**First code**:

private static boolean isRooted() {

return findBinary("su");

}

public static boolean findBinary(String binaryName) {

boolean found = false;

if (!found) {

String[] places = {"/sbin/", "/system/bin/", "/system/xbin/", "/data/local/xbin/",

"/data/local/bin/", "/system/sd/xbin/", "/system/bin/failsafe/", "/data/local/"};

for (String where : places) {

if ( new File( where + binaryName ).exists() ) {

found = true;

break;

}

}

}

return found;

}

It is working properly. But i have heard that the filename "su" can be changed and also a file with the name "su" can be created in the nonrooted devices. In that cases, this source is not dependable.so i want to know some other way to find the rooted device other than searching for "su". I have used the following code

Public static boolean checkRootMethod1()

{

String buildTags = android.os.Build.TAGS;

if (buildTags != null && buildTags.contains("test-keys")) {

return true;

}

return false;

}

**Note :** If Build.TAGS contains "test-keys" it does not mean the device is rooted.

It is not working properly. For rooted devices it works as expected. But for SOME unrooted devices also it shows as rooted. Since the output is varying for different devices.

so you can use second code

**Second code**:

/\*\*

\* Checks if the device is rooted.

\*

\* @return <code>true</code> if the device is rooted, <code>false</code> otherwise.

\*/

public static boolean isRooted() {

// get from build info

String buildTags = android.os.Build.TAGS;

if (buildTags != null && buildTags.contains("test-keys")) {

return true;

}

// check if /system/app/Superuser.apk is present

try {

File file = new File("/system/app/Superuser.apk");

if (file.exists()) {

return true;

}

} catch (Exception e1) {

// ignore

}

// try executing commands

return canExecuteCommand("/system/xbin/which su")

|| canExecuteCommand("/system/bin/which su") || canExecuteCommand("which su");

}

// executes a command on the system

private static boolean canExecuteCommand(String command) {

boolean executedSuccesfully;

try {

Runtime.getRuntime().exec(command);

executedSuccesfully = true;

} catch (Exception e) {

executedSuccesfully = false;

}

return executedSuccesfully;

}

**Third code:**

you could try to do

Process proc = Runtime.getRuntime ().exec ( "su" );

if that throws an exception or proc is null then they don't have root

**note:** Can[ Process proc = Runtime.getRuntime ().exec ( "su" );] fail ?

yes there are softwares available that simulate that phone is not rooted by not granting you access of super user .

**Forth code:**

public static boolean checkRooted()

{

try

{

Process p = Runtime.getRuntime().exec("su", null, new File("/"));

DataOutputStream os = new DataOutputStream( p.getOutputStream());

os.writeBytes("pwd\n");

os.writeBytes("exit\n");

os.flush();

p.waitFor();

p.destroy();

}

catch (Exception e)

{

return false;

}

return true;

}

**Fifth code:**

The code in CodeMonkey's post works on most devices, but at least on Nexus 5 with Marshmallow it doesn't, because the which command actually works even on non-rooted devices. But because su doesn't work, it returns a non-zero exit value. This code expects an exception though, so it has to be modified like this:

private static boolean canExecuteCommand(String command) {

try {

int exitValue = Runtime.getRuntime().exec(command).waitFor();

if (exitValue != 0) return false;

else return true;

} catch (Exception e) {

return false;

}

}

**Sixth code: Modules**

**Appcelerator Titanium Rooted Module**

This module provides a function to check if a device has been Jailbroken (iOS), or Rooted (Android). Check the Releases tab for downloads.

var Rooted = require('com.collinprice.rooted');

if (Rooted.isRooted()) {

Ti.API.info("DEVICE IS ROOTED!!!");

} else {

Ti.API.info("DEVICE IS NOT ROOTED.....");

}

**You can refer to this link:** [**https://github.com/collinprice/com.collinprice.rooted**](https://github.com/collinprice/com.collinprice.rooted)

# RootFinder Module

The **RootFinder** module provides an interface to detect hacked/rooted devices from Titanium

# You can refer to this link: <https://github.com/mattapperson/RootFinder>

**Seventh code:**

There is another reliable way to check whether the Android Phone is rooted or not. For that, you need to download **Android Debug Bridge (ADB)**. After that, you have to open "cmd" and open the path of **ADB** in same. Now, you have to run following commands:

1. adb devices : This command will list the attached Android devices.
2. shell serial\_number: This command will open shell/terminal for a particular android device.
3. su : This command is to get super-user access of any terminal. Now, after running this command if you get an error "Permission Denied" than your device is not rooted and if you do not get any error than your device is rooted.

**Reference**:

1. [Android Debug Bridge download](http://adbdriver.com/downloads/)
2. [ADB commands](http://www.linuxtopia.org/online_books/android/devguide/guide/developing/tools/android_adb_commandsummary.html)

**Reference**:

[1] <https://stackoverflow.com/questions/19288463/how-to-check-if-android-phone-is-rooted>

[2] <https://stackoverflow.com/questions/24523606/how-to-find-the-rooted-device-programmatically>

[3] [https://stackoverflow.com/questions/35425742/how-to-check-if-the-android-device-is-rooted-or-not-in- appcelerator](https://stackoverflow.com/questions/35425742/how-to-check-if-the-android-device-is-rooted-or-not-in-%20%20%20%20%20appcelerator)