Android Security Paper

2026

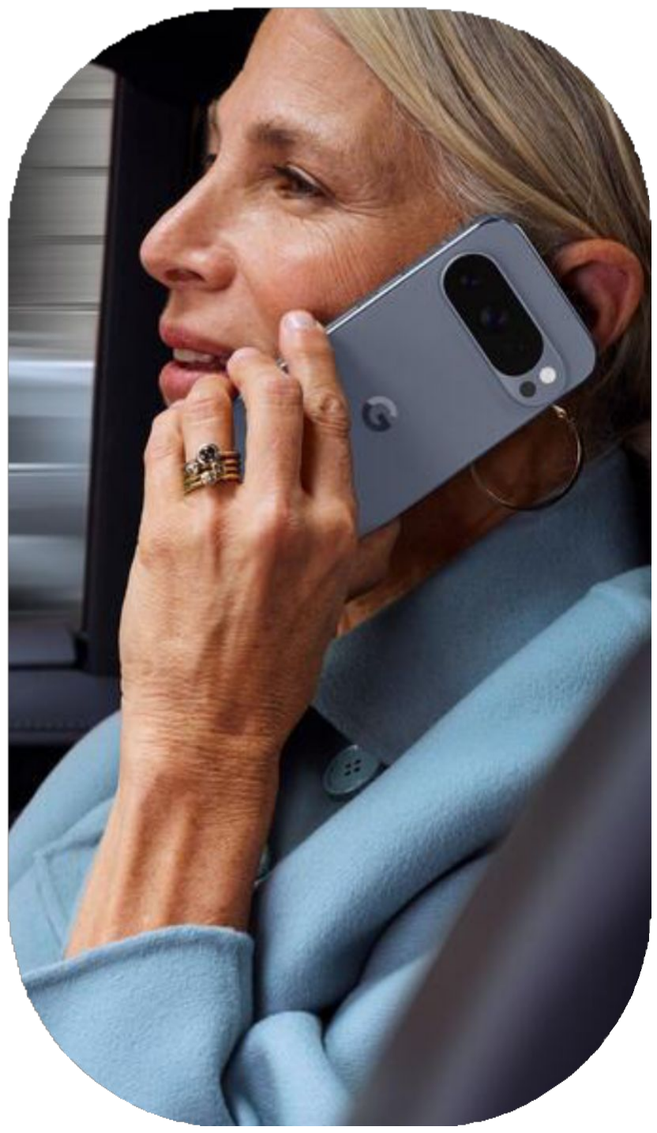
Contents

About the Android Security Paper

In today's modern world, mobile devices play a critical role in both our work and personal lives.
They accompany us everywhere — whether at home, on the go, or at the office — which means protecting them against cyber threats has never been more important.

That's why we've updated the Android Security Paper detailing our latest security measures designed to help protect your devices.

We're firmly committed to helping you navigate the complexities of the modern digital landscape. By combining Zero Trust principles, enhanced privacy features, and advanced security capabilities, Android continues to set the standard for a secure and user-friendly mobile platform.



About Android

When Android was being developed, the state of the art in consumer operating system security was provided by memory management systems.

Both Windows and Unix workstations had protected memory features that were used to provide robust security between users of a device: multiple users could have their own logins to the device, with fully protected separation of all their apps and data, as if they were on different devices.

However, within an individual user's logged-in session, security was more limited. Protected memory was used to prevent applications from gaining highly privileged access to the kernel or accidentally corrupting each other, but with very little security between the applications themselves. A single user could install applications on their device that would function independently from other users, while still having substantial access to the device's capabilities.

> In these operating systems, installing an application essentially gave it the same privileges as the user. This posed a security risk, as users had to implicitly trust every application they installed. The web addressed this problem by treating web pages as inherently untrusted, limiting their capabilities.

Compare using a web-based email service versus installing an email app. You can safely try a web-based email with little worry, as it has limited access to your device unless you explicitly grant it more. Installing an email app, however, requires careful consideration. You're essentially trusting the app developer with potentially gaining full access to your device. Even after uninstalling the app. in some cases, your device might not function the same way it did before.

The lack of security in traditional desktop apps was already pushing users away. This issue was even more critical for mobile devices, which were becoming increasingly central to people's lives. Users were less willing to trust apps with full access to their personal data.

Several approaches emerged to address the growing security concerns on mobile platforms:

Leverage Web Technology

This option utilized the web's robust security model. However, it required significant modifications to support mobile application functionalities and enhance security, It also lacked support for native code, posing challenges for performance-intensive apps like games. Additionally. web technology, initially designed for desktops, was too resource-intensive for mobile hardware at the time.

Adopt a Virtual Environment

Technologies like Java offered a closer fit for mobile app development. However, they shared a critical limitation with the web-based approach: apps had to be written in a specific language, and native code was unsupported.

Implement Security Gatekeeping

This involved code signing, app source restrictions, or other methods to ensure users could only install safe apps. Apps could still run with extensive device access, similar to desktop apps, and could utilize native code. The key difference was a gatekeeper vetting all apps before installation. This approach was gaining traction in carrier stores and other distribution channels. Android aimed to be an open and secure platform, not just for mobile devices, but for a wide variety of devices. This meant more than being open source; Android also wanted to emulate the openness of traditional desktop operating systems, Users should be able to freely install apps without fear of malicious activity. Android's vision was clear: if mobile was the future of computing, controlling app installations and compromising security and privacy should not be in the hands of a few powerful entities.

The goal of an open Android platform ruled out the option of restricting app sources for security.

Instead, Android itself has a stronger security model; preventing installed apps from gaining excessive control over the device. To create an open and secure Android platform, we needed to allow native code in apps. Relying solely on virtualized languages was not feasible, as it limited the performance needed for gaming, media, and other demanding applications. Moreover, these virtualized languages often had security vulnerabilities due to the complexity of managing multiple security domains.

> Therefore, Android implemented a security solution based on the Linux kernel, utilizing protected memory and a unique process isolation mechanism.

Another developing area in operating systems was capability-based systems where processes initially have no access rights (to things like files or hardware). They are then granted specific capabilities depending on their intended function. However, at that time, these implementations were mainly research projects or tailored for embedded systems, not general-purpose operating systems.

Widely used for years in millions of security-sensitive environments, Linux has become a stable and secure kernel trusted by many corporations and security professionals. This is due to continuous research, attacks, and fixes by thousands of developers. Modern Android devices must use the latest long-term support (LTS) kernel, which receives regular security updates and bug fixes.

The traditional relationship between users, apps, and system services in Linux is represented in the image:

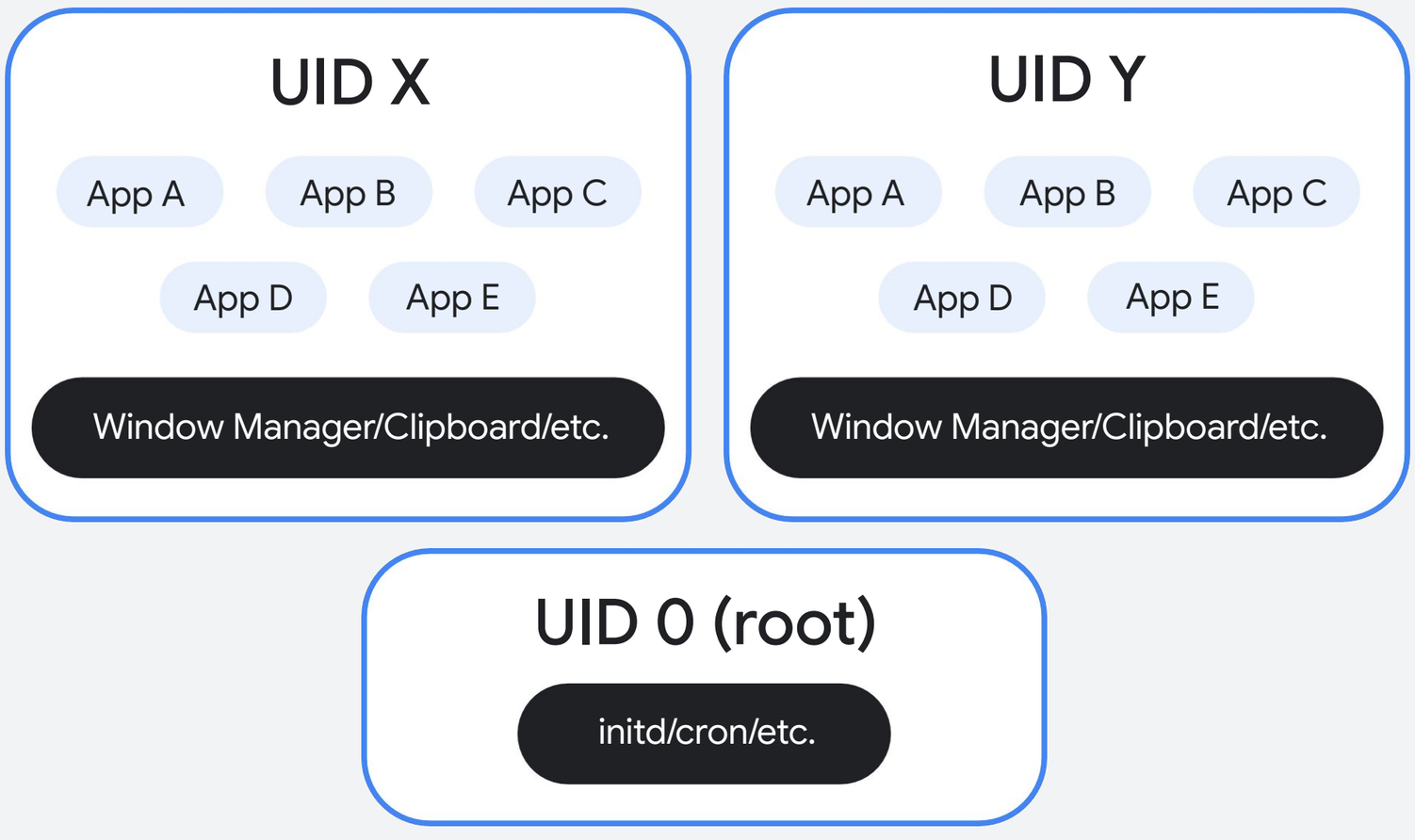


Figure 1. Relationship between users, apps, and system services in Linux.

Rather than treating applications as full-fledged users, Android assigns a UID to each app. This ensures apps remain isolated from each other at the kernel level. Android also limits the amount of code running with root privileges (UID 0).

For instance, a dedicated service called installd handles essential filesystem management, and most system components operate within their own isolated UID environments.

The resulting security architecture looks like this:

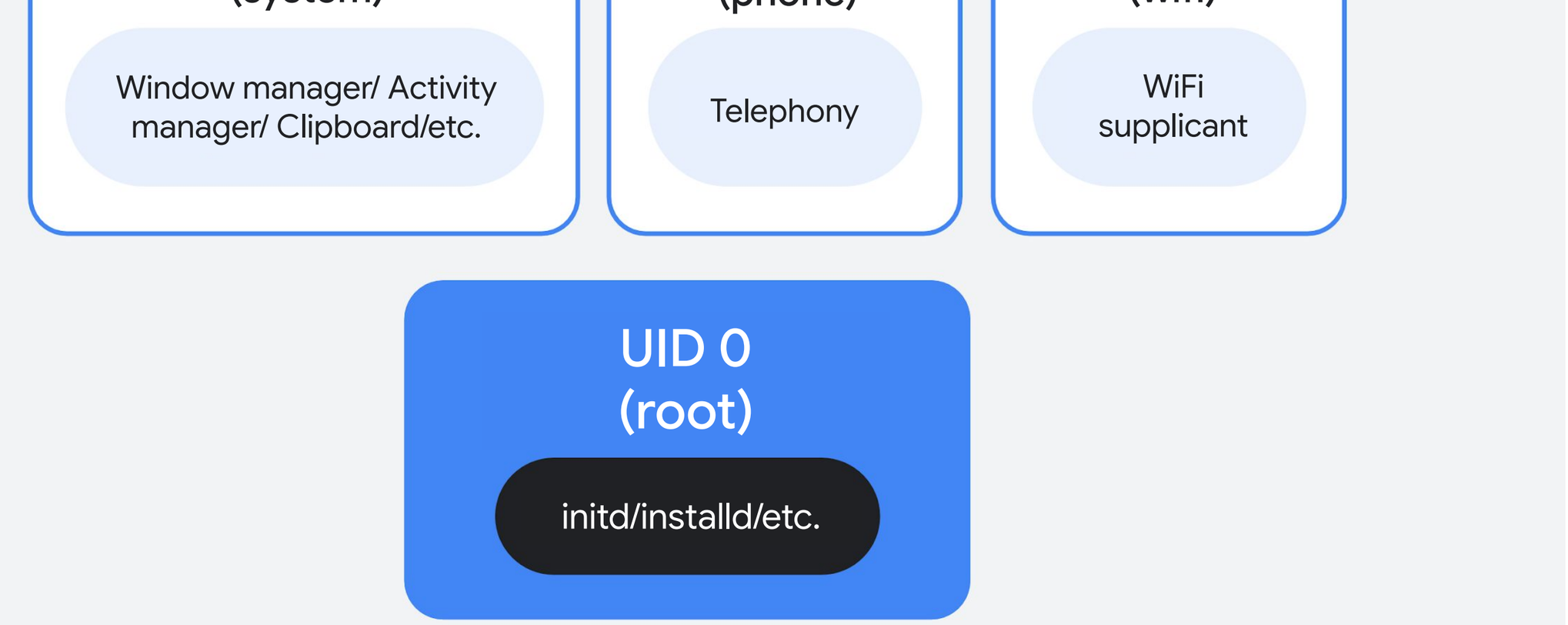


Figure 2. Security architecture

Any interactions between UIDs must be explicitly allowed by Android, much like a capability-based system.

To define the sandboxes for applications, there must be the concept of a secure identity for that application. A well-established way to do this is with a signing certificate, where the application provides a public key identifying who its source is, and a private key to sign the code. The operating system then uses the public key to verify that the application came from the author by verifying it against its signing certificate. In traditional implementations of code signing, the public certificate is chained to a root certificate. This root certificate can come only from a limited set of Certificate Authorities who are responsible for controlling the public certificates charged to them, so their identities can be trusted.

In the development of mobile platforms before Android's creation, the Certificate Authority was often the platform owner, device manufacturer, carrier, or some combination of the three. For each app being installed, the platform would verify the app was signed by a certificate granted by one of these certificate authorities, effectively making them gatekeepers for which apps the user was allowed to install.

Android deliberately avoids built-in gatekeepers. Instead, it relies on a robust system to securely identify apps. While apps still use cryptographic certificates, there's no central authority verifying them. Each app's certificate is self-contained, only confirming if two apps share the same author, not who the author is.

This lets Android ensure key security measures like 'this app update is from the original developer,' without external restrictions on app authorship.

Apps are typically designed to run within the Android Runtime and interact with the operating system through a framework that defines system services, platform APIs, and message formats. Developers can use various programming languages, like Java, Kotlin, C/C++, and Rust, all operating within the same application sandbox. Importantly, Android's security model isn't affected by the choice of language. Both native and VM code within an app share the same sandbox and have the same security restrictions.

The overall layers of the Android software stack can be visualized as below; though the actual security sandboxing layers are much more fine-grained.

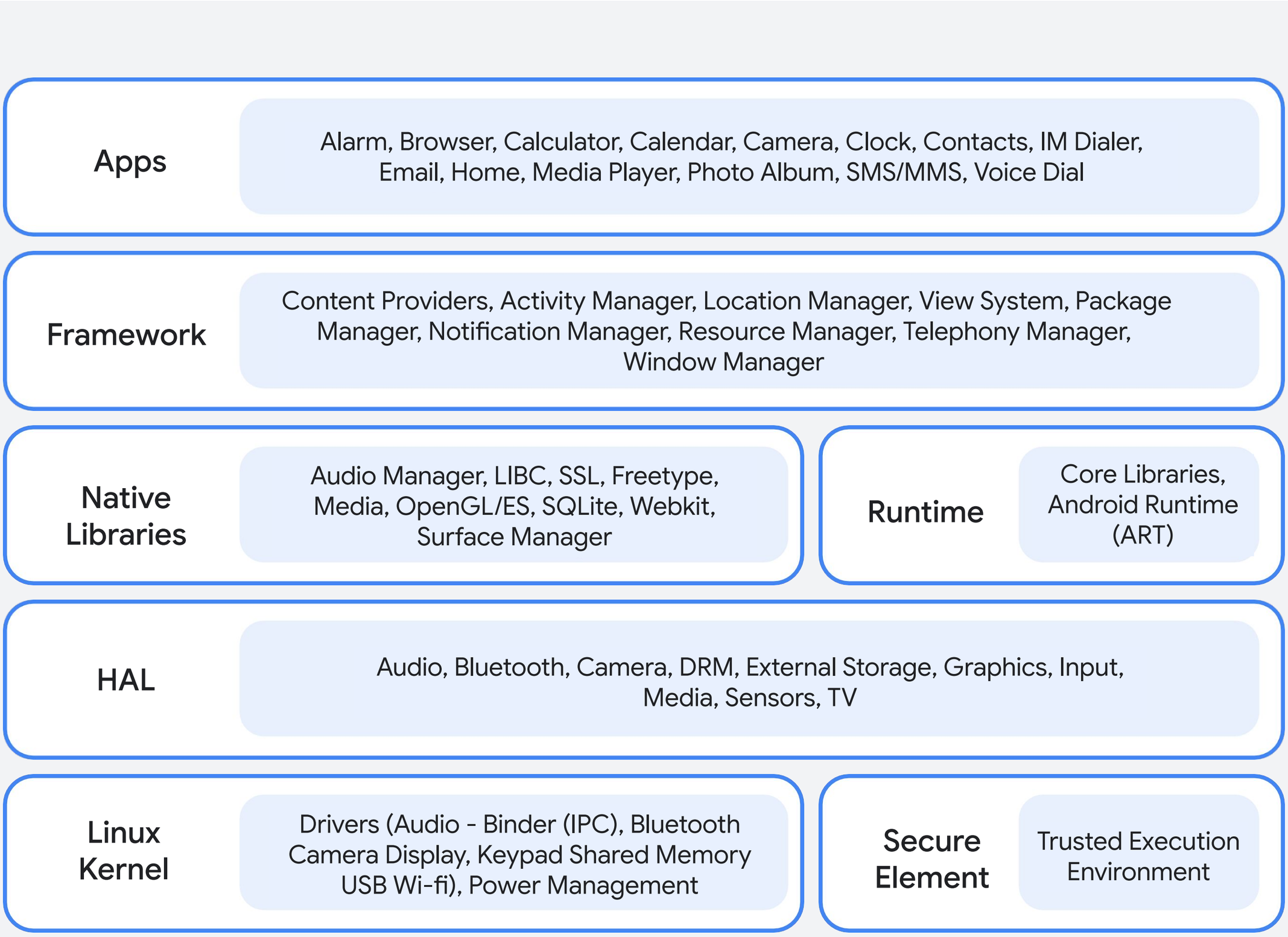


Figure 3. The Android software stack

Android is an open-source software stack designed for a wide array of devices and form factors. It incorporates industry-leading security features, and the Android team collaborates with developers and device manufacturers to maintain a safe platform and ecosystem. A robust security model is vital for fostering a thriving ecosystem of apps, devices, and cloud services built on and around Android.

Consequently, Android evolves by continually undergoing rigorous security testing throughout its entire development lifecycle.

Applications running on Android are signed and isolated within application sandboxes linked to their signature. The application sandbox defines the privileges available to each app.

What's new in Android 16

Empowering the Modern Mobile Workplace

Security Enhancements

Android 16 focuses on hardening the entire device and protecting against increasingly sophisticated threats like physical theft and scams.

Advanced Protection (Device-Level)

This is arguably the most significant security update. It's a full-system security mode that can be enabled with a single toggle.

Hardened Device
It blocks the sideloading of apps, ensures that scam and web protections are on and cannot be disabled, deactivates 2G networks (which are susceptible to interception), and is designed to help prevent the device from reconnecting to known insecure Wi-Fi networks.

USB Data Lockdown
It turns off USB data transfers unless the phone is unlocked.

Theft Protection
It enforces an inactivity reboot if the device is locked for an extended period (e.g., 72 hours).

Identity Check

This feature, which is rolling out more broadly, requires biometric authentication (like fingerprint or face unlock) for highly sensitive actions, even if an attacker knows your PIN/password.

Protected Actions
This includes changing your device PIN, modifying saved passkeys, disabling theft protection, or accessing critical Google account settings.

Location-Aware
The extra authentication can be set to only kick in when you are **outside a trusted location** (like your home), reducing inconvenience while maximizing security when you're out.

Enhanced Anti-Scam Protection

AI-powered on-device behavioral analysis is being integrated across calls and messaging apps to protect you from social engineering scams.

In-Call Protections
The system will now block high-risk actions—like granting Accessibility permissions to a newly downloaded app, disabling Google Play Protect, or sideloading an app—while you are on a call with a number not in your contacts.

Expanded Scam Detection
Detection has been expanded to cover a wider variety of sophisticated scams (e.g., crypto scams, financial impersonation, tech support scams).

Privacy Controls

The focus on privacy shifts toward contextual intelligence and tighter control over permissions and sensitive data displayed.

AI-Powered Contextual Privacy Controls

Android 16 introduces an AI framework that analyzes your usage patterns and context to dynamically adjust an app's data access.

Note: The accuracy of AI-driven redaction and access controls may vary based on app behavior and user context.

Adaptive Consent
For example, a navigation app might get unrestricted location access during your daily commute, but the system could prompt you for re-confirmation if the app requests location data unexpectedly at midnight.

Transparency
Users are provided with detailed logs explaining why the AI restricted or granted access, and they have the option to override the AI's recommendations.

Smart Redaction on Lock Screen

The system uses AI to detect sensitive information (like **OTPs/one-time passwords**, banking alerts, or personal messages) in notifications and automatically **hides/redacts** them on the lock screen unless the phone has been recently unlocked or is in a low-risk scenario.

Hardware-Backed Zero-Trust Architecture (Enterprise-Focused)

For corporate and high-security users, Android 16 moves towards enforcing a zero-trust model at the hardware level, ensuring that every request—from the device, app, or user—must be validated before granting access.

Quantum-Resistant Cryptography

The platform is being upgraded with new algorithms to strengthen defenses in communication and stored data against theoretical attacks from future quantum computers.

These new security and privacy features are critically important for enterprise customers as they directly address the biggest risks facing corporate mobility: **physical theft, social engineering, and maintaining strict compliance.**

Here is a breakdown of why each major feature is vital for the enterprise environment:

Advanced Protection

In an enterprise context, this feature acts as an enforced baseline for maximum security, directly mitigating multiple attack vectors.

Mitigates Targeted Attacks (High-Risk Users)

For executives, IT admins, and employees handling highly sensitive data (i.e., high-risk users), the inability to sideload apps or connect to insecure networks is a non-negotiable security requirement. It ensures the device resists sophisticated, targeted malware attacks.

Prevents Data Exfiltration (USB Data Lockdown)

A common method of data theft involves an attacker quickly connecting a stolen/lost device to a computer to pull data. By requiring the phone to be unlocked for any USB data transfer, the feature prevents rapid data extraction, giving the IT security team more time to remotely wipe the device.

Identity Check (Biometric Re-authentication)

This feature is essential for protecting the corporate identity and the device's role as a trusted authenticator.

Countering PIN/Shoulder-Surfing Theft

If a phone is stolen, the thief often observes the user entering their PIN first. Identity Check reduces the utility of a compromised PIN in unauthorized hands for critical, sensitive actions (like changing the screen lock, accessing saved corporate passkeys, or factory resetting the device), as it requires the user's **biometric signature** as a second factor.

Securing the Credentials Vault

Enterprise environments are increasingly using passwordless solutions like **passkeys**. Identity Check ensures that access to these high-value passkeys is protected by hardware-backed biometrics, preventing an attacker who has the device PIN from compromising multiple corporate accounts.

Conditional Access (Zero-trust)

By linking the extra biometric requirement to location (e.g., outside a trusted office network), it integrates with a **Zero-Trust** security model. The device's trust level is dynamically assessed based on its context, ensuring that access to sensitive corporate apps or credentials is only granted after the highest level of user assurance.

Enhanced Anti-Scam Protection

Social engineering is one of the top causes of corporate breaches. This feature is a powerful, on-device defense layer.

Blocks Social Engineering Exploits

Scammers frequently instruct victims to grant **Accessibility Services** permissions to malicious apps, which then hijack the phone. By blocking this high-risk action specifically while the user is on a call with an unknown number, the feature prevents the most common form of mobile device takeover used in corporate espionage and financial fraud.

Reduces End-User Risk (The Human Factor)

This protection minimizes the risk of human error by building ‘guardrails’ directly into the OS, protecting employees from their own potential lapse in judgment under pressure from a convincing scammer.

Contextual Privacy and Redaction

These features are crucial for managing data leakage and simplifying compliance with regulations (like GDPR or CCPA).

Prevents Data Leakage from Notifications

The Smart Redaction on Lock Screen protects sensitive corporate data (e.g., internal server alerts, confidential email snippets, OTPs for corporate systems) from being read by unauthorized people through ‘shoulder-surfing’ while the device is locked or in a public space.

Ensures Compliance

The move toward **AI-Powered Contextual Privacy** gives IT teams a future-proof framework to ensure that apps, particularly those handling regulated data (finance, health), are only accessing resources when absolutely necessary and contextually appropriate, aiding in compliance audits.

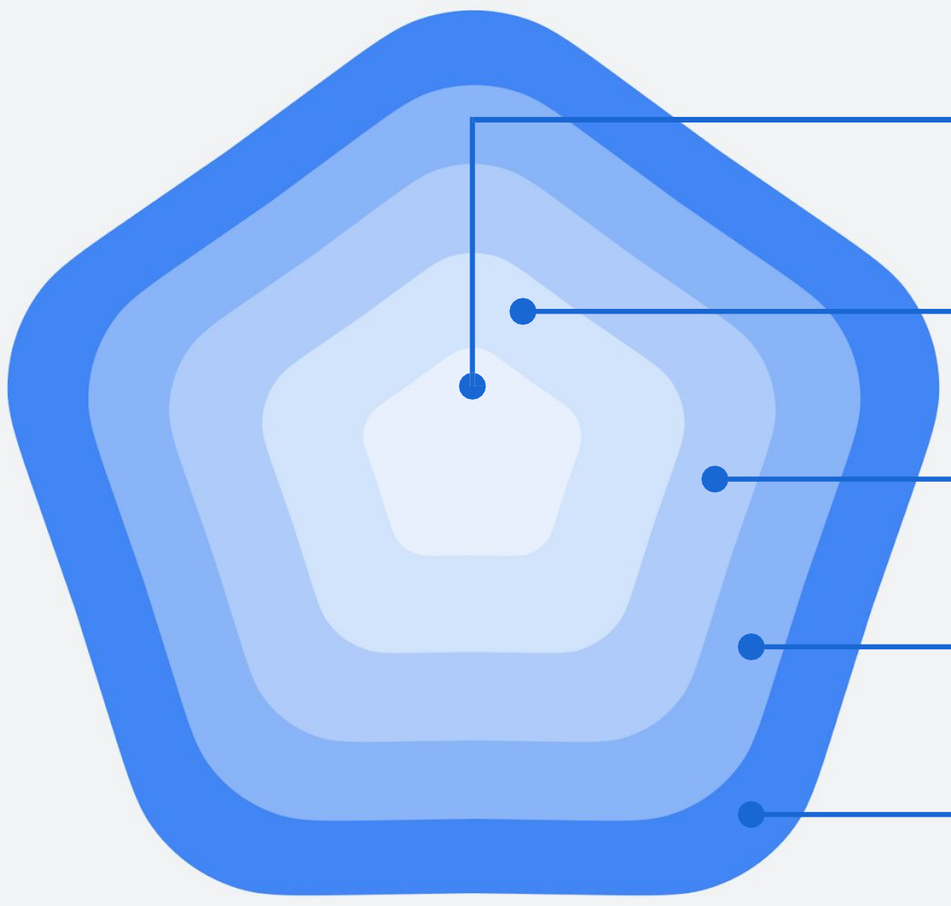
In short, Android 16 makes managed Android devices **more resilient to physical theft, more resistant to sophisticated malware, and better equipped to handle the human element of security risks,** all of which are paramount for protecting corporate intellectual property and maintaining a compliant security posture.

Security by design

Android uses a combination of hardware and software protections to create strong defenses. Security starts at the hardware level with lock screen credentials and/or combined with biometrics. Once installed, the Android operating system is unchangeable, or immutable. This means attackers cannot permanently modify the system.

For example, Verified Boot ensures the integrity of the system software as a device boots, while hardware-assisted encryption and key management protect data at rest.

Layers of trust



Firmware Integrity and Trust
Key management, Attestation, TrustZone, Baseband Security, Verified boot

OS Integrity and Trust
Verified boot, defense-in-depth

Application Integrity and Trust
Malware prevention, App sandboxing, and safe APIs

User Integrity and Trust
Authentication, Biometrics

Supply Chain Security
Vulnerability remediation, Remote key provisioning

Figure 4. Security by design

Built-in software protection is crucial for Android device security.

Google Play Protect, the world's most widely used threat detection service, actively scans over 340 billion apps daily for harmful behavior. This scanning covers all applications, including public apps from Google Play, system apps updated by device manufacturers and carriers, and even apps installed from sources other than the official app store.

Application sandboxing isolates and protects Android and all user apps and data, preventing malicious apps from accessing private information. Android also safeguards access to internal operating system components, making it harder to exploit vulnerabilities. Mandatory, always-on encryption helps keep data safe, even if a device is lost or stolen. Encryption is further protected by Keystore keys, which store cryptographic keys securely, making them difficult to extract. Developers can easily leverage the Android Keystore capabilities by using Jetpack, a suite of libraries to help developers follow best practices, reduce boilerplate code, and write code that works consistently across Android versions safely.

Overall, Android uses both hardware and software capabilities to keep devices safe and data private.

Android’s security is based on a design philosophy called **Defense-in-Depth**, which layers multiple independent security controls to ensure that the failure of one does not lead to the compromise of the entire system.

This model is engineered from the hardware foundation up through the operating system and application layers.

Core Philosophy: Defense-in-Depth

The fundamental philosophy behind Android’s security is separation of privilege, code integrity, and user control of every action and every piece of code. This is achieved by:

Least Privilege

Every process (especially third-party apps) is granted only the minimum permissions necessary to perform its legitimate function. By default, applications have no privileges outside their own data.

Isolation and Containment

Every app runs in its own **sandbox**, preventing unauthorized interaction and limiting the damage an exploited app can cause.

System Integrity

Hardware and software checks ensure that the device’s code and data have not been tampered with from the moment the device powers on.

Key Components of the Secure Architecture

Android’s security model is implemented through several interconnected components:

The Application Sandbox

This is the most critical element of app security. The Linux kernel’s user-based protection is repurposed to isolate apps.

Process Isolation

Each Android application is assigned a unique **User ID (UID)** and runs as a separate process. The Linux kernel enforces this separation, meaning App A cannot read or write to App B’s data directory or memory space.

Data Isolation

By default, files created by an application are accessible only to that application’s UID.

Mandatory Access Control (SELinux)

Security-Enhanced Linux (SELinux) is a system-wide mandatory access control (MAC) layer that restricts permissions even for processes running as root or with high privileges.

Containment Policy

SELinux defines a strict policy dictating what every process is **allowed** to access (files, network sockets, other processes). This policy is **mandatory** and cannot be bypassed by a malicious app.

Hardware-Backed Security: TEE and Keystore

Android leverages hardware-based security for handling the most sensitive data.

Trusted Execution Environment (TEE)

The TEE is a physically isolated processing environment on the device's main processor. It runs a secure OS (like Trusty) separate from the main Android OS.

Android Keystore

This API leverages the TEE to securely generate, store, and manage **cryptographic keys**. Crucially, the key material **never leaves the TEE**. Keys can be used for encryption/decryption by the main OS, but they cannot be stolen, making them highly resistant to software-based attacks. This is essential for features like File-Based Encryption and biometric authentication.

Integrity and Chain of Trust

These mechanisms ensure the device boots and runs trusted software.

Verified Boot

Establishes a **chain of trust** starting from a hardware-protected root of trust. It cryptographically verifies that every stage of the boot process (bootloader, kernel, system partition) is uncorrupted and comes from a trusted source (the OEM). If tampering is detected, the device will be blocked from booting or enter a limited-function mode.

File-Based Encryption (FBE)

Modern Android encrypts user data at the file level. FBE uses encryption keys that are tied to the user's lock screen credentials, ensuring that user data at rest is secure even if an attacker gains physical access to the storage.

Application Lifecycle Security

Security is enforced before and after an app is installed.

App Signing

Every application must be signed. This signature identifies the author and allows the system to verify that the app hasn't been altered or corrupted since it was published.

Google Play Protect

This integrated service scans all apps (on the Play Store and on-device) for malware and Potentially Harmful Apps (PHAs). It can remotely disable or remove harmful applications, providing an essential **real-time protection layer.**

Hardware-backed security

Android leverages underlying hardware features to enable strong device security.

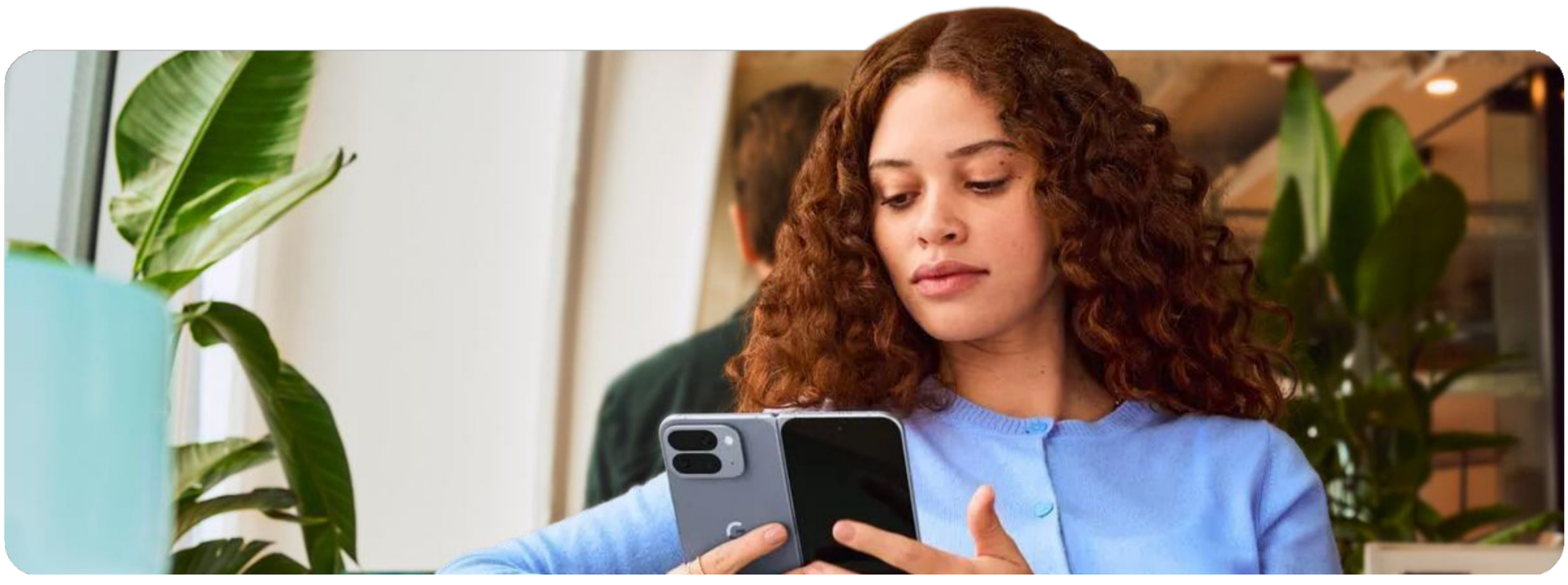
Trusted Execution Environment

The ARM based processor on Android devices provides a Trusted Execution Environment (TEE), often implemented as TrustZone. This secondary, isolated environment virtualizes the main processor, creating a secure trusted execution environment for confidential operations like device unlock, credential verification, and biometric operations.

Android's main operating system, the Rich Execution Environment (REE), is considered ‘untrusted.’ It cannot access sensitive areas of RAM, hardware registers, or write-once fuses where manufacturers store secret data, such as device-specific cryptographic keys. Any operations on a device requiring this data are delegated to the TEE.

A TEE environment consists of a small, separate operating system (TEE OS) and mini-apps that provide critical security services to Android. Although running on the same processor, ARM hardware isolates the Android kernel and apps from the TEE. This hardware-enforced isolation adds another layer of defense, protecting critical user data and device secrets. Google Pixel devices utilize the open-source TEE OS called Trusty.

Only the TEE can access device-specific keys needed to decrypt protected content. The REE only sees encrypted content, providing strong security and protection against software-based attacks.



The TEE is vital for various security-critical operations, including:

Lock screen passcode verification
Available on devices with secure lock screens, this is handled by the TEE unless a more secure environment, like the Titan M security chip, is present.

Fingerprint template matching and Face Unlock
Available on devices with fingerprint sensors and secure camera hardware.

KeyStore key protection and management
Available on devices with secure lock screens.

Digital Rights Management (DRM)
An extensible framework for apps to manage rights-protected content.

Protected Confirmation
Uses a hardware-protected Trusted UI for high-assurance transactions, an optional feature for Android devices.

> The TEE's role in handling these sensitive operations underscores its importance in maintaining Android's overall security.

Protected Confirmation

Android Protected Confirmation leverages a hardware-protected user interface (Trusted UI) to perform critical transactions outside the operating system on modern Android devices.

This is very important to help confirm a user's intentions, for example, when they initiate a sensitive transaction like making a banking payment or approving an insulin pump injection. App developers can utilize the feature to display a prompt to the user, asking them to approve a short statement that reaffirms their intent to complete a sensitive transaction.

If the user approves the action, the app can use a key from the Android Keystore to sign the message shown in the prompt that is cryptographically authenticated and tamper-proof when conveyed to the relying party. This provides users with more assurance that a critical action has been executed securely and helps developers verify a user's intent with a very high degree of confidence. Transactions utilizing Protected Confirmation have higher protection and security relative to other forms of secondary authentication, for example, an SMS code.

Android Keystore System and Related Features

Android Keystore System

The **Android Keystore System** handles the generation, storage, and usage of cryptographic keys within a secure environment such as the TEE or a secure element.

Key Benefit

Think of it as a digital safe for your app's secret keys. The app can ask the safe to sign or decrypt data, but the safe never gives the actual secret key away.

Keystore Key Attestation

Key Attestation is a critical security feature that provides a **cryptographic proof** of metadata about a newly-created key, and the device it was created on.

How it works

The Keystore generates a **signed certificate** (an attestation certificate) for a newly created asymmetric key. This certificate is signed by a chain of signing keys that are only available within the secure environment, rooted at a known Google key. The certificate chain, and the metadata within it, can be verified by a remote server.

What it Proves:

1. The key is genuinely backed by secure hardware.
2. The key's security properties (e.g., whether it requires **biometric authentication** to use) cannot be changed.
3. The state of the device at key creation time, including OS version, security patch level, bootloader unlock status and verified boot information.

Key Benefit

Allows a remote service (like a banking server) to confirm that the app is using a key that resides in the device's most secure component, not a software imitation, and that the device complies with OS integrity requirements

Version Binding

Version Binding is a security measure that **ties a key's availability to the version of the Android OS and/or the security patch level** of the device.

How it works

When a key is created, the current OS and patch level are cryptographically bound to the key. If the device is rolled back to an **older** version (a downgrade attack), the key cannot be decrypted, making it unusable.

Key Benefit

It protects against attackers who might try to exploit a security vulnerability in a previous OS version by forcing a downgrade.

RKP (Remote Key Provisioning)

Remote Key Provisioning (RKP) is a process that allows a device to **securely generate attestation keys and receive certificate chains remotely** from a provisioning server, enhancing **attestation security** and scale.

How it works

Instead of relying solely on the device's locally provisioned root keys for attestation (which could be cloned by manufacturers/attackers), RKP allows the device to generate new, unique attestation keys on demand and have a remote server vouch for their authenticity.

Key Benefit

It provides greater trust for services like Google Play Integrity by ensuring that the attestation keys are unique, up-to-date, and tied directly to the genuine hardware. This feature significantly enhances the difficulty of cloning or faking device identity.

KeyChain

The **KeyChain** is the system API that allows apps to request and use **system-wide credentials** (like Wi-Fi certificates, VPN keys, and user-installed certificates).

Difference from Keystore

While the Keystore holds keys for specific apps, KeyChain deals with keys and certificates that are shared and managed system-wide, often installed by the user or an administrator.

Key Benefit

It provides a standard, secure way for users and system services to manage and access their personal credentials without exposing the private keys to every application.

StrongBox

StrongBox is the **highest security level** for key storage and operations within the Android Keystore System.

Technology

It is a hardware-backed KeyMint implementation that resides in **a dedicated, tamper-resistant security chip** (a secure element or discrete chip)that is physically separate from the main CPU and the TrustZone (TEE).

Key Benefit

Provides the highest assurance level for storing cryptographic keys, protecting high-value credentials like biometric data or payment keys.

Why it's ‘Stronger’

It has its own isolated memory, CPU, and storage, making it resistant to advanced software and physical attacks that might compromise the standard **Trusted Execution Environment (TEE).**

Key Decryption on Unlocked Devices (File-Based Encryption)

This refers to how Android manages keys for **File-Based Encryption (FBE),** which is the default since Android 7.

How it works

In FBE, the device's data is secured using various keys. When the device is **unlocked** (e.g., after the user enters their PIN/password after a boot-up), a specific key set, called the **Credential Encrypted Storage (CES)** key, is decrypted and made available.

Key Benefit

This key set is used to decrypt **user-specific** data like emails, text messages, and app private data, ensuring this highly sensitive data is only accessible after the user has successfully authenticated.

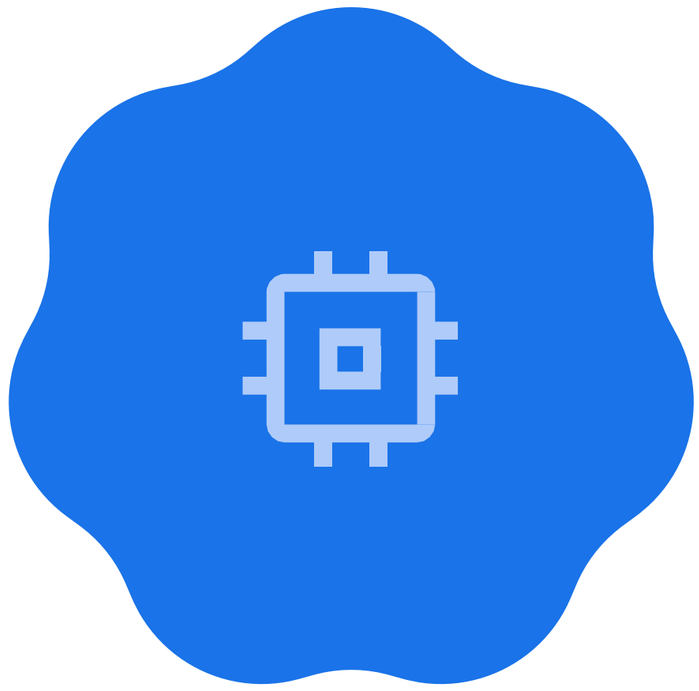
Summary: Android Keystore Security

Concept	What it is	Primary Security Benefit
Android Keystore System	A hardware-backed system for securely storing and using cryptographic keys.	**Isolation:** Keeps keys out of reach of the main OS and malware.
StrongBox	A dedicated, tamper-resistant hardware chip for key storage and operations.	**Highest Assurance:** Protection against advanced physical and software attacks.
Key Attestation	A cryptographically signed certificate proving properties about a key and the device it was generated on.	**Verifiability:** Allows remote servers to trust the security of the key and the device it was generated on.
Version Binding	Tying a key's validity to the Android OS and/or patch level.	**Anti-Downgrade:** Prevents keys from being exploited after an OS rollback to an older vulnerable version.
RKP (Remote Key Provisioning)	Generating and securely receiving attestation keys remotely from a server.	**Anti-Cloning:** Ensures attestation keys are unique and difficult to fake.
KeyChain	API for managing system-wide, shared credentials (certificates, VPN keys).	**Centralization:** Securely manages and grants access to user-installed credentials.
Key Decryption (FBE)	The process of unlocking user data keys after the user authenticates.	**Data Protection:** Ensures sensitive user data is locked until the device is unlocked.

Memory safety

Memory safety bugs, and errors in handling memory in native programming languages like C and C++, are an industry-wide problem with negative consequences to software stability, security, and ultimately user experience.

Throughout the industry, across different companies and products, these bugs represent a large fraction of reported security vulnerabilities. Additionally, we have provided guidance on how to Eliminate Memory Safety Vulnerabilities at the Source.



To improve the security and user experience of the operating system, Android has been investing in the development of technologies to address this problem:

Rust

Android 12 introduced Rust as a language for platform development. Rust provides memory and thread safety at performance levels similar to C/C++. Rust is the preferred choice for new native projects in the Android platform.

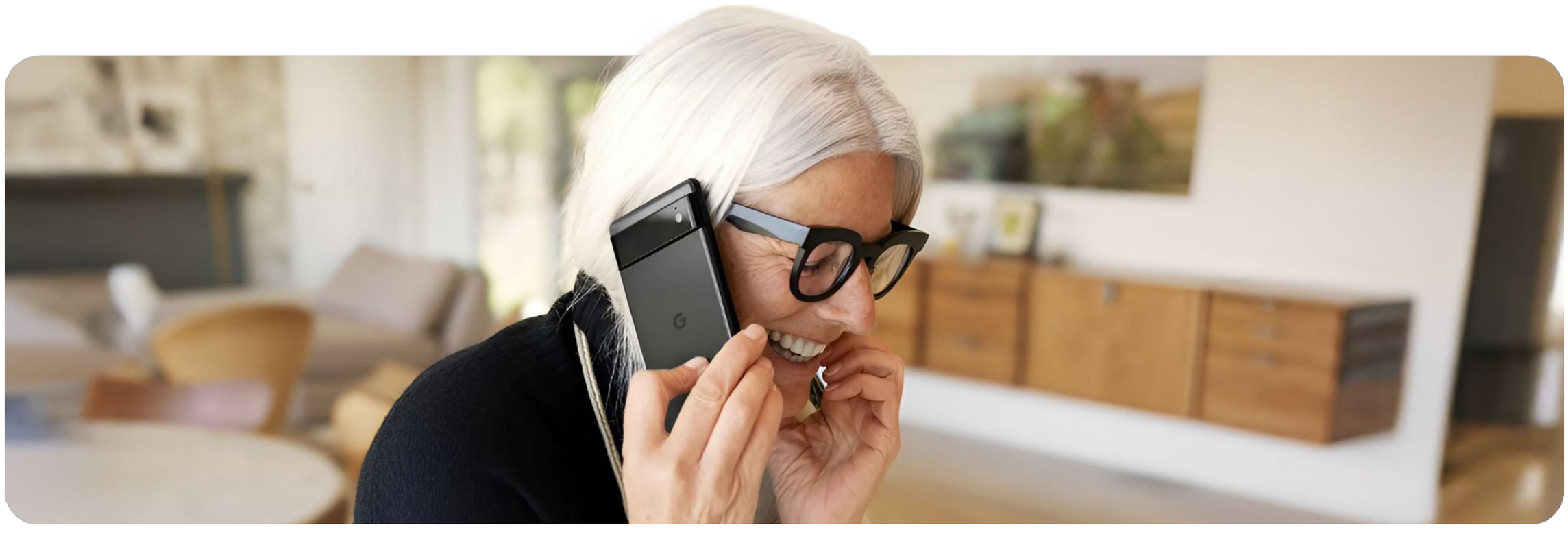
Anti-Exploitation and Sanitizers

The security features below utilize various techniques to **prevent memory corruption vulnerabilities** and **detect undefined behavior,** which are common targets for attackers seeking to exploit code and gain unauthorized access.

For **enterprise customers,** these features are crucial because they ensure the **robustness and integrity** of the devices and operating system, protecting sensitive corporate data and resources from sophisticated cyberattacks.

Feature	Description	Enterprise Importance
MTE (Memory Tagging Extension)	A hardware-assisted feature (on Arm v9 architecture) that tags each memory allocation and corresponding pointers with a small, random value. At runtime, the CPU checks that the pointer and memory tags match on every memory access.	**Highly effective against major memory-safety bugs** like **use-after-free** and **buffer overflows** in production. Reduces the attack surface for advanced persistent threats (APTs) targeting native code.
Control Flow Integrity (CFI)	A security mechanism that enforces the predetermined execution flow of a compiled binary. It prevents an attacker from redirecting the program's execution to an unintended, potentially malicious, location in memory.	**Mitigates control-flow hijacking attacks,** which are often used to gain code execution after exploiting a vulnerability like a buffer overflow. This protects the core operating system and critical enterprise apps.
ASLR/KASLR (Address Space Layout Randomization / Kernel ASLR)	Techniques that randomly arrange the address space positions of key data areas (executable base, stack, heap, libraries) for **ASLR** (userspace programs) and the **kernel** for **KASLR.**	**Frustrates exploit development** by making it difficult for an attacker to predict the exact memory addresses of code or data they need to hijack execution. This adds a necessary layer of unpredictability to the system's defenses.

Feature	Description	Enterprise Importance
SCUDO	A **hardened memory allocator** used internally by Android. It's designed with a strong focus on security, featuring integrity checks and delayed freelists to prevent common memory corruption vulnerabilities.	**Mitigates memory-based attacks** by making heap corruption exploits—such as double-free or use-after-free—much harder to successfully execute, ensuring the stability and security of system processes.
UBSan (Undefined Behavior Sanitizer)	A compiler-based sanitizer that detects various types of **Undefined Behavior (UB)** at runtime, which includes but is not limited to: signed integer overflow, division by zero, and array subscript out of bounds where bounds are known at compile time.	**Catches programming mistakes** that can lead to security flaws or application crashes, particularly in native code. Ensures the Android system behaves predictably and correctly, reducing the risk of denial-of-service or privilege escalation attacks.
BoundSan	A specific check provided by UBSan (Bounds Sanitizer). It performs **bounds checks** for array indexing, primarily in cases where the array bounds can be statically determined during compilation.	**Prevents buffer over-reads and over-writes** for statically sized arrays. While less comprehensive than a full memory sanitizer, it provides a low-overhead, effective defense against a common class of memory corruption bugs.



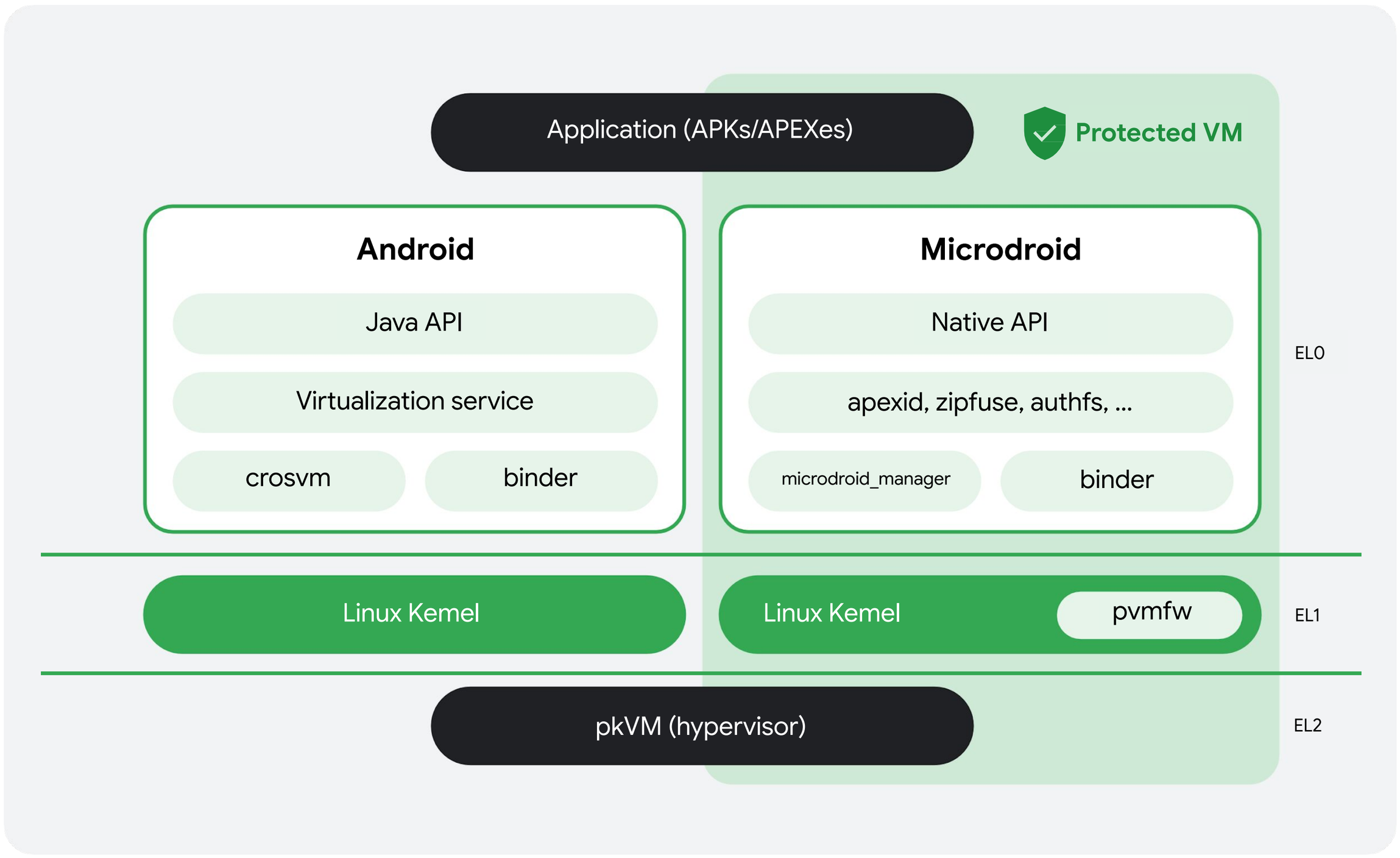
Feature	Description	Enterprise Importance
Integer Overflow Sanitization	A specific check provided by UBSan (Integer Sanitizer, or IntSan) that detects arithmetic operations that would result in a **signed integer overflow.**	**Prevents a class of vulnerabilities** where attackers exploit unexpected numerical wrapping (overflow or underflow) to bypass security checks, manipulate memory allocations, or cause application crashes. This is critical for data integrity and financial/transactional security.
GWP-Asan and KFENCE	**GWP-ASan** is a sampled allocator framework for userspace that detects **use-after-free** and **heap-buffer-overflows** with low performance overhead, enabling detection in production. **KFENCE** is its equivalent for the kernel (K=Kernel Fence).	These are vital for **post-release security monitoring.** They enable the detection and reporting of rare, difficult-to-find memory bugs in the wild, allowing Google and OEMs to quickly identify and patch vulnerabilities that might otherwise be exploited, thereby improving the long-term security posture of enterprise devices.

In summary, Android has fortified its security against exploits through compiler-based mitigations, making bugs harder to exploit and preventing certain classes of bugs from becoming vulnerabilities.

Hypervisor and Virtualization

The Android Virtualization Framework (AVF): Enterprise Security and Isolation

The Android Virtualization Framework (AVF) unifies various hypervisors to offer a standardized set of APIS for running isolated, sandboxed workloads. This framework is essential for the next generation of Isolated Execution Environments (IEEs) used in modern Android applications.



Core Benefits for the Enterprise

AVF provides a novel isolation primitive, offering security superior to standard operating system mechanisms alone, This is critical for enterprise use cases because:

Compromise Resilience

A Protected Virtual Machine (pVM) maintains its security and integrity even if the main Android operating system is compromised, which is vital for protecting sensitive corporate data and credentials.

Compliance and Reduced Privilege

AVF enables isolation without requiring the elevated privilege levels associated with technologies like TrustZone. This cleaner separation of concerns can assist enterprises in meeting strict compliance and security auditing requirements.

Standardized Security

By providing a standardized virtual machine environment, AVF reduces fragmentation and abstracts the complexities of various platform implementations. This simplifies security policies and compliance for Enterprise Mobility Management (EMM) solutions.

Manageability and Efficiency

The framework improves the updatability and portability of applications by allocating resources on demand. This streamlines patching vulnerabilities via OTA updates and enhances the overall security posture of managed devices.

Key Components of the AVF

The AVF is composed of several architectural layers, each playing a crucial role in delivering secure virtualization.

The Hypervisor (EL2)

The Hypervisor is the foundational layer responsible for creating a secure sandbox for the virtual machine and ensuring its isolation from other processes.

Google's Hypervisor

Google provides its own open-source hypervisor called **pKVM** (protected Kernel Virtual Machine).

Integration

pKVM is integrated into the Android Common Kernel and is built upon the field-tested Linux KVM hypervisor, with added capabilities to restrict access to designated ‘protected’ VMS.

Enterprise Importance

The hypervisor's guaranteed isolation is the bedrock of the entire security model. For enterprises, this isolation means that critical apps (e.g., VPN clients, secure containers, cryptographic key stores) running in a pVM are inherently protected from malware or exploits targeting the main Android user space.

Virtual Machine Monitor (VMM)

The VMM is a process that runs in the host Android user space and manages the virtual platform of the VM,

Implementation

Crosvm is a widely used VMM built on Linux’s KVM, designed for simplicity, security, and speed.

Role

The VMM allocates CPU and controls the kernel resources utilized by the VM. Each VM has its own VMM process, and terminating the VMM ends the VM.

Security

Crucially, the hypervisor ensures that the VMM cannot violate any isolation properties.

Enterprise Importance

The VMM is the mechanism for managing the isolated environment. It ensures that resource allocation and lifecycle management (starting/stopping the secure environment) are handled efficiently while still respecting the fundamental security boundaries set by the hypervisor,

Microdroid: The Guest OS

Microdroid is a minimal, Android-based operating system designed to run inside a virtual machine to execute native code

Developer Focus

Its purpose is to help developers seamlessly transfer parts of their application into a pVM, offering a familiar environment with access to a subset of Android APIs

OEM Flexibility

OEMs have the flexibility to utilize alternative operating systems within the PVM if Microdroid is not suitable.

Enterprise Importance

Microdroid enables the creation of highly secure, lightweight, and dedicated execution environments for specific enterprise components—like FIPS-validated cryptographic modules or proprietary app logic—without the overhead or attack surface of a full OS.

PVM Firmware: Secure Bootloader

The PVM Firmware acts as a secure bootloader for isolated virtual environments.

Integrity Verification

Before the guest OS starts, the firmware verifies the integrity and authenticity of the guest OS, ensuring that only trusted VM images are executed.

Trusted Environment

It operates in a protected guest environment, isolated from the host Android kernel, preventing tampering from the host system.

Updates

Guest OS images are distributed as APEX modules, and the firmware itself as an Android partition, enabling streamlined over-the-air (OTA) updates and efficient patching of vulnerabilities.

Enterprise Importance

The secure boot process and verification are mandatory for enterprise compliance, This feature guarantees supply chain integrity for the secure environment, preventing the loading of malicious or non-compliant guest operating systems onto a corporate device.

Enhancing Trust with Attestation and Secret Management

VM Attestation

While PVMs provide isolation, verifying their integrity and contents is essential for applications handling sensitive data, **Remote attestation** assures a server that a client's VM is genuine, running valid components, and operating on a trusted device.

The process involves two steps using the trusted intermediary **RKP VM** (Remote Key Provisioning VM):

Step 1:

RKP VM Attestation

The RKP VM is verified periodically against a trusted RKP server.

Step 2:

Client VM Attestation

The trusted RKP VM verifies the client VWs DICE chain (Dynamic Root of Trust for Measurement) and issues a certificate rooted in the server's trust chain, which the client VM can then use for secure interactions.

Enterprise Importance

This feature is crucial for zero-trust architectures and accessing corporate resources, The attestation certificate provides cryptographic proof to a corporate backend that the device is not only running a secure container (pVM) but that the software inside the container has not been tampered with, This is vital for Conditional Access policies.

Secretkeeper HAL

The Secretkeeper HAL (Hardware Abstraction Layer) was introduced to solve the challenge of rollback protection for protected VMs. Traditionally, rollback protection relies on tamper-evident storage accessed by the Android Bootloader (ABC), which PVMs cannot access directly,

Role

Secretkeeper provides **DICE Policy gated storage**, which Microdroid VMS use to store their secrets.

Security

This ensures that only the original VM instance or its authorized upgrades can access the secrets, protecting them against rollback attacks on boot images.

Enterprise Importance

Rollback protection is a fundamental requirement for securing stored data and credentials. For enterprises, this means that even if an attacker gains control of the main Android OS, they cannot force the secure PVM back to an older, vulnerable version to access stored corporate secrets or cryptographic keys.

Why This Matters for Security and Business

AVF's design provides fundamental benefits that are crucial for businesses and zero-trust security models:

Protection from Compromise (Isolation)

Problem

If the main Android OS gets infected with malware, sensitive corporate data (like VPN credentials) is at risk.

AVF Solution

Data and applications running inside the PVM maintain their security and integrity **even if the main Android OS is fully compromised**. It's a layer of defense against sophisticated attacks.

Guaranteed Trust (Attestation)

Problem

A corporate server needs proof that the device trying to access company resources (like email) is genuinely secure and hasn't been tampered with.

AVF Solution

The PVM can cryptographically prove its integrity to a remote server using a process called **Remote Attestation**. This allows the business to enforce **Conditional Access**—only proven, secure devices can access sensitive resources.

Future-Proof Security (Certification)

Recent update

The pKVM hypervisor recently achieved **SESIP Level 5 certification**.

Meaning

This is the highest level of security assurance. It means the system is resistant to attack by highly skilled, well-funded, and knowledgeable adversaries. This high security bar is essential for supporting future critical applications, such as on-device AI processing of ultra-personalized data.

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

Mandatory encryption on Android protects user data if an Android device is lost or stolen. Android uses file-based encryption (FBE), which enables different directories to be encrypted with different keys.

With file-based encryption, the device boots directly to the lock screen and is fully usable almost immediately when unlocked.

Apps can use two kinds of storage locations:

- 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 TEE 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 keys can only be unlocked 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. 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 CE and DE keys. Each user's CE key is protected by that user's lockscreen PIN, pattern, or password. Encryption keys are 256 bits long and are generated randomly on-device.

An additional layer of encryption called metadata encryption protects filesystem metadata such as directory layouts, file sizes, permissions, and creation/modification times. The metadata encryption key is protected by Keymaster and Verified Boot.

Adiantum

Adiantum is an encryption method designed for devices running Android 9 and higher whose CPUs lack AES instructions. Adiantum 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) requires that all new Android devices be encrypted using one of the allowed encryption algorithms.

Backup encryption

Devices that run Android 9 and higher support end-to-end encrypted backup, a capability whereby the backup data is encrypted on the device using a device and user specific key. The backup server has no ability to decrypt the backup archive. The backup is encrypted with a randomly generated key that is further encrypted with a hash of the user's lockscreen PIN, pattern, or password, This encrypted key is securely shared with a cohort of secure enclaves located across Google's data centers. None of the data shared with the secure enclave is known to Google. and the device verifies the identity of the secure enclave by checking its root of trust.

> With this secure enclave, there is a limited number of incorrect attempts strictly enforced by the custom firmware. By design, this means that no one (including Google) can access a user's backed-up application data without specifically knowing their PIN, pattern, or password.

Operating system security

Android utilizes a 'defense in depth' approach 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 consumers and enterprises face.

Sandboxing

Enforcement of Android's security model starts with sandboxing of applications and system services. Hardware components like a TEE help further isolate sensitive processes and data such as cryptographic operations and key storage. Process isolation provides the foundation for sandboxing of userspace processes.

Every app applies Linux's powerful multi-user isolation to each individual app and is thus isolated from the operating system components and other apps. SELinux enforces Mandatory Access Control policies and is one of the many defense-in-depth techniques.

- **Kernel restrictions** are imposed on what actions the kernel may take and puts limitations on userspace access to kernel entry points such as device drivers.
- **System process sandboxing** ensures separation of all processes such as the media frameworks. telephony stack, Wi-Fi services, and Bluetooth components.
- **Application sandboxing** uses SELinux combined with a unique user ID to isolate apps from each other and the system. This sandbox keeps the application and its data secure.
- **Isolated processes** are anonymous uids with strong SELinux restrictions to provide system-supported isolation for Web render proceeds, PDF rendering, and other sensitive execution.
- **Other areas of separation** include the TEE and userspace components. For example, the Android Keymint integrates the keystore into the TEE, which guards cryptographic key storage from exposure and tampering. An attacker cannot read key material stored in the Keymint even if the kernel is fully compromised. Android 9 and above devices with dedicated tamper-resistant hardware can store keys in the StrongBox Keymint. This implementation mitigates against the most sophisticated attacks such as cold boot memory attacks, power analysis, and other invasive attacks that could allow privileged escalation.

SELinux

Android uses Security-Enhanced Linux (SELinux) to strictly control what every process can access, even those with root privileges, This helps Android protect system services, limit access to app data and system logs, isolate potentially harmful apps. and safeguard users from security flaws.

SELinux operates on a **default deny** basis so if something isn't explicitly allowed. it's blocked. Android includes SELinux and a security policy for core components. Any denied actions are logged using Linux tools like dmesg (kernel messages) and logcat (system messages).

SELinux also enforces separation between the core Android framework and device-specific vendor components. These run in separate processes and communicate through approved Hardware Abstraction Layers (HALs),

Unix permissions

Android uses UIDS to further isolate application resources. Android assigns a unique user ID to each app and runs each app in a separate process. Apps are not allowed to access each other's files or resources just as different users on Linux are isolated from each other.

Network Security

Key Protections for Android Devices

The proliferation of Android devices in the enterprise —through corporate-owned or Bring Your Own Device (BYOD) programs—necessitates a robust security framework. We outline the critical importance of several core networking and credential management features for securing enterprise data and maintaining compliance, specifically focusing on **DNS over TLS/HTTPS, TLS by default, Randomized MAC Addresses, Wi-Fi and Cellular Security, VPN configurations,** and **Certificate Management.**

Network Traffic Security

Secure network communication is a foundation of modern mobile security, protecting data both in transit and from prying eyes.

DNS over TLS (DOT) & DNS over HTTPS (DOH)

Importance

Traditional **DNS queries** are unencrypted, allowing network operators, ISPs, or malicious actors to monitor a device's web traffic (which sites/services are being accessed) and potentially hijack requests (DNS spoofing).

Enterprise Value

DoT/DoH **encrypts** the DNS lookup process, preventing eavesdropping and tampering. This significantly enhances user **privacy** and denies attackers a crucial data point for phishing or targeted attacks.

TLS (Transport Layer Security) by Default

Importance

Enterprise applications and services must mandate the use of **HTTPS** (which uses TLS) over unencrypted **HTTP**.

Enterprise Value

Ensuring all application traffic (API calls, data synchronization) is encrypted by default protects sensitive data (e.g., login credentials, customer records) from passive interception via man-in-the-middle (MITM) attacks on untrusted networks,

Certificate Transparency for TLS certificates

Importance

Requires all TLS certificates to be logged to a public log, for apps that opted in.

Enterprise Value

Allows enterprises to detect misissued certificates for their domains.

Randomized MAC Address

Importance

A device's **MAC address** is a unique hardware identifier. If a device uses its real, static MAC address. it can be passively tracked across different Wi-Fi networks and access points, creating a detailed activity profile of the user.

Enterprise Value

MAC randomization makes it much harder to link a device to its activity across different physical locations, enhancing **user privacy** and making device-based corporate surveillance significantly more difficult.

Wireless and Cellular Security

Securing the device's physical connection layers is essential to prevent network-level exploitation.

Wi-Fi Security

Importance

Corporate Android devices connect to various Wi-Fi networks (corporate, home, public). Using weak or outdated Wi-Fi protocols (like WEP or WPA) exposes data.

Enterprise Value

If an enterprise's infrastructure can support it, they can use standards such as **WPA3** and implement **802.1X EAP** (Extensible Authentication Protocol) for corporate network access,

Cellular Security

Importance

Devices are susceptible to attacks like **IMSI catchers** (Stingrays) or **baseband exploits** when relying on cellular data.

Enterprise Value

Android's continuous security patching and features like enhanced modem security (though largely managed by the OEM) are critical. Enterprise Mobility Management (EMM) solutions must ensure devices are always updated to protect against known vulnerabilities in the radio stack. Android additionally offers configurable network modes (disabling 2G and unencrypted mobile network connections) to reduce attack surface, and prevent interception of user communications and valuable metadata. On devices with supporting hardware, Android 16 offers full transparency into the security configuration of the mobile network, and notifications about PII (IMSI, IMEI) disclosures over-the-air.

Virtual Private Networks (VPN)

VPNs provide a protected tunnel for device traffic back to the corporate network.

VPN Service Modes

Importance

An enterprise needs flexible control over how the VPN is used. This includes **per-app VPN** (only specific work apps use the tunnel) versus **full-device VPN** (all traffic is tunneled).

Enterprise Value

Per-app VPN is crucial for work profiles, ensuring that only corporate data is routed through the secure tunnel while respecting user privacy for personal activities. Full-device VPN is often required for corporate-owned devices accessing sensitive internal resources.

VPN Lockdown Modes

Importance

A VPN tunnel may drop (e.g., due to poor signal), which could cause sensitive traffic to 'leak' onto the public network unencrypted.

Enterprise Value

Lockdown mode prevents the device from establishing any network connection (Wi-Fi or cellular) if the VPN connection is not active. This is an essential **data loss prevention (DLP)** mechanism for highly sensitive corporate devices.

Identity and Credential Management

Securely provisioning and managing digital identities is non-negotiable for enterprise access.

Certificate Enrollment/ Management:

Importance

Digital certificates are the backbone of secure authentication, used for VPN access. EAP-based Wi-Fi. mutual TLS for applications, and email signing. Manually installing and managing these is error-prone and insecure.

Enterprise Value

EMM solutions can integrate with **Public Key Infrastructure (PKI)** to silently and securely push, renew, and revoke certificates to Android devices and work profiles. This automates the process. enforces strong, non-phishable authentication, and ensures **compliance** with access control policies.

Conclusion

For enterprises leveraging Android, these features move beyond mere 'nice-to-haves' and become **mandatory security controls**. They collectively ensure **data confidentiality** (TLS. DoT/DoH. VPN). **network access integrity** (Certificate Management, Wi-Fi Security), and **user privacy** (MAC Randomization), Implementing and managing these settings through a centralized EMM platform is critical for maintaining a secure and compliant mobile fleet.

User Privacy

Core Principles and Enhancements

A foundational commitment of Android is to protect user privacy.

Recent Android versions have introduced significant changes to limit data access, particularly for background applications. This includes restricting access to device sensors, limiting information from Wi-Fi scans, and introducing new permission categories for phone calls. thereby strengthening privacy for all applications regardless of their target SDK.

Android lO onwards has continuously strengthened privacy and control mechanisms, offering users and IT administrators clearer visibility into how location and other data are accessed by applications.

Managing Device Identifiers and Networking

To safeguard user privacy. Android restricts the visibility of sensitive device identifiers such as IMEI, IMSI, and serial numbers. As of Android IO and later, only applications with specific permissions or carrier privileges are granted access to these identifiers. To be granted these permissions, the applications must undergo a specific review to ensure compliance with our privacy policies.

Furthermore, Android 10 and newer versions use random MAC addresses by default when devices search for new Wi-Fi networks, This practice of assigning a randomized MAC address upon connecting to a Wi-Fi network significantly enhances user privacy.

Restricting Background Sensor Access

Apps running in the background on Android have restricted access to user input and sensor data, including:

- Inability for the app to utilize the microphone or camera.
- Sensors that report continuous data. like accelerometers and gyroscopes, are disabled.
- One-shot or on-change sensors that only report data periodically or upon state change are also blocked.

If an app requires sensor event detection, it must use a foreground service.

Enhanced Location Control

Apps utilize location APIs to provide relevant services, such as navigation assistance. Android gives developers the tools to request necessary location permissions while ensuring users have full control over what they permit.

Applications that rely on location services must request permission from the user unless those apps have been pregranted location. We have specific requirements for 'dangerous permissions' when it comes to pregrants: In Android 10 and above, a dialog prompts users to choose between granting access only while the app is in use or granting continuous access.

If a user grants an app constant location access, Android sends an initial reminder notification when the app first uses this background access.

Android 12 introduced the option for users to approve access to their approximate location instead of their precise location. This allows applications to function without compromising user comfort by exposing excessive information.

Visual and Managerial Privacy Tools

Android 12 improves transparency with Privacy Indicators that display when the camera or microphone is actively being used by an app. Users retain control through runtime permissions, granting or denying microphone and camera access before an app can record.

The privacy dashboard offers users key insights into data access, detailing which apps have used location, camera, or microphone data. This information empowers users to manage or revoke application permissions effectively.

Users can also use the **Permission manager** to review which applications share the same permission settings and adjust permissions as needed.

Storage Access and Scoped Storage

To reduce data clutter and enhance user control. apps targeting Android IO and newer versions default to restricted file access, a feature known as scoped storage. Apps can freely manage their own dedicated directory but require specific permissions to create files in shared storage areas.

Accessing media files (images, videos, or audio) created by other applications in shared storage necessitates corresponding runtime permissions (READ_MEDIA_IMAGS, READ_MEDIA_VIDEO, or READ_MEDIA_AUDIO).

For photo or video location metadata, the ACCESS_MEDIA_LOCATION permission is additionally required. Even with read access, apps need explicit user consent to modify or delete media files.

EMM IT administrators can also prevent users from accessing external storage (e.g., an SD card) on managed devices, mitigating potential data loss or theft.

Secure Environments

Android's Private Compute Core (PCC), introduced in Android 12, is an open-source, secure environment isolated from the main OS and other applications, PCC allows on-device processing for features such as Smart Reply, live caption, and Now Playing, eliminating the need to send data to the cloud, thus adding a new layer of privacy.

Android Work Profile creates an independent, sandboxed and company-controlled space on a device, isolating corporate applications and data from personal content. This is applicable to both BYOD and company-owned devices used for both work and personal use. Personal apps and data in the personal profile remain outside of IT control.

When a Work Profile is set up, the EMM Device Policy Controller (DPC) clearly presents the terms of use and data collection information to the user. Setup requires the user to review and accept the user license agreement. Particularly impactful policies, such as those which modify network connections to enable the enterprise to monitor traffic, are persistently indicated to the user.

Developers are urged to ensure their apps conform to the latest privacy changes introduced in Android IO and subsequent versions, which include changes to the permissions model and restrictions on data access.

Advanced User Controls and Features

The **Private space** feature allows users to create a separate, authenticated space on their device for sensitive applications, utilizing a separate user profile, Users can select either the device lock or a unique credential for the private space.

Apps in the private space appear in a separate launcher container and are hidden from the recents view. notifications, settings. and other apps when the space is locked. User-generated content and accounts are also segmented between the private and main spaces. The photo picker and system share sheet can bridge content access between spaces when unlocked.

Existing apps cannot be migrated to the private space; users must install new. separate copies of apps within the private space.

Locking the private space stops the associated profile, rendering the apps inactive and preventing them from executing foreground or background tasks, including displaying notifications.

On Android 9 and later, the **Lockdown mode** option can be enabled via the power button menu to restrict device access further. This mode disables biometric unlocking. Smart Lock. and notifications on the lock screen. On managed devices, Enterprise IT administrators have the capability to remotely lock the Work Profile and remove the encryption key from memory.

Android 15 introduces **screen recording detection** support for apps. This allows developers to take necessary precautions to protect user privacy and notify users when their screen is being recorded or viewed.

Enterprise Benefits of Android's Data Protection and Privacy

For enterprise customers. the comprehensive suite of privacy and data protection features in Android offers significant advantages:

Reduced Risk of Data Leakage

Features like Scoped Storage, restricted background sensor access, and Private Compute Core minimize the attack surface and prevent unauthorized access to sensitive corporate data by rogue or background applications.

Enhanced Regulatory Compliance

The granular control over permissions, clear visibility into data access (via Privacy Indicators and Dashboard), and mechanisms like Lockdown mode help enterprises meet stringent data protection regulations and compliance standards.

Secure BYOD and Corporate-Owned Devices

The Android Work Profile ensures a strict separation between corporate data and personal user activity, which is crucial for both Bring Your Own Device (BYOD) and company-owned. personally enabled devices, allowing IT to control the work environment without compromising user privacy.

IT Control and Management

Enterprise Mobility Management (EMM) administrators gain powerful tools, such as the ability to remotely lock the Work Profile, restrict external storage access, and leverage the Permission manager to enforce company security policies across the fleet.

App security

Mobile apps are now essential tools for user productivity and communication. Android ensures these applications are secure through a multi-layered security approach. This robust protection allows users to confidently download apps for both personal and professional use, safeguarding them from malware, security vulnerabilities, and various attacks.

Google security services

The Android ecosystem prioritizes app security through several key services and collaborations:

- **Google Play Protect and the Play Integrity API:** These services run on Google Mobile Services (GMS) certified devices to identify malware and compromised devices, which are often used to deliver exploitation code.
- **Low PHA Installation Rate:** Android devices that utilize managed Google Play maintain a very low Potentially Harmful Application (PHA) installation rate, approximately 0.009%.
- **App Defense Alliance:** Security within the Google Play Store is further bolstered by the App Defense Alliance. a partnership involving Google and other industry security experts.

IT administrators have the ability to manage application access by creating allow-lists and block-lists for the managed Google Play Store. This provides precise control over which applications can be installed on devices. Furthermore, for corporate devices utilizing a Work Profile. these controls also apply to the Google Play Store within the personal profile, ultimately reducing the risk of unauthorized or undesirable applications being installed.

The Play Integrity API is designed to assist app and SDK developers in verifying the authenticity of interactions and server requests. It confirms that these requests originate from a genuine Android device running a legitimate version of the app. By identifying potentially fraudulent activities, such as those from untrustworthy environments or tampered app binaries, the app's backend server can take suitable measures to mitigate abuse and defend against attacks.

Enterprise Mobility Management (EMM) platforms can leverage on-device services and the Play Integrity API to enhance application security. This enables EMM customers to prevent users from sideloading unauthorized applications, ensuring that installation is restricted to Google Play or other approved app stores. By receiving signals via the Play Integrity API from these on-device services, EMMs can effectively detect and mitigate potential security compromises on user devices.

App signing

Android mandates that all apps be digitally signed. APK key rotation, introduced in Android 9, allows apps to change their signing key upon update by moving from the v2 to the v3 APK signature scheme. The previous key is added to the app's signing lineage and can grant capabilities for interacting with other apps.

To resolve rotation issues, Android 13 introduced APK signature scheme v3.1. This scheme uses the new key in the v3.1 block and the original key in the v3.O block for compatibility. New rotations via apksigner default to v3.1 for Android 13+. V3.1 also allows verified SDK-targeted signing configs for stricter controls in newer Android versions.

Android verifies app updates by checking the signing lineage: an update is allowed if the existing version's key attests to the new one.

Apps with the same key, or those sharing a lineage and the SHARED USER_ID capability, can run as a single application using sharedUserId in the manifest.

Android signature-based permissions enable functionality sharing between apps with a common signer and the PERMISSION capability granted to the previous key, facilitating secure data exchange.

Google Play developers use Play App Signing, delegating key management to Google, This ensures developers can regain access if the upload key is lost. Google securely stores keys on Google Cloud Platform and handles key upgrades and rotation complexities across Android versions.

> **Note:**
>
> The sharedUserId attribute is deprecated in Android 11. Use sharedUserMaxSdkVersion to limit its use for new installs.

App permissions

Android's Permission System: Protecting Privacy and Data Access

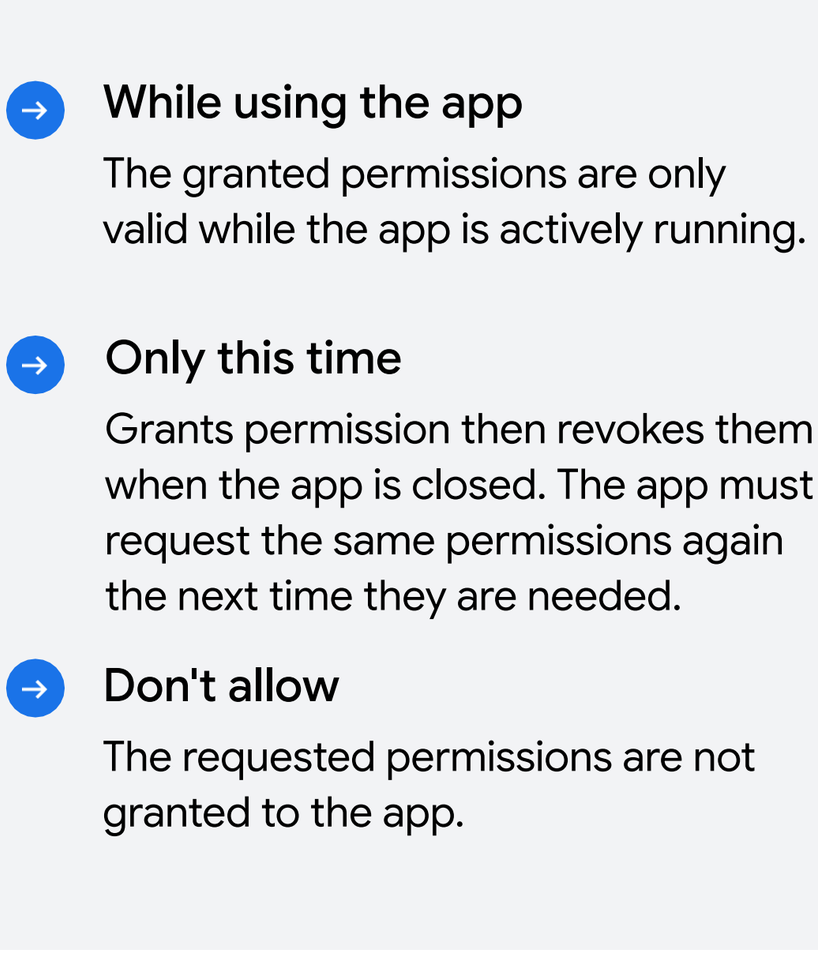
Permissions in Android apps function as gatekeepers, ensuring that applications only access the resources and data they absolutely require. This system is foundational to the Android security architecture, which operates on the principle that, by default, no app has permission to perform actions that could negatively affect other apps, the operating system. or the user.

This architecture is designed to safeguard user privacy and provide transparency. Apps must explicitly request permission for any actions that are not permitted by default, such as:

- Accessing sensitive user data (e.g.. contacts, SMS, or emails).
- Accessing system features like the camera and internet.
- Modifying another application's files.
- Maintaining the device's wake state.

Android utilizes **runtime permissions**. Unlike older install-time permissions, this approach displays a clear dialog to the user when the app actually needs access to a resource, allowing the user to approve or deny the request at that moment. This method gives users greater control and streamlines the app installation process.

When a permission is requested, users have options to select from:



While using the app

The granted permissions are only valid while the app is actively running.

Only this time

Grants permission then revokes them when the app is closed. The app must request the same permissions again the next time they are needed.

Don't allow

The requested permissions are not granted to the app.

Evolving User Privacy and Control in Android

Android has consistently introduced mechanisms to enhance user privacy and control over sensitive data and app permissions:

Enhanced Notification Controls (Android 13+)

Starting with Android 13 (API level 33), users must grant a runtime permission for apps to send non-essential notifications, giving them greater authority over the notifications they receive.

Approximate Location Sharing (Android 12+)

Android 12 introduced a feature allowing users to restrict apps to accessing only their approximate location, rather than precise data.

One-Time Permissions and Auto-Reset (Android 11+)

Android 11 enhanced user control by introducing one-time permissions for the microphone, camera, and location, This version also added permission auto-reset for apps (targeting Android 6 or higher) that haven't been used for several months.

Stricter Background Location Access (Android 10+)

Android 10 required apps to request a dedicated background location permission for continuous location access, implementing stricter control.

Foreground App Restrictions (Android 9+)

Starting with Android 9, only apps running in the foreground or foreground services can access the camera, microphone, or device sensors.

App Security and Privacy Enhancements

Google Play Protect: Comprehensive Device Protection

Google Play Protect offers robust security for Android devices, regardless of where apps originate (Play Store or third-party sources). It actively monitors for security threats, removes malicious applications, and helps users make safer installation choices.

Key Features of Play Protect

- **Automatic and Ubiquitous Security:** It is included on all devices with Google Play, safeguarding over 3 billion devices from malware and other threats.
- **Continuous Monitoring:** The system performs daily scans of your device for security risks and harmful activity, alerting you to the presence of malware.
- **Automated Remediation:** Malicious applications are automatically disabled or removed to prevent device harm and improve future threat detection capabilities.
- **Threat Analysis for Unknown Apps:** Users have the option to submit unknown applications to Google for in-depth security analysis.
- **Pre-Installation Scanning:** Play Protect examines apps installed from outside the Play Store prior to installation to block known malware.
- **Real-time Detection:** Real-time threat scans are recommended for any app. regardless of source, that has not been previously analyzed.

Data Privacy and Transparency

Enhanced Data Sharing Transparency

Starting in Android 14, system runtime permission dialogs for apps that share location data with third parties now feature a clickable section. This highlights the app's data-sharing practices and provides context, such as the reasons for sharing data.

Device Identifier Controls

Access to certain privacy-sensitive identifiers is now restricted or requires specific runtime permission. This includes:

- **Wi-Fi MAC Address:** APIs providing access to the Wi-Fi MAC address have been removed, except in the case of fully managed devices.
- **Enterprise Device Control:** Device policy controllers (DPCs) for enterprise devices can deny permissions on the user's behalf. This ensures greater privacy control for the user and more granular data access control for organizations managing those devices.

These changes aim to give users and enterprises better control over the use of device identifiers and data access.

A Robust Security Ecosystem for Developers and Users

Google Play ensures a secure environment through two main approaches: providing developers with advanced security tools and maintaining a rigorous app review process.

Play Integrity API: Empowering Developers

The Play Integrity API offers developers a powerful set of tools to strengthen app security and deliver a secure user experience.

Key features include:

- Verification that requests originate from a genuine app, installed from Google Play, and running on a trusted Google Play Certified Android device. This process uses Android Key Attestation for a strong guarantee of system integrity.
- To check if other apps running on the device could potentially capture the screen, display overlays, or control the device, This protects users from social engineering or scams during sensitive actions like money transfers. Developers can then prompt users to close risky apps or restrict access.
- Verification of the status of and whether it has detected any risky apps installed. This is particularly useful for sensitive actions and enforcing strict device policies in enterprise settings.
- And it indicates a device's recent activity level without disclosing specific numbers to protect user privacy. Developers can limit access to protected functionality if the device exhibits suspicious activity due to a high volume of recent requests.

Google Play App Review: Protecting the Ecosystem

The Google Play Store maintains user trust and safety with a multi-layered review process to prevent the distribution of harmful apps.

Developer and App Vetting

- Developers' real-world identities are verified when they create a developer account.
- Further checks, including automated analysis for potential harmful behaviors, are conducted when an app is submitted.

If suspicious activity is detected, a security analyst manually reviews the app. Apps that violate policies lead to developer account suspension, effectively weeding out malicious actors before they reach users.

Remediation and API Requirements

- Flagged apps trigger immediate notifications and guidance for developers, prioritizing swift action to address security issues. In some cases, updates may be blocked until necessary security improvements are implemented.
- Risk mitigation is reinforced by encouraging the use of updated APIs. Newer Android versions provide security and performance enhancements. which apps must explicitly support.
- Both new and updated public apps are required to target at least to meet current API requirements, driving the adoption of the latest security standards, Developers can find detailed guidance in the Google Play Developers documentation.

These comprehensive efforts ensure a secure and trustworthy Android ecosystem, benefiting users with confidence in their apps and developers with a safe environment to build and distribute their products.

Jetpack Security for Enhanced Data Protection

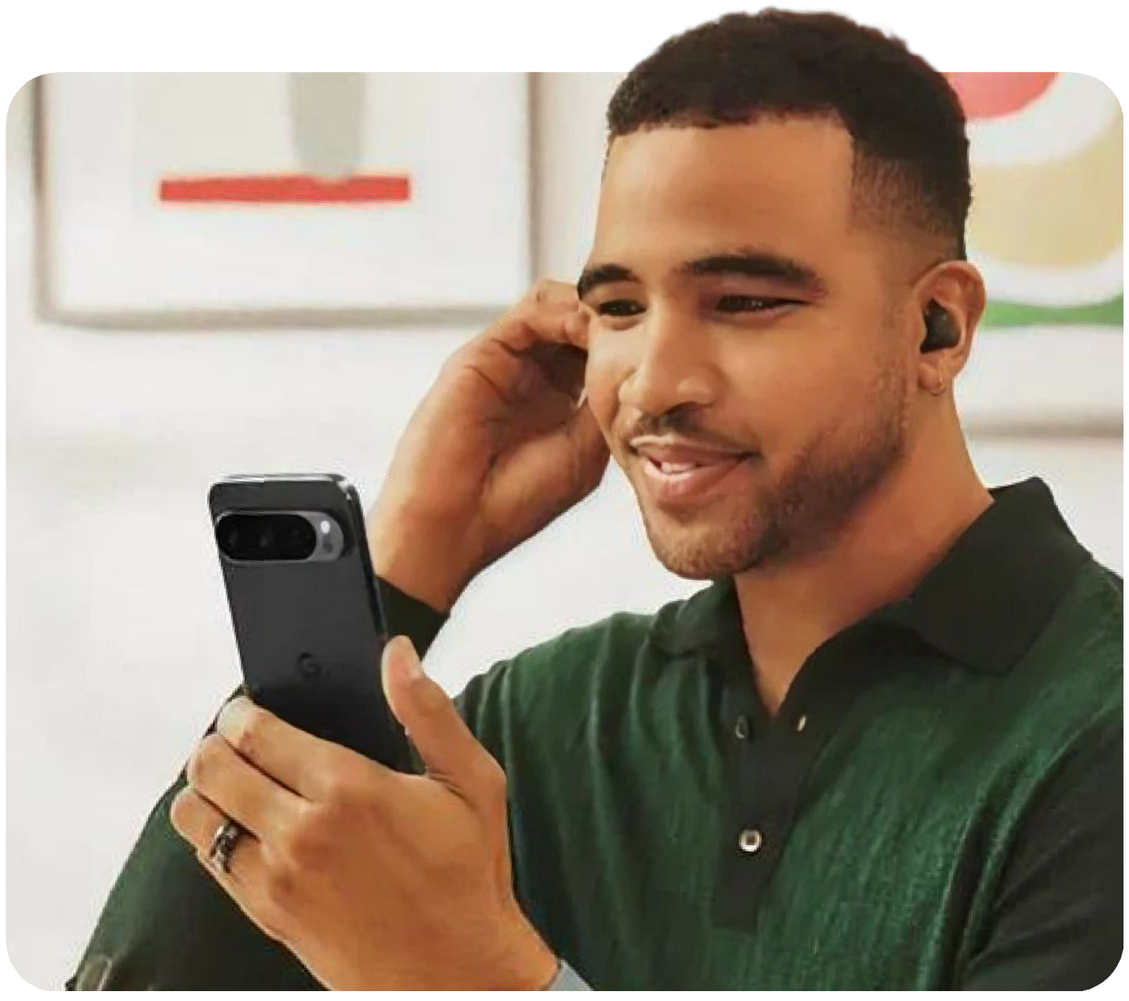
The Jetpack Security library allows Android developers to leverage the Android KeyStore for superior data protection. This tool is highly recommended for all applications, particularly those managing sensitive data or device policies.

Key features include:

- Simplified Key Creation: The **MasterKeys** utility makes it easy to generate robust AES 256 GCM encryption keys.
- Easy Data Encryption: Provides a straightforward way to encrypt files and shared preferences.
- Customizable Security: Developers have access to advanced options for tailoring key authorization settings to specific requirements.

User Authentication on Android Devices

User authentication is the fundamental process of verifying that a user is who they claim to be. On Android devices, this typically involves a user providing credentials —such as a **PIN, password,** or **pattern**—that the device verifies against stored data.



This initial layer of security ensures that only authorized individuals can access the device and the corporate data it holds. Robust authentication is the first line of defense against unauthorized access, data breaches, and corporate espionage.

Biometrics

Biometric authentication leverages a user's unique biological and physical characteristics for verification. This method offers a significant advantage over traditional passwords because biometric data is difficult to replicate or guess, providing a higher level of security and convenience. The most common biometrics on modern Android devices include **fingerprint authentication** and **face authentication**.

Fingerprint Authentication
This method scans the unique ridges and valleys of a user's finger. The device securely stores a mathematical representation of the fingerprint (not the actual image) and compares it to the live scan during verification, It's often favored for its **speed** and **reliability**.

Face Authentication
This method analyzes the unique features and dimensions of a user's face, often using 3D mapping (for higher security implementations) or 2D image analysis. Once authenticated, the user can gain access.

Additional Authentication Methods

Beyond the core password/PIN and biometrics. Android offers several additional **authentication methods** that can be employed. often as a fallback or for higher security needs:

Smart Lock
This feature allows a device to remain unlocked in certain. trusted scenarios, such as when it's connected to a specific Bluetooth device (like a smartwatch) and when it's in a specific trusted location.

Two-Factor or Multi-Factor Authentication (2FA/MFA)
While not strictly an on-device unlock mechanism, 2FA/MFA is critical for application and service access, It requires the user to provide two or more verification factors, such as 'something they know' (a password) and 'something they have' (a temporary code from an authenticator app).

Importance for Enterprise Customers

Robust user authentication is critical for enterprise customers deploying Android devices for several key reasons:

Data Security and Compliance

Enterprise customers manage sensitive corporate data, including customer records, financial information, and proprietary secrets, Strong authentication ensures that this data is protected from unauthorized access, which is a requirement for meeting regulatory standards like **HIPAA** or **GDPR** and many Gov't use cases.

Enhanced Productivity and User Experience

Biometric methods like fingerprint and face authentication are significantly **faster** and more **convenient** than typing a complex password multiple times a day. This reduces friction for employees, leading to higher productivity and better adoption of the deployed devices.

Support for Corporate Policies (BYOD/COPE)

In both Bring Your Own Device (BYOD) and Corporate-Owned, Personally-Enabled (COPE) models, the enterprise must enforce strict security policies. Device management solutions can mandate the use of strong PINs combined with biometrics, ensuring a consistent and high level of security across the entire fleet of devices, isolating corporate data effectively.

Android security updates

Monthly device updates are a crucial aspect of Android security. Google releases Android Security Bulletins every month to inform users, partners, and customers about the latest fixes.

These security updates are available for Android versions for a period of three and a half years from their initial release date. Device manufacturers can choose to extend this support period by upgrading the Android version on their devices.

Android OS leverages a feature called Project Treble to accelerate the delivery of security fixes, privacy enhancements, and consistency improvements. Treble enables device manufacturers and silicon vendors to develop and deploy Android updates more rapidly than was previously possible. Modern Android devices are Treble-compliant and fully benefit from its architecture.

For fully managed devices. IT admins can install system updates manually using a system update file on Android 10 and above devices. This manual control offers several advantages to IT admins. They can test an update on a small group of devices before deploying it widely, thereby minimizing potential risks.

Additionally, they can avoid duplicate downloads on networks with limited bandwidth, saving valuable resources. Finally. IT admins can stagger installations or schedule updates for times when devices are not in use, minimizing disruption to users.

Device manufacturer partner updates

Security updates for Pixel devices are sent directly from Google every month. You can also update Pixel devices manually using firmware images from the Google Developer site. Many other device makers like Nokia, Samsung, LG, Motorola, and Zebra have a similar update schedule and provide their own security bulletins.

You can check if your device is up-to-date by looking at the Security Patch Level. This is in your device settings and is also available in the attestation certificate chain. Enterprise Mobility Management (EMM) partners can use an API to see which security update is installed and enforce policies for devices that are out of date.

Google Play system updates

Google Play System Updates provide a quicker and easier way to deliver important updates to your Android device. Key parts of the Android system are now modular, so they can be updated individually, just like apps, through the Google Play Store or directly from your device manufacturer.

These updates come in the form of APK or APEX files. APEX files load earlier during device startup. which can be important for security and performance improvements. This means you can get critical updates without waiting for a full operating system upgrade. And, of course, all updates are cryptographically signed to ensure their security.

Conscrypt

The Conscrypt module plays a crucial role in enhancing Android's security by delivering accelerated security improvements and bolstering device protection through regular updates via Google Play System Updates.

It leverages Java code and a native library to provide the Android TLS implementation, along with a substantial portion of Android's cryptographic functionality, such as key generators, ciphers, and message digests. While Conscrypt is available as an open-source library, it incorporates specific optimizations when integrated into the Android platform.

Furthermore, the Conscrypt module utilizes BoringSSL, a native library that serves as a Google fork of OpenSSL. BoringSSL is widely employed in numerous Google products for cryptography and TLS, notably Google Chrome. The Conscrypt module is distributed as an APEX file encompassing the Conscrypt Java code and a Conscrypt native library that dynamically links to Android NDK libraries, Importantly, the native library also includes a copy of BoringSSL that has undergone validation through NIST's Cryptographic Module Validation Program (CMVP), further reinforcing its security posture. The most recent certified version is identified by certificate #4735.

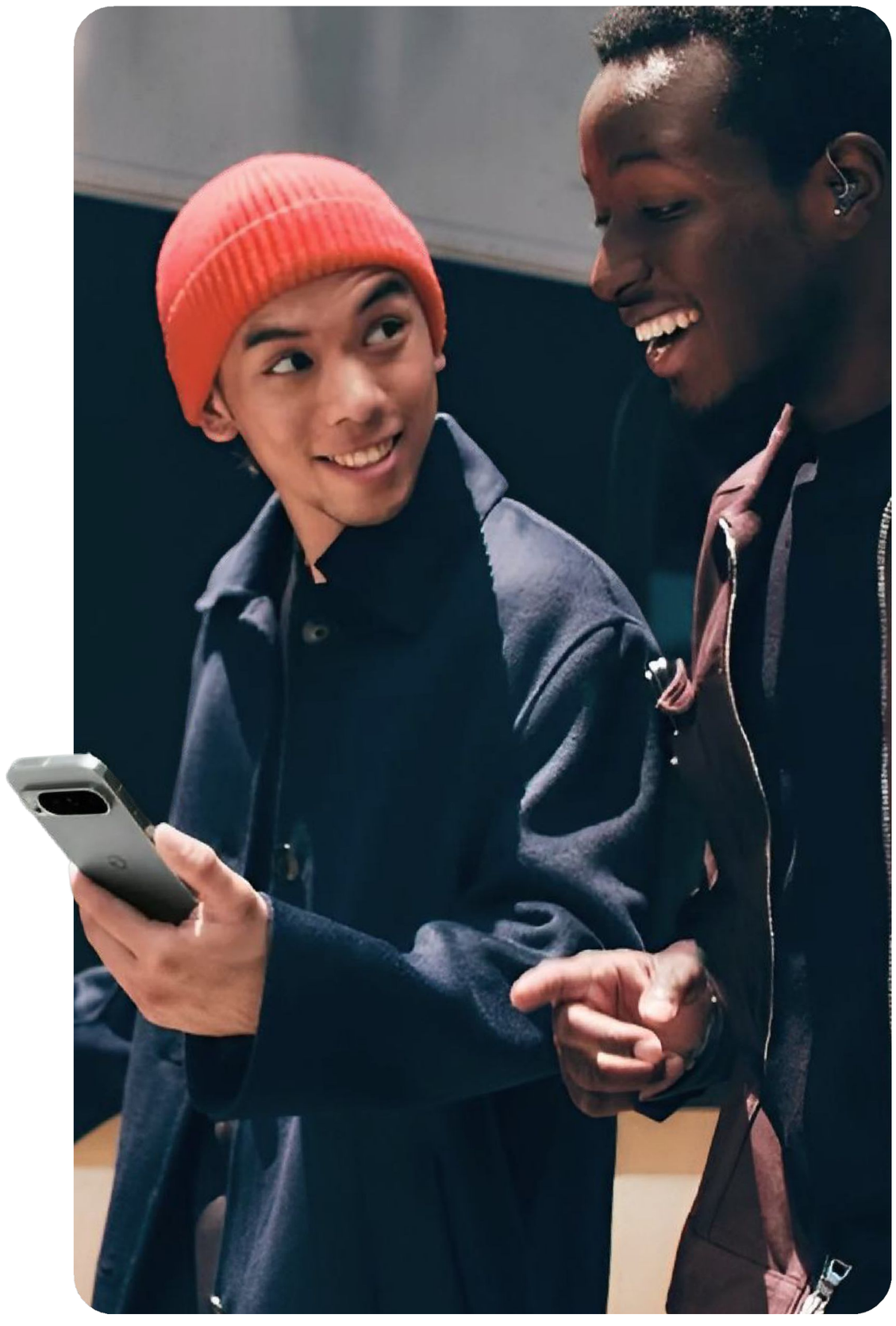
App management

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

Managed Google Play

On devices managed by an Enterprise Mobility Management (EMM) provider, the IT admins can control which work apps may be installed. On GMS devices, managed Google Play can be used for application management. Managed Google Play is an enterprise version of Google Play that allows IT admins to easily find. deploy, and manage work apps while minimizing the threat of malware with Google Play Protect, Managed Google Play provides APIs and iframes to EMM partners that allow their customers to manage apps on Android devices.

Using Managed Google Play, organizations can build a customized mobile application storefront for their teams, featuring public and private applications that are available to their employees. This eliminates the need to sideload any applications onto devices. Managed Google Play is available for all fully managed devices and devices with a Work Profile, whether they are personally owned (BYOD) or company owned (COPE).



Organizations have two methods of identifying allowed applications:

Allowlist

Users may be restricted to a specific allow-list of permitted applications in the company policy (default behavior). This protects company data by blocking unknown applications from being installed.

Blocklist

When using an EMM that uses Android Management API, IT admins also have the option of blocking one or more apps. Users may only install applications that are not explicitly marked as 'blocked' in the organization's application policy. EMMs using Android Management API may also set application block lists in the personal usage policies of a company owned device with Work Profile,

Installation of apps in the Work Profile on BYOD & COPE or fully managed devices is possible via two main mechanisms:

- Users may install permitted applications on-demand through the managed Google Play application, in their organization's custom store front.
- The organization may push an application to a device using their EMM. Organizations can **silently** (without user interaction) install applications on to fully managed devices and inside of the work profile.

Additionally, IT admins can enforce update preferences through managed Google Play. IT admins can push an urgent update, such as security updates, to devices automatically as **high priority**.

Private apps

With managed Google Play, enterprise customers and developers can publish apps and target them privately (that is, they're only visible and installable by users within that enterprise). Private apps are logically separated in Google's cloud infrastructure from public Google Play for consumers.

There are three modes of delivery for private and web apps:

Google-hosted

By default. Google hosts the APK in its secure, global data centers, This is the recommended option to take full advantage of Google's enterprise-grade security, including SSL downloads and malware security scanning. Google-hosting also allows organizations to take advantage of Google Play app signing. With Google Play app signing. Google manages and protects the app's signing key on behalf of an organization and uses it to sign optimized distribution APKs. Google Play app signing stores the app signing key on Google's secure key enclaves and offers upgrade options to increase security.

Web apps

A web app turns a web page into an Android app. making it easier to find and simpler to use on mobile devices. A web app looks like a native app in a device's launcher, and when the user opens it, Android renders the web page in the selected display mode (minimal IJI, standalone or full screen).

In all cases, Google Play stores the app metadata (such as, title, description, graphics, and screenshots). Private apps are held to Play Policies for preventing mobile unwanted software and malware. Private apps cannot be made public.

Externally-hosted

Enterprise customers host APKs on their own servers, accessible only on their intranet or via VPN. When managed Google Play makes a request to download an APK from an external server, the request includes a cookie containing a JSON Web Token (JWT). We recommend that the organization decodes the JWT to authenticate the download.

Managed configurations

Managed configurations allow an organization's IT admin to control the availability of features, configure settings, or set credentials within an app via their EMM's management console. As an example, an app may have an option to only sync data when a device is connected to Wi-Fi, or allowlist or blocklist specific URLs in the web browser. App settings exposed through managed configurations are managed by the developer.

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

Applications from unknown sources

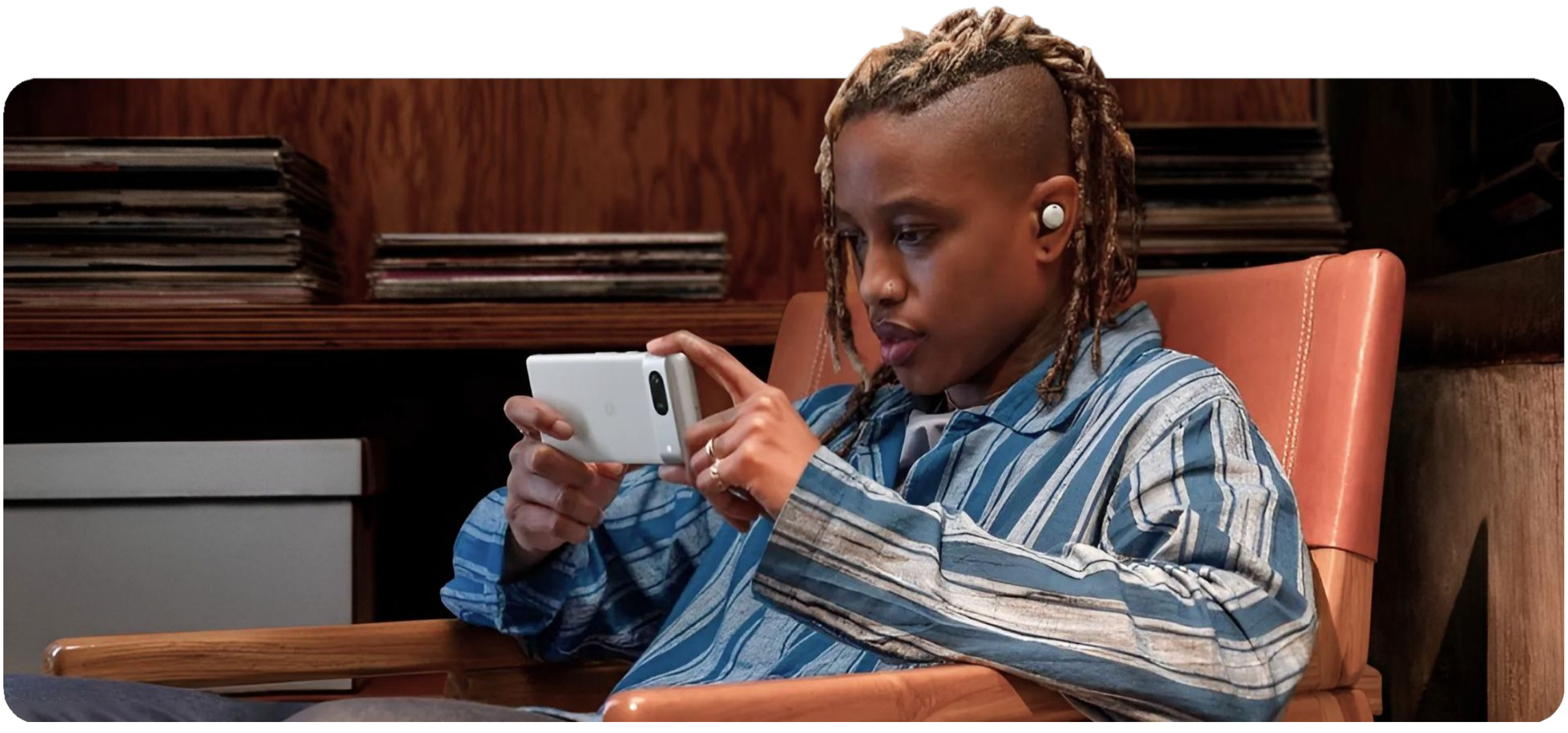
IT admins may need to prevent the installation of applications from outside Google Play or trusted installers, According to the Android Transparency Devices & data can be at an increased risk when apps are installed from unverified sources.

To prevent the installation of apps from unknown sources, IT admins deploying fully managed devices and Work Profiles can set a restriction using their EMM, On devices using a Work Profile, these restrictions apply only to the Work Profile by default. However, IT admins can also set a device- wide policy to also prevent apps from unknown sources being installed into the personal profile.

Summary: Why This Benefits Android Enterprise Customers

This ecosystem provides three primary advantages for business customers:

Benefit	Impact
Enhanced Security	**Google Play Protect** scans every app for malware, while the ability to block 'unknown sources' and use 'Allowlists' ensures only trusted software touches corporate data.
Operational Efficiency	**Silent installs** and **Managed Configurations** mean employees don't have to waste time setting up their own apps or entering server urls; everything is 'ready to work' out of the box.
Privacy & Flexibility	The **Work Profile** allows for strict corporate app management on the same device used for personal life, ensuring company data stays secure without overstepping into the user's private photos or apps.



Security Programs

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

Android Enterprise Partner Program

The Android Enterprise Partner Program empowers partners to build, sell, and support Android products, services, and solutions specifically designed for the enterprise market, Collaborating with a validated partner guarantees that they fulfill essential requirements across three key pillars: Partner Expertise, Product Excellence, and Performance.

Android Enterprise Recommended

Within the Android Enterprise Partner Program, the Android Enterprise Recommended technical product validation establishes a higher standard for enterprise-ready devices and solutions. Devices that satisfy the advanced validation requirements represent the top tier of Android Enterprise specifications, encompassing hardware. deployment processes, and user experience. Organizations can confidently choose devices from this curated list, knowing they meet the rigorous criteria for inclusion in the Android Enterprise Recommended program. Furthermore, participating device manufacturers gain access to enhanced technical support and specialized training.

Android Enterprise Recommended enterprise mobility management solutions have the most advanced management features and deliver a consistent deployment experience, Knowing that these solutions are backed by enhanced training and technical support allows organizations to choose a mobility solution with confidence.

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.

Android partner vulnerability initiative

The Android Partner Vulnerability Initiative (APVI) aims to manage security issues specific to Android OEMs. The APVI is designed to drive remediation and provide transparency to users about issues we have discovered at Google that affect device models shipped by Android partners.

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.

Advanced Protection Program

The Advanced Protection Program safeguards users with high visibility and sensitive information from targeted online attacks. New protections are automatically added to defend against today's wide range of threats.

Here are some of the protections:

- Protects your account from phishing by requiring the use of a passkey or a security key to verify your identity and sign in to your Google Account
- Provides extra protection from harmful downloads by utilizing Safe Browsing and Google Play Protect
- Keeps your personal information secure by allowing only Google apps and verified third-party apps to access your Google Account data, and only with your permission

App Defense Alliance

> The mission of the App Defense Alliance is to protect Android users from bad apps through shared intelligence and coordinated detection between alliance partners.

Together, with its members, ESET, Lookout, Zimperium, McAfee and Trend Micro. the alliance has been able to reduce the risk of app-based malware and better protect Android users.

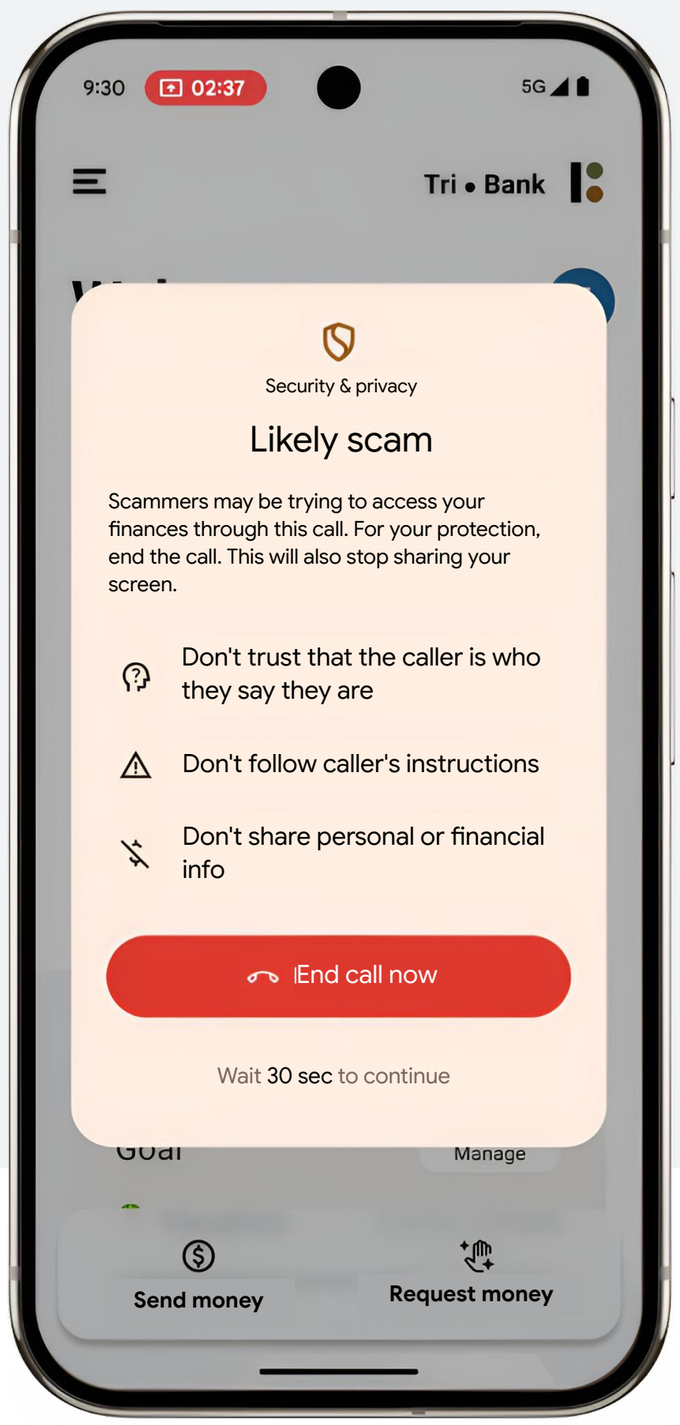
In 2022, the App Defense Alliance expanded to include App Security Assessments where authorized lab partners perform testing services for apps distributed through the Play Store, or Google Partners connecting to Google Cloud Services. Building on the success of the App Defense Alliance, in 2023 Google partnered with Microsoft and Meta as steering committee members in the newly restructured ADA under the Joint Development Foundation, part of the Linux Foundation family. The Alliance supports industry-wide adoption of app security best practices and guidelines, as well as countermeasures against emerging security risks.

Bank Scam Warnings

Screen sharing and remote access scams are becoming increasingly common, with fraudsters often impersonating banks, government agencies, and other trusted institutions—using legitimate screen sharing apps to guide users to perform costly actions such as mobile banking transfers. To better protect Android users from these attacks, this year we piloted new in-call protections for banking apps, starting in the UK.

If you are on a phone call with an unknown contact, while sharing your screen, and open a supported banking app. your Android device will warn you about the potential dangers and give you the option to end the call and stop screen sharing with one tap.

Following a successful initial pilot with Monzo, Revolut, and Natwest, this feature has been enabled automatically for additional banks in the UK on Android 11+ devices. We also expanded the pilot to Brazil, India, and the US, starting with selected partner banks and fintechs in each region.



Industry standards and certifications

Devices running Android and the cloud services they utilize comply with various industry standards and have received numerous security certifications which demonstrate our strong commitment to the highest security standards. ISO and 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.

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, confidentiality. and privacy.

Independent credentialed auditors perform thorough audits 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.

Government grade security

NIST FIPS 140-3/140-2 CMVP & 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 standards 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. The goal of the Cryptographic Module Validation Program (CMVP) is to promote the use of validated cryptographic modules and provide Federal agencies with a security metric to use in procuring equipment containing validated cryptographic modules.

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.

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). In partnership with NIST, NIAP approves Common Criteria Testing Laboratories to conduct security evaluations in private sector operations across the U.S. 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 (AOSP), which enables device manufacturers with the ability to attain certification in much less time. The Android Management API (AMAPI) client has also achieved certification as part of the mobile device evaluation.

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 detect and prevent PHAs or malware. Google Safe Browsing in Chrome and WebView 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.