# System and kernel security

8-10 minutes

---

At the operating system level, the Android platform provides the security of the Linux kernel, as well as a secure inter-process communication (IPC) facility to enable secure communication between apps running in different processes. These security features at the OS level ensure that even native code is constrained by the Application Sandbox. Whether that code is the result of included app behavior or an exploitation of an app vulnerability, the system is designed to prevent the rogue app from harming other apps, the Android system, or the device itself. See Kernel Configuration for measures you can take to strengthen the kernel on your devices. See the Android Compatibility Definition Document (CDD) for required settings.

## Linux security

The foundation of the Android platform is the Linux kernel. The Linux kernel has been in widespread use for years, and is used in millions of security-sensitive environments. Through its history of constantly being researched, attacked, and fixed by thousands of developers, Linux has become a stable and secure kernel trusted by many corporations and security professionals.

As the base for a mobile computing environment, the Linux kernel provides Android with several key security features, including:

- A user-based permissions model
- Process isolation
- Extensible mechanism for secure IPC
- The ability to remove unnecessary and potentially insecure parts of the kernel

As a multiuser operating system, a fundamental security objective of the Linux kernel is to isolate user resources from one another. The Linux security philosophy is to protect user resources from one another, Linux:

- Prevents user A from reading user B's files
- Ensures that user A doesn't exhaust user B's memory
- Ensures that user A doesn't exhaust user B's CPU resources
- Ensures that user A doesn't exhaust user B's devices (for example, telephony, GPS, and Bluetooth)

## Application Sandbox

Android's app security is enforced by the Application Sandbox, which isolates apps from each other and protects apps and the system from malicious apps. For more details, see Application Sandbox.

## System partition and safe mode

The various integrity protected partitions contain Android's kernel as well as the operating system libraries, app runtime, app framework, and apps. This partition is set to read-only. When a user boots the device into safe mode, third-party apps may be launched manually by the device owner but aren't launched by default.

## File system permissions

In a UNIX-style environment, file system permissions ensure that one user can't alter or read another user's files. In the case of Android, each app runs as its own user. Unless the developer explicitly shares files with other apps, files created by one app can't be read or altered by another app.

## Security-Enhanced Linux

Android uses Security-Enhanced Linux (SELinux) to apply access control policies and establish mandatory access control (mac) on processes. See Security-Enhanced Linux in Android for details .

## Verified Boot

Android 7.0 and later supports strictly enforced Verified Boot, which means compromised devices can't boot. Verified Boot guarantees the integrity of the device software starting from a hardware root of trust up to the system partition. During boot, each stage cryptographically verifies the integrity and authenticity of the next stage before executing it.
See Verified boot for more details.

## Cryptography

Android provides a set of cryptographic APIs for use by apps. These include implementations of standard and commonly used cryptographic primitives such as AES, RSA, DSA, and SHA. Additionally, APIs are provided for higher level protocols such as SSL and HTTPS.

Android 4.0 introduced the KeyChain class to allow apps to use the system credential storage for private keys and certificate chains.

## Device rooting

By default, on Android only the kernel and a small subset of the core services run with root permissions. SELinux still constrains user space processes running as root. Verified Boot prevents a user or service with root permissions from permanently modifying the operating system.

The ability to modify an Android device they own is important to developers working with the Android platform. On many Android devices, users have the ability to unlock the bootloader in order to allow installation of an alternate operating system. These alternate operating systems may allow an owner to gain root access for purposes of debugging apps and system components or to access features not presented to apps by Android APIs.

On some devices, a person with physical control of a device and a USB cable is able to install a new operating system that provides root privileges to the user. To protect any existing user data from compromise the bootloader unlock mechanism requires that the bootloader erase any existing user data as part of the unlock step. Root access gained via exploiting a kernel bug or security hole can bypass this protection.

Encrypting data with a key stored on-device doesn't protect the app data from root users on rooted devices. Apps can add a layer of data protection using encryption with a key stored off-device, such as on a server or a user password. This approach can provide temporary protection while the key isn't present, but at some point the key must be provided to the app and it then becomes accessible to root users.

A more robust approach to protecting data from root users is through the use of hardware solutions. OEMs may choose to implement hardware solutions that limit access to specific types of content such as DRM for video playback, or the NFC-related trusted storage for Google wallet. In the case of a lost or stolen device, storage encryption ensures that user data can't be accessed without knowing the user's lockscreen credential.

## User security features

### Storage encryption

The CDD requires that all devices that launch with Android 10 or higher, and most devices that launch with Android 6.0 or higher, enable storage encryption out of the box.

Android's current implementation of storage encryption is file-based encryption in combination with metadata encryption. File-based encryption transparently encrypts file contents and names on the userdata partition, using different keys for different directories. It provides credential-encrypted and device-encrypted storage directories for each user, including work profiles.

Metadata encryption complements file-based encryption. It encrypts all blocks on the userdata partition that aren't already encrypted by file-based encryption, using a key not bound to any user's lockscreen credential but still protected by Verified Boot.

### Lockscreen credential protection

Android can be configured to verify a user-supplied lockscreen credential (PIN, password, or pattern) prior to providing access to a device. In addition to preventing unauthorized use of the device, the lockscreen credential protects the cryptographic key for credential-encrypted data. Use of a lockscreen credential and/or credential complexity rules can be required by a device administrator.

## Device administration

Android 2.2 and later provide the Android Device Administration API, which provides device administration features at the system level. For example, the built-in Android email app uses the APIs to improve Exchange support. Through the email app, Exchange administrators can enforce lockscreen credential policies — including alphanumeric passwords or numeric PINs — across devices. Administrators can also remotely wipe (that is, restore factory defaults on) lost or stolen handsets.

In addition to use in apps included with the Android system, these APIs are available to third-party providers of Device Management solutions. Details on the API are provided at Device Administration.