Android Root Detection Techniques

Introduction

I have taken a look at a lot of Mobile Device Management (MDM) solutions lately to figure out how they are detecting rooted Android devices. Through some research I have discovered that many of these MDM solutions use similar methods to detect rooted devices. This usually involves looking for specific packages and files, directory permissions, and running certain commands. I won't be disclosing which MDMs use which methods, but I will provide a list of packages, files, folders, and commands that I have found to be used in root detection. All the commands I will be running are on a stock rooted Nexus 4 running Android 4.2.2.

Default Files & Configurations

The first root detection checks are for default files and configurations that should be present on a non-rooted device. These may also be present in rooted devices with non-custom roms.

1. Checking the BUILD tag for test-keys. By default, stock Android ROMs from Google are built with release-keys tags. If test-keys are present, this can mean that the Android build on the device is either a developer build or an unofficial Google build. My Nexus 4 is running stock Android from Google's (Android Open Source Project) AOSP. This is why my build tags show release-keys.

   ```
   root@android:/ # cat /system/build.prop | grep ro.build.tags
   ro.build.tags=release-keys
   ```

2. Checking for Over The Air (OTA) certs. By default, Android is updated OTA using public certs from Google. If the certs are not there, this usually means that there is a custom ROM installed which is updated through other means. My Nexus 4 has no custom ROM and is updated through Google. Updating my device however, will probably break root.

   ```
   root@android:/ # ls -l /etc/security/otacerts.zip
   ls -l /etc/security/otacerts.zip
   -rw-r--r-- root     root         1733 2008-08-01 07:00 otacerts.zip
   ```

Installed Files & Packages

There are many files and packages that MDMs look for when detecting if a device is rooted. I have compiled a list of ones that I know for sure are being detected.

1. Superuser.apk. This package is most often looked for on rooted devices. Superuser allows the user to authorize applications to run as root on the device.
2. Other packages. The following list of packages are often looked for as well. The last two facilitate in temporarily hiding the su binary and disabling installed applications.

   ```
   com.noshufou.android.su
   ```

```
com.thirdparty.superuser
eu.chainfire.supersu
com.koushikdutta.superuser
com.zachspong.temprootremovejb
com.ramdroid.appquarantine
```

3. The following command lists packages that are currently installed on your device.

```
root@android:/ # pm list packages
package:com.android.backupconfirm
package:com.android.bluetooth
package:com.android.browser.provider
package:com.android.calculator2
package:eu.chainfire.supersu
```

4. Any chainfire package. One MDM looks for any package that is developed by chainfire. The most notable one being SuperSU.
5. Cyanogenmod.superuser. If the Cyanogenmod ROM is installed, the cyanogenmod.superuser activity may be in the com.android.settings package. This can be detected by listing the activities within com.android.settings.
6. Su Binaries. The following list of Su binaries are often looked for on rooted devices.

```
/system/bin/su
/system/xbin/su
/sbin/su
/system/su
/system/bin/.ext/.su
/system/usr/we-need-root/su-backup
/system/xbin/mu
```

Directory Permissions

Sometimes when a device has root, the permissions are changed on common directories. I have never seen this personally, but it is being checked for.

1. Are the following directories writable.

```
/data
/
/system
/system/bin
/system/sbin
/system/xbin
/vendor/bin
/sys
/sbin
/etc
/proc
```

```
/dev
```

2. Can we read files in /data. The /data directory contains all the installed application files. By default, /data is not readable.

Commands

A few MDMs execute common commands to detect if a device is rooted.

1. Su. Execute su and then id to check if the current user has a uid of 0 or if it contains (root).

```
shell@android:/ $ su
shell@android:/ # id
uid=0(root) gid=0(root)
groups=1003(graphics),1004(input),1007(log),1009(mount),1011(adb),1015(sdca
rd_rw),1028(sdcard_r)
```

2. Busybox. If a device has been rooted, more often then not Busybox has been installed as well. Busybox is a binary that provides many common linux commands. Running Busybox is a good indication that a device has been rooted.

```
root@android:/ # busybox df
Filesystem           1K-blocks      Used Available Use% Mounted on
tmpfs                   958500        32    958468   0% /dev
tmpfs                   958500         0    958500   0% /mnt/secure
tmpfs                   958500         0    958500   0% /mnt/asec
tmpfs                   958500         0    958500   0% /mnt/obb
```

Conclusion

This is probably no where near a complete list, but it does show the many different ways root can be detected on Android devices. Blacklisting packages and binaries seems to be the simplest and most effective way to detect root. This is especially true if your device is running a stock ROM from Google that has been rooted like mine where the only difference is the addition of su and a couple packages. At some point in the future I would like to create an app that will provide all these checks before installing an MDM. I touch on bypassing AirWatch root detection in my blog here: https://www.netspi.com/blog/entryid/192/bypassing-airwatch-root-restriction, however, AirWatch has made some changes so it may not work anymore depending on your environment.