Android Enterprise Security White Paper

Updated January, 2020

Table of Contents

Introduction

Android uses industry-leading security practices and works closely with the entire ecosystem to help keep devices safe. Our robust, multi-layered approach to security is critical to supporting enterprises, which must contend with ongoing threats. Organizations require strong security to protect their own data while also giving employees the flexibility to use mobile devices for essential productivity tasks and preserve users' privacy.

This security white paper outlines the Android approach to mobile security for business and government customers, and details the strengths of the Android platform, the range of management APIs available to enforce control, and the role of Google Play Protect in detecting threats.

Android offers a multi-layer security strategy with unique ways to keep data and devices safe. Beyond hardware and operating system protections, Android uses multi-profile support and device-management options that enable the separation of work and personal data, keeping company data secure. Google Play Protect offers built-in malware protection, identifying Potentially Harmful Applications and continually working to keep data and devices safe.

This white paper also details how the open source Android platform enables best-in-class enterprise security by leveraging the collective intelligence of the Android ecosystem. This information assists organizations in their decisions to implement Android and take advantage of its robust security technology.

Android Operating System

Android is an open source software stack created for a wide array of devices with different form factors. Android incorporates industry-leading security features and the Android team works with developers and device OEMs to keep the Android platform and ecosystem safe. A robust security model is essential to enable a vigorous ecosystem of apps and devices built on and around the Android platform and supported by cloud services. As a result, through its entire development lifecycle, Android has been subject to a rigorous security program.

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. Applications running on Android are signed and isolated into application sandboxes associated with their application signature. The application sandbox defines the privileges available to the application. Apps are generally built to execute in the Android Runtime and interact with the operating system through a framework that describes system services, platform Application Programming Interfaces (APIs), and message formats. A variety of high-level and lower-level languages, such as Java, Kotlin, and C/C++, are allowed and operate within the same application sandbox.

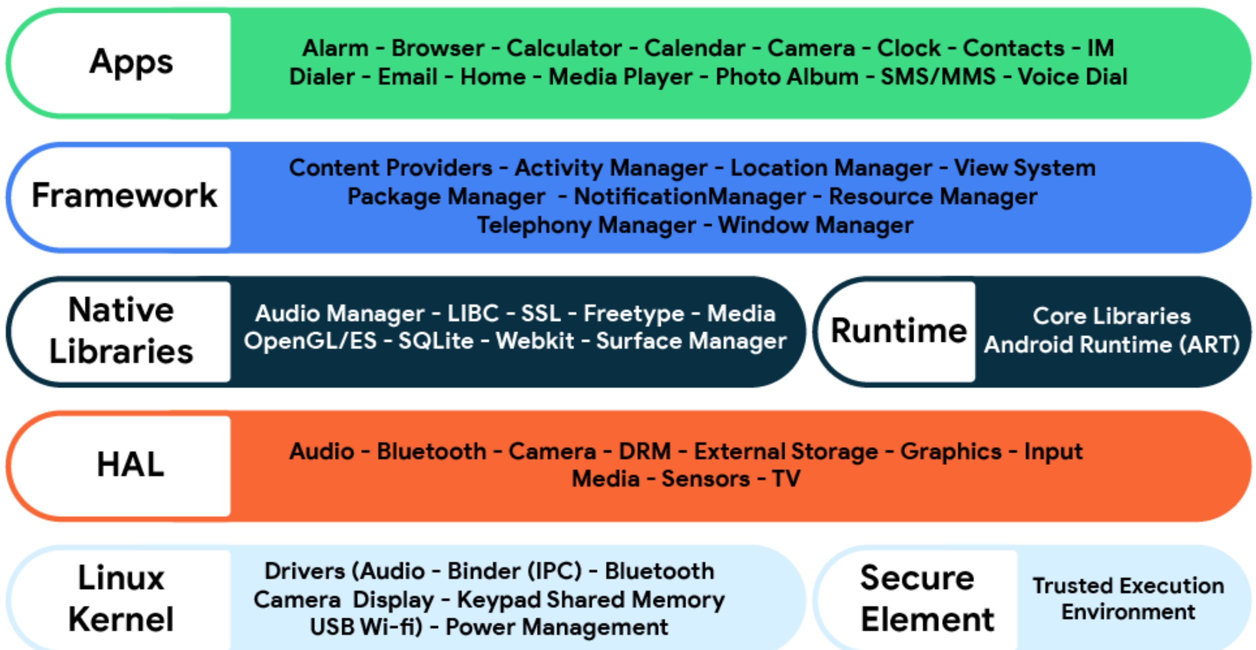


Figure 1. The Android software stack

Security by design

Android uses hardware and software protections to achieve strong defenses. Security starts at the hardware level, where the user is authenticated with lock screen credentials. Verified Boot ensures the system software has not been tampered with, and hardware-assisted encryption and key handling help protect data in transit and at rest.

At the software layer, built-in protection is essential to helping Android devices stay safe. Google Play Protect is the world's most widely-deployed threat detection service, actively scanning over 50 billion apps on-devices every day to monitor for harmful behavior. Play Protect scans all applications including public apps from Google Play, system apps updated by OEMs and carriers, and sideloaded apps.

Application sandboxing isolates and guards Android apps, preventing malicious apps from accessing private information. Android also protects access to internal operating system components, to help prevent vulnerabilities from becoming exploitable. Mandatory, always-on encryption helps keep data safe, even if devices fall into the wrong hands. Encryption is protected with Keystore keys, which store cryptographic keys in a container, making it more difficult to extract from a device. Developers can leverage the Android KeyStore with Jetpack Security safely and easily. Adiantum provides encryption capabilities for lower-powered devices that do not have Advanced Encryption Standard (AES) instructions as part of the CPU, potentially allowing businesses to use lower cost devices without forfeiting cryptographic security. In total, Android leverages hardware and software to keep devices safe.

Hardware-backed Security

Android supports several hardware features that enable strong device security.

Verified Boot

Verified Boot is Android's secure boot process that verifies system software before running it. This makes it more difficult for software attacks to persist across reboots, and provides users with a safer state at boot time.

Each Verified Boot stage is cryptographically signed. Each phase of the boot process verifies the integrity of the subsequent phase, prior to executing that code. As of Android 7.0, full boot of a compatible device with a locked bootloader proceeds only if the OS satisfies integrity checks.

The Verified Boot state is used as an input in the process to derive disk encryption keys. If the Verified Boot state changes (e.g. the user unlocks the bootloader), then the secure hardware prevents access to data used to derive the disk encryption keys that were used when the bootloader was locked.

Verified Boot on compatible devices running Android 9.0 and above require rollback protection. This means that a kernel compromise (or physical attack) cannot put an older, more vulnerable, version of the OS on a system and boot it. Additionally, rollback protection state is stored in tamper-evident storage.

Enterprises can check the state of Verified Boot using KeyStore key attestation. This retrieves a statement signed by the secure hardware attesting to many attributes of Verified Boot along with other information about the state of the device. KeyStore key attestation is required for compatible devices shipped with Android 8.0 onwards.

Trusted Execution Environment

Android devices that support a lock screen have a secondary, isolated environment called a Trusted Execution Environment (TEE). This enables further separation from any untrusted code. The capability is typically implemented using secure hardware such as ARM TrustZone® technology.

The TEE is responsible for some of the most security-critical operations on the device, including:

1. **Lock screen passcode verification**: available on devices that support a secure lock screen and ship with Android 7.0 and above. Lock screen verification is provided by TEE unless an even more secure environment is available.
2. **Fingerprint template matching**: available on devices that have a fingerprint sensor and ship with Android Marshmallow 6.0 and above.
3. **Protection and management of KeyStore keys**: available on devices that support a secure lock screen that ship with Android 7.0 and above.
4. **Protected Confirmation:** leverages a hardware-protected user interface called Trusted UI to facilitate high assurance to critical transactions, available on devices running Android 9.0 and above.
5. **Digital Rights Management (DRM):** an extensible framework that lets apps manage rights-protected content according to the license constraints associated with the content.

Android Keystore System

The Android Keystore system is a foundation of data protection on devices. It stores cryptographic keys in a container, making it more difficult to extract them from the device. Android KeyStore mitigates unauthorized use of key material on the Android device by making apps specify authorized uses of their keys and then enforcing these restrictions outside of the apps' processes.

Devices with Android 9.0 and above use the hardware-backed Keymaster 4, which offers additional protections against tampering. Keymaster introduced ID attestation in Android 8.0 (Keymaster 3) and Key Attestation in Android 7.0 (Keymaster 2). Key attestation enables off-device services to verify that keys used in apps are stored in a device's hardware-backed keystore. ID attestation allows a device to provide proof of its hardware identifiers, such as serial number or IMEI.

Keystore supports symmetric cryptographic primitives such as AES (Advanced Encryption Standard) and HMAC (Keyed-Hash Message Authentication Code) and asymmetric cryptographic algorithms. Access controls are specified during key generation and enforced for the lifetime of the key. Keys can be restricted to be usable only after the user has authenticated, and only for specified purposes or with specified cryptographic parameters.

For devices that support a secure lock screen and ship with Android 7.0 or above, KeyStore must be implemented in secure hardware. This guarantees that even in the event of a kernel compromise, KeyStore keys are not extractable from the secure hardware.

KeyStore key attestation

Compatible devices that ship with Android 8.0 and higher support Key Attestation, which empowers a server to gain assurance about the properties of keys. Devices that support Google Play are provisioned at the factory with an attestation key generated by Google. The secure hardware on such devices can sign statements with the provisioned key, which attests to properties of keys protected by the secure hardware, such as the fact that the key was generated and can't leave the secure hardware. Key Attestation better secures the location of important properties about the device, such as the OS version, patch level, and whether it passed Verified Boot.

Learn more about verifying hardware-backed keys with Key Attestation.

KeyChain

The KeyChain class provides access to private keys and their corresponding certificate chains in credential storage. KeyChain is often used by Chrome, Virtual Private Networks (VPNs), and enterprise apps to access keys imported by the user or by the mobile device management app.

Whereas the KeyStore is for non-shareable app-specific keys, KeyChain is for keys that are meant to be shared within the profile. For example, a mobile device management agent can import a key that Chrome will use for an enterprise website.

Android 10 introduces several improvements to the KeyChain API. When an app calls KeyChain.choosePrivateKeyAlias, devices now filter the list of certificates a user can choose from based on the issuers and key algorithms specified in the call. KeyChain no longer requires a device to have a screen lock before keys or certificate authority (CA) certificates can be imported.

Jetpack Security

Developers can leverage the Android KeyStore with Jetpack Security. MasterKeys allows developers to create a safe AES 256 GCM key out of the box or for advanced use cases that specify settings to control key authorization. Jetpack Security also provides higher level crypto abstractions for encrypting files (EncryptedFile) and SharedPreferences (EncryptedSharedPreferences). It is recommended that Jetpack Security be used by all Device Policy Controllers (DPCs), which control local device policies and system applications on devices, enterprise apps, public apps, and private apps.

Key decryption on unlocked devices

Android 9.0 introduced the unlockedDeviceRequired flag. This option determines whether the Keystore requires the screen to be unlocked before allowing usage of the specified key. These types of keys are well suited for encrypting sensitive data to store on disk, such as health or enterprise data. The flag provides users a higher assurance that the data cannot be decrypted while the device is locked should their phone be lost or stolen.

Version Binding

In Keymaster 2 and 3, all keys are also bound to the operating system and patch level of the system image. This ensures that an attacker who discovers a weakness in an old version of system or TEE software cannot roll a device back to the vulnerable version and use keys created with the newer version. In addition, when a key with a given version and patch level is used on a device that has been upgraded to a newer version or patch level, the key is upgraded before it can be used, and the previous version of the key invalidated. In this way, as the device is upgraded, the keys will "ratchet" forward along with the device, but any reversion of the device to a previous release will cause the keys to be unusable.

Tamper-resistant hardware support

As of Android 8.0, compatible devices can optionally use tamper-resistant hardware to verify the lock screen passcode. If verification succeeds, the tamper-resistant hardware returns a high entropy secret that can be used to derive the disk encryption key.

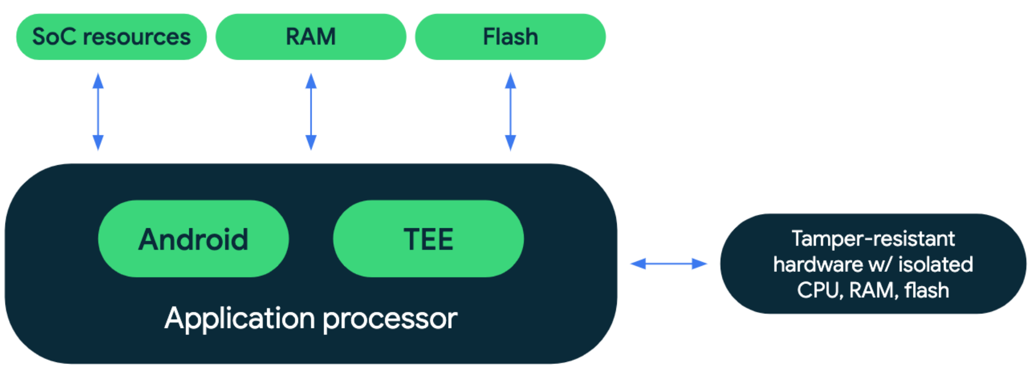


Figure 2. Security hardware provides numerous protections on the device.

Biometrics

In Android 9.0 and above, the BiometricPrompt API system provides biometric authentication dialogs to be used on behalf of an application. This creates a consistent look, feel, and placement for the dialog, and gives users a greater confidence they're authenticating with biometrics using a trusted credential tracker.

This API is used in conjunction with the Android Keystore system. Protecting biometric data is accomplished through a hardware security module in the form of Strongbox Keymaster, which securely stores and handles cryptographic keys on a device.

Devices can use biometric authentication to safeguard private information and essential corporate data accessible through devices used in an enterprise setting. The BiometricPrompt API is accessible to developers for integrating biometric authentication into their apps.

The Android framework includes face and fingerprint biometric authentication. Android can be customized to support other forms of biometric authentication, as well, such as Iris. All biometric implementations must meet security specifications and have a strong rating in order to participate in the BiometricPrompt class.

Biometric-based unlock modalities are typically evaluated on the basis of a False Accept Rate (FAR). Android uses two additional metrics to help device manufacturers evaluate their security: the Imposter Accept Rate (IAR) and Spoof Accept Rate (SAR).

Additionally, Android device manufacturers can access recommendations of system security best practices for using biometric authentication. Biometric sensors are classified based on their spoof and imposter acceptance rates and on the security of the biometric pipeline. Test methodology is available to assist in measuring the implementation of these unlock methods.

Fingerprint authentication

On devices with a fingerprint sensor, users can enroll one or more fingerprints and use those fingerprints to unlock the device and perform other tasks. Android uses the Fingerprint Hardware Interface Definition Language (HIDL) to connect to a vendor-specific library and fingerprint hardware, such as a fingerprint sensor.

Android supports Gatekeeper for fingerprint and PIN/pattern/password authentication. The Gatekeeper subsystem performs this authentication in a Trusted Execution Environment, enrolling and verifying passwords via an Hash-Based Message Authentication Code (HMAC) with a hardware-backed secret key.

The Android Compatibility Definition Document specifies the implementation requirements for biometric security.

Face authentication

Face authentication allows users to unlock their device simply by looking at the front of their device. Android 10 adds support for a new face authentication stack that can securely process camera frames, preserving security and privacy during face authentication on supported hardware. Android 10 also provides a method for security compliant implementations to enable application integration for transactions, such as online banking or other services.

Additional authentication methods

Android supports the Trust Agent framework to unlock the device. Google Smart Lock uses that framework to allow a device to remain unlocked as long as it stays with the user, as determined by certain user presence or other signals.

However, Smart Lock does not meet the same level of assurance as other unlock methods on Android and is not allowed to unlock auth-bound KeyStore keys. Organizations can disable Trust Agents using the KEYGUARD_DISABLE_TRUST_AGENTS flag in the EMM policies.

Protected Confirmation

Android Protected Confirmation leverages a hardware protected user interface (Trusted UI) to perform critical transactions outside the operating system in devices that run Android 9.0 or above. This protects operations from fraudulent apps or a compromised operating system. When an app invokes Protected Confirmation, control is passed to the Trusted UI, where transaction data is displayed and user confirmation of the data's correctness obtained.

Once confirmed, the intention is cryptographically authenticated and tamper-proof when conveyed to the relying party. In total, the transaction has higher protection and security relative to other forms of secondary authentication.

Several use cases exist for Android Protected Confirmation, such as person to person money transfers, user authentication, or other innovations such as confirming correct insulin pump injections.

Operating System Security

Android leverages multi-layered defenses to help keep the operating system secure. With each version of Android, the operating system is further hardened to have the right defenses for the ongoing threats that enterprises face.

Device integrity

Device integrity features protect the mobile device from running a tampered and/or compromised operating system. Android adopts several measures to guarantee device integrity, including sandboxing, SELinux, Seccomp, Unix permissions, and a Hardware Abstraction Layer.

Sandboxing

Android runs all apps inside sandboxes to prevent malicious or buggy app code from compromising other apps or system components. Because the application sandbox is enforced in the kernel, it encompasses the entire app regardless of the specific development environment, APIs, or programming language used. By default, apps can't interact with each other and have limited access to the operating system.

Similarly, system components run in least-privileged sandboxes in order to prevent compromises in one component from affecting others. For example, remotely reachable components, like the media server and WebView, are isolated in their own restricted sandbox.

SELinux

Android uses Security-Enhanced Linux (SELinux) to enforce mandatory access control (MAC) over all processes, including those with root/superuser privileges. SELinux enables Android to better protect and confine system services, control access to app data and system logs, reduce the effects of malicious software, and helps protect users from potential flaws in code on mobile devices.

SELinux operates on the principle of default denial: Anything not explicitly allowed is denied. Android includes SELinux and a corresponding security policy for components in AOSP. Disallowed actions are prevented and all attempted violations are logged via Linux tools: dmesg prints the message buffer of the kernel, and logcat, is a command-line tool that dumps a log of system messages.

With the Android system architecture, SELinux is used to enforce a separation between the Android framework and the device-specific vendor components such that they run in different processes and communicate with each other via a set of allowed vendor interfaces implemented as Hardware Abstraction Layers (HALs).

Seccomp filter

In conjunction with SELinux, Android uses Seccomp to further restrict access to the kernel by blocking access to certain system calls. As of Android 7.0, Seccomp was applied to processes in the media frameworks. As of Android 8.0, a Seccomp filter is applied to all apps, enforcing a whitelist of system calls which are allowed. Apps may optionally provide their own seccomp filter to further reduce the set of allowed system calls.

Unix permissions

Android uses Linux/Unix permissions to further isolate application resources. Android assigns a unique user ID (UID) to each application and runs each user in a separate process. By default, apps cannot access each other's files or resources just as different users on Linux are isolated from each other.

Anti-exploitation

Android offers exploit protection such as Control Flow Integrity and Integer Overflow Sanitization. New compiler-based mitigations have been added to make bugs harder to exploit and prevent certain classes of bugs from becoming vulnerabilities. This expands existing compiler mitigations, which direct the runtime operations to safely abort the processes when undefined behavior occurs.

User and Data Privacy

Protecting user privacy is fundamental to Android.

In Android 9.0, privacy highlights included limiting background apps' access to device sensors, restricting information retrieved from Wi-Fi scans, and new permission rules and permission groups related to phone calls, phone state, and Wi-Fi scans. These changes affect all apps running on Android 9, regardless of target SDK version.

Android 10 extends the privacy and controls that users have over data and app capabilities. In total, they provide users and IT administrators with better clarity about how data and user location can be accessed.

The work profile creates a separate, self-contained profile on Android devices that isolates corporate data from personal apps and data. This can be added to personal devices in a BYOD setting or on a company-owned device used for both work and personal purposes.

With this separate profile, the user's personal apps and data are outside IT control.

To provide clear visibility to the user, when a work profile is applied to a device the DPC presents the terms of use and details the data that is captured and recorded. The user must review and accept the user license agreement to set up the work profile.

Users can view work profile settings via Settings > Accounts.

Developers are encouraged to ensure their apps are compliant with the latest privacy changes. Android 10 places restrictions on accessing data and system identifiers, accessing camera and networking information, and makes several changes to the permissions model.

Restricting access to device identifiers

Devices that run Android 8.0 and above use random MAC addresses when probing new networks, while not currently associated to a network. On Android 9.0, the device can use a randomized MAC address when connecting to a Wi-Fi network if enabled by a developer option. In Android 10, the system transmits randomized MAC addresses by default. Additionally, device IMEI and serial numbers are unable to be accessed.

Location control

Apps can provide relevant information to the user using location APIs. For example, if an app helps the user navigate a delivery route, it needs to continually access the device location to provide the right assistance. Location is useful in many scenarios — Android provides tools for developers to request the necessary permissions while granting users choice in what they allow.

Apps that use location services must request location permissions so the user has visibility and control over this access. In Android 10, users see a dialog to notify them that an app wishes to access their location. This request can be for access only while using the app or all the time.

The user can choose to allow an app all-the-time access to device location. When an app accesses device location in the background for the first time after the user makes this choice, the system schedules a notification to send to the user. This notification reminds the user that they've allowed the app to access device location all the time.

Learn more about location updates.

External storage access

To give users more control over their files and to limit file clutter, apps targeting Android 10 and higher are given scoped access into an external storage device, or scoped storage, by default. Such apps can see only their app-specific directory—accessed using Context.getExternalFilesDir()—and specific types of media. Developers are encouraged to use scoped storage as a best practice.

EMM administrators are able to prevent their organization's users from accessing external storage, such as an SD card, connected to their device to further mitigate the potential for any data loss.

Limited access to background sensors

Android 9.0 limits the ability for background apps to access user input and sensor data. If an app is running in the background on a device running Android 9.0 and above, the system applies the following restrictions to the app:

- Application cannot access the microphone or camera.
- Sensors that use the continuous reporting mode, such as accelerometers and gyroscopes, don't receive events.
- Sensors that use the on-change or one-shot reporting modes don't receive events.

If an app needs to detect sensor events on devices running Android 9.0, it must use a foreground service.

Lockdown mode

A user can enable a lockdown option to further restrict access to the device. This mode displays a power button option that turns off Smart Lock, biometric unlocking, and notifications on the lock screen. It can be enabled via Settings > Lock screen preferences >

Lockdown mode. Enterprise administrators can remotely lock the work profile and evict the encryption key from memory on enterprise devices by leveraging this capability.

Network security

In addition to data-at-rest security—protecting information stored on the device—Android provides network security for data-in-transit to protect data sent to and from Android devices. Android provides secure communications over the Internet for web browsing, email, instant messaging, and other Internet apps, by supporting Transport Layer Security (TLS).

DNS over TLS

Android 9.0 and above includes built-in support for DNS over TLS. Users or administrators can enable a Private DNS mode in the Network and internet settings. Android 10 further extends the capabilities for administrators to prevent users from changing DNS settings, thus preventing DNS query leakage.

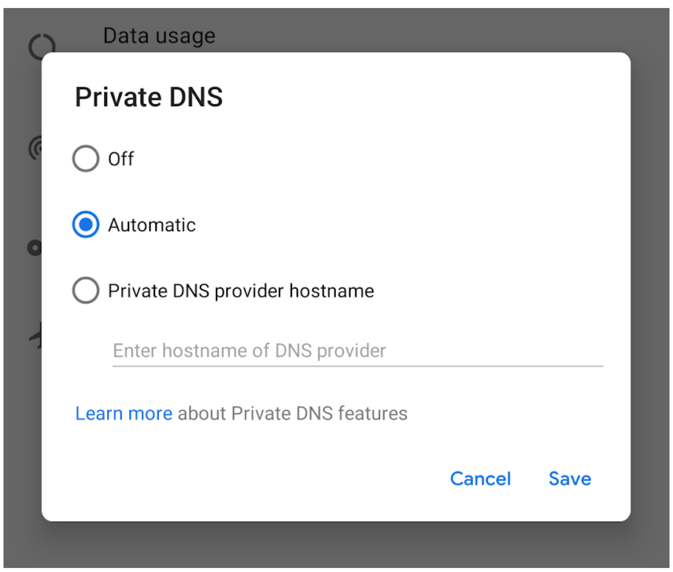


Figure 3. The Private DNS feature in the settings is enabled by default, with an option to input a private DNS provider hostname.

Devices automatically upgrade to DNS over TLS if the configured DNS server supports it, but users can turn it off if they wish.

TLS by default

Android helps keep data safe by protecting network traffic that enters or leaves a device with Transport Layer Security (TLS).

On Android 9.0 and above, the defaults for Network Security Configuration block all cleartext (unencrypted HTTP) traffic. Developers must explicitly opt-in to specific domains to use cleartext traffic in their applications. Android Studio also warns developers when their app includes a potentially insecure Network Security Configuration.

To prevent accidental unencrypted connections, the *android:usesCleartextTraffic* manifest attribute enables apps to indicate that they do not intend to send network traffic without encryption.

Android 10 uses TLS 1.3 by default for all TLS connections. TLS 1.3 is a major revision to the TLS standard with performance benefits and enhanced security. TLS v1.3 is also more private as it encrypts more of the handshake process and offers stronger security by no longer supporting certificates signed with SHA 1. Benchmarks indicate secure connections can be established as much as 40 percent faster with TLS 1.3 compared to TLS 1.2.

Learn more about TLS 1.3 implementation.

Wi-Fi

Android 10 supports the Wi-Fi Alliance's Wi-Fi Protected Access version 3 (WPA3) and Wi-Fi Enhanced Open standards. WPA3 and Wi-Fi Enhanced Open improve overall Wi-Fi security, providing better privacy and robustness against known attacks.

WPA3 is a new WFA security standard for personal and enterprise networks, taking advantage of modern security algorithms and stronger cipher suites.

WPA3 has two parts: personal and enterprise. WPA3-Enterprise offers stronger authentication and link-layer encryption methods, and an optional 192-bit security mode for sensitive security environments. WPA3-Personal uses simultaneous authentication of equals (SAE) instead of pre-shared key (PSK), providing users with stronger security protections against attacks such as offline dictionary attacks, key recovery, and message forging.

Wi-Fi Enhanced Open is a new WFA security standard for public networks based on opportunistic wireless encryption (OWE). It provides encryption and privacy on open, non-password-protected networks in areas such as cafes, hotels, restaurants, and libraries. Enhanced Open doesn't provide authentication.

Android also supports the WPA2-Enterprise (802.11i) protocol, also designed for enterprise networks and can be integrated into a broad range of Remote Authentication Dial-In User Service (RADIUS) authentication servers. The WPA2-Enterprise protocol support uses AES-128-CCM authenticated encryption.

In Android 10, QR codes and NFC data used for device provisioning can contain Extensible Authentication Protocol (EAP) config and credentials—including certificates. When a person scans a QR code or taps an NFC tag, the device automatically authenticates to a local Wi-Fi network using EAP and starts the provisioning process without any additional manual input.

Learn more about EAP WI-Fi provisioning.

VPN

Android supports securely connecting to an enterprise network using VPN:

- **Always-on VPN**—The VPN can be configured so that apps don't have access to the network until a VPN connection is established, which prevents apps from sending data across other networks.
    - In Android 7.0 and above, Always-on VPN supports VPN clients that implement VpnService. The system automatically starts the VPN after the device boots. Always-on VPN can be enabled for apps in enterprise use cases through DevicePolicyManager#setAlwaysOnVpnPackage. Device owners and profile owners can require work apps to always connect through a specified VPN. Additionally, users can manually set Always-on VPN clients that implement VpnService methods using **Settings>More>VPN**. The option to enable Always-on VPN from settings is available only if the VPN client targets API level 24 or higher.
- **Per User VPN**—On multi-user devices, VPNs are applied per Android user, so all network traffic is routed through a VPN without affecting other users on the device. VPNs are applied per work profile, which allows an IT administrator to specify that only their enterprise network traffic goes through the enterprise-work profile VPN—not the user's personal network traffic.
- **Per Application VPN**—Android 5.0 introduced support to facilitate VPN connections on allowed apps and to prevent VPN connections on disallowed apps.

In Android 10, VPN apps can set an HTTP proxy for their VPN connection. A VPN app must configure a ProxyInfo instance with a host and port, before calling VpnService.Builder.setHttpProxy(). The system and many networking libraries use this proxy setting but the system doesn't force apps to proxy HTTP requests.

VPN service modes

VPN apps can also now discover if the service is running because of always-on VPN and if lockdown mode is active. New methods added in Android 10 can help developers adjust the user interface. For example, developers may disable the disconnect button in the VPN application when always-on VPN controls the lifecycle of the service.

VPN Lockdown modes

Lockdown modes allow administrators to block network traffic that does not use the VPN and exempt an app that allows it to use any available network if the VPN is down or unreachable.

Administrators can also explicitly deny access to all networks for an app and this only allows communication to take place over the VPN.

Third-party apps

Google is committed to increasing the use of TLS in all apps and services. As apps become more complex and connect to more devices, it's easier for apps to introduce networking mistakes by not using TLS correctly.

As of December 2019, 80% of Android apps are encrypting traffic by default. The percentage is even greater for apps targeting Android 9 and higher, with 90% of them encrypting traffic by default.

Android 7.0 and above supports Network security configuration, which lets apps easily customize their network security settings in a safe, declarative configuration file without modifying app code. These settings can be configured for specific domains, such as opting out of cleartext traffic. This helps prevent an app from accidentally regressing due to changes in URLs made by external sources, such as backend servers.

This safe-by-default setting reduces the application attack surface while bringing consistency to the handling of network and file based application data.

Certificate handling

As of Android 7.0, all new devices must ship with the same certificate authority (CA) store.

Certificate authorities are a vital component of the public key infrastructure used in establishing secure communication sessions via TLS. Establishing which CAs are trustworthy—and by extension, which digital certificates signed by a given CA are trustworthy—is critical for secure communications over a network.

These protections are further improved through preventing apps that target Android 9.0 from allowing unencrypted connections by default. This follows a variety of changes made over the years to better protect Android users.

With Android 7.0 and above, compatible devices trust only the standardized system CAs maintained in AOSP. Apps can also choose to trust user- or admin- added CAs. Trust can be specified across the whole app or only for connections to certain domains.

When device-specific CAs are required, such as a carrier app needing to securely access components of the carrier's infrastructure (e.g. SMS/MMS gateways), these apps can include the private CAs in the components/apps themselves. For more details, see Network Security Configuration.

Application security

Apps are an integral part of any mobile platform, and users increasingly rely on mobile apps for core productivity and communication tasks. Android provides multiple layers of application protection, enabling users to download apps for work or personal use to their devices with the peace of mind that they're getting a high level of protection from malware, security exploits, and attacks.

Application signing

Android requires that all apps be digitally signed with a developer key prior to installation. APK key rotation, supported in Android 9.0, gives apps the ability to change their signing key as part of an APK update. To support key rotation, the APK signature scheme has been updated from v2 to v3 to allow old and new keys to be used.

Android uses the corresponding certificate to identify the application's author. When the system installs an update to an application, it compares the certificate in the new version with the one in the existing version, and allows the update if the certificate matches.

Android allows apps signed with the same key to run in the same process, if the apps so request, so that the system treats them as a single application. Android provides signature-based permissions enforcement, so that an application can expose functionality to another app that's signed with the same key. By signing multiple apps with the same key, and using signature-based permissions, an app can share code and data in a secure manner.

App permissions

Permissions protect the privacy of Android users and provide transparency about what resources or information apps wish to access. For apps to access system features, such as camera and the web, or user data, such as contacts and SMS, an Android app must explicitly request permission. These permission prompts are designed so the user has clear visibility into the request and the opportunity to approve or deny it.

A central design point of the Android security architecture is that no app, by default, has permission to perform any operations that would adversely impact other apps, the operating system, or the user. This includes reading or writing the user's private data (such as contacts or emails), reading or writing another app's files, performing network access, keeping the device awake, and others.

Apps that target API level 23 (Android 6.0) and above use runtime permissions. These dialogs request the user grant access to the specified permission. This approach streamlines the app install and update process, since the user does not need to grant permissions when they install or update the app. It also gives the user more control over the app's functionality; for example, a user could choose to give a camera app access to the camera but not to the device location. Users can revoke permissions at any time, even if the app targets a lower API level.

Android 8.0 and above also includes improvements to give users better control over the use of identifiers. Privacy-sensitive persistent device identifiers are either no longer accessible or gated behind a runtime permission. For example, APIs that access the Wi-Fi MAC address have been removed except on fully managed devices.

On enterprise devices, device management apps (DPCs) can deny permissions on behalf of the user using the setPermissionPolicy API, a feature of managed Google Play.

Google Play Protect

Google Play Protect is a powerful threat detection service that actively monitors a device to protect it, its data, and apps from malware. The always-on service is built into all devices that have Google Play, protecting more than 2.5 billion devices.

The Google Play Protect Verify Apps service scans devices once everyday for harmful behavior or security risks. If it detects an app containing malware, it notifies the user. Google Play Protect may also remove or disable malicious apps automatically as part of its prevention initiative and use the information it gathers to improve the detection of Potentially Harmful Applications (PHAs). In addition, the user can opt to have unknown apps sent to Google for analysis.

Google Play app review

The Play Store has policies in place to protect users from malicious actors trying to distribute PHAs.

Developers are validated in two stages. They are first reviewed when they create their developer account based on their profile and credit cards. Developers are then reviewed further with additional signals upon app submission.

Before applications become available in Google Play they undergo an application review process to confirm they comply with Google Play policies. Google has developed an automated application risk analyzer that performs static and dynamic analysis of APKs to detect potentially harmful app behavior. When Google's application risk analyzer discovers something suspicious, it flags the offending app, and refers it to a security analyst for manual review. Google suspends developer accounts that violate developer program policies.

A developer is notified immediately if their app is flagged for a security issue. They receive details about how to improve the app and links to support page details for additional guidance. This notification usually includes a timeline for delivering the improvement. In some cases, security improvements to apps must be made before a developer can publish any further updates.

Another key element in minimizing risk is the use of updated APIs. Encouraging developers to use the most recent APIs encourages support for the most updated features, creating optimal security and performance. Both new apps and app updates must target at least Android 9.0, or API level 28, to meet API requirements.

Every new Android version introduces changes that bring significant security and performance improvements – and enhance the user experience of Android overall. Some of these changes only apply to apps that explicitly declare support through their targetSdkVersion manifest attribute, also known as the target API level. Reference the Google Play Developers documentation for more details on updating to the proper target API level requirement.

SafetyNet

SafetyNet is a set of services and APIs that developers may use to protect apps against security threats. They can mitigate against device tampering, bad URLs, PHAs, and fake users.

The SafetyNet Attestation API is an anti-abuse API that allows app developers to assess the Android device their app is running on. This API provides a cryptographically-signed attestation, assessing the device's integrity. In order to create the attestation, the API examines the device's software and hardware environment, looking for integrity issues, and comparing it with the reference data for approved Android devices. The generated attestation is bound to the nonce that the caller app provides. The attestation also contains a generation timestamp and metadata about the requesting app.

The SafetyNet Safe Browsing API offers services to determine if a URL has been marked as a known threat by Google. SafetyNet implements a client for the Safe Browsing Network Protocol v4, developed by Google. Both the client code and the v4 network protocol were designed to preserve users' privacy and keep battery and bandwidth consumption to a minimum. Developers can use this API to take full advantage of Google's Safe Browsing service on Android in the most resource-optimized way, and without implementing its network protocol.

The SafetyNet service also includes the SafetyNet reCAPTCHA API, which protects apps from malicious traffic. This API uses an advanced risk analysis engine to protect apps from spam and other abusive actions. If the service suspects that the user interacting with the app might be a bot instead of a human, it serves a CAPTCHA that a human must solve before the app can continue executing.

The SafetyNet Verify Apps API allows an app to interact programmatically with Google Play Protect, to check whether there are known potentially harmful apps installed. If an app handles sensitive user data, such as financial information, developers should confirm that the current device is protected against malicious apps and is free of apps that may impersonate it or perform other malicious actions. If the security of the device doesn't meet the minimum security posture, developers can disable functionality within the app to reduce the danger to the user.

Data protection

Android uses industry-leading security features to protect user data. The platform provides developer tools and services to aid in securing the confidentiality, integrity, and availability of user data.

Encryption

Encryption protects user data if an Android device is lost, and is mandatory on all devices supporting Android 6.0 and higher. Android supports two methods for device encryption: file-based encryption (FBE) and legacy full-disk encryption.

File-Based Encryption

File-based encryption enables storage areas to be encrypted with different keys and has been available for use on devices since Android 7.0. New devices running Android 10 out of the box are now required to use file-based encryption as the default.

With FBE, the device boots directly to the normal lock screen, and the device is fully usable almost immediately when unlocked. Devices using FBE offer two kinds of storage locations to apps:

- Device Encrypted (DE) storage is available once the device boots, before the user unlocks the device. This storage is protected by a hardware secret and software running in the Trusted Execution Environment that checks that Verified Boot is successful before decrypting data.
- Credential Encrypted (CE) storage is available only after the user has unlocked the device. In addition to the protections on DE storage, CE storage keys can only be derived after unlocking the device, with protection against brute force attacks in hardware.

Most apps store all data in CE storage and run only after credentials are entered, but apps such as alarm clocks or accessibility services such as Talkback can take advantage of the Direct

Boot APIs and run before credentials are entered, using DE storage while CE is unavailable.

On devices with more than one user, each user has their own encryption keys, with CE keys bound to that user; this improves on FDE, which has only a single key bound to the first user, which unlocks all user data on the device. Encryption keys are 256 bits long and generated randomly on-device.

Devices running Android 9.0 and higher can use adoptable storage and FBE. An additional layer of encryption protects information, such as directory layouts, file sizes, permissions and creation/modification times (collectively this is known as file system metadata).

Android 9.0 also introduced support for metadata encryption of the main user data partition where hardware support is present, using a single key protected by Keymaster and Verified Boot.

Full-Disk Encryption

Devices running Android 5.0 to 9.0 may use full-disk encryption instead of file-based encryption. Full-disk encryption encodes all user data on an Android device using a single encryption key. As with file-based encryption, all user-created data is automatically encrypted before committing it to disk and all reads automatically decrypt data before returning it to the calling process.

Android full-disk encryption is based on dm-crypt, which is a kernel feature that works at the block device layer. The encryption algorithm is AES-128 with cipher-block chaining (CBC) and ESSIV:SHA256. The master key is encrypted with AES-128 via calls to the BoringSSL library. Some devices may use AES-256.

Upon first boot, the device creates a randomly generated 128-bit master key and then hashes it with a default password and stored salt. This hash is then passed through a keyed function based on RSA in the Trusted Execution Environment, to prevent offline password guessing. When the user updates their passcode, the hash is regenerated without regenerating the master key.

Backup encryption

Devices that run Android 9.0 and above can now support enhanced backup encryption, a new capability whereby the backed-up application data on a device can only be decrypted by a key that is randomly generated on that same device.

Then, the randomly generated decryption key is securely shared with a custom-built security chip known as Titan, which is located at a Google datacenter, together with a hash of the user's lockscreen PIN, pattern, or password. None of this data shared with the Titan chip is known to

Google, and the device verifies the identity of the Titan chip by checking its root of trust before sharing the data with the chip.

With this Titan chip, there is a limited number of incorrect attempts strictly enforced by the custom firmware, which cannot be updated without erasing the contents of the chip. By design, this means that no one (including Google) can access a user's backed-up application data without specifically knowing their passcode.

Android security updates

Monthly device updates are an important tool to keep Android users safe. Every month, Google publishes Android Security Bulletins to update users, partners, and customers on the latest fixes. These security updates are available for Android versions for three years from the date of release.

Android 8.0 re-architected the Android OS framework adding a feature called project Treble, which accelerates the delivery of security fixes, privacy enhancements, and consistency improvements. Launched with Android 8.0 in 2017, it has enabled OEMs and silicon vendors to develop and deploy Android updates faster than what was previously possible. All devices that launch with Android 9.0 and later are Treble-compliant and take full advantage of the Treble architecture.

Administrators of fully managed devices can install system updates via a system update file in Android 10 devices. With manual system updates, IT administrators can:

- Test an update on a small number of devices before installing them widely
- Avoid duplicate downloads on bandwidth-limited networks.
- Stagger installations, or update devices only when they're not being used.

OEM parter updates

Security-critical fixes are pushed to all Pixel devices monthly directly from Google's over-the-air servers. Pixel firmware images are also available on the Google Developer site for manual update and flashing. Many OEM partners follow a similar cadence in their security updates. Many OEMs also publish their security bulletins:

- Google
- Nokia
- Samsung
- LG
- Motorola

Users can find out whether they're running a recently patched device with the Security Patch Level, a value indicating the security patch level of a build. It's available through the attestation certificate chain, which contains a root certificate that is signed with the Google attestation root key. It is also visible in the device settings. EMM partners have the capability to call an API to detect which security update is installed and impose compliance rules for outdated devices.

Google Play System Updates

In Android 10, Google Play System Updates offer a simpler and faster method to deliver updates. Key Android system components are modularized, and end-user devices receive the components from the Google Play Store or through a partner-provided over-the-air (OTA) mechanism.

The components are delivered as either APK or APEX files — APEX is a new file format which loads earlier in the booting process. Important security and performance improvements that previously needed to be part of full OS updates can be downloaded and installed similarly to an app update. Updates delivered from Google Play System Updates are secured by being cryptographically signed.

Google Play System Updates can also deliver faster security fixes for critical security bugs by modularizing media components, which accounted for nearly 40% of recently patched vulnerabilities, and allowing updates to Conscrypt, the Java Security Provider.

Conscrypt

The Conscrypt module accelerates security improvements and improves device security through regular updates via Google Play System Updates. It uses Java code and a native library to provide the Android TLS implementation as well as a large portion of Android cryptographic functionality such as key generators, ciphers, and message digests. Conscrypt is available as an open source library, though it has some specializations when included in the Android platform.

The Conscrypt module uses BoringSSL, a native library that is a Google fork of OpenSSL and which is used in many Google products for cryptography and TLS (most notably Google Chrome). The version of BoringSSL that ships on Android 10 has gone through the National Institute of Standards and Technology's (NIST) Cryptographic Algorithm Validation Program (CAVP).

Adiantum

Adiantum is an encryption method designed for devices running Android 9.0 and higher whose CPUs lack AES instructions. This provides encryption to such devices with little performance overhead and enables a class of lower-powered devices to use strong encryption. The Android Compatibility Definition Document (CDD) reflects that all new Android devices be encrypted using one of the allowed encryption algorithms.

Device and profile management

Android Enterprise Device Use Cases

Android Enterprise offers APIs and other tools for developers to integrate support for Android into their enterprise mobility management (EMM) solutions.

Employee-owned devices: Personal devices can be set up with a work profile—a feature built into Android 5.1 and above that allows work apps and data to be stored in a separate, self-contained space within a device. An employee can continue to use their device as normal; all their personal apps and data remain on the device's primary profile.

The employee's organization has full management control over a device's work profile, but has no visibility or access to a device's personal profile. This distinct separation gives enterprises control over corporate data and security without compromising employee privacy.

Company-owned devices for knowledge workers: Organizations can exercise full management control over devices running Android 5.0 and above that they own and issue to employees. There are two deployment options available for these types of company-owned devices: **fully managed** (Android 5.0 and above) and **fully managed with a work profile** (Android 8.0 and above).

- Fully managed deployments are for company-owned devices that are used exclusively for work purposes. Organizations can enforce the full range of management policies on the entire device, including device-level policies that are unavailable to work profiles.
- Fully managed devices with work profiles are for company-owned devices that are used for both work and personal purposes. The organization still manages the entire device. However, the separation of work data and apps into a work profile allows organizations to enforce two separate sets of policies. For example:
    - A stronger set of policies for the work profile that applies to all work apps and data.
    - A more lightweight set of policies for the personal profile that applies to the user's personal apps and data.

Learn more about the capabilities available to device and profile owners, and about work profiles on fully managed devices.

Company owned devices for dedicated use: Dedicated devices are a subset of company-owned devices that serve a specific purpose. Android comes with a broad set of management features that allow organizations to configure devices for everything from employee-facing factory and industrial environments, to customer-facing signage and kiosk purposes.

Dedicated devices are typically locked to a single app or set of apps. Android 6.0 and above offers granular control over a device's lock screen, status bar, keyboard, and other key features, to prevent users from enabling other apps or performing other actions on dedicated devices.

Integrating Android

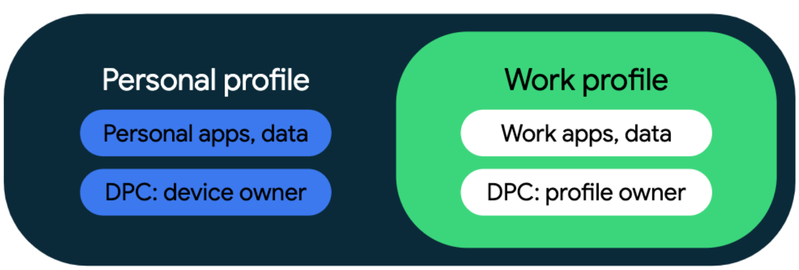
EMM Console: EMM solutions typically take the form of an EMM console—a web application that allows IT administrators to manage their organization, devices, and apps. To support these functions for Android, organizations integrate their console with the APIs and UI components provided by Android Enterprise.

Device Policy Controller: All Android devices that an organization manages through an EMM console must install a Device Policy Controller (DPC) app during setup. A DPC is an agent that applies the management policies set in an EMM console to devices. On a device with a work profile, the DPC controls the creation and policies of the work profile, or the profile owner (PO). A device that is fully managed, or a device owner (DO) profile that is device wide, is also controlled by a DPC.

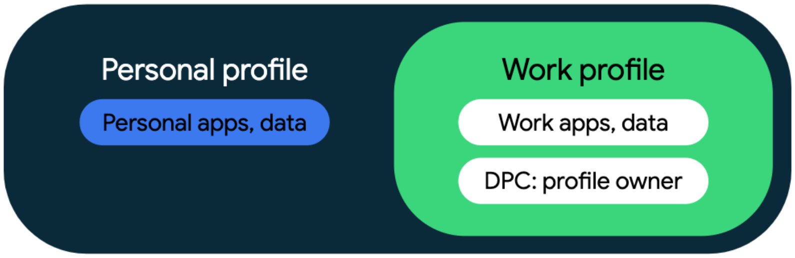
The DPC runs in one of two main modes:

1. **Device Owner:** runs in the primary profile and has the ability to manage a device in fully managed device mode. This is appropriate for company-owned devices.
2. **Profile Owner:** runs in and manages only the work profile.

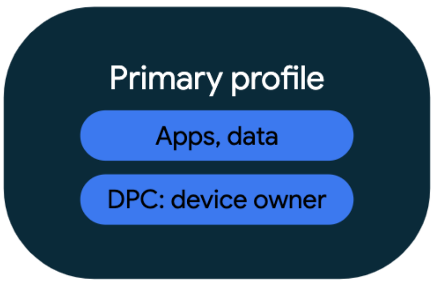
For example, a typical personally enabled work device configuration looks like this:



A typical BYOD configuration would not have a DPC running in the personal profile and looks like the following:



When only work apps and data are present on the device, such as in a typical dedicated device configuration, then only the primary profile exists with a DPC running.



Android 10 introduces new features and APIs for fully managed devices — manual system updates, extending QR code and NFC provisioning to include credentials for an EAP Wi-Fi network, and support for DNS over TLS.

OEMConfig

OEMConfig is an Android standard that enables device makers to create custom device features for immediate and universal support from EMMs. Instead of integrating enterprise APIs from each OEM to support their custom features such as control of barcode scanners or enabling extra security features, EMMs can easily use an OEM-built application that configures all of the unique capabilities of a device.

OEMConfig takes advantage of managed configurations, enabling developers to provide built-in support for the configuration of apps. For example, an app may have the option to only sync data when a device is connected to Wi-Fi. With such abilities, IT administrators can specify the managed configuration and apply them to devices.

The managed configurations iframe is an embeddable UI that lets IT administrators save, edit, and delete an app's managed configuration settings. Developers can, for example, display a button (or similar UI element) in an app's details or settings page that opens the iframe.

Within the iframe, an IT administrator can set configurations and save them as a configuration profile. Each time an IT administrator saves a new configuration profile, the iframe returns a unique identifier called mcmID. This makes it possible for IT administrators to create multiple profiles for the same app.

Device policies

Those developing a DPC or apps for managed Google Play can refer to Android Developers documentation for new APIs, features, and behavior changes.

Most capabilities available to the DPC are accessible via the DevicePolicyManager APIs and user restrictions in UserManager.

They can prevent sharing of files from the work profile or device, such as:

1. DISALLOW_BLUETOOTH_SHARING: disallows transferring files via Bluetooth.
2. DISALLOW_USB_FILE_TRANSFER: disallows sending files via USB.
3. DISALLOW_OUTGOING_BEAM: disallows beaming out data from apps using NFC.
4. DISALLOW_MOUNT_PHYSICAL_MEDIA: disallows mounting physical external media.

Device owner and profile owner mode also have a lot of control over other aspects of the device or profile. Below are some of the available policies:

1. DISALLOW_DEBUGGING_FEATURES: disallows access to debugging capabilities.
2. DISALLOW_AUTOFILL: disallows autofill services.
3. Setting device passcode policy using APIs such as setPasswordQuality().
4. Disabling less secure unlock methods using setKeyguardDisabledFeatures().
5. Disabling the camera using setCameraDisabled().
6. Setting permitted accessibility services using setPermittedAccessibilityServices().
7. Setting permitted input methods using setPermittedInputMethods().
8. Disabling screen capture using setScreenCaptureDisabled().
9. Automatically accepting/denying some runtime permissions with setPermissionPolicy().
10. If the device is lost, DPC can lock (lockNow()) or wipe (wipeData()) the device.
11. Disable backups using setBackupServiceEnabled().
12. Disallow adding a personal account using DISALLOW_MODIFY_ACCOUNTS. This makes it harder to copy corporate data to personal cloud accounts.
13. Require Google Play Protect to be enabled and enforce app verification across all users on the device using ENSURE_VERIFY_APPS.
14. Require only installing apps from known sources such as the Play store using DISALLOW_INSTALL_UNKNOWN_SOURCES.
15. Install keys and certificates into the profile-wide KeyChain using installKeyPair(), and control access to those keys. These can be used as machine certificates to identify the device.

16. Set always on VPN using setAlwaysOnVpnPackage().

As a general guide, DO controls the primary profile and PO controls the work profile. However, there are circumstances whereby PO can enable a global user restriction. Google provides an open-source app, Test DPC, for testing enterprise functionality in the DPC app. Test DPC is available from github or Google Play. The Test DPC may used to:

- Simulate features in Android
- Set and enforce policies
- Set app and intent restrictions
- Set up work profiles
- Set up fully managed Android devices

Fully Managed Device provisioning

The lifetime of the DPC is always tied to the lifetime of the device or profile it manages. IT managers, or the end user, must enroll a device into fully managed device mode, which provisions the device policy client as a device owner. Provisioning must occur during the initial setup of a new device, or after a factory reset. In the case of DO, it can only be provisioned during initial device setup and only be removed by the DO itself.

A number of options exist to provision a device into fully managed device mode:

- Zero-touch enrollment - after creating a configuration in the zero-touch portal, the IT administrator can ship a device directly to an end-user. Enrollment is automatic at first boot, or after factory reset, and is enforced to prevent the user from breaking out of the zero-touch enrollment process.
- NFC / QR code - An administrator provisioning large numbers of devices or an employee setting up their own single device can perform an NFC bump using a programmed NFC tag or scan a QR code to install the necessary DPC and initiate the enrollment process.
- G Suite or Cloud Identity account - With this provisioning method, the DPC guides the user through the provisioning steps after the user adds their Google Account during the initial device setup or via settings.

A device in fully managed mode can have a policy added that prevents a user from factory resetting a device.

Work profile security

Work profile mode is initiated when the DPC initiates a managed provisioning flow. The work profile is based on the Android multi-user concept, where the work profile functions as a separate Android user segregated from the primary profile. The work profile shares common UI real estate with the primary profile. Apps, notifications, and widgets from the work profile have a badge icon to distinguish them from the personal apps and notifications.

With the work profile, enterprise data does not intermix with personal application data. The work profile has its own apps, its own downloads folder, its own settings, and its own KeyChain. It is encrypted using its own encryption key, and it can have its own passcode to gate access.

The work profile is provisioned upon installation, and the user can only remove it by removing the entire work profile. Administrators can also remotely instruct the device policy client to remove the work profile, for instance, when a user leaves the organization or a device is lost. Whether the user or an IT administrator removes the work profile, user data in the primary profile remains on the device.

Android 10 introduces new provisioning and attestation features for company-owned devices that only require a work profile. During the provisioning of a company-owned device, a new intent extra allows DPCs to initiate work profile or fully managed setup. After a work profile is created or full management is established, DPCs must launch policy compliance screens to enforce any initial policies.

A device with a work profile can be configured with factory reset protection so that if the device is incorrectly reset, the organization has the ability to reset the factory reset protection, which is a feature that prevents device theft.

Separate work challenge

Devices running Android 7.0 and above support a separate work challenge to enhance security and control. The work challenge is a separate passcode that protects work apps and data. Administrators managing the work profile can choose to set the password policies for the work challenge differently from the policies for other device passwords. Administrators managing the work profile set the challenge policies using the usual DevicePolicyManager methods, such as setPasswordQuality() and setPasswordMinimumLength(). These administrators can also configure the primary device lock, by using the DevicePolicyManager instance returned by the DevicePolicyManager.getParentProfileInstance() method.

As part of setting up a separate work challenge, users may also elect to enroll fingerprints to unlock the work profile more conveniently. Fingerprints must be enrolled separately from the primary profile as they are not shared across profiles.

As with the primary profile, the work challenge is verified within secure hardware, ensuring that it's difficult to brute-force. The passcode, mixed in with a secret from the secure hardware, is used to derive the disk encryption key for the work profile, which means that an attacker cannot derive the encryption key without either knowing the passcode or breaking the secure hardware.

Cross profile data sharing

While data in the work profile is segregated by default from the user's personal data, there are instances where sharing is useful. Android allows sharing between profiles in ways that can be managed by the DPC. For example:

1. Disallow copy & paste between profiles: DISALLOW_CROSS_PROFILE_COPY_PASTE
2. Allow the primary profile to handle web links from the work profile: ALLOW_PARENT_PROFILE_APP_LINKING
3. Allow widgets from the work profile, such as a calendar widget, to be added on the home screen: addCrossProfileWidgetProvider()
4. Set whether work profile Caller ID is shown in primary profile: setCrossProfileCallerIdDisabled()
5. Set whether work profile contacts are shown in primary profile: setCrossProfileContactsSearchDisabled()
6. Set which apps can see notifications from the work profile: setPermittedCrossProfileNotificationListeners()
7. Set whether apps in the primary profile using the ACTION_SEND intent may share into the work profile using the DISALLOW_SHARE_INTO_MANAGED_PROFILE user restriction available as of Android 9.0. Note that, to reduce the risk of data leakage, the opposite direction is not allowed by default, though it can be enabled by the DPC.

IT administrators can also control cross profile intents using the addCrossProfileIntentFilter and clearCrossProfileIntentFilters methods available in Android 5.0 and higher. By default, during work profile creation the system automatically configures the following intents to be forwarded to the primary profile:

- **Telephony intents**: as of Android 7.0, administrators can also whitelist a dialer for work, which allows a "business" phone account to make and receive work calls instead of forwarding telephony intents to the primary profile dialer. In this case, all calls from the work dialer are inserted into the work call log.
- **Home intent**: to invoke the launcher in the primary profile since it doesn't run in the work profile.
- **Get content**: the user has the option to resolve in either the primary or work profile.
- **Open document**: the user has the option to resolve in either the primary or work profile.
- **Picture**: the user has the option to resolve in either the primary or work profile if an app that can handle camera exists in the work profile.
- **Set clock**: the user has the option to resolve in either the primary or work profile.
- S**peech recognition**: the user has the option to resolve in either the primary or work profile.

Application management

Android Enterprise provides IT administrators with powerful tools to deploy, configure and manage applications on a variety of device form factors.

Enterprise Mobility Management apps

The EMM app, acting as a DPC, controls which work apps may be installed. On a fully managed device, the EMM app can call the PackageInstaller APIs directly to silently install, uninstall, and update apps. It can also listen for broadcasts such as ACTION_PACKAGE_ADDED, ACTION_PACKAGE_REMOVED, and ACTION_PACKAGE_REPLACED to be notified of changes to installed apps.

On devices that ship with Google Play, an EMM can delegate app management to Google Play. Through managed Google Play, an enterprise version of Google Play, IT administrators can easily find, deploy, and manage work apps while ensuring that malware and other threats are neutralized.

Managed Google Play

Managed Google Play provides APIs to EMM vendors that allow them to manage apps on Android devices. Organizations can build a customized and secure mobile application storefront for their teams, featuring public and private applications, which can be delivered to devices directly from the managed Google Play store. This eliminates the need to sideload any applications to devices.

Installation of apps in either the work profile or on fully managed devices is possible via direct user request in the managed Google Play Store app (pull), or as a result of a call to the EMM API (push). The APIs provide functionality for use (indirectly) by EMM-managed enterprise administrators as follows:

- An IT administrator can remotely install or remove apps on managed Android devices. This action is limited to devices or profiles that are under management by the EMM.
- An IT administrator can define which users see which apps. A user running the Play Store app within the work profile only sees apps whitelisted for them. The user can install these apps, but not others.
- Enterprise administrators can see which users have apps installed or provisioned, and the number of licenses purchased and provisioned.

Managed Google Play also provides additional app management options for IT administrators. With the managed Google Play iFrame, administrators can approve apps in the managed Google Play store directly from the EMM console. By using managed configurations, administrators can whitelist specific apps for employee use, and selectively approve only the permissions they want their apps to use.

Additionally, administrators can enforce update preferences through managed Google Play. While the recommendation is for users to leave auto updates enabled, administrators can push an urgent update out to devices automatically.

Private apps

With managed Google Play, an enterprise customer can publish apps and target them privately (i.e., they're only visible and installable by users within that enterprise). Private apps are logically separated in Google's cloud infrastructure from Google Play for consumers. There are two modes of delivery for private apps:

- Google hosted: By default, Google hosts the APK in its secure, global data centers.
- Externally-hosted: Enterprise customers host APKs on their own servers, accessible only on their intranet or via VPN. Details of the requesting user and their authorization is provided via a JSON Web Token (JWT) with an expiry time. The JWT is signed by Google using the key pair associated with the specific app in Play, and should be verified before trusting the authorization contained in the JWT.

In both cases, Google Play stores the app metadata—title, description, graphics, and screenshots. In all cases, apps must comply with all Google Play policies.

Managed configurations

Managed configurations allow an organization's IT administrator to remotely specify settings for apps. This capability is useful for organization-approved apps that are deployed to a work profile. Managed configurations allow an IT administrator to remotely control the availability of features, configure settings, or set in-app credentials, via the Google Play EMM API. As an example, an app may have an option to only sync data when a device is connected to Wi-Fi, or whitelist or blacklist specific URLs in the web browser. Managed configuration options can be changed by the developer and updated in Google Play where the EMM will pickup on the changes for new and existing app deployments.

Google Chrome is an example of an enterprise-managed app that implements policies and configurations that can be fully managed according to enterprise policies and restrictions.

Applications from unknown sources

Administrators may need to prevent the installation of applications from outside Google Play, or apps from unknown sources. Devices and data can be at increased risk when such apps are installed from unverified sources.

To prevent the installation of apps from unknown sources, administrators of fully managed devices and work profiles can add the DISALLOW_INSTALL_UNKNOWN_SOURCES user restriction.

When the administrator of a work profile adds DISALLOW_INSTALL_UNKNOWN_SOURCES, the restriction only applies to the work profile. However, the administrator of a work profile can place a device-wide restriction by setting a managed configuration for Google Play.

Programs

A number of Google-backed initiatives and collaborations, help advance the Android ecosystem and support partners and customers in their use of Android in enterprise settings.

Android Enterprise Recommended

The Android Enterprise Recommended program sets an elevated standard for enterprise devices and services. Validated devices in the program meet a set of specifications for hardware, deployment, security updates, and user experience. Security updates are delivered to devices within 90 days of release from Google. In addition, OEMs receive an enhanced level of technical support and training. Organizations may select devices from the curated list with confidence that they meet a common set of criteria required for inclusion in the Android Enterprise Recommended program.

Additional key requirements include:

- Knowledge worker devices run Android 8.0 or above
- Rugged devices run Android 7.0 or above
- Support for bulk deployment methods
- Delivery of Android security updates within 90 days of release from Google (30 days recommended)

Learn more about the program's device requirements.

Android Security Rewards Program

The Android Security Rewards (ASR) program incentivizes researchers to find and report security issues, providing key assistance to Android security efforts. This program covers security vulnerabilities discovered in the latest available Android versions for Pixel phones and tablets.

Google Play Security Rewards

The Google Play Security Reward Program is available for researchers to report any vulnerabilities discovered in apps hosted on Google Play. All Google's apps are included in the program, and developers of popular Android apps are invited to opt-in.

App Security Improvement Program

The App Security Improvement Program is a service that helps Google Play developers improve the security of their apps. The program provides tips and recommendations for building more secure apps and identifies potential security issues and mitigations when apps are uploaded to Google Play.

App Defense Alliance

The App Defense Alliance is a collaboration between Google and ESET, Lookout, and Zimperium. It was created to further enhance the safety of the Google Play Store by working with partners to quickly find PHAs and take action to protect users.

Integrating Google Play Protect detection systems with each partner's scanning engines generates new app risk intelligence as apps are being queued to publish. Partners will analyze that dataset and act as another set of eyes prior to an app going live on the Play Store.

Industry Standards and Certifications

Android has received numerous security certifications which demonstrate our strong commitment to the highest security standards.

SOC certification

Android Enterprise has received ISO 27001 certification and SOC 2 and 3 reports for information security practices and procedures for Android Management API, zero-touch enrollment and managed Google Play. This designation ensures these services meet strict industry standards for security and privacy.

Granted by the International Organization for Standardization, ISO 27001 outlines the requirements for an information security management system. It specifies best practices and details a list of security controls regarding information risk management.

The SOC 2 and 3 reports are based on American Institute of Certified Public Accountants (AICPA) Trust Services principles and criteria. To earn this, auditors assess an organization's information systems relevant to security, availability, processing integrity and confidentiality or privacy.

An independent assessor performed a thorough audit to ensure compatibility with the established principles. The entire methodology of documentation and procedures for data management are reviewed during such audits, and must be made available for regular compliance review.

Learn more about these security designations.

Government Grade Security

FIPS 140-2 CAVP

Federal Information Processing Standards (FIPS) are standards and guidelines for Federal computer systems that are developed by the National Institute of Standards and Technology (NIST) in accordance with the Federal Information Security Management Act (FISMA) and approved by the Secretary of Commerce. Although FIPS are developed for use by the federal government, many in the private sector voluntarily use these standards as well. The National Institute of Standards and Technology's (NIST) Cryptographic Algorithm Validation Program (CAVP) provides validation testing of approved cryptographic algorithms and their individual components.

Common Criteria/NIAP Mobile Device Fundamentals Protection Profile

Common Criteria is a driving force for the widest available mutual recognition of security products with 31 participating countries. The National Information Assurance Partnership (NIAP) serves as the U.S. representative to the Common Criteria Recognition Arrangement (CCRA). The U.S. government uses the NIAP to certify devices for official use. Google Pixel 3 and Pixel 3 XL have achieved this strict certification. This certification process has enabled the Android team to build some of the requirements to achieve this certification directly into the Android Open Source Project, which enables OEMs the ability to attain certification in much less time.

DISA Security Technical Implementation Guide (STIG)

The Security Technical Implementation Guides (STIGs) are the configuration standards for Department of Defense Information Assurance (IA) and IA-enabled devices/systems. The STIGs contain technical guidance to "lock down" information systems/software that might otherwise be vulnerable to a malicious computer attack. The Google Pixel Android 10 STIG provides a standard implementation for configuring and locking down any Android device using Android Enterprise management controls.

Conclusion

The open source development approach of Android is a key part of its security. Developers, device manufacturers, security researchers, SoC vendors, academics, and the wider Android community create a collective intelligence that locates and mitigates vulnerabilities for the entire ecosystem.

With Android, multiple layers of security support the diverse use cases of an open platform while also enabling sufficient safeguards to protect user and corporate data. Additionally, Android platform security keeps devices, data, and apps safe through tools like app sandboxing, exploit mitigation and device encryption. A broad range of management APIs gives IT departments the tools to help prevent data leakage and enforce compliance in a variety of scenarios. The work profile enables enterprises to create a separate, secure profile on users' devices where apps and critical company data are kept secure and separate from personal information.

Google Play Protect, the world's most widely deployed mobile threat protection service, delivers built-in protection on every device. Powered by Google machine learning, it works to catch and block harmful apps and scan the device to root out any PHAs or malware. Google Safe Browsing in Chrome protects enterprise users as they navigate the web by warning of potentially harmful sites.

Enterprises rely on smart devices for critical business operations, collaboration, and accessing proprietary data and information. Google continues to invest in resources to further strengthen the security of the Android platform, and we look forward to further contributions from the community and seeing how organizations will use Android to drive business success.