**2.1 Root Detection Techniques**

* **Android File System and Directory Structure**

Most people are familiar with the file layout on Windows and are happy navigating the

Windows file System. Windows uses a drive letter for each physical drive or partition e.g. C:

drive. A Physical drive will have at least 1 partition but can have more than one. This is true for both Windows and Android. Each disk partition has a root directory which contains files and folders (directories). The root of the C drive is C:\ and of the F drive is F:\ etc.

Android uses the **Linux file** system structure which has a single root. The diagram below shows the Structure:

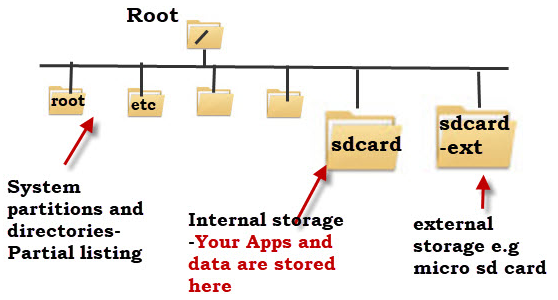


Figure 1: Android File System Structure

The system partitions and directories are protected and unless your device is rooted you don’t normally have access to these although some file managers will display them. Physical disks and partitions appear under the root as directories, and do not have a drive letter as in Windows. Android doesn’t normally come with a default file manager, and so you will need to install a file manager App like Astro file manager, to locate and manage files and folders.

Partitions in android are directories under the root. In windows you can reach partition using the

drive name followed by ‘:’ (e.g. c:/), in android: (/drive-name).

* **Root Access**

Rooting is the process of allowing users of smartphones, tablets and other devices running the

Android mobile operating system to attain privileged control (known as root access) over various

Android subsystems. As Android uses the Linux kernel, rooting an Android device gives similar

access to administrative (superuser) permissions as on Linux or any other Unix-like operating

system such as FreeBSD or macOS.

Rooting is often performed with the goal of overcoming limitations that carriers and hardware manufacturers put on some devices. Thus, rooting gives the ability (or permission) to alter or replace system applications and settings, run specialized applications ("apps") that require administrator-level permissions, or perform other operations that are otherwise inaccessible to a normal Android user. On Android, rooting can also facilitate the complete removal and replacement of the device's operating system, usually with a more recent release of its current operating system.

* **Root Detection**

After taking a look at a lot of Mobile Device Management (MDM) solutions lately to figure out how they are detecting rooted Android devices. Many of MDM solutions use similar methods to detect rooted devices. This usually involves looking for specific packages and files, directory permissions, and running certain commands. Disclosing which MDMs use which methods will not provide, but a list of packages, files, folders, and commands that are found to be used in root detection will provide.

* **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.

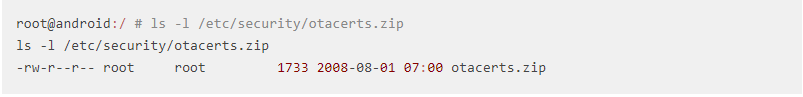
1. **Checking the BUILD tag for test-keys**.

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. This is why my build tags show release-keys.



1. **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. Updating my device however, will probably break root.



* **Installed Files & Packages**

There are many files and packages that MDMs look for when detecting if a device is rooted. A list of ones is compiled to 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.

1. **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

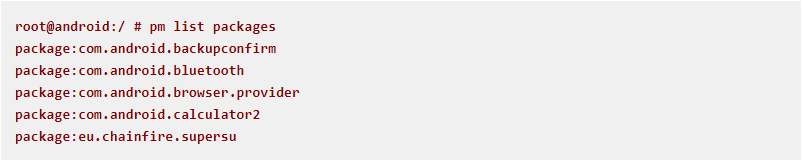
**eu**.chainfire.supersu

**com**.koushikdutta.superuser

**com**.zachspong.temprootremovejb

**com**.ramdroid.appquarantine

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



1. Any **chainfire package**. One MDM looks for any package that is developed by chainfire. The most notable one being SuperSU.
2. **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.
3. **Su Binaries**. The following list of Su binaries are often looked for on rooted devices.

* **Directory Permissions**

Sometimes when a device has root, the permissions are changed on common directories.

1. **Are the following directories writable**

/data

/

/system

/system/bin

/system/sbin

/system/xbin

/vendor/bin

/sys

/sbin

/etc

/proc

/dev

1. **Can we read files in /data**

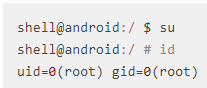
The /data directory contains all the installed application files. By default, /data is not readable.

* **Commands**

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

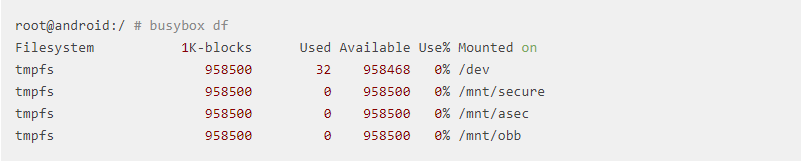
1. **Su**

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



1. **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.



* **SafetyNet**

It is an Android API that creates a profile of the device using software and hardware

information. This profile is then compared against a list of white-listed device models that

have passed Android compatibility testing.

SafetyNet is not well documented, and may change at any time: When you call this API, the service downloads a binary package containing the device validation code from Google, which is then dynamically executed using reflection. An analysis showed that the checks performed by SafetyNet also attempt to detect whether the device is rooted, although it is unclear how exactly this is determined.

To use the API, an app may the **SafetyNetApi.attest()** method with returns a **JWS message** (it is a way to authenticate (but not necessarily encrypt) information in a highly serializable, machine-readable format. That means that it is information, along with proof that the information hasn't changed since being signed. It can be used for sending information from one web site to another, and is especially aimed at communications on the web. It even contains a compact form optimized for applications like URI query parameters) with the Attestation Result, and then check the following fields:

* ctsProfileMatch: Of "true", the device profile matches one of Google's listed devices that have passed Android compatibility testing.
* basicIntegrity: Of "true", The device running the app likely wasn't tampered with.

The attestation result looks as follows.

{

"nonce": "R2Rra24fVm5xa2Mg",

"timestampMs": 9860437986543,

"apkPackageName": "com.package.name.of.requesting.app",

"apkCertificateDigestSha256": ["base64 encoded, SHA-256 hash of the

certificate used to sign requesting app"],

"apkDigestSha256": "base64 encoded, SHA-256 hash of the app's APK",

**"ctsProfileMatch": true,**

**"basicIntegrity": true,**

}

REFERENCES

1. Android File System and Directory Structure Explained

<http://www.stevesandroidguide.com/android-files/> Retrieved Nov. 2017.

1. Rooting (Android)

<https://en.wikipedia.org/wiki/Rooting_(Android)> Retrieved Nov. 2017.

1. Testing Root Detection

[https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05j-Testing-Resiliency- Against-Reverse-Engineering.md#safetynet Retrieved Nov. 2017.](https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05j-Testing-Resiliency-%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20Against-Reverse-Engineering.md#safetynet   Retrieved Nov. 2017.  )

1. Android Root Detection Techniques

<https://blog.netspi.com/android-root-detection-techniques/> Retrieved Nov. 2017.