Promon's approach to root & jailbreak detection

Understand the risks of rooted and jailbroken devices—and how Promon helps safeguard your app

Rooting/jailbreaking basics

Rooting or jailbreaking is the process of removing the default controls on a smartphone to gain admin-level access. Rooting is done on Android devices and jailbreaking on iOS.

Rooted/jailbroken devices are more susceptible to software vulnerabilities and malware injection. This is because admin permissions allow users to analyze or alter an app's behavior or internal logic, risking data exfiltration, code injection, data breaches, and IP loss.

There are several methods to help you detect rooted/jailbroken devices, but none are foolproof. You should combine these methods with additional defenses that also protect the app from the risks that rooted/jailbroken devices create.

4 common jailbreak/root detection methods

Detecting rooting/jailbreaking apps

Some apps help non-technical users easily root or jailbreak their devices. These include SuperSU, Magisk Manager, KingRoot, Cydia, Sileo, and more. Scanning for the presence of these apps on the user's device is one common detection method.

Identifying altered files

Rooting/jailbreaking creates certain files on devices, including directories, binaries, or packages. These include **/system/bin/su binary** or **Superuser.apk**, or **/system/xbin/su** on Android or **/private/var/tmp/cydia.log**, **/Applications/Cydia.app**, and **/etc/apt** on iOS.

Finding evidence of hooking frameworks

Hooking frameworks can modify an app's behavior at runtime. This requires elevated privileges, so finding evidence of these hooking frameworks is another common rooting/jailbreaking detection technique you can use.

Sandbox inspection [iOS only]

Sandboxes on iOS devices generally restrict an app's ability to access files and data on the wider mobile device. You can test the integrity of these sandboxes to detect jailbreaking because these are often weakened on jailbroken devices.

Why root/jailbreak detection isn't perfect

Cat-and-mouse game

Rooting and jailbreaking tools use a range of constantly evolving techniques to hide their activity. This creates a constant cat-and mouse game between jailbreakers and the tools designed to detect them.

Elevated privileges

By definition, rooting/jailbreaking provides elevated privileges. This gives technically savvy phone users a wide range of tools to hide evidence of their activities.

False positives

Many rooting/jailbreaking techniques cause false positives, which risk penalizing devices that in reality pose little threat.

Multiple methods

There are countless ways to jailbreak/root a device. While the most common (e.g. using a rooting app) are comparatively easy to detect, others (like rooting via a custom Android kernel) can be fiendishly difficult.

Tightening sandboxes

Android and iOS sandboxes are becoming more restrictive, making it harder for apps to analyze activity on the user's device. This is an issue, because most root/jailbreak detection tools rely on this analysis.

Root detection is a nice-to-have (and we have it)

Since root detection isn't perfect, we consider it a nice-to-have, rather than a non-negotiable. But root detection is a valuable part of a broader defense, so we include it alongside other protections, including:

- **App integrity checks:** Protect the binary code and executable files of your app so you can easily identify and prevent attempts to modify its behavior.
- **Code injection prevention:** Prevent hackers and malicious actors from injecting malicious code into your app.
- **Hooking framework detection:** Detect and prevent hooking frameworks designed to modify your app's behavior at runtime.
- **Code obfuscation:** Make it difficult or impossible for hackers to reverse engineer the underlying logic of your app, preventing data exfiltration, vulnerability exploits, static analysis, and more.

Together, these features help protect your app **even when it runs on rooted/jailbroken devices**.

Promon for root detection

Promon SHIELD®

Access **runtime protections** to defend your app against code injection, runtime analysis, reverse engineering, and other attacks. Find out more >

IP Protection Pro™

Utilize **code obfuscation** to safeguard your mobile app from reverse engineering, tampering, and other security threats. Find out more >