System and kernel security

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 applications 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 application behavior or an exploitation of an application vulnerability, the system is designed to prevent the rogue application from harming other applications, the Android system, or the device itself. See Kernel Configuration (https://source.android.com/devices/tech/config/kernel.html) for measures you can take to strengthen the kernel on your devices. See the Android Compatibility Definition Document (CDD) (https://source.android.com/compatibility/cdd.html) 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. Thus, Linux:

- Prevents user A from reading user B's files
- Ensures that user A does not exhaust user B's memory
- Ensures that user A does not exhaust user B's CPU resources
- Ensures that user A does not exhaust user B's devices (e.g. telephony, GPS, Bluetooth)

The Application Sandbox

Android's application 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 (https://source.android.com/security/app-sandbox).

System Partition and Safe Mode

The system partition contains Android's kernel as well as the operating system libraries, application runtime, application framework, and applications. This partition is set to read-only. When a user boots the device into Safe Mode, third-party applications may be launched manually by the device owner but are not launched by default.

Filesystem Permissions

In a UNIX-style environment, filesystem permissions ensure that one user cannot alter or read another user's files. In the case of Android, each application runs as its own user. Unless the developer explicitly shares files with other applications, files created by one application cannot be read or altered by another application.

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 (https://source.android.com/security/selinux/index.html) for details.

Verified boot

Android 6.0 and later supports verified boot and device-mapper-verity. 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.

Android 7.0 and later supports strictly enforced verified boot, which means compromised devices cannot boot.

See Verified boot (https://source.android.com/security/verifiedboot/index.html) for more details.

Cryptography

Android provides a set of cryptographic APIs for use by applications. 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 (http://developer.android.com/reference/android/security/KeyChain.html) class to allow applications to use the system credential storage for private keys and certificate chains.

Rooting of Devices

By default, on Android only the kernel and a small subset of the core applications run with root permissions. Android does not prevent a user or application with root permissions from modifying the operating system, kernel, or any other application. In general, root has full access to all applications and all application data. Users that change the permissions on an Android device to grant root access to applications increase the security exposure to malicious applications and potential

application flaws.

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 applications and system components or to access features not presented to applications 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 does not protect the application data from root users. Applications 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 is not present, but at some point the key must be provided to the application 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, full filesystem encryption on Android devices uses the device password to protect the encryption key, so modifying the bootloader or operating system is not sufficient to access user data without the user's device password.

User Security Features

Filesystem Encryption

Android 3.0 and later provides full filesystem encryption, so all user data can be encrypted in the kernel.

Android 5.0 and later supports full-disk encryption (https://source.android.com/security/encryption/full-disk.html). Full-disk encryption uses a single key—protected with the user's device password—to protect the whole of a device's userdata partition. Upon boot, users must provide their credentials before any part of the disk is accessible.

Android 7.0 and later supports file-based encryption (https://source.android.com/security/encryption/file-based.html). File-based encryption allows different files to be encrypted with different keys that can be unlocked independently.

More details on implementation of filesystem encryption are available in the Encryption (https://source.android.com/security/encryption/index.html) section.

Password Protection

Android can be configured to verify a user-supplied password prior to providing access to a device. In addition to preventing unauthorized use of the device, this password protects the cryptographic key for full filesystem encryption.

Use of a password and/or password 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 application uses the APIs to improve Exchange support. Through the Email application, Exchange administrators can enforce password 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 applications 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 (https://developer.android.com/guide/topics/admin/device-admin.html).