com/reference/android/R.styleable.html#AndroidManifestUsesPermission) etiquetas que declaran los permisos que sus necesidades de aplicación.

Por ejemplo, una aplicación que necesita para controlar los mensajes SMS entrantes se especifican:

Menores xmlns: android = manifiestos "http://schemas.android.com/apk/res/android~~HEAD=NNS"

 paquete = "com.android.app.myapp">

 <Usos de permiso de android: name = "android.permission.RECEIVE\_SMS "/>

 ...

</ Manifest>

En el momento de la aplicación de instalación, los permisos solicitados por la aplicación se le concedió por el instalador de paquetes, basado en los controles contra las firmas de las aplicaciones que declaran esos permisos y / o interacción con el usuario. *No se aceptan* cheques con el usuario se hace mientras se ejecuta una aplicación: o bien se le concedió un permiso en particular cuando se instala, y se puede utilizar esta característica como se desee, o el permiso no fue concedido y cualquier intento de utilizar la característica de un error sin preguntar al usuario .

Muchas veces un fracaso permiso se traducirá en una [SecurityException](http://developer.android.com/reference/java/lang/SecurityException.html) ser lanzado de nuevo a la aplicación. Sin embargo, esto no se garantiza que se producen en todas partes. Por ejemplo, el [sendBroadcast(Intent)](http://developer.android.com/reference/android/content/Context.html#sendBroadcast%28android.content.Intent%29) método comprueba los permisos de los datos se está entregando a cada receptor, después de la llamada al método ha regresado, por lo que no recibirá una excepción si hay fallas de permisos. En casi todos los casos, sin embargo, un fallo de autorización se imprimirá en el registro del sistema.

Los permisos proporcionados por el sistema Android se puede encontrar en [Manifest.permission](http://developer.android.com/reference/android/Manifest.permission.html) . Cualquier aplicación puede definir y hacer cumplir sus propios permisos, así que esto no es una lista completa de todos los permisos posibles.

Un permiso en particular puede ser aplicado en varios lugares durante la operación de su programa:

* En el momento de una llamada en el sistema, para evitar que una aplicación de ejecución de ciertas funciones.
* Al iniciar una actividad, para evitar que las aplicaciones del lanzamiento de las actividades de otras aplicaciones.
* Tanto el envío y recepción de emisiones, para controlar quién puede recibir su difusión o quién puede enviar una transmisión para usted.
* Al acceder y operar en un proveedor de contenido.
* Unirse o iniciar un servicio.

**La declaración de Permisos y Cumplimiento**

Para hacer cumplir sus propios permisos, primero debe declarar en su AndroidManifest.xml utilizando uno o más [<permission>](http://developer.android.com/reference/android/R.styleable.html#AndroidManifestPermission) etiquetas.

Por ejemplo, una aplicación que quiere controlar quién puede iniciar una de sus actividades podría declarar un permiso para esta operación de la siguiente manera:

Menores xmlns: android = manifiestos "http://schemas.android.com/apk/res/android~~HEAD=NNS"

 paquete = "com.me.app.myapp">

 <Permiso de android: name = "com.me.app.myapp.permission.DEADLY\_ACTIVITY "

 android: label = "@ cadena / permlab\_deadlyActivity"

 android: description = "@ cadena / permdesc\_deadlyActivity"

 android: permissionGroup = "android.permission grupo.COST\_MONEY "

 android: ProtectionLevel = "peligroso" />

 ...

</manifest>

El [<protectionLevel>](http://developer.android.com/reference/android/R.styleable.html#AndroidManifestPermission_protectionLevel) atributo es necesario, diciendo al sistema cómo el usuario debe ser informado de las aplicaciones que requieren el permiso, o que pueda mantener ese permiso, como se describe en la documentación relacionada.

El [<permissionGroup>](http://developer.android.com/reference/android/R.styleable.html#AndroidManifestPermission_permissionGroup) atributo es opcional, y sólo se utiliza para ayudar a los permisos de visualización del sistema para el usuario. Por lo general, se desea establecer que se trata de un grupo con sistema estándar (incluida en el [android.Manifest.permission\_group](http://developer.android.com/reference/android/Manifest.permission_group.html) ) o en casos más raros que uno se define por sí mismo. Se prefiere utilizar un grupo existente, ya que esto simplifica la interfaz de usuario permiso se muestra al usuario.

Tenga en cuenta que tanto una etiqueta y la descripción debe ser suministrado por el permiso. Estos son recursos de cadena que se pueden mostrar al usuario cuando se está viendo una lista de permisos ( [android:label](http://developer.android.com/reference/android/R.styleable.html#AndroidManifestPermission_label) ) o detalles sobre el permiso único ( [android:description](http://developer.android.com/reference/android/R.styleable.html#AndroidManifestPermission_description) ). La etiqueta debe ser corto, unas pocas palabras que describen la pieza clave de la funcionalidad de la autorización está protegiendo. La descripción deberá ser un par de frases que describen lo que el permiso le permite al titular que hacer. Nuestra convención para la descripción es de dos frases, la primera descripción de la autorización, la segunda advertencia al usuario de lo que las cosas malas pueden suceder si una aplicación se concede el permiso.

He aquí un ejemplo de una etiqueta y una descripción para el permiso CALL\_PHONE:

 <string name="permlab\_callPhone"> llamar directamente a números de teléfono </ string>

 <string name="permdesc\_callPhone"> Permite que la aplicación llame a

 números de teléfono sin su intervención. Las aplicaciones maliciosas pueden

 hacer que las llamadas inesperadas en su factura de teléfono. Nótese que esto no hace

 permitir que la aplicación para llamar a números de emergencia. </ string>

Usted puede ver los permisos definidos en la actualidad en el sistema con la aplicación de configuración y la estructura de comandos adb shell pm list permissions . Para utilizar la aplicación Configuración, vaya a Configuración> Aplicaciones. Elija una aplicación y desplácese hacia abajo para ver los permisos que la aplicación utiliza. Para los desarrolladores, la opción del Banco Asiático de Desarrollo '-s' muestra los permisos de una forma similar a cómo el usuario va a ver:

$ Adb shell pm lista de permisos-s

Todos los Permisos:

Red de comunicación: Wi-Fi de vista del Estado, crear conexiones Bluetooth, completa

Acceso a Internet, punto de vista de la red estatal

Su ubicación: acceso a los comandos adicionales del proveedor de ubicación, muy bien (GPS),

fuentes simuladas de localización para las pruebas, gruesa (basada en la red) la ubicación

Servicios que le cuestan dinero: enviar mensajes SMS, llamar directamente a números de teléfono

...

**Permisos de ejecución, de AndroidManifest.xml**

Alto nivel de permisos que restringen el acceso a los componentes de todo el sistema o la aplicación se puede aplicar a través de su AndroidManifest.xml . Todo lo que esto requiere es que incluye un [android:permission](http://developer.android.com/reference/android/R.attr.html#permission) atributo en el componente deseado, nombrando el permiso que se utilizará para controlar el acceso a la misma.

[**Activity**](http://developer.android.com/reference/android/app/Activity.html) permisos (aplicado a la [<activity>](http://developer.android.com/reference/android/R.styleable.html#AndroidManifestActivity) etiqueta) restringir quién puede iniciar la actividad asociada. El permiso se comprueba durante [Context.startActivity()](http://developer.android.com/reference/android/content/Context.html#startActivity%28android.content.Intent%29) y [Activity.startActivityForResult()](http://developer.android.com/reference/android/app/Activity.html#startActivityForResult%28android.content.Intent,%20int%29) , si el llamador no dispone del permiso requerido a continuación [SecurityException](http://developer.android.com/reference/java/lang/SecurityException.html) se lanza de la llamada.

[**Service**](http://developer.android.com/reference/android/app/Service.html) permisos (aplicado a la [<service>](http://developer.android.com/reference/android/R.styleable.html#AndroidManifestService) etiqueta) restringir quién puede iniciar o enlazar con el servicio asociado. El permiso se comprueba durante [Context.startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) , [Context.stopService()](http://developer.android.com/reference/android/content/Context.html#stopService%28android.content.Intent%29) y [Context.bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) , si el llamador no dispone del permiso requerido a continuación [SecurityException](http://developer.android.com/reference/java/lang/SecurityException.html) se lanza de la llamada.

[**BroadcastReceiver**](http://developer.android.com/reference/android/content/BroadcastReceiver.html) permisos (aplicado a la [<receiver>](http://developer.android.com/reference/android/R.styleable.html#AndroidManifestReceiver) etiqueta) restringir quién puede enviar emisiones asociadas al receptor. El permiso se comprueba *después de* [Context.sendBroadcast()](http://developer.android.com/reference/android/content/Context.html#sendBroadcast%28android.content.Intent%29) regresa, ya que el sistema trata de ofrecer la emisión presentado al receptor dado. Como resultado, un fracaso permiso no resultará en una excepción ser lanzado de nuevo a la persona que llama, no se acaba de entregar la intención. De la misma manera, un permiso puede ser suministrado a [Context.registerReceiver()](http://developer.android.com/reference/android/content/Context.html#registerReceiver%28android.content.BroadcastReceiver,%20android.content.IntentFilter,%20java.lang.String,%20android.os.Handler%29) para controlar que puede transmitir a un receptor mediante programación registrado. En sentido contrario, un permiso puede ser suministrado al llamar [Context.sendBroadcast()](http://developer.android.com/reference/android/content/Context.html#sendBroadcast%28android.content.Intent,%20java.lang.String%29) para restringir los objetos BroadcastReceiver se les permite recibir la transmisión (ver más abajo).

[**ContentProvider**](http://developer.android.com/reference/android/content/ContentProvider.html) permisos (aplicado a la [<provider>](http://developer.android.com/reference/android/R.styleable.html#AndroidManifestProvider) etiqueta) restringir quién puede acceder a los datos en un [ContentProvider](http://developer.android.com/reference/android/content/ContentProvider.html) . (Proveedores de contenido tienen una instalación importante de seguridad adicional a disposición de los llamó [permisos URI](http://developer.android.com/guide/topics/security/security.html#uri) que se describen más adelante.) A diferencia de los otros componentes, hay dos atributos de permisos separados que se pueden establecer: [android:readPermission](http://developer.android.com/reference/android/R.attr.html#readPermission) restringe quién puede leer desde el proveedor, y [android:writePermission](http://developer.android.com/reference/android/R.attr.html#writePermission) restringe quién puede escribir en él. Tenga en cuenta que si un proveedor está protegido con un tanto de lectura y escritura permiso, sujetándolo sólo por el permiso de escritura no significa que usted puede leer de un proveedor. Los permisos se comprueban la primera vez que un proveedor de recuperar (si es que no tienen ni permiso, una SecurityException será lanzado), y como usted realizar operaciones en el proveedor. Usando [ContentResolver.query()](http://developer.android.com/reference/android/content/ContentResolver.html#query%28android.net.Uri,%20java.lang.String%5B%5D,%20java.lang.String,%20java.lang.String%5B%5D,%20java.lang.String%29) requiere la celebración de el permiso de lectura, utilizando [ContentResolver.insert()](http://developer.android.com/reference/android/content/ContentResolver.html#insert%28android.net.Uri,%20android.content.ContentValues%29) , [ContentResolver.update()](http://developer.android.com/reference/android/content/ContentResolver.html#update%28android.net.Uri,%20android.content.ContentValues,%20java.lang.String,%20java.lang.String%5B%5D%29) , [ContentResolver.delete()](http://developer.android.com/reference/android/content/ContentResolver.html#delete%28android.net.Uri,%20java.lang.String,%20java.lang.String%5B%5D%29) requiere el permiso de escritura. En todos estos casos, no la celebración de los resultados de los permisos necesarios en una [SecurityException](http://developer.android.com/reference/java/lang/SecurityException.html) ser arrojado fuera de la convocatoria.

**La aplicación de los permisos al enviar transmisiones**

Además de hacer cumplir el permiso que se puede enviar a un Intenciones registrado [BroadcastReceiver](http://developer.android.com/reference/android/content/BroadcastReceiver.html) (como se describió anteriormente), también puede especificar un permiso necesario para enviar una emisión. Al llamar a [Context.sendBroadcast()](http://developer.android.com/reference/android/content/Context.html#sendBroadcast%28android.content.Intent,%20java.lang.String%29) con una serie de permiso, se requiere que la aplicación de un receptor debe tener este permiso con el fin de recibir su difusión.

Tenga en cuenta que un receptor y un locutor puede requerir un permiso. Cuando esto sucede, ambas comprobaciones de permisos debe pasar de la intención de ser entregados al destino asociado.

**Aplicación de otro permiso**

Arbitrariamente permisos específicos se puede aplicar en cualquier llamada a un servicio. Esto se logra con la [Context.checkCallingPermission()](http://developer.android.com/reference/android/content/Context.html#checkCallingPermission%28java.lang.String%29) método. Llame con una serie de permiso deseado y se devolverá un número entero que indica si el permiso ha sido concedido en el proceso de llamadas actual. Tenga en cuenta que esto sólo se puede utilizar cuando se está ejecutando una llamada que viene de otro proceso, generalmente a través de un interface IDL publicada de un servicio o de alguna otra manera dada a otro proceso.

Hay un número de otras formas útiles para comprobar permisos. Si usted tiene el pid de otro proceso, puede utilizar el método de contexto [Context.checkPermission(String, int, int)](http://developer.android.com/reference/android/content/Context.html#checkPermission%28java.lang.String,%20int,%20int%29) para comprobar el permiso en contra de que la EPI. Si usted tiene el nombre del paquete de otra aplicación, puede utilizar el método directo PackageManager [PackageManager.checkPermission(String, String)](http://developer.android.com/reference/android/content/pm/PackageManager.html#checkPermission%28java.lang.String,%20java.lang.String%29) para saber si ese paquete en particular se ha concedido un permiso específico.

**URI Permisos**

El sistema de permisos estándar descrito hasta el momento no suele ser suficiente cuando se utiliza con los proveedores de contenido. Un proveedor de contenido puede querer protegerse a sí mismo con el de leer y escribir, mientras que sus clientes directos también tienen que entregar específica URI a otras aplicaciones para que puedan operar. Un ejemplo típico es los archivos adjuntos en una aplicación de correo. Acceso al correo electrónico deben ser protegidos por los permisos, ya que se trata de datos sensibles del usuario. Sin embargo, si un URI para un archivo adjunto de imagen se da a un visor de imágenes, visor de imágenes que no tienen permiso para abrir el archivo adjunto, ya que no tiene motivos para celebrar un permiso para acceder a todo el correo electrónico.

La solución a este problema es por URI permisos: al iniciar una actividad o devolver un resultado de una actividad, la persona que llama puede establecer [Intent.FLAG\_GRANT\_READ\_URI\_PERMISSION](http://developer.android.com/reference/android/content/Intent.html#FLAG_GRANT_READ_URI_PERMISSION) y / o [Intent.FLAG\_GRANT\_WRITE\_URI\_PERMISSION](http://developer.android.com/reference/android/content/Intent.html#FLAG_GRANT_WRITE_URI_PERMISSION) . Esto concede el permiso de actividad que reciba acceder a la URI de datos específica en la intención, independientemente de si tiene algún permiso para acceder a los datos del proveedor de contenidos que corresponden a la intención.

Este mecanismo permite una capacidad común de estilo de modelo en el que la interacción del usuario (abrir un archivo adjunto, al seleccionar un contacto de una lista, etc) impulsa ad-hoc concesión de permisos de grano fino. Esto puede ser un centro clave para la reducción de los permisos que necesita la aplicación sólo a aquellos directamente relacionados con su comportamiento.

La concesión de permisos de URI de grano fino, sin embargo, requieren algún tipo de cooperación con el proveedor de contenido de la celebración de los URI. Se recomienda que los proveedores de contenido poner en práctica este mecanismo, y declarar que lo apoyan a través del [android:grantUriPermissions](http://developer.android.com/reference/android/R.styleable.html#AndroidManifestProvider_grantUriPermissions) atributo o [<grant-uri-permissions>](http://developer.android.com/reference/android/R.styleable.html#AndroidManifestGrantUriPermission) etiqueta.

Más información se puede encontrar en la [Context.grantUriPermission()](http://developer.android.com/reference/android/content/Context.html#grantUriPermission%28java.lang.String,%20android.net.Uri,%20int%29) , [Context.revokeUriPermission()](http://developer.android.com/reference/android/content/Context.html#revokeUriPermission%28android.net.Uri,%20int%29) , y [Context.checkUriPermission()](http://developer.android.com/reference/android/content/Context.html#checkUriPermission%28android.net.Uri,%20int,%20int,%20int%29) métodos.