Curbing Android Permission Creep

Encouraging Least Privilege in development

Timothy Vidas

Nicolas Christin

Lorrie Faith Cranor

W2SP 2011

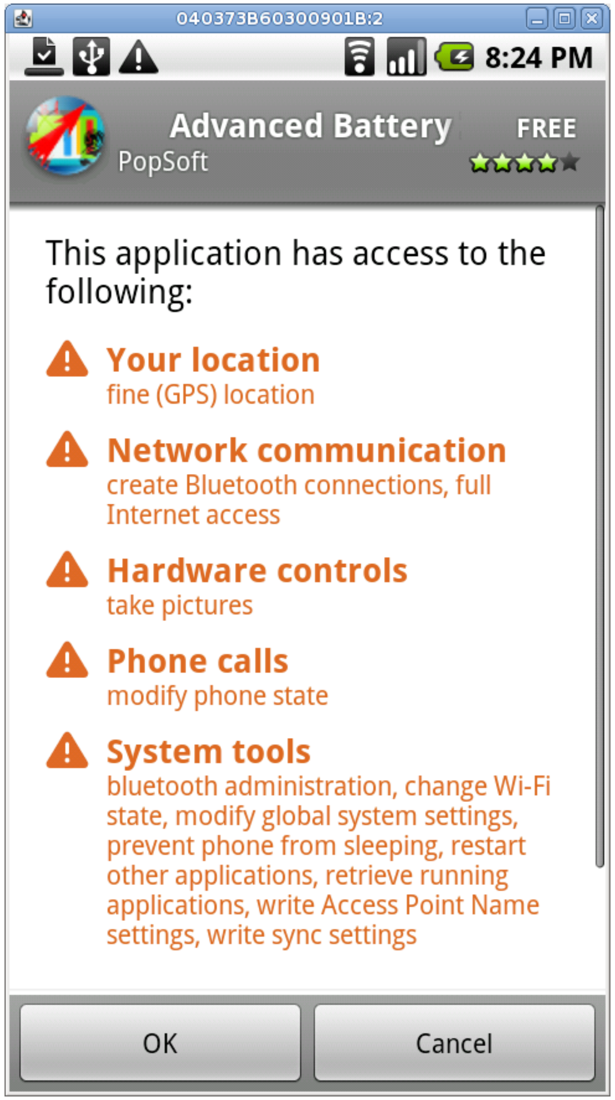
Carnegie Mellon

CyLab Usable Privacy and Security Laboratory

http://cups.cs.cmu.edu/

Motivation

- Developers specify required permissions
- Applications must be granted permissions at install time by the user
  - it's "all or nothing"
- Why does a Battery app require internet access?



This application has access to the following:

Your location
fine (GPS) location

Network communication
create Bluetooth connections, full Internet access

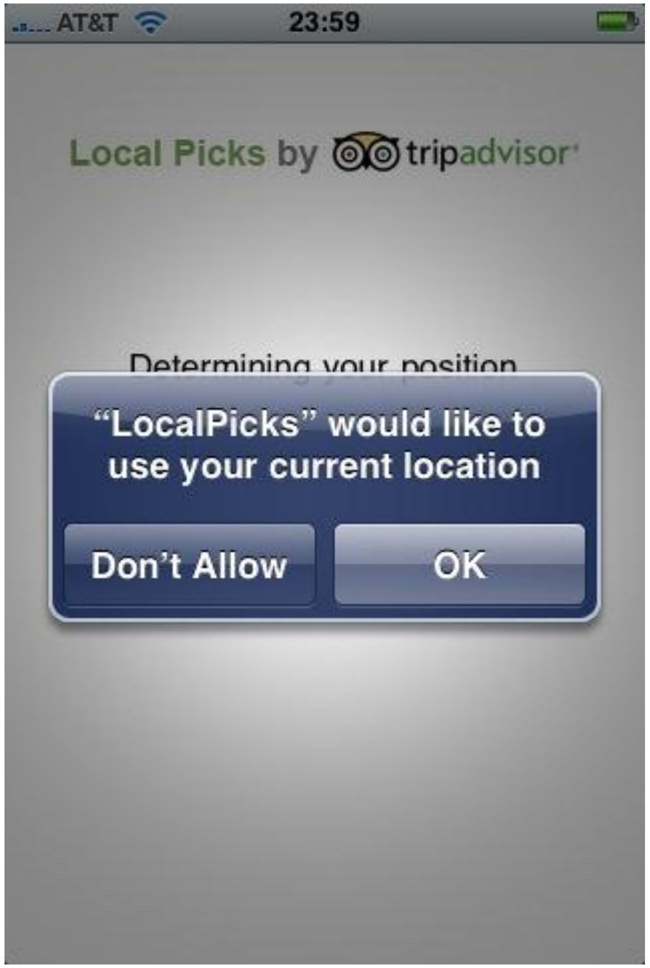
Hardware controls
take pictures

Phone calls
modify phone state

System tools
bluetooth administration, change Wi-Fi state, modify global system settings, prevent phone from sleeping, restart other applications, retrieve running applications, write Access Point Name settings, write sync settings

Motivation

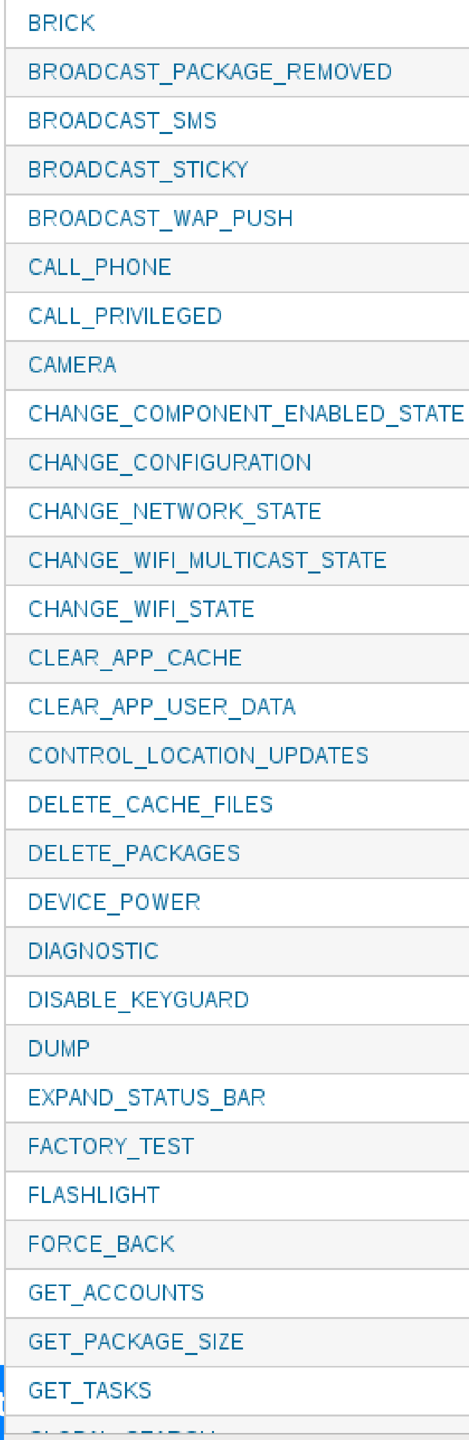
- Unlike the iPhone's model, users are not prompted when restricted objects are accessed
- If an app attempts access w/o having already been granted a permission, it crashes



"LocalPicks" would like to use your current location

Android Permissions

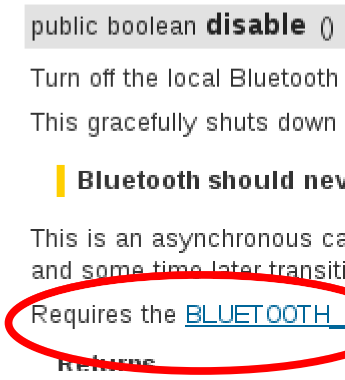
- More than 100 application level permissions govern access to resources
  - Some are very broad:
    - INTERNET
  - Some are very specific
    - READ_SMS
  - Some are very obtuse
    - ACCESS_SURFACE_FLINGER
  - Some are very generic / ambiguous
    - DIAGNOSTIC



BRICK
BROADCAST_PACKAGE_REMOVED
BROADCAST_SMS
BROADCAST_STICKY
BROADCAST_WAP_PUSH
CALL_PHONE
CALL_PRIVILEGED
CAMERA
CHANGE_COMPONENT_ENABLED_STATE
CHANGE_CONFIGURATION
CHANGE_NETWORK_STATE
CHANGE_WIFI_MULTICAST_STATE
CHANGE_WIFI_STATE
CLEAR_APP_CACHE
CLEAR_APP_USER_DATA
CONTROL_LOCATION_UPDATES
DELETE_CACHE_FILES
DELETE_PACKAGES
DEVICE_POWER
DIAGNOSTIC
DISABLE_KEYGUARD
DUMP
EXPAND_STATUS_BAR
FACTORY_TEST
FLASHLIGHT
FORCE_BACK
GET_ACCOUNTS
GET_PACKAGE_SIZE
GET_TASKS

Android Permissions

- As a developer, knowing when to use them is not always clear
- API documentation has some guidance
- Source provides some
- Debugger…



public boolean **disable** ()

Turn off the local Bluetooth

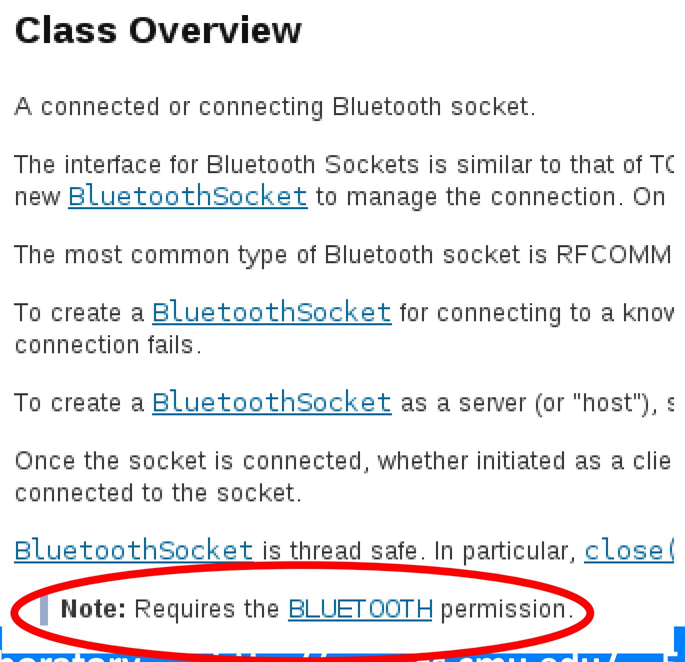
This gracefully shuts down

Bluetooth should nev

This is an asynchronous ca
and some time later transiti

Requires the BLUETOOTH

Returns



Class Overview

A connected or connecting Bluetooth socket.

The interface for Bluetooth Sockets is similar to that of TC
new BluetoothSocket to manage the connection. On

The most common type of Bluetooth socket is RFCOMM

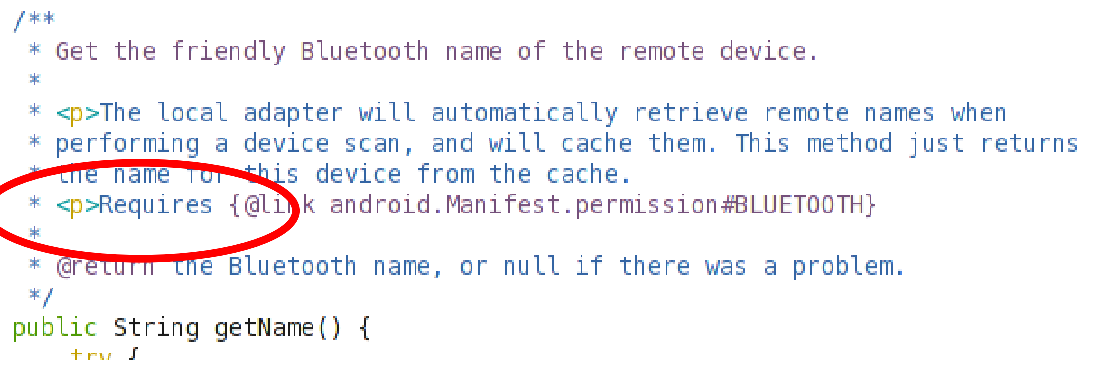
To create a BluetoothSocket for connecting to a know
connection fails.

To create a BluetoothSocket as a server (or "host"), s

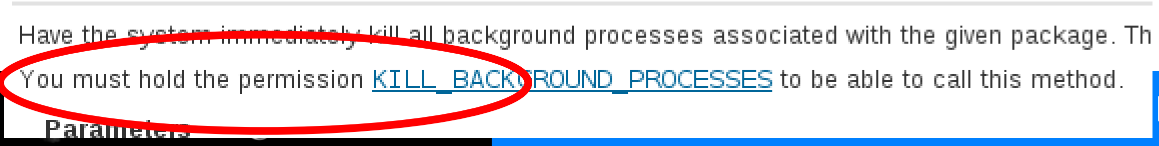
Once the socket is connected, whether initiated as a clie
connected to the socket.

BluetoothSocket is thread safe. In particular, close(

Note: Requires the BLUETOOTH permission.



```
/**
 * Get the friendly Bluetooth name of the remote device.
 *
 * <p>The local adapter will automatically retrieve remote names when
 * performing a device scan, and will cache them. This method just returns
 * the name for this device from the cache.
 * <p>Requires {@link android.Manifest.permission#BLUETOOTH}
 *
 * @return the Bluetooth name, or null if there was a problem.
 */
public String getName() {
    try {
```



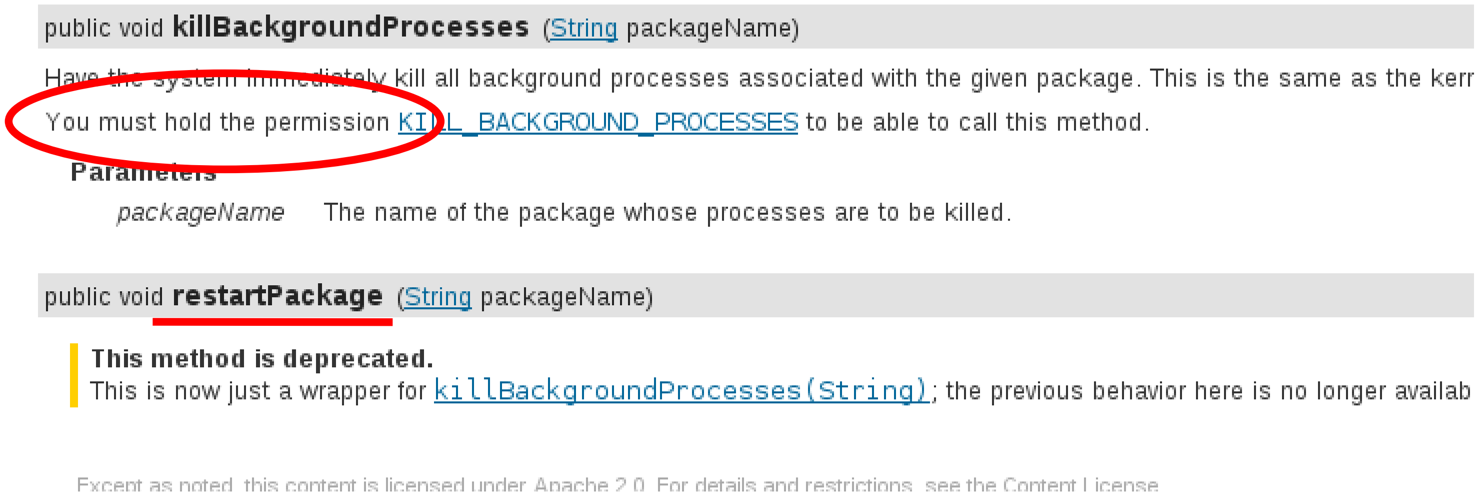
Have the system immediately kill all background processes associated with the given package. Th

You must hold the permission KILL_BACKGROUND_PROCESSES to be able to call this method.

Parameters

Android Permissions

- What permission does restartPackage() need?



public void **killBackgroundProcesses** (String packageName)

Have the system immediately kill all background processes associated with the given package. This is the same as the kerr

You must hold the permission KILL_BACKGROUND_PROCESSES to be able to call this method.

Parameters

packageName The name of the package whose processes are to be killed.

public void **restartPackage** (String packageName)

This method is deprecated.
This is now just a wrapper for `killBackgroundProcesses(String)`; the previous behavior here is no longer availab

Except as noted, this content is licensed under Apache 2.0. For details and restrictions, see the Content License.

killMethod = ActivityManager.class.getMethod("killBackgroundProcesses", String.class);

- What does killMethod() need?

Further Motivation

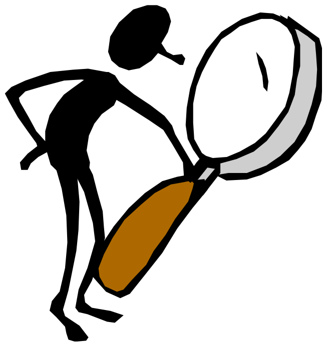
- Downloaded ~34,000 applications from the Android Market
  – More than 4% contained *duplicate* permission specifications
  – Every category contained some applications that contained duplicates

Note: these applications are *packaged applications* that typically have no associated source code

Android Development

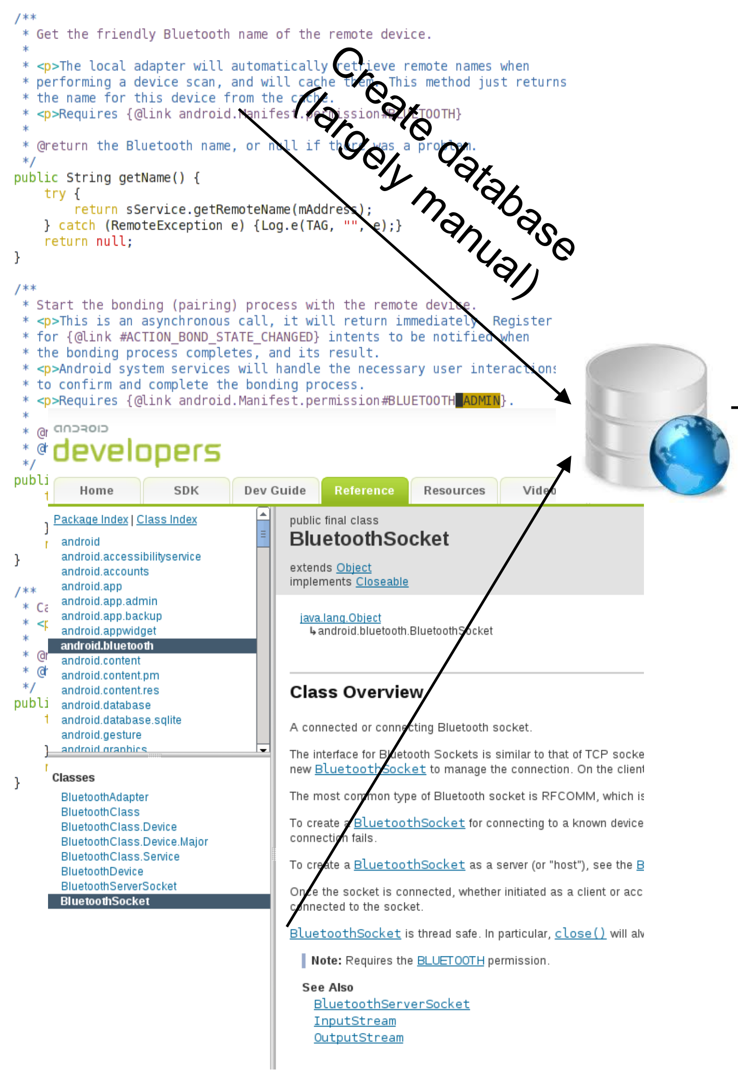
- Android Development Tools (ADT) is an Eclipse extension that facilitates development
  - Developers can choose to not use an IDE, but ADT integrates all the additional tools such as the emulator, debug bridge, etc

Developer Aid



- Enable Eclipse to notify the user that they are including an unnecessary permission (or missing a necessary one)
- Performed via static code analysis along with a permission-API database
- Does not require any alteration to existing devices or the Android framework

Permission Check Tool



```
/**
 * Get the friendly Bluetooth name of the remote device.
 *
 * <p>The local adapter will automatically retrieve remote names when
 * performing a device scan, and will cache them. This method just returns
 * the name for this device from the cache.
 * <p>Requires {@link android.Manifest.permission#BLUETOOTH}
 *
 * @return the Bluetooth name, or null if there was a problem.
 */
public String getName() {
    try {
        return sService.getRemoteName(mAddress);
    } catch (RemoteException e) {Log.e(TAG, "", e);}
    return null;
}

/**
 * Start the bonding (pairing) process with the remote device.
 * <p>This is an asynchronous call, it will return immediately. Register
 * for {@link #ACTION_BOND_STATE_CHANGED} intents to be notified when
 * the bonding process completes, and its result.
 * <p>Android system services will handle the necessary user interactions
 * to confirm and complete the bonding process.
 * <p>Requires {@link android.Manifest.permission#BLUETOOTH_ADMIN}.
```

android developers

Home | SDK | Dev Guide | Reference | Resources | Video

Package Index | Class Index

android
android.accessibilityservice
android.accounts
android.app
android.app.admin
android.app.backup
android.appwidget
android.bluetooth
android.content
android.content.pm
android.content.res
android.database
android.database.sqlite
android.gesture
android.graphics

Classes

BluetoothAdapter
BluetoothClass
BluetoothClass.Device
BluetoothClass.Device.Major
BluetoothClass.Service
BluetoothDevice
BluetoothServerSocket
BluetoothSocket

public final class
BluetoothSocket

extends Object
implements Closeable

java.lang.Object
↳android.bluetooth.BluetoothSocket

Class Overview

A connected or connecting Bluetooth socket.

The interface for Bluetooth Sockets is similar to that of TCP socke
new BluetoothSocket to manage the connection. On the client

The most common type of Bluetooth socket is RFCOMM, which is

To create a BluetoothSocket for connecting to a known device
connection fails.

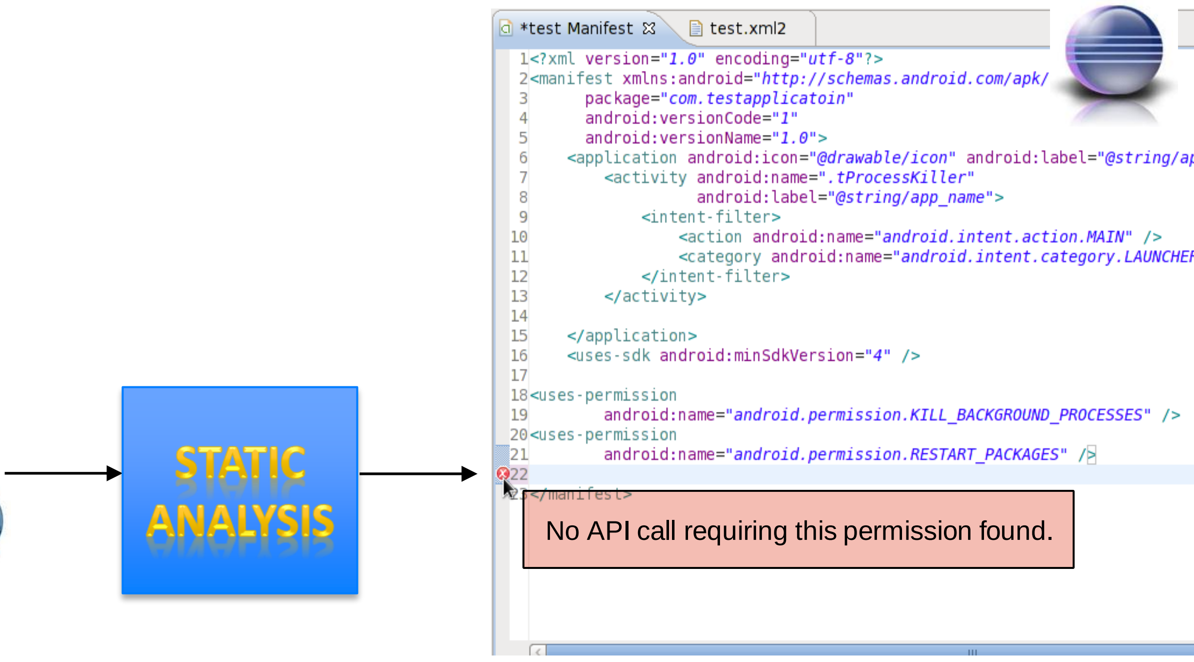
To create a BluetoothSocket as a server (or "host"), see the B

Once the socket is connected, whether initiated as a client or acc
connected to the socket.

BluetoothSocket is thread safe. In particular, close() will alw

Note: Requires the BLUETOOTH permission.

See Also
BluetoothServerSocket
InputStream
OutputStream



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/
      package="com.testapplicatoin"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/ap
        <activity android:name=".tProcessKiller"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHEF
            </intent-filter>
        </activity>

    </application>
    <uses-sdk android:minSdkVersion="4" />

<uses-permission
        android:name="android.permission.KILL_BACKGROUND_PROCESSES" />
<uses-permission
        android:name="android.permission.RESTART_PACKAGES" />

</manifest>
```

No API call requiring this permission found.

- End result is an Eclipse notification denoting an error when a permission is not needed.

Tool Analysis

- Without large sets of source code, it is difficult to perform extensive empirical analysis of a source code oriented tool
- Even so, several open source applications were found to have extraneous permissions
  – Quite likely due to common circumstances such as the removal of a feature and forgetting to remove the associated permission in the manifest

Conclusions

- The tool is a functional proof of concept that is compatible with Android ADT Eclipse extension
- The tool highlights permission discrepancies so that the developer has to “go out of the way” in order to include additional permissions

Future Work

- Utilize a more robust permission database
- Extend the Eclipse plugin to be even more user friendly such as using an Eclipse nature to provide real time nudges
- Create corpora of source and packaged applications to facilitate future research projects
- Adjust the definition of proper operation, to include required permissions that the user deems unnecessary (not the source code)

Cylab Usable Privacy and Security Laboratory

http://cups.cs.cmu.edu/

Carnegie Mellon