Android Security Paper

2024

Android

Contents

About the Android Security Paper

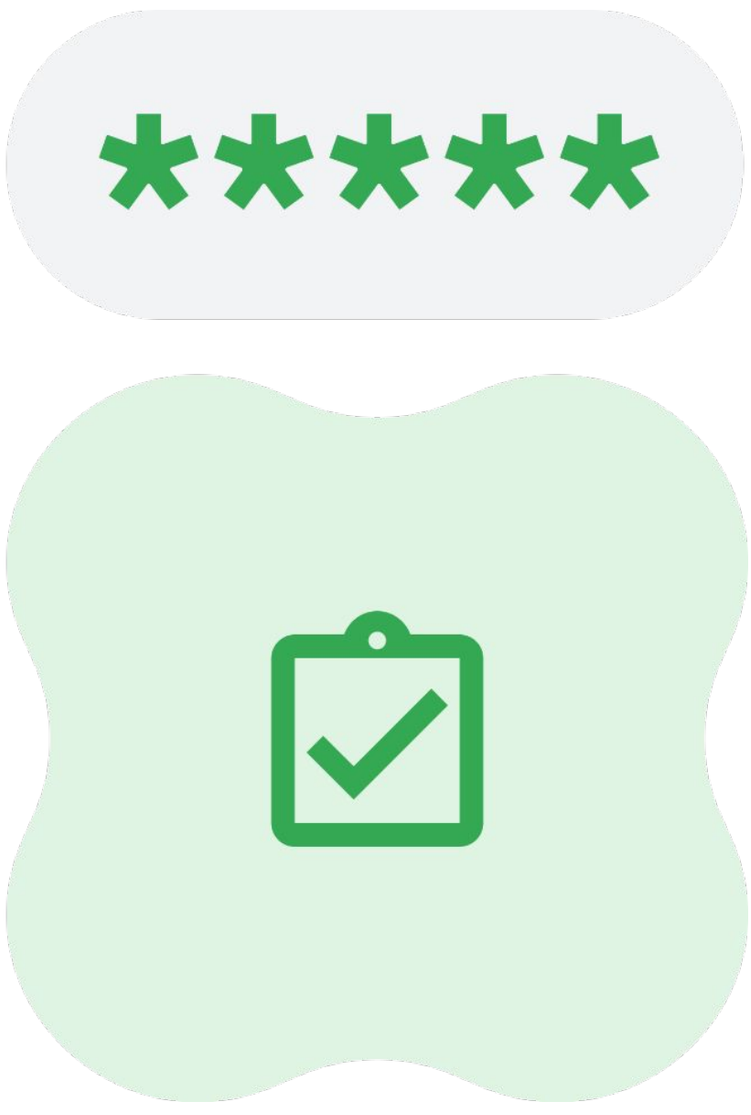
In today's modern world, mobile devices play a critical role in both our work and personal lives. They accompany us everywhere – whether at home, on the go, or at the office – which means protecting them against cyber threats has never been more important.

And because mobile devices play such a critical role in our lives, they're an attractive target for bad actors—with 83% of all phishing sites specifically targeting mobile devices.

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.

Keep reading to learn more.



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 needed 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:

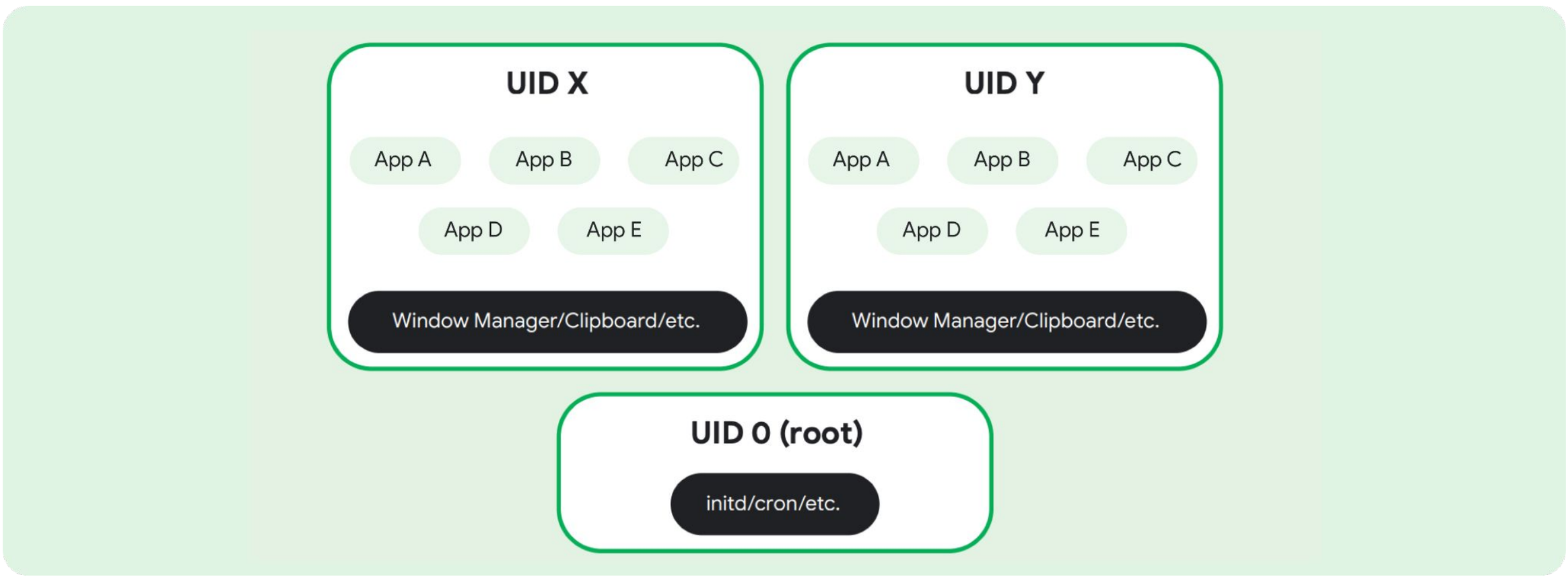


Figure 1. Relationship between users, apps, and system services in Linux.

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:

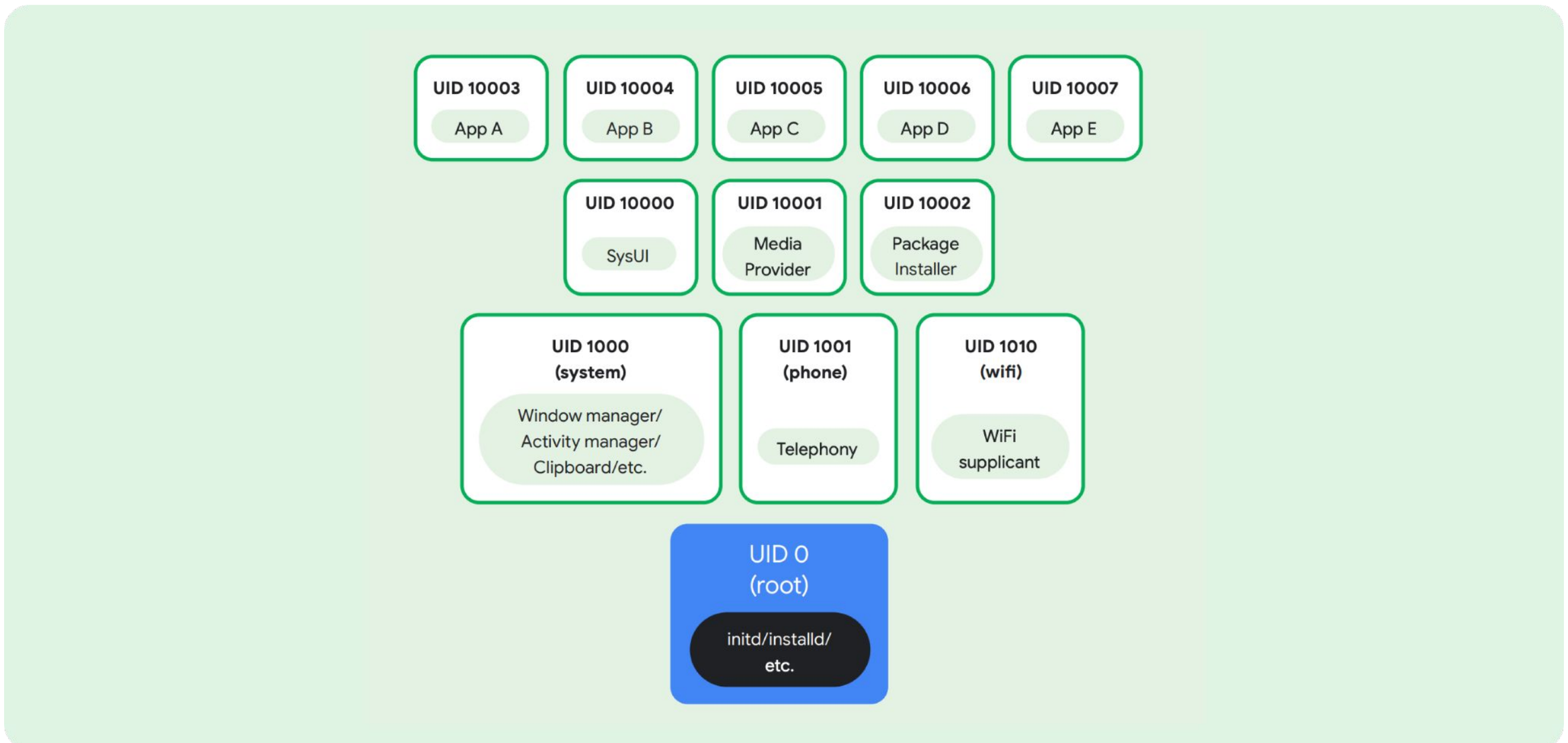


Figure 2. Security architecture

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

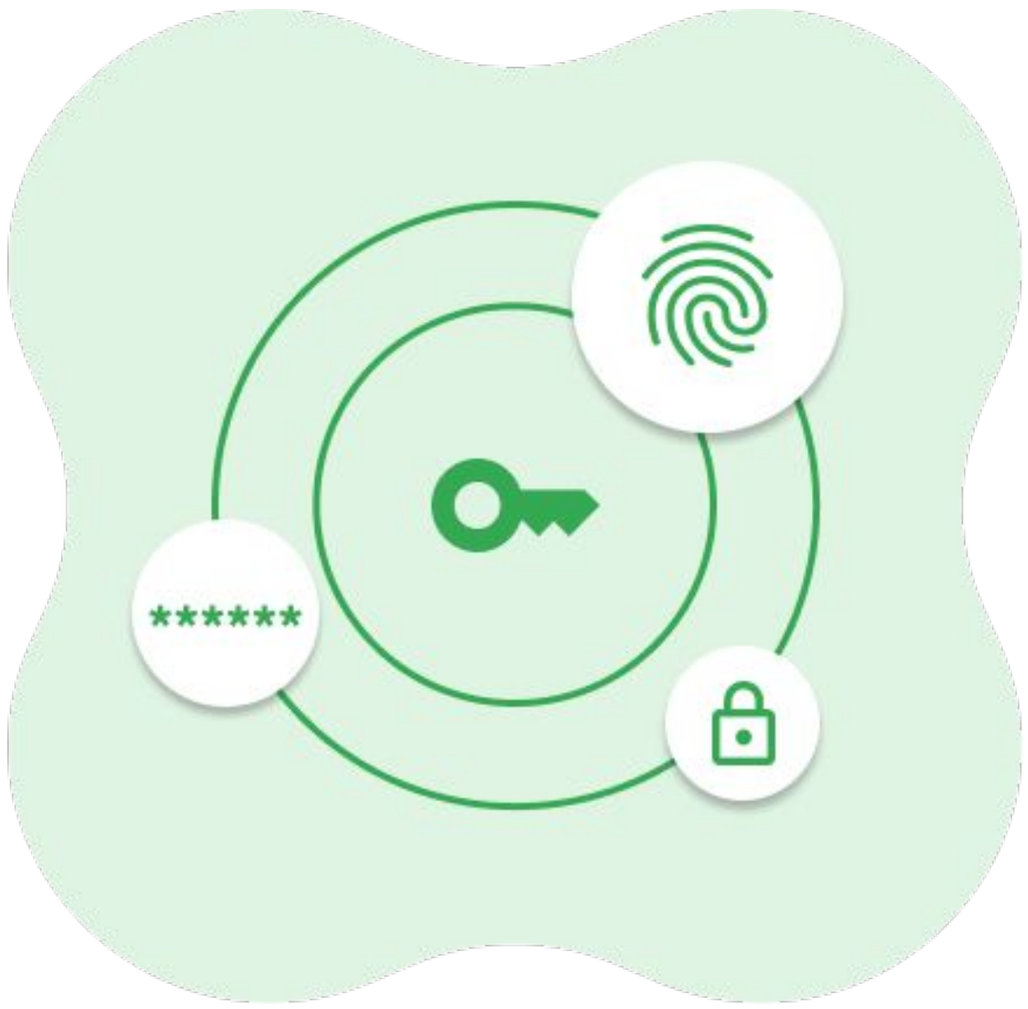
To define the sandboxes for applications, there must be the concept of a secure identity for that application. A well-established way to do this is with a signing certificate, where the application provides a public key identifying who its source is, and a private key to sign the code. The operating system then uses the public key to verify that the application came from the author it claims 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.



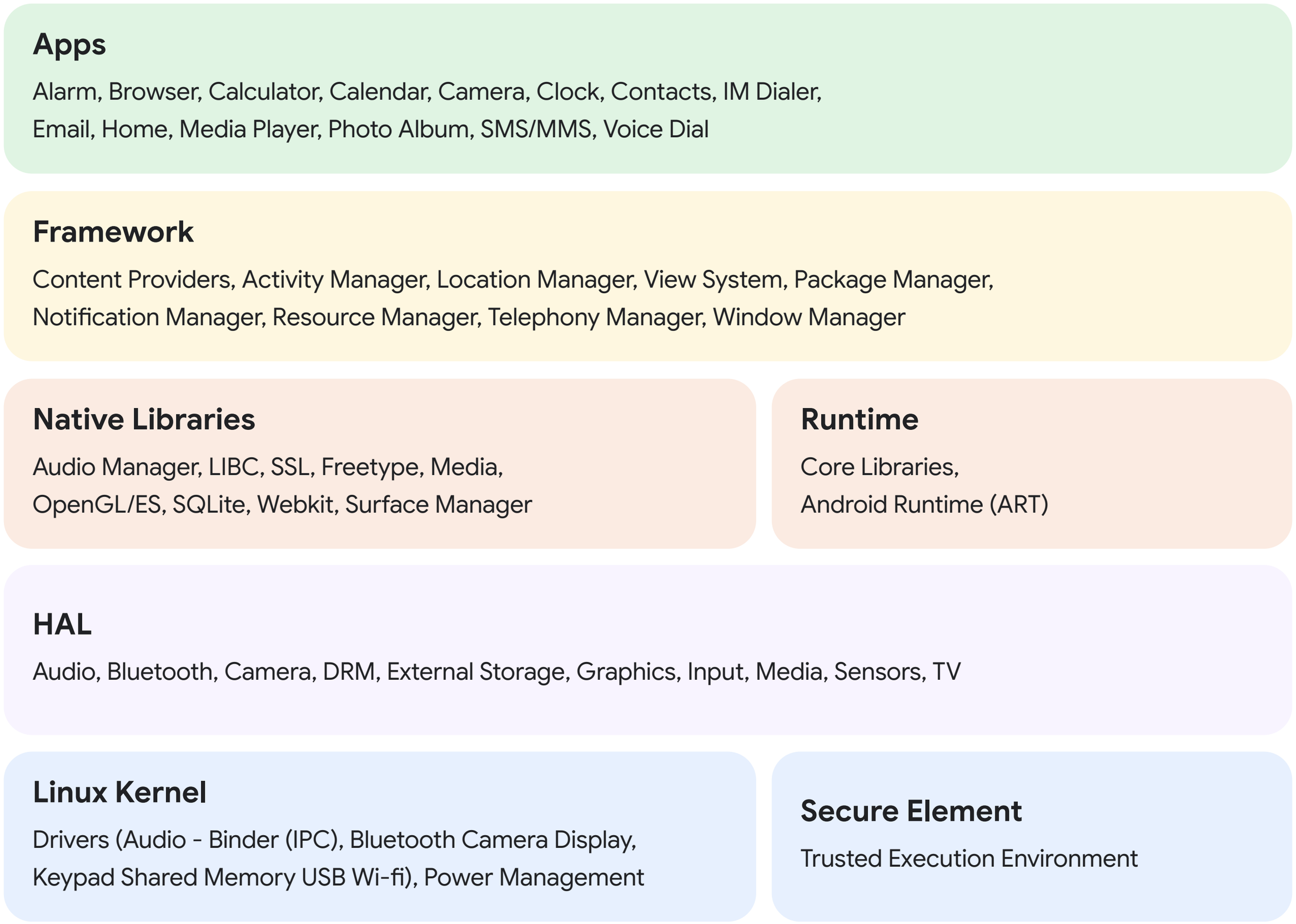


Figure 3. The Android software stack

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 15

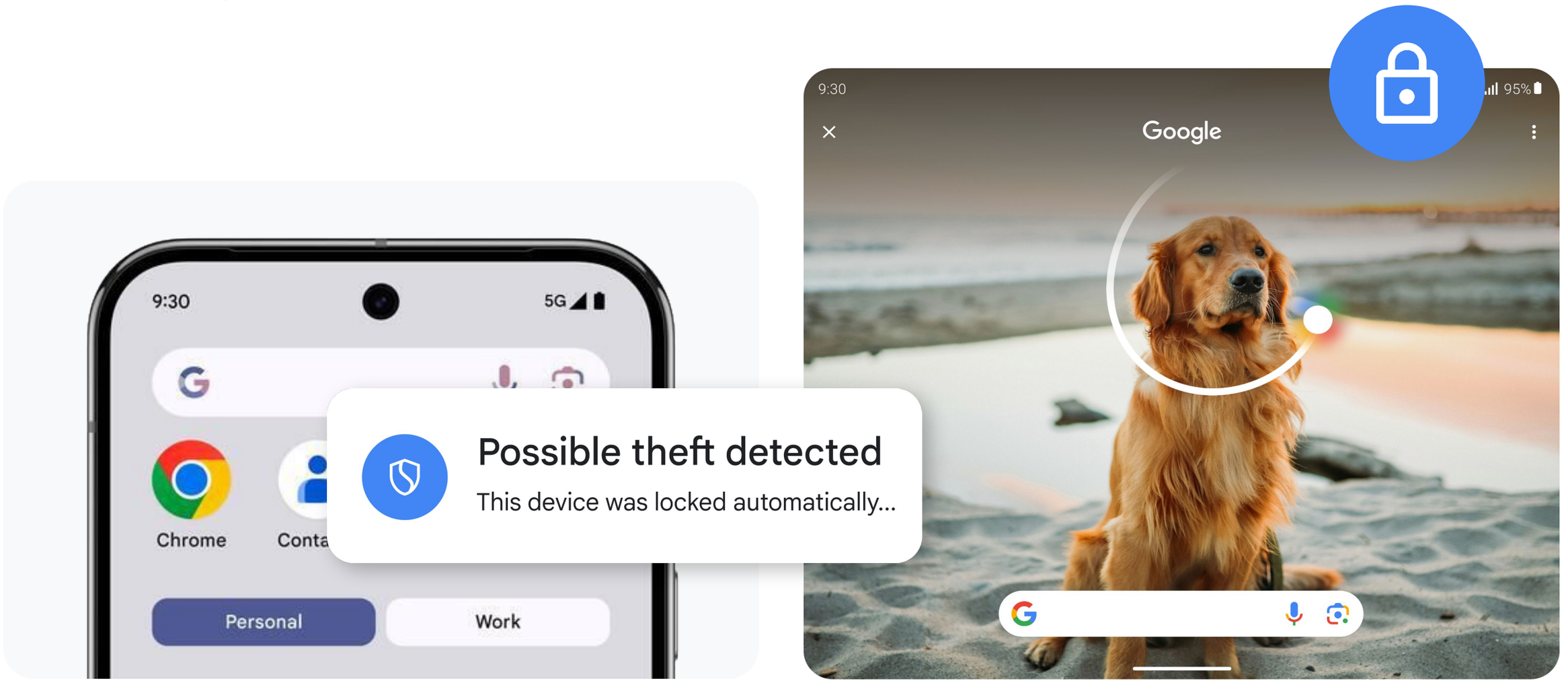
Empowering the Modern Mobile Workplace

In the wake of the modern workplace's shift toward distributed and edge work models, businesses have fully embraced the flexibility and productivity these work arrangements offer. In this modern workplace, securing and managing mobile business devices is more important than ever.

Android 15 addresses the needs of distributed and edge work use cases with enhanced device protection and more management options for company-owned devices. Android 15 introduces several new features to protect devices from theft and unauthorized access. Android theft protection uses machine learning to lock the device screen when it suspects a theft has occurred. Additionally, Google's machine learning and AI features come with the controls to manage them on your terms, to include turning them off.

AICore offers on-device generative AI capabilities on selected Android devices and is compliant with Private Compute Core - a specialized sandbox to privately process data on-device and provide functionality without calling apps gaining access to the underlying data. Additionally, you can learn more about AI Privacy on our Security Blog.

For company-owned devices, Android 15 provides IT admins with more control over personal profiles while still preserving user privacy. This includes the ability to choose what apps are available in personal profiles on company-owned devices, as well as the ability to manage and deploy eSIMs.



Android 15 prioritizes the security and privacy of both employees and their devices. Key features include:

Android theft protection

A suite of theft protection features, designed to protect users before, during and after a theft, is available through a combination of Android 15 and Google Play services updates. The suite includes features like Theft Detection Lock, which uses machine learning to automatically lock a device when a theft motion is detected, and Offline Device Lock, which activates when a device is off the network for too long. Remote Lock allows a user to lock their device screen by entering their phone number at android.com/lock from any device.[1]

Private space

Give users more control to organize and hide sensitive apps with private space—a separate folder that requires a password or biometrics to unlock.[2]

NIAP audit logging requirements

Ensure compliance with industry standards and facilitates security reviews through improved audit logging.

[1]Theft Detection Lock, Offline Device Lock, and Remote Lock requires Android 10+ and an internet connection. Android Go devices are not supported. Support may vary based on your device model. The user must be using the phone while it is unlocked. All theft protection features will be available in October.

[2]Private space on COPE devices are subject to the same security requirements as the personal profile. IT admins will be able to block the user from having a private space and remove an existing private Space in COPE. Private space is not available on Fully Managed devices.

Android 15 introduces enhanced management capabilities for company-owned devices, streamlining IT operations and improving control. Key features include:

01

Simplified eSIM management

Empowers IT admins with granular control over eSIMs. Enabling them to streamline device setup and management by programmatically adding, removing, and provisioning eSIMs, which simplifies large-scale deployments.

02

Security restrictions for apps outside the Work Profile on company-owned devices

Apply a limited set of security restrictions in a privacy preserving way to selected apps outside the Work Profile. IT admins will be able to apply their existing personal app policies to the new private space feature. In the future, additional privacy-safe security configurations for core apps will be made available, and backportable to Android 15.[3]

03

Enforce the default app selection for calls, messaging, and web browsing when setting up company-owned devices

Default selection for the dialer, messaging, and browser for the personal profile on company-owned devices allows IT admins to set these defaults and prevent the end user from changing them.[4]

04

Controls for Circle to Search on Android Work Profile

Circle to Search is a new way for employees to search within apps by circling, highlighting, scribbling, or tapping. Admin controls are available for Circle to Search on fully managed devices or within Android Work Profile.[5]

[3]AMAPI managed devices will have the ability from Android 15 onward. Managed configurations apply only to company-owned, personally enabled (COPE) devices.
[4]Available only on company-owned, personally enabled (COPE) devices. IT admins can only make an app the default if it's already in the user's personal profile. To ensure OEM defaults for dialer and browser are set, this feature should be configured prior to set up. To enforce defaults after set up, the control must be combined with an app allowlist.
[5]Circle to Search requires internet connection and compatible apps and surfaces. Results may vary depending on visual matches. For Android Enterprise managed devices, the feature is available on fully managed devices and devices with Android Work Profile. For company-owned, personally enabled (COPE) devices, Circle to Search is subject to the IT admin's ability to turn off screen capture, which will disable the feature. For employee-owned devices with an Android Work Profile, Circle to Search within the personal profile remains unaffected by IT admin policies. Available on Pixel P8, P8 Pro, P6 series, P7 series, Pixel Fold, Pixel Tablet, Samsung S24 series, S23 series (incl. FE), S22 series, S21 series, Z Flip 3/4/5, Tab S9 series, Tab S8 series.

Improvements to customer sign-up & account governance

Launched in 2024, the updated Android Enterprise sign-up process now provides organizations with a Managed Google domain for enhanced IT security. Organizations register with their actual company email, replacing the need for throwaway Gmail accounts. IT admins gain an enterprise-grade Managed Google account to configure Android Enterprise and other Google services within the Google admin console. These accounts bolster security with two-factor authentication (2FA), improved recovery, and single sign-on (SSO) support using third-party identity providers (OIDC & SAML).

Once IT admins verify ownership of their organization's domain, they can deploy Managed Google Accounts to all employees. This can be done manually or by synchronizing accounts from their existing identity provider. Employees then benefit from enhanced account security (like 2FA and SSO), the ability to sign in to Android devices with work credentials, and increased productivity through access to Google services and multi-device experiences. IT admins always have full control over which services are available.

Security by design

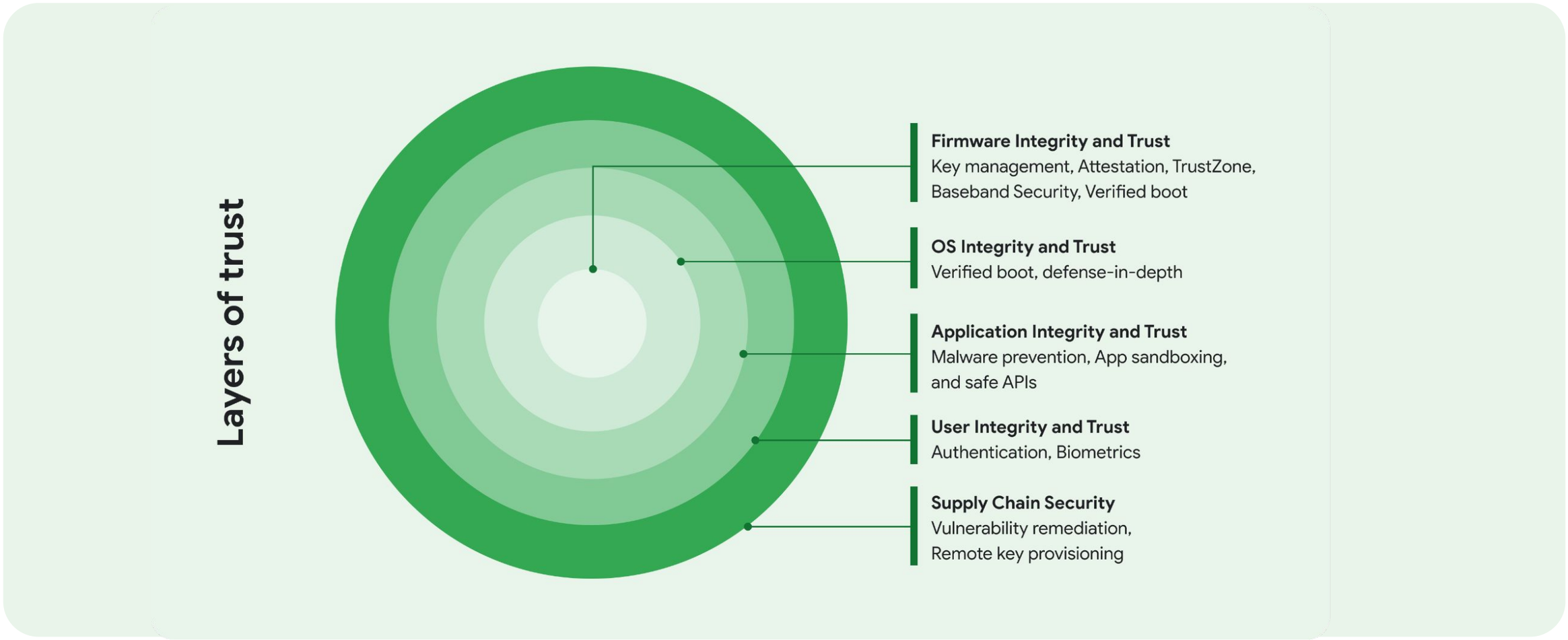


Figure 4. Security by design

Android uses a combination of hardware and software protections to create strong defenses. Security starts at the hardware level, where the user is authenticated 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 both data in transit and data at rest.

> **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 200 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, 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.

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.**

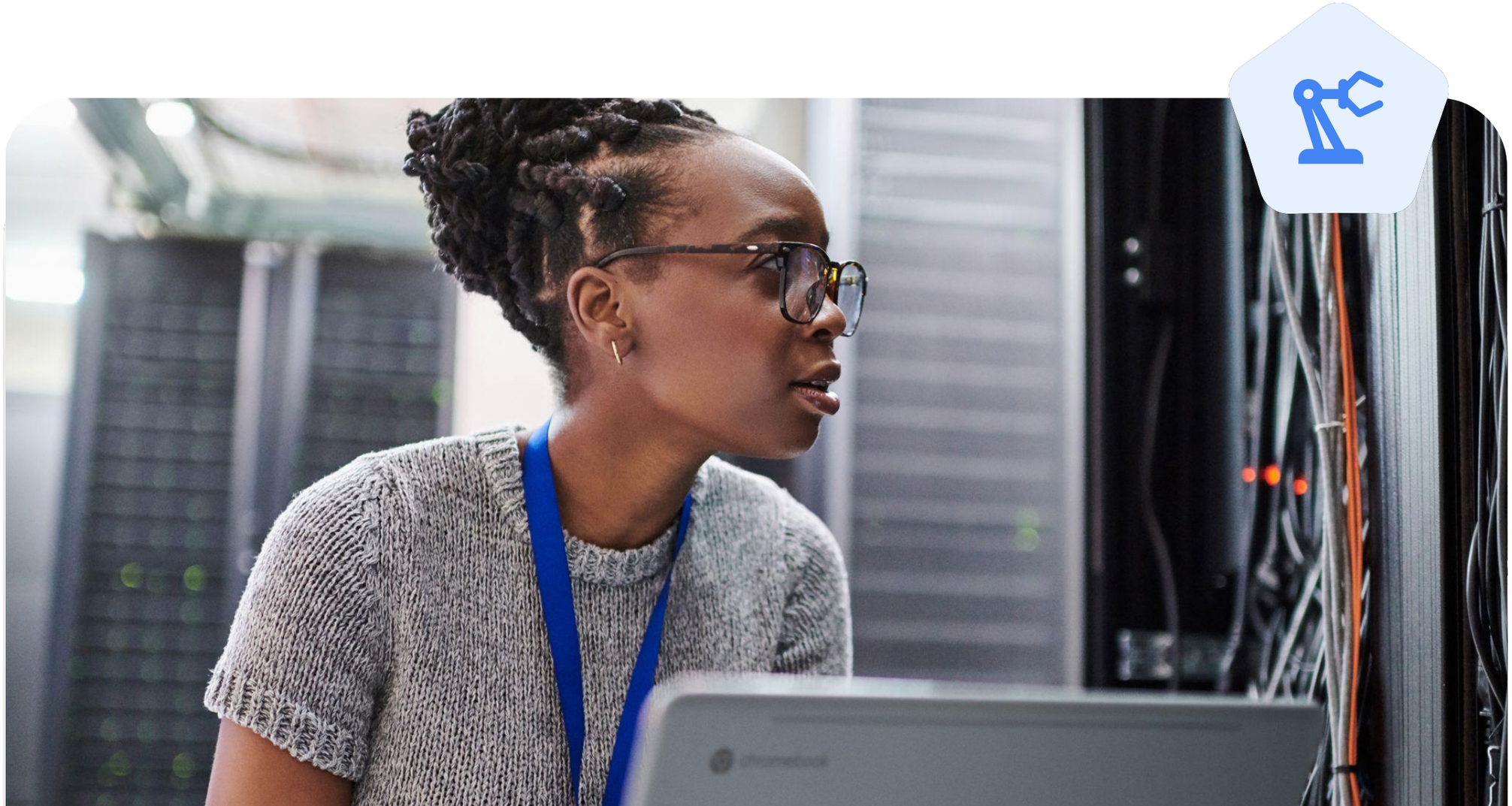
This is very important to help confirm a user's intentions, for example, when they initiate a sensitive transaction like making a banking payment or approving an insulin pump injection. App developers can utilize the feature to display a prompt to the user, asking them to approve a short statement that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

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:

Sanitizers such as HWASan help find memory safety bugs in pre-release testing.

GWP-ASan and KFENCE allow probabilistic detection of memory safety bugs in production.

Rust is a modern native programming language that is memory safe.

Sanitizers

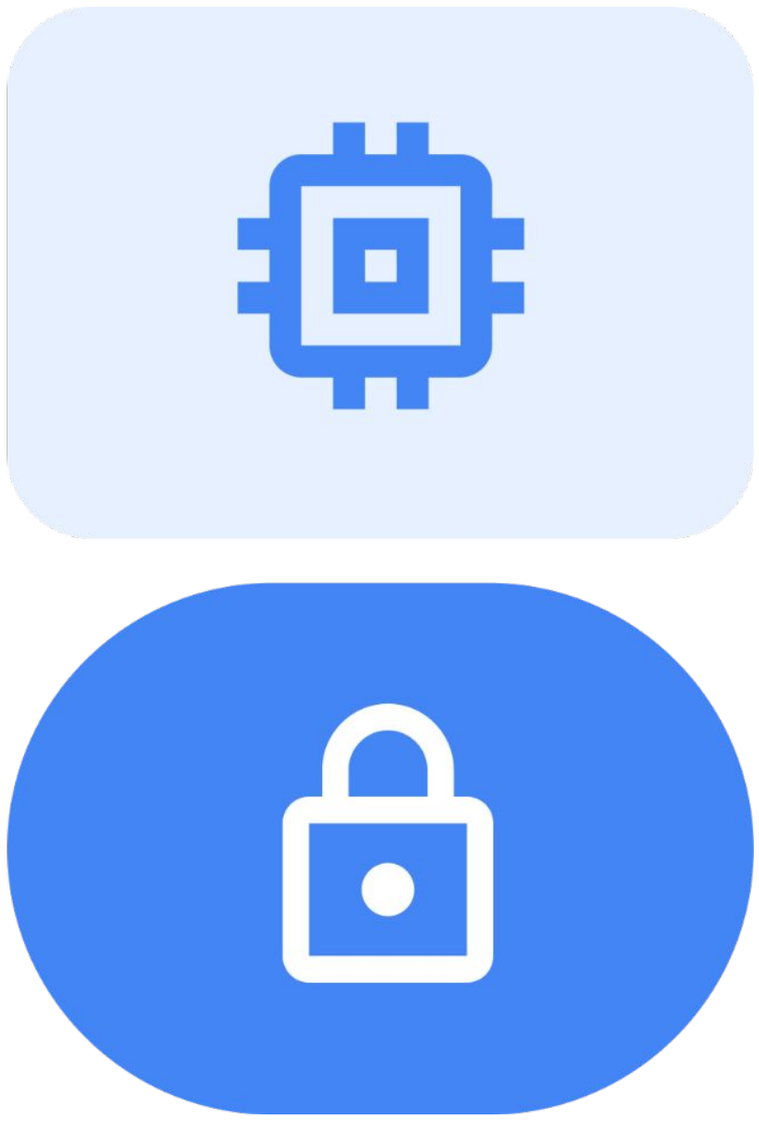
Android has supported HWASan since Android 10. This tool has been used extensively for Android platform development, preventing many bugs from making it into Android releases. The app developer workflow for using HWASan has been significantly improved in Android 14 and we hope it will gain more widespread usage.

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.

GWP-ASan and KFENCE

GWP-ASan and KFENCE are probabilistic memory detection tools for production usage in userspace and the kernel, respectively. When enabled, a small number of allocations are guarded against memory safety bugs. Even with a small sample rate for the guarded allocations, when deployed at scale they can effectively detect memory safety bugs. This is used for some 1P apps and is available for 3P applications. Android 14 introduced a "recoverable" GWP-ASan mode that is enabled by default for all 3P applications.



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 runs in its own UID and is thus isolated from the operating system components and other apps. SELinux enforces Mandatory Access Control policies and is the primary means by which the isolation among processes, apps and system services is achieved.

- **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.
- **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 Linux/Unix permissions 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.

Anti-Exploitation

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

Android uses BoundsSanitizer (BoundSan), which adds instrumentation to insert bounds checks around array accesses. These checks are added if the compiler cannot prove at compile time that the access will be safe and if the size of the array will be known at runtime. BoundSan is deployed in Bluetooth, media codecs, and other components throughout the platform.

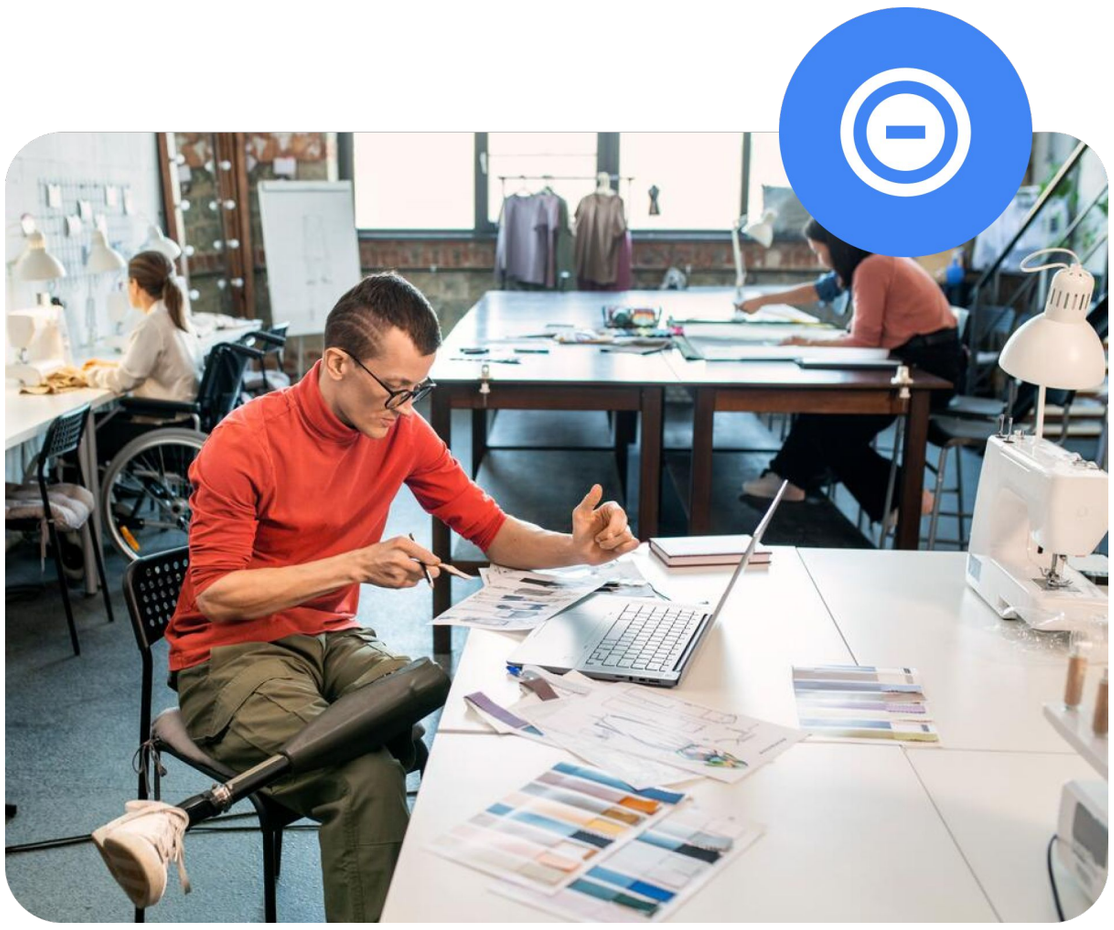
Unintended integer overflows can cause memory corruption or information disclosure vulnerabilities in variables associated with memory accesses or memory allocations. To combat these types of overruns, Clang, which is the compiler used to build Chrome, was added to support UndefinedBehaviorSanitizer (UBSan) to signed and unsigned integer overflow sanitizers to harden the media framework. UBSan covers more components to improve build system support.

This is designed to add checks around arithmetic operations/instructions—which might overflow—to safely abort a process if an overflow does happen. These sanitizers can mitigate an entire class of memory corruption and information disclosure vulnerabilities where the root cause is an integer overflow, such as the original Stagefright vulnerabilities. As a side effect, components hardened with these mitigations also have better quality and stability.

Android also uses Scudo, a hardened memory allocator which employs multiple defense-in-depth strategies to detect and prevent use-after-free, double-free, and bounds-violations. This provides additional hardening of the platform and prevents memory unsafe errors from becoming exploits.

> **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.**

Android's BoundsSanitizer and the expanded UndefinedBehaviorSanitizer provide higher capabilities to detect and prevent memory-related issues. Additionally, the hardened memory allocator Scudo further enhances platform security by mitigating vulnerabilities like use-after-free and double-free errors.



User and data privacy

Protecting user privacy is fundamental to Android. Limiting background apps' access to device sensors, restricting information retrieved from Wi-Fi scans, and implementing new permission groups related to phone calls and phone states help ensure more user privacy. This affects all apps, regardless of target SDK version or version of Android.

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

Android Work Profile creates a separate, self-contained enterprise controlled space on Android devices that isolates corporate apps and data from personal apps and data. Work Profiles can be added to personal devices in a BYOD setting or on a company-owned device used for both work and personal purposes. With this separate profile, the user's personal apps and data in the personal profile are outside of IT control.

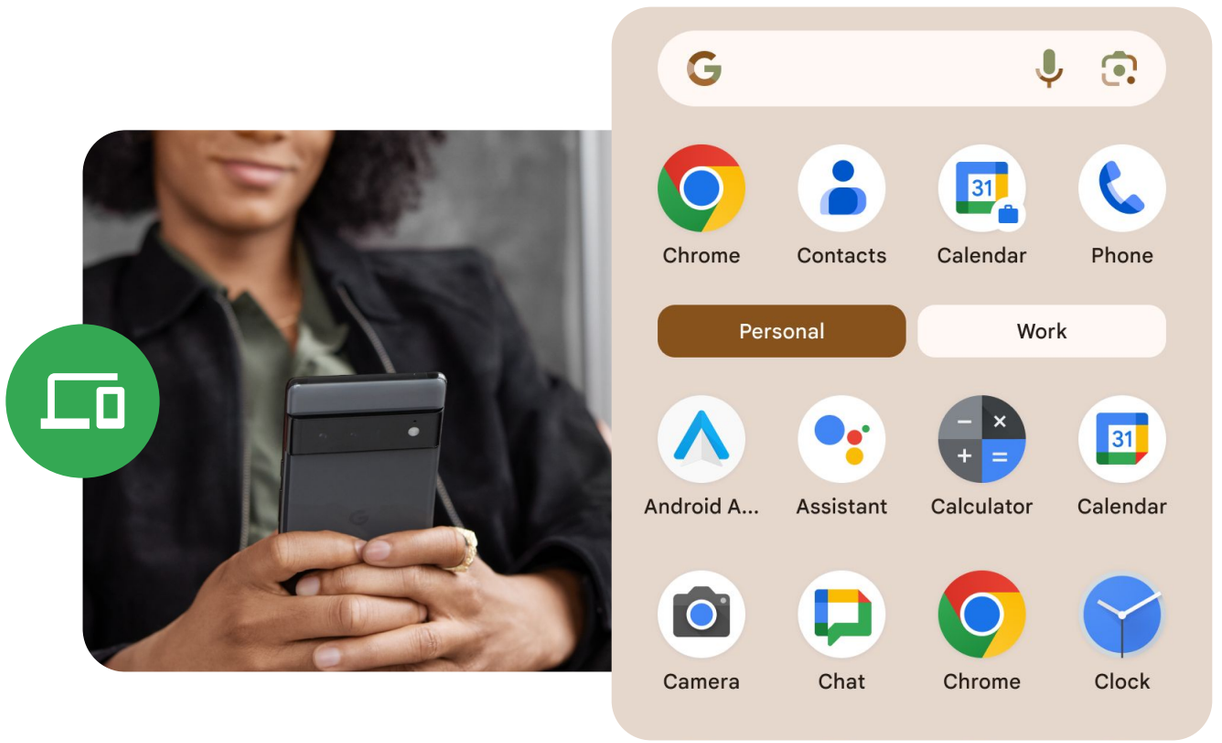
To provide clear visibility to the user, when a Work Profile is applied to a device, the EMM Device Policy Controller (DPC) presents the terms of use and provides information relevant to data collection. The user must review and accept the user license agreement to set up the Work Profile.

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

Restricting access to device identifiers

To protect the privacy of its users, Android limits the visibility of device identifiers. Android limits access to device identifiers like IMEI, IMSI, and serial numbers to protect user privacy. In Android 10 and higher, only certain apps with special permissions or carrier privileges can access these identifiers.

Android uses random MAC addresses when searching for new networks on Android 10 and higher by default. This enables a randomized MAC address assigned when connecting to a Wi-Fi network to enhance privacy.



Android

Location control

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

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

When users grant apps continuous access to their device's location, the system sends a reminder notification the first time an app uses this access in the background.

In Android 12, users gained the ability to grant an app access to their approximate location, rather than their precise location. A lot of apps require location permissions in order to operate properly but these permissions expose more information than a lot of users are comfortable sharing. Giving users the ability to choose between approximate and precise location allows apps to function without the app knowing the user's precise location.

Privacy indicators

Runtime permissions in Android let users decide when apps can access their device's microphone and camera. Before an app can record, the system prompts the user to grant or deny this permission. Additionally, Android 12 enhances transparency by showing indicators when an app uses the camera or microphone.

Storage access

To enhance user control over their files and reduce clutter, apps designed for Android 10 and newer versions have restricted file access capabilities by default. This is known as scoped storage. Apps can freely access their own dedicated directory, but they need permission to create files in shared storage areas.

To access media files (like images, videos, or audio) created by other apps in shared storage, an app needs the corresponding permission (READ_MEDIA_IMAGES, READ_MEDIA_VIDEO, or READ_MEDIA_AUDIO). The user must grant these permissions at runtime.

To access location metadata for photos or videos, an app need an additional permission: ACCESS_MEDIA_LOCATION. Even with read access, an app needs explicit user approval to modify or delete any media files.

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

Limited access to background sensors

Android apps running in the background have limited access to user input and sensor data:

- The app cannot use the microphone or camera.
- Sensors like accelerometers and gyroscopes that continuously report data won't function.
- Sensors that only report data when something changes or once (on-change or one-shot) are also disabled.

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

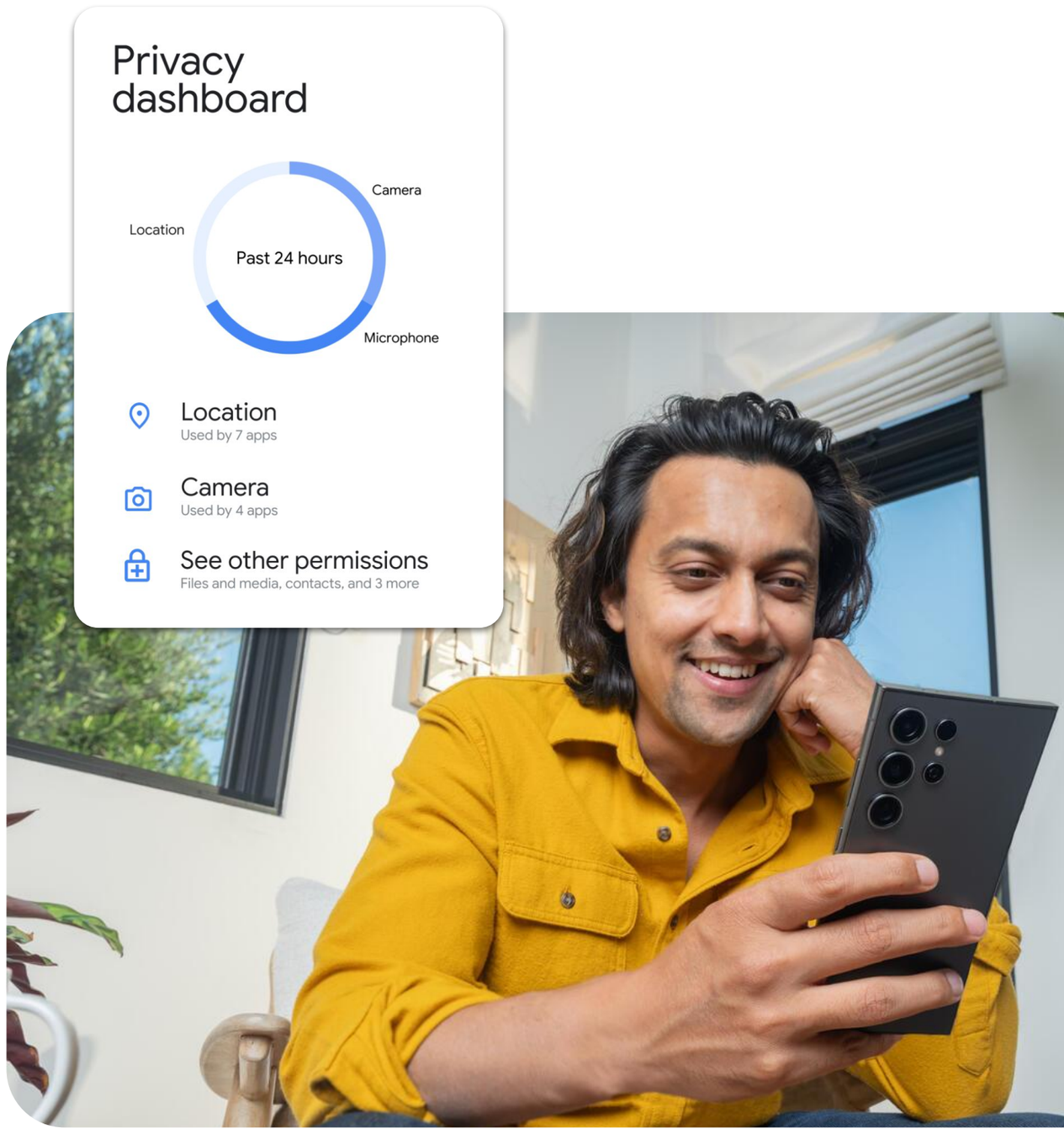
Privacy dashboard and permission manager

Android's privacy dashboard provides users with valuable insights into how their data is being accessed by apps. The dashboard provides users a clear overview of which apps have accessed their location, camera, microphone, and other device data. Giving users this information helps them make informed decisions about allowing or revoking permissions to their apps.

Users can also check which apps have the same permission setting and change an app's permission in the Permission manager.

Lockdown mode

On Android 9 and above, a user can enable a lockdown option to further restrict access to the device. This mode displays a power button option that turns off Smart Lock, biometric unlocking, and notifications on the lock screen. On managed devices, Enterprise IT admins can remotely lock the Work Profile and evict the encryption key from memory.



Private Compute Core

Android's Private Compute Core is an open source, secure environment that is isolated from the rest of the operating system and apps. Private Compute Core (PCC), introduced in Android 12, brings a new layer of privacy to the Android ecosystem. PCC enables on-device processing for features like live caption, now playing, Smart Reply, AI core and many others by eliminating the need to send data to the cloud for processing.

Screen recording detection

Screen recording is a common use case for capturing contents on users devices. Android 15 adds support for apps to detect that they are being recorded. This enables developers to take appropriate action to safeguard user privacy and inform the user that their screen is being viewed or recorded.

Private space

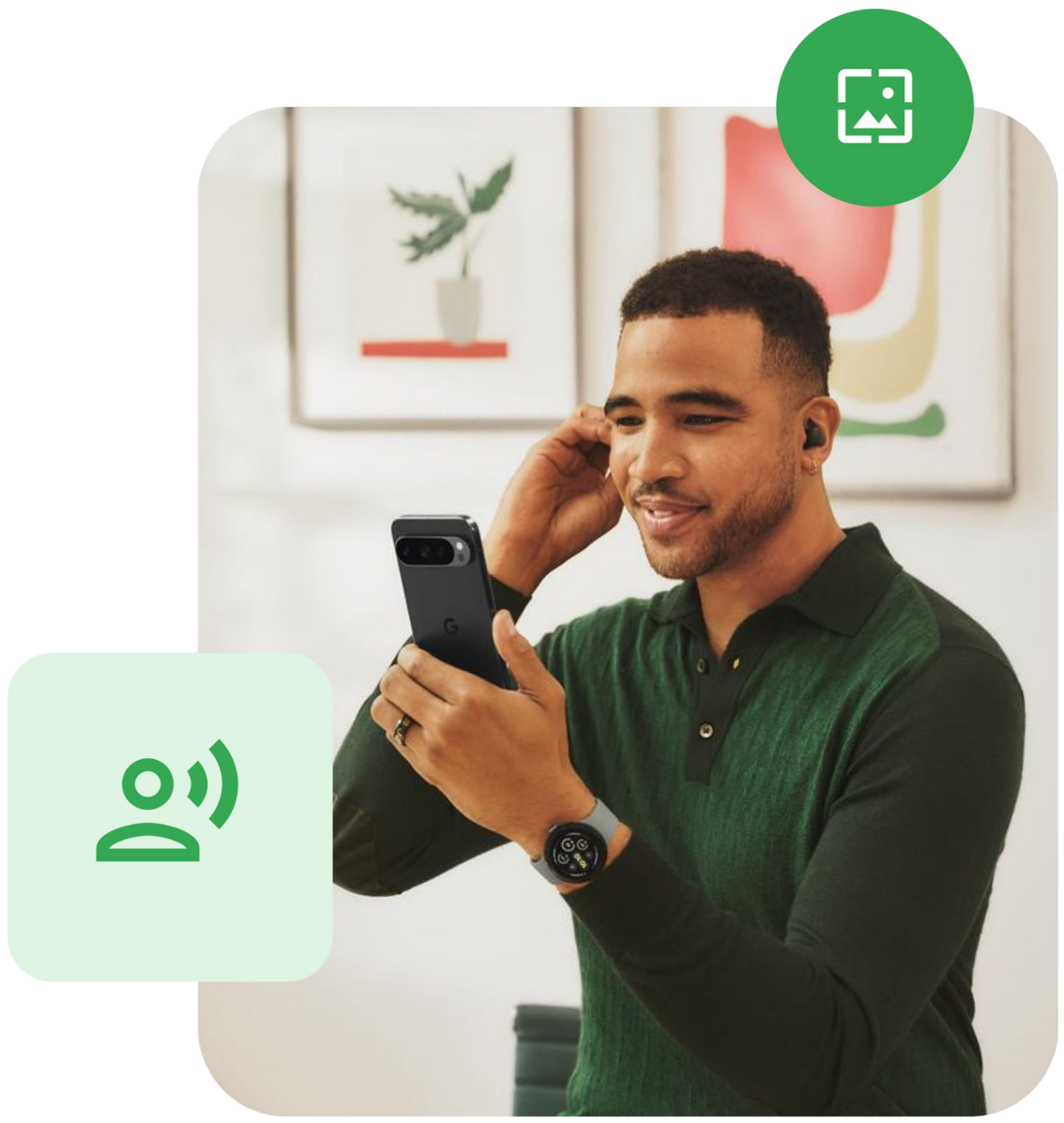
Private space lets users create a separate space on their device where they can keep sensitive apps away from prying eyes, under an additional layer of authentication. The private space uses a separate user profile. The user can choose to use the device lock or a separate lock credential for the private space.

Apps in the private space show up in a separate container in the launcher, and are hidden from the recents view, notifications, settings, and from other apps when the private space is locked. User-generated and downloaded content (such as media or files) and accounts are separated between the private space and the main space. The system share sheet and the photo picker can be used to give apps access to content across spaces when the private space is unlocked.

Users can't move existing apps and their data into the private space. Instead, users select an install option in the private space to install an app.

Apps in the private space are installed as separate copies from any apps in the main space (new copies of the same app).

When a user locks the private space, the profile is stopped. While the profile is stopped, apps in the private space are no longer active and can't perform foreground or background activities, including showing notifications.



Hypervisor and Virtualization

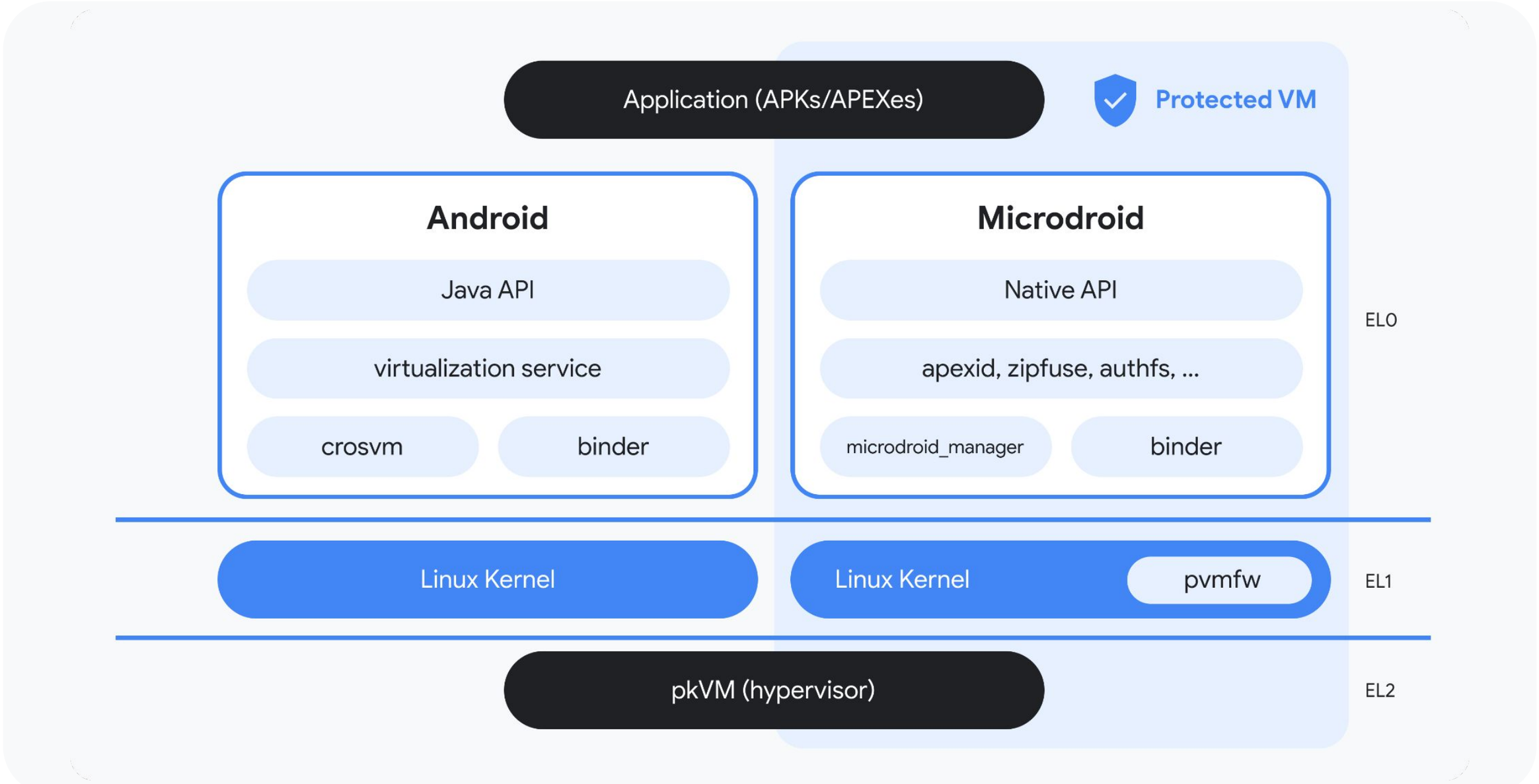


Figure 5. AVF framework

The Android Virtualization Framework (AVF) unifies various hypervisors within a single framework, offering standardized APIs for executing sandboxed workloads. Protected virtual machines (pVMs) powered by AVF offer a novel isolation primitive, superior to the standard mechanisms offered by the operating system alone, while remaining readily accessible to Android developers. These pVMs represent the next generation of Isolated Execution Environments (IEE), essential for numerous Android applications today. As the hypervisor guarantees the isolation and confidentiality of a pVM, a pVM maintains its security and integrity, even if the Android system is compromised.

The Android Virtualization Framework offers several key advantages. It enables isolation without requiring elevated privileges, unlike TrustZone, where isolation is coupled with higher privilege levels. Additionally, it reduces fragmentation by providing a standardized virtual machine environment that abstracts the complexities of various implementations. Finally, it improves the updatability and portability of applications running within protected virtual machines (pVMs) by allocating resources on demand. These features contribute to a more secure, flexible, and efficient Android ecosystem.

The Android Virtualization Framework comprises several key components. The foundation of this framework is the Hypervisor, the underlying technology responsible for its operation. Google provides its own hypervisor known as pKVM (protected Kernel Virtual Machine), while AVF also accommodates third-party hypervisors. The Virtual Machine Monitor (VMM) works in conjunction with the hypervisor, facilitating the management and execution of virtual machines.

Within these virtual machines, an operating system is required, and Android offers Microdroid, an Android-based OS specifically designed for this purpose. However, OEMs retain the flexibility to utilize alternative operating systems within the pVM if they so choose. Finally, the VirtualizationService API provides developers with the necessary tools and interfaces to interact with and leverage the capabilities of the framework.

Hypervisor

The hypervisor creates a secure environment, or sandbox, for the virtual machine and ensures its isolation from other processes. AVF is compatible with various hypervisors, each providing the necessary features to implement the AVF APIs.

pKVM, or protected kernel-based virtual machine, is an open-source hypervisor created by Google. It is integrated into the Android Common Kernel and distributed as part of the Generic Kernel Image (GKI). pKVM is built upon the field-tested Linux KVM hypervisor, with added capabilities to restrict access to designated "protected" virtual machines. The integration of the hypervisor with the Linux kernel enables seamless communication between them. This facilitates the use of existing tools, device drivers, and development processes, streamlining development.

Virtual machine monitor

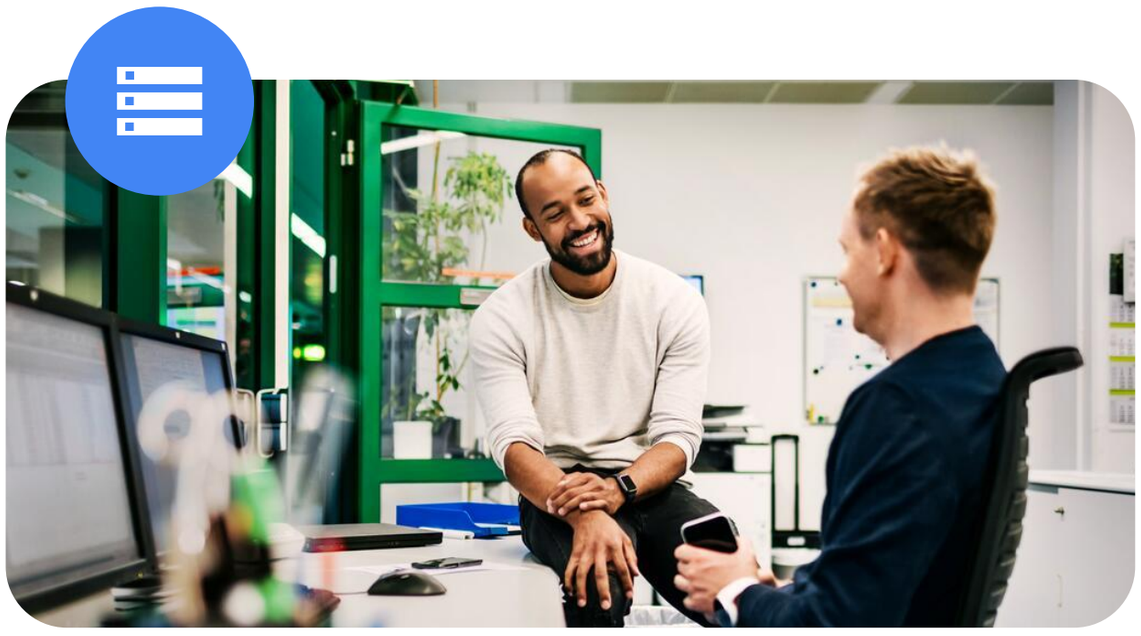
A virtual machine monitor (VMM), a process within the host Android user space, manages the virtual machine's (VM) virtual platform. It allocates CPU resources to the VM and controls the kernel resources the VM utilizes. Each virtual machine has its own VMM process, and terminating this process ends the virtual machine. The hypervisor ensures that the VMM cannot violate any isolation properties.

Crosvm is a virtual machine monitor (VMM) built upon Linux's KVM hypervisor, prioritizing simplicity, security, and speed. It helps provide a secure environment for running native applications.

Microdroid

Microdroid is a minimal OS based on Android, designed to run within a virtual machine and execute native code built against Bionic, communicating via Binder. Its purpose is to facilitate developers in seamlessly transferring parts of their app into a pVM.

Microdroid loads app code from APKs, permits importing APEXes from Android, and provides access to a subset of the Android APIs. In essence, Microdroid offers developers a familiar environment equipped with the tools they've come to rely on within the full Android OS.



PVM Firmware:
Secure bootloader for isolated virtual environments

Secure verification

Before the guest operating system starts, the hypervisor loads the pVM firmware into the pVM. This firmware verifies the integrity and authenticity of the guest operating system. This ensures that only trusted VM images execute within a secure virtual environment.

Trusted Execution Environment

The pVM firmware operates in a protected guest environment that is inaccessible to the host Android kernel. While having fewer privileges than the hypervisor, this isolation effectively prevents tampering from the host system or other applications, ensuring integrity without compromising the core security components of the system.

Streamlined updates

Guest operating system images are distributed as APEX modules, and pVM firmware as an Android partition. This facilitates over-the-air (OTA) updates without requiring platform-specific modifications. This streamlined process enables efficient patching of vulnerabilities, bolstering the overall security of Android devices.

VM Attestation: Enhancing trust and security in protected virtual machines

> **In mobile device security, especially when using sensitive machine learning models, trust is paramount.**

While Protected VMs offer isolation, verifying their integrity and contents is essential. This is where remote attestation becomes critical.

Remote attestation assures a server that a client's virtual machine (VM) is genuine, running valid components, and operating on a trusted device. However, direct attestation is challenging due to the dynamic nature of client VMs and their boot stage records, or DICE chain. To address this, a hardcoded RKP (Remote Key Provisioning) VM serves as a trusted intermediary. The process involves two main steps:

RKP VM Attestation

The RKP VM is periodically verified against a trusted server (RKP server) using its DICE chain and secure boot mechanisms, establishing it as a reliable platform for further validation.

Client VM Attestation

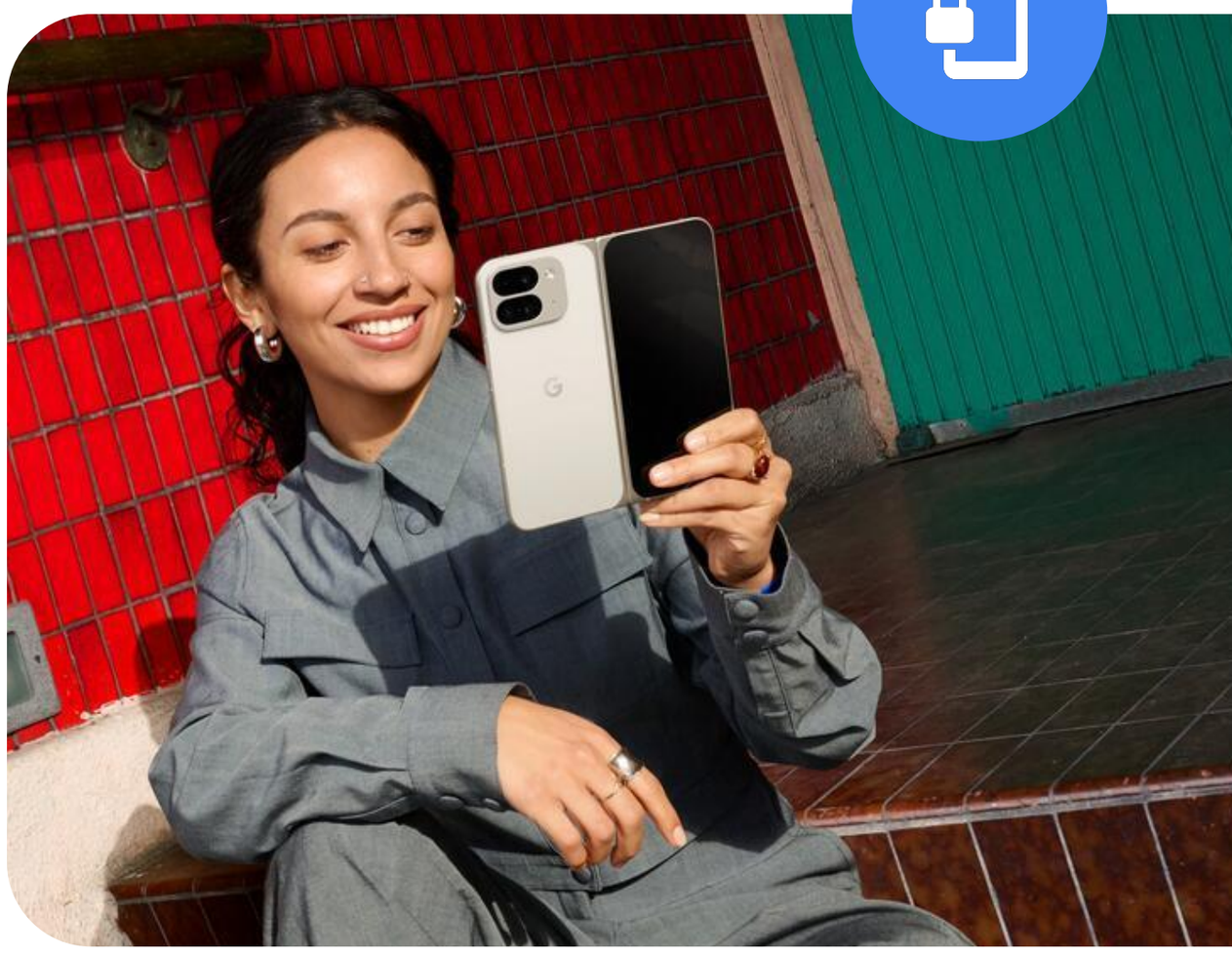
The client VM generates a challenge, a public/private key pair, and a Certificate Signing Request (CSR) containing its DICE chain. The trusted RKP VM verifies the client VM's DICE chain and issues a certificate rooted in the server's trust chain. The client VM can then use this certificate for secure interactions with other services.

Secretkeeper

Another key element in the security architecture is the Secretkeeper. Traditionally in Android, each boot stage is responsible for rollback protection of the next boot image in the process. The Android Bootloader (ABL) has access to tamper-evident storage to ensure this protection.

However, protected VMs do not have direct access to such tamper-evident storage. To enable VM core components and payload upgrades while preserving VM secrets and associated data, and preventing access to potentially downgraded images, the rollback protection scheme in Android was enhanced.

The Secretkeeper HAL was introduced to provide DICE Policy gated storage. Microdroid VMs utilize it to store their secrets, ensuring that only that VM instance or its upgrades can access the secrets, thereby protecting against rollback of boot images.



App security

Apps are an integral part of any mobile platform. Users increasingly rely on mobile apps for core productivity and communication tasks.

> **Android protects your apps with multiple layers of security, enabling users to download apps for work or personal use to their devices with the peace of mind that they're getting a high level of protection from malware, security exploits, and attacks.**

Google security services

Google Play Protect and Play Integrity API are services on Google Mobile Services (GMS) certified devices that help detect malware and device compromise. Exploitation code is often delivered to devices via malware. Android devices using managed Google Play have a Potentially Harmful Application (PHA) installation rate around 0.009%. Google Play Store security is further enhanced through the work of the App Defense Alliance, a collaboration with industry security partners.

In addition, IT admins can create allow-lists and block-lists for the managed Google Play Store to provide greater specificity over which apps are allowed on devices. On company-owned devices with a Work Profile, these controls extend to the Google Play store in the personal profile too. These resources all help to reduce the likelihood of unwanted apps from being installed.

The Play Integrity API helps app and SDK developers check that interactions and server requests are coming from their genuine app binary running on a genuine Android device. By detecting potentially risky and fraudulent interactions, such as from tampered app versions and untrustworthy environments, the app's backend server can respond with appropriate actions to mitigate attacks and reduce abuse.

Customers using an Enterprise Mobility Management (EMM) platform can also use these services to prevent users from sideloading applications. They can ensure users only install applications from Google Play or trusted app stores. EMMs can receive the signals via the Play Integrity API from these on-device services to help detect and mitigate compromises.

App signing

Android requires all apps to be digitally signed with a developer key. APK key rotation, introduced in Android 9, lets apps change their signing key during updates. This is made possible by updating the APK signature scheme from v2 to v3, allowing both old and new keys.

When an app rotates its key, the previous key attests to the new one, becoming part of the app's signing lineage. The previous keys can be granted certain capabilities to allow the app to interact with other apps still signed by previous keys that are still trusted.

Since its introduction, key rotation-related issues have surfaced. To address this, APK signature scheme v3.1 was launched with Android 13. This allows the rotated key to sign the APK in the v3.1 block and the original key in the v3.0 block, ensuring compatibility with older Android versions. New key rotations using apksigner will default to v3.1 for Android 13 and later. Apps that have already rotated their key can specify Android's SDK version as the minimum for rotation. The v3.1 scheme also supports verified SDK targeted signing configs, allowing for stricter capabilities in newer releases while maintaining flexibility for older ones.

Android verifies app updates by comparing certificates. With key rotation, the system checks the signing lineage, allowing updates if the existing version's key attests to the new one.

Apps signed with the same key, or previously in the app's lineage with the SHARED_USER_ID capability, can run in the same process, treated as a single application by the system. This is done via sharedUserId in the manifest.

Android's signature-based permissions allow apps with the same key to share functionality. Even with key rotation, this is possible if both apps share a common signer and the declaring app granted the PERMISSION capability to the previous key. This enables secure code and data sharing.

Developers on Google Play use Play App Signing, delegating key management to Google. This ensures developers can regain update access if they lose their upload key. Google securely stores these keys using Google Cloud Platform's service. Developers can request key upgrades, letting Google handle the complexities of key rotation across Android versions.

> **Note:** sharedUserId is deprecated in Android 11. Apps can specify the sharedUserMaxSdkVersion to control its use in new installs.

App permissions

> **In the world of Android apps, permissions act as gatekeepers, ensuring that apps only access the data and resources they genuinely need.**

Permissions safeguard Android users' privacy and provide transparency about the data and resources apps request access to. Android apps must explicitly request permission to access system features like the camera and internet. They also need permission to access user data such as contacts and SMS. Permission prompts are designed to give users clear visibility into the request and the opportunity to approve or deny it.

The Android security architecture is built on a fundamental principle: no app has automatic permission to perform harmful actions. This means apps cannot, by default, engage in activities that would negatively impact other apps, the operating system, or the user. In other words, apps need explicit permission before they can access sensitive user data (like contacts or emails), modify another app's files, connect to the network, or even keep the device awake.

Android uses runtime permissions, which display a dialog for the user to grant access at the time the app needs it. This approach gives users more control than install-time permissions and streamlines the installation process.

When an app requests permissions, users can choose between:

While using the app

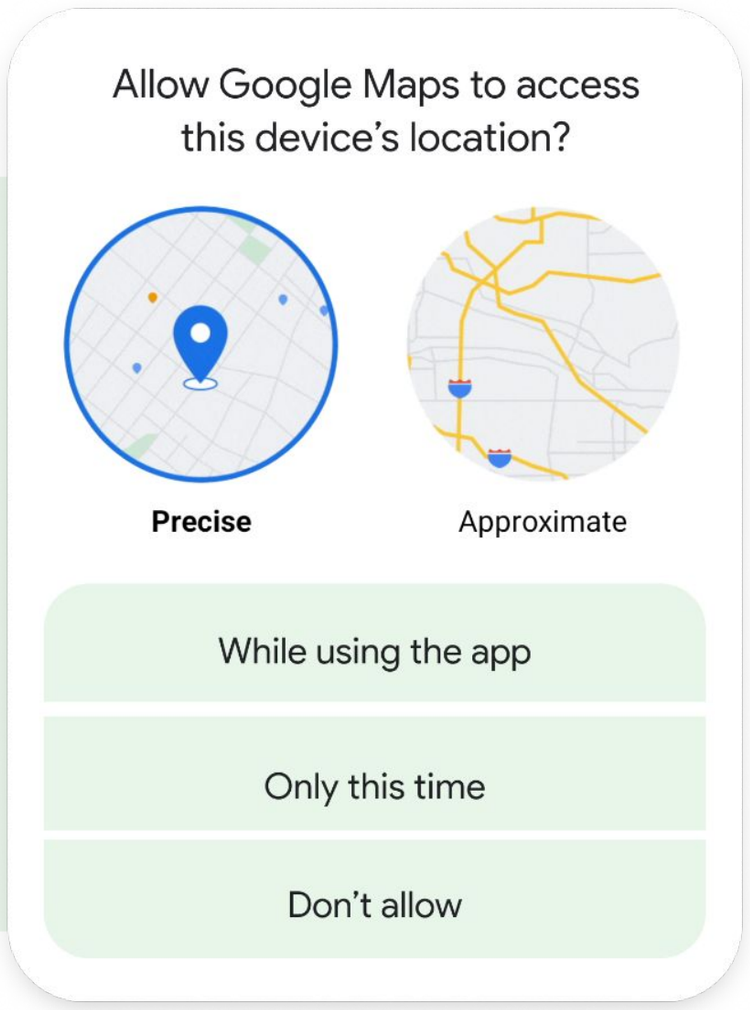
The granted permissions are only valid while the app is actively running.

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.



These options are presented for each permission requested by the app. Granting location permissions doesn't automatically grant storage access. For example, a user can give a camera app access to the camera but not to the device's location. Users can revoke permissions at any time, even if the app targets a lower API level.

Enhanced user privacy controls

Android 13 (API level 33) and higher supports a runtime permission for sending non-exempt notifications from an app (notifications that aren't critical to the app's core functionality). This gives users even more control over which permission notifications they see.

Android has implemented several mechanisms to enhance user privacy and control over sensitive data access:

Foreground app restrictions

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

Stricter background location access

Android 10 introduced stricter background location access, requiring an additional background location permission.

One-time permissions

Android 11 further enhanced user control with one-time permissions for location, camera, and microphone, allowing users to grant permissions only for a single use.

Permission auto-reset

Android 11 also introduced permission auto-reset for apps that target Android 6 or higher and haven't been used for a few months.

Approximate location

Android 12 introduced the option for users to grant apps access only to their approximate location.

Data sharing transparency

In Android 14, starting with apps that share location data with third parties, the system runtime permission dialog now includes a clickable section that highlights the app's data-sharing practices, including information such as why an app may decide to share data with third parties.

Users also have access to provide better control over the use of device identifiers.

Device Identifiers

Privacy-sensitive identifiers are either no longer accessible or require specific permission at runtime.

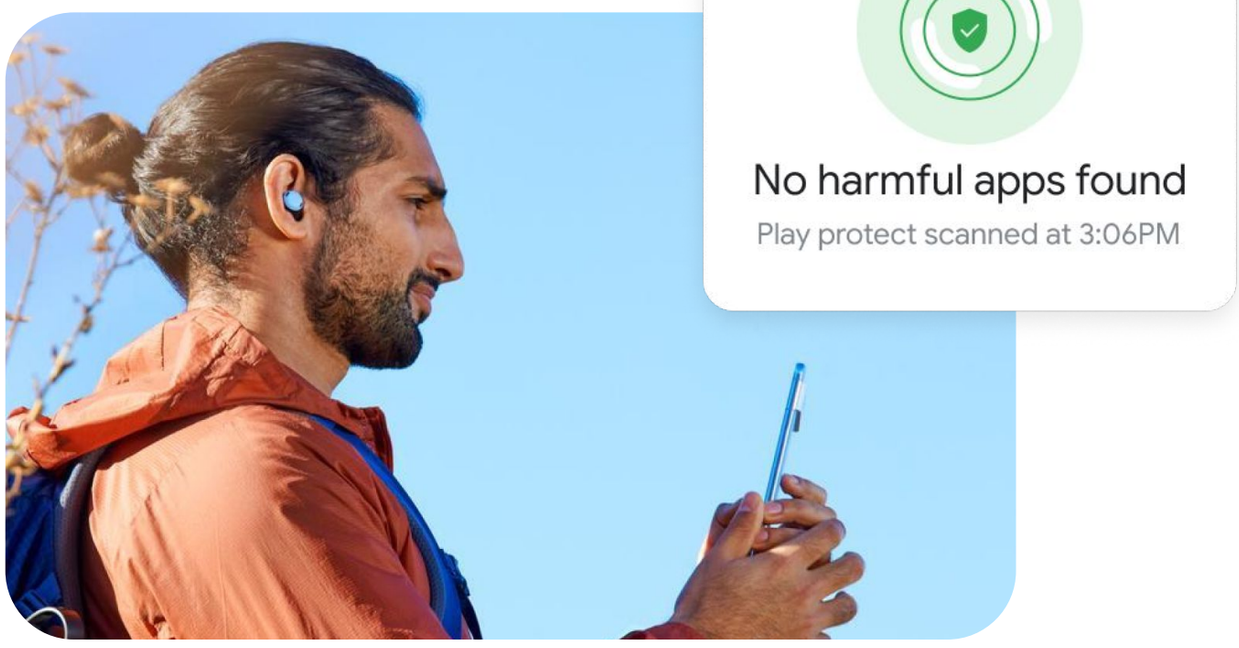
Wi-Fi MAC Address

APIs accessing the Wi-Fi MAC address have been removed, except on fully managed devices.

For enterprise devices, device policy controllers (DPCs) can deny permissions on behalf of the user, ensuring better control over privacy settings. These controls prioritize user privacy and give enterprises more control over data access on their managed devices.

Google Play Protect: Powerful protection for your device

Google Play Protect provides comprehensive security for Android devices, whether apps come from the Play Store or other sources. It actively monitors for threats, removes harmful apps, and helps you make informed decisions about app installations.



Built-in security

Google Play Protect is automatically included on devices with Google Play, actively safeguarding more than 3 billion devices from threats like malware.

Daily scans

It scans your device daily for harmful activity and security risks, notifying you if an app contains malware.

Automatic removal

It can automatically remove or disable malicious apps to prevent harm and improve future threat detection.

Unknown app analysis

You can choose to send unknown apps to Google for further analysis.

Protection beyond Google Play

Google Play Protect checks apps from outside the Play Store before installation, stopping known malware.

Real-time scanning

Real-time threat detection scans are suggested for apps that have not been scanned previously from any installation source.

Enhanced security for developers

The Play Integrity API empowers developers to verify that user actions and server requests are originating from their genuine app, installed from Google Play, and running on a trusted Google Play Certified Android device. This verification process now includes a strong guarantee of system integrity using Android Key Attestation, ensuring a higher level of confidence in the app's environment.

Furthermore, the API introduces an app access risk verdict, enabling developers to check if other apps running on the device could potentially capture the screen, display overlays, or control the device. This valuable information helps protect against social engineering attacks or scams, especially during sensitive actions like money transfers. Developers can take appropriate action, such as prompting the user to close risky apps or restricting access to sensitive features until the risk is mitigated.

Additionally, developers can now verify the status of Google Play Protect and whether it has detected any risky apps installed on the device. This is particularly beneficial for safeguarding sensitive actions or enforcing strict device policies in enterprise environments.

Lastly, the API provides an attestation counter indicating the device's recent activity level, without disclosing specific numbers to protect user privacy. This allows developers to limit access to protected functionality if the device has exhibited a high volume of requests in the past hour, which could be a sign of suspicious activity.

> **The Play Integrity API equips developers with a robust set of tools to bolster app security, defend against emerging threats, and provide a secure user experience**

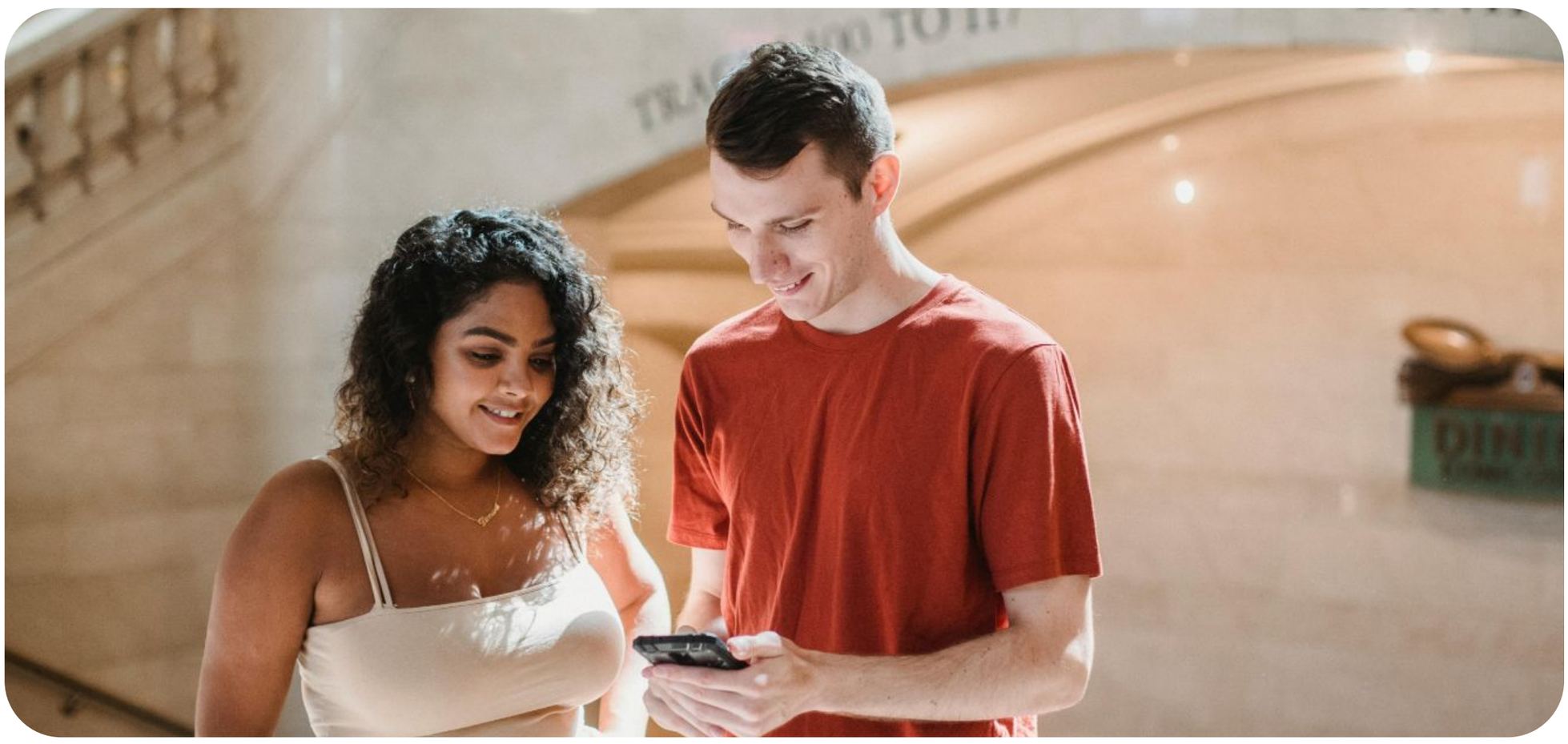
Google Play app review

The Google Play Store takes user protection seriously with policies that prevent malicious actors from distributing harmful apps. This is important because it helps maintain the trust and safety of the Android ecosystem, ensuring users can confidently download and use apps without fear of encountering malware or other security risks.

Developers are thoroughly vetted through a two-stage process.

01 First, their real-world identity is verified when creating a developer account.

02 Then, further checks are conducted upon app submission, including automated analysis to detect potential harmful behaviors.



If anything suspicious is found, a security analyst manually reviews the app. Developers whose apps violate policies face account suspension. This multi-layered approach helps weed out bad actors and their malicious apps before they can reach users.

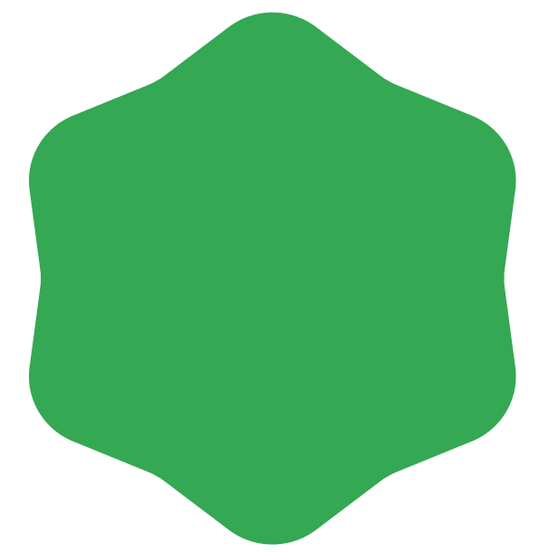
If an app is flagged, developers receive immediate notification, guidance on improvements, and a timeline for addressing the issue. This prioritizes swift action to enhance app security, which is crucial for protecting users from potential threats.

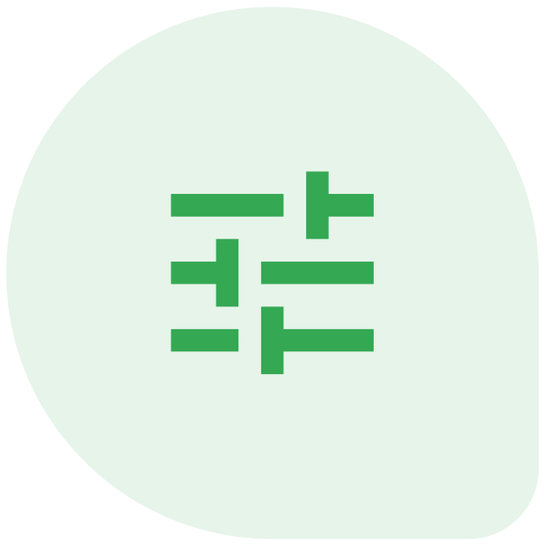
In certain cases, updates may be blocked until necessary security improvements are made. This ensures that users are not exposed to new vulnerabilities while developers work on fixing their apps. Encouraging the use of updated APIs is another crucial aspect of risk mitigation. This ensures apps support the latest features, optimizing security and performance.

This is important because newer Android versions often include security enhancements and performance optimizations that help protect users and improve their overall experience. Both new and updated public apps must target at least Android 13 to meet API requirements, further promoting the adoption of the latest security standards.

Each new Android version brings security, performance, and user experience enhancements. Some changes are exclusive to apps that explicitly declare support through their targetSdkVersion. Developers can refer to Google Play Developers documentation for detailed guidance on updating to the appropriate target API level. This is important because it incentivizes developers to keep their apps up-to-date with the latest Android features and security improvements, ultimately benefiting users with a safer and more enjoyable app experience.

Overall, the Google Play Store's efforts to protect users from malicious apps through developer vetting, app reviews, and API requirements are vital for maintaining a secure and trustworthy Android ecosystem. This benefits both users, who can confidently download and use apps, and developers, who can build and distribute their apps in a safe and supportive environment.





Jetpack security

Android developers can use the Android KeyStore with Jetpack Security for enhanced data protection. This allows them to easily create strong AES 256 GCM encryption keys or use advanced options for specific needs. Jetpack Security also simplifies encrypting files and shared preferences. It's recommended for all types of apps, especially those managing device policies or sensitive data.

Android KeyStore and Jetpack Security

Work together for robust data encryption.

MasterKeys

Makes creating strong encryption keys simple.

Advanced options

Advanced options: Developers can customize key authorization settings.

File & shared preferences encryption

Easy-to-use encryption for different data types.

Recommended for all apps:

Especially for those handling device policies or sensitive information.

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.

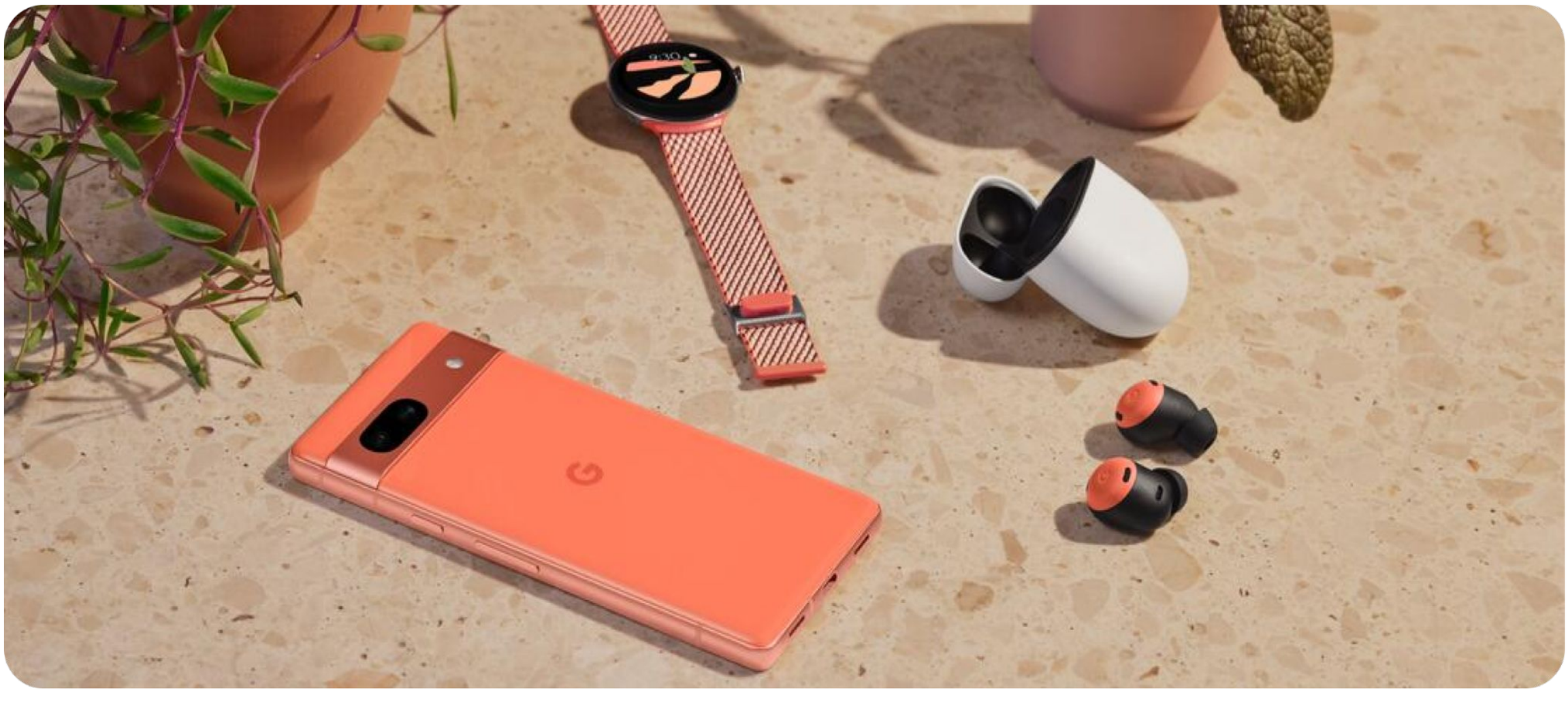
For fully managed devices, IT admins can install system updates manually using a system update file on Android 10 and above devices. This manual control offers several advantages to IT admins. They can test an update on a small group of devices before deploying it widely, thereby minimizing potential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

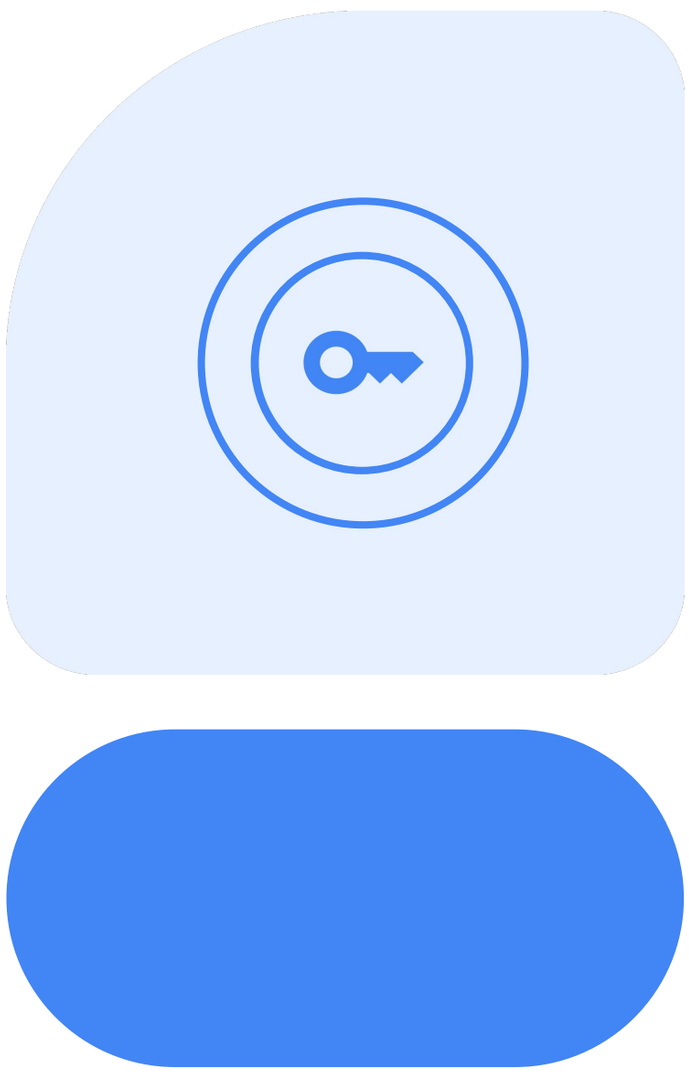
These updates come in the form of APK or APEX files. APEX files load earlier during device startup, which can be important for security and performance improvements. This means you can get critical updates without waiting for a full operating system upgrade. And, of course, all updates are cryptographically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

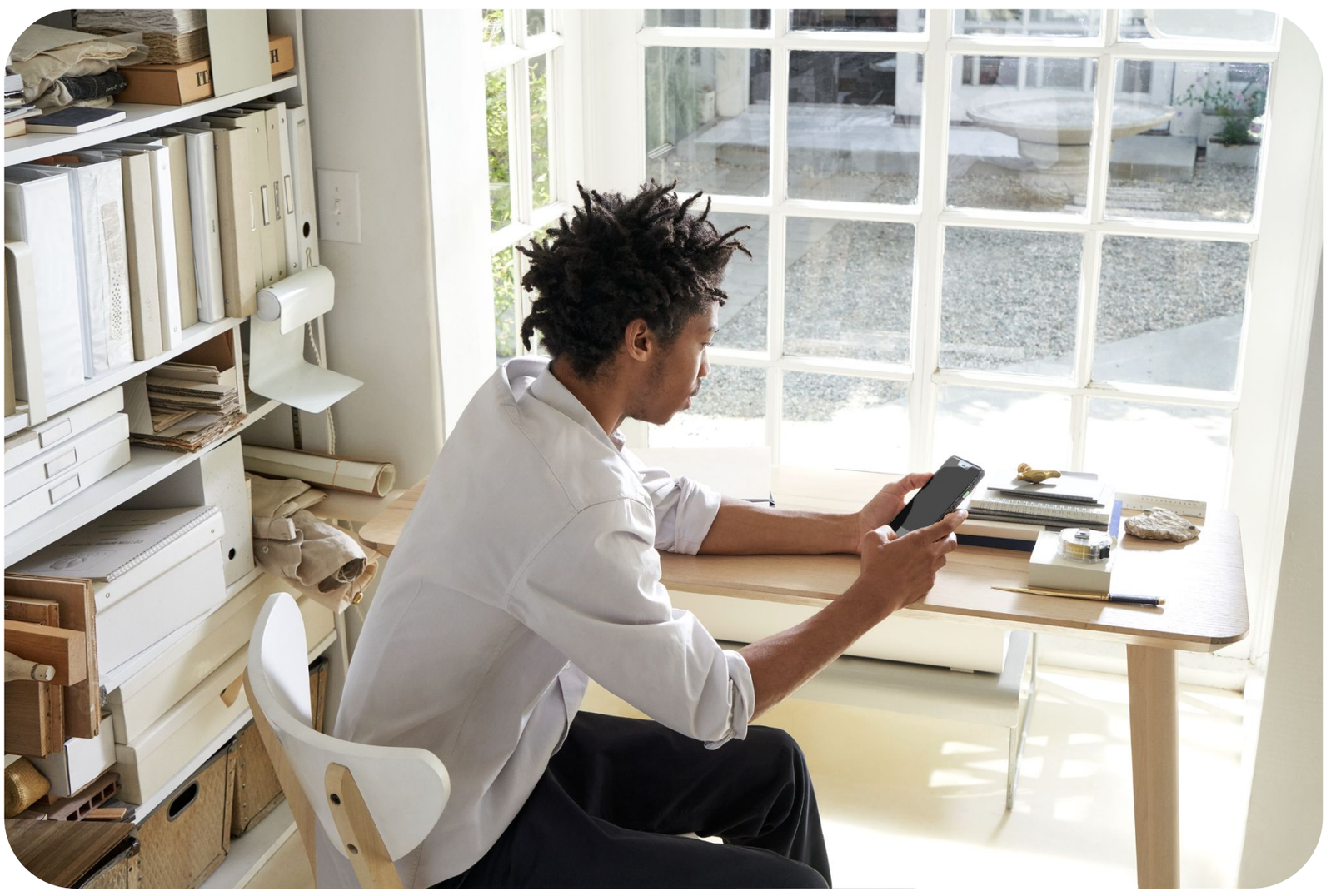
Conscrypt

> **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 ofidentifying 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.

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.

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 UI, 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.

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.

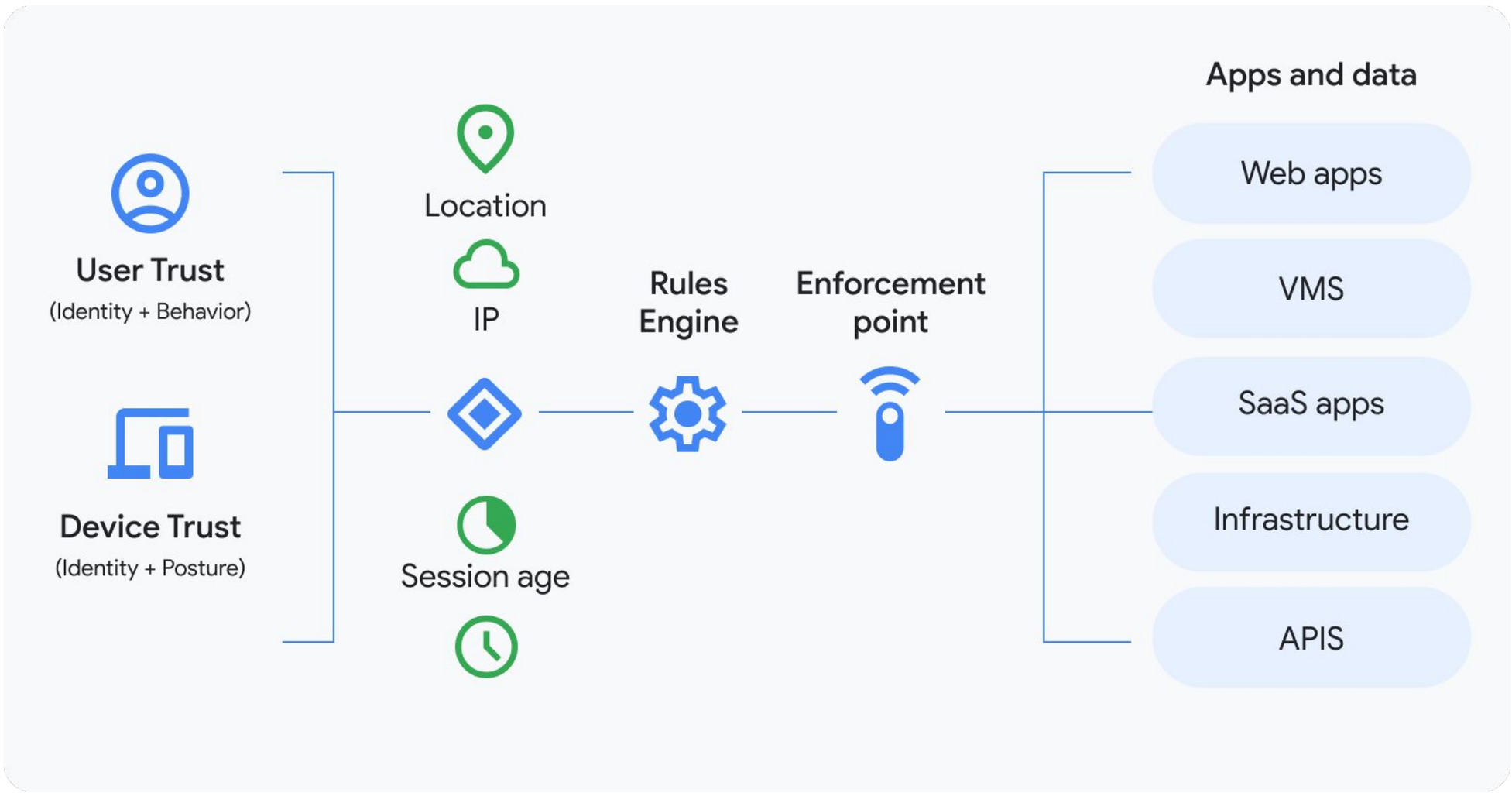
Applications from unknown sources

IT admins may need to prevent the installation of applications from outside Google Play or trusted installers. According to the Android Transparency Report, 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.

Identity and Zero Trust from Android Enterprise

Zero Trust from Android Enterprise is a security model that enables a mobile and remote workforce to securely connect to company resources from any location. Devices are vetted before being granted access to company resources. Companies can use tight, granular controls to specify the level of access whether the devices are connected to a corporate network, from home, or elsewhere. An effective Zero Trust implementation requires numerous device signals, context and controls to make intelligent decisions about access.



Establishing device trust is critical for Zero Trust implementations and this is an area where Android stands out against other mobility platforms. Android has a wealth of platform features and APIs that our enterprise mobility management and security partners leverage to safeguard backend services and resources. Android provides a variety of device signals that IT admins can use in building systems to verify the security and integrity of devices. In a Zero Trust model, these signals are used to assess whether a device should be allowed to access corporate information. There are currently more than 100 unique device trust signals available across 30 APIs on Android devices.

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.

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.

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.

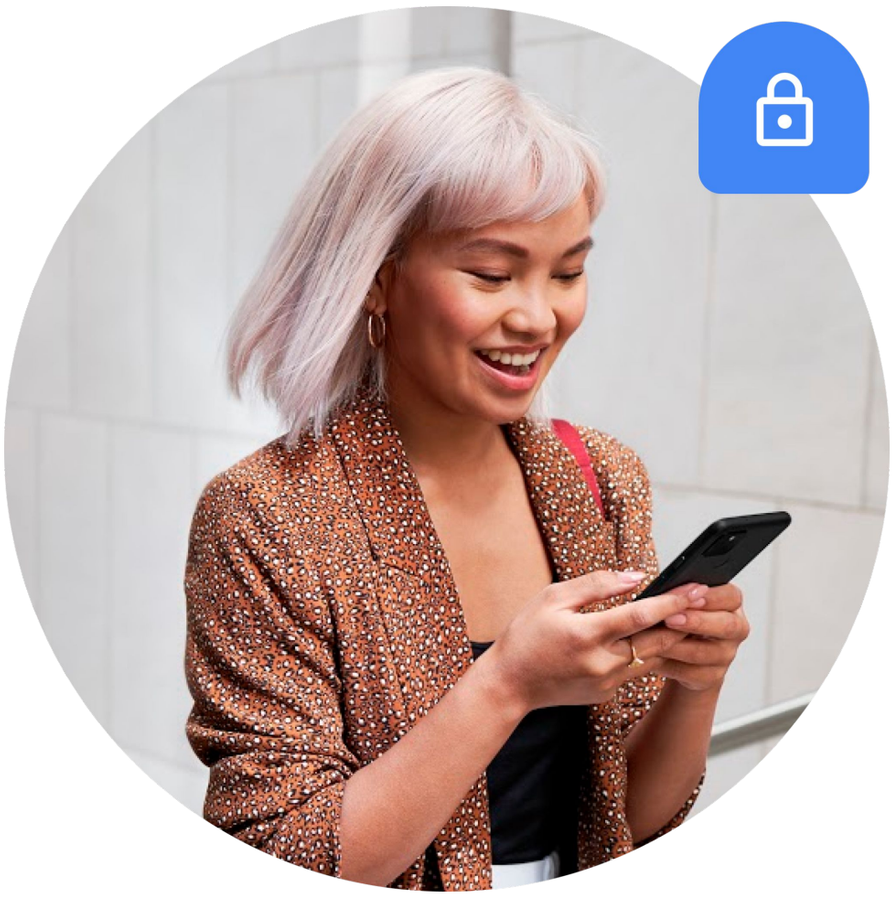
Independent credentialed auditors perform thorough audits to ensure compatibility with the established principles.

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.



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.

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.

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.

Contributors

Thank you to the following people, and so many others, for contributions to this paper and the work it represents.

Al Chappelle
Amy van den Berghe
Andrew Scull
Armelle Laine
Brian Wood
Brooke Davis
Colin Hacker
David Still
Dom Elliot
Eileen Lucey
Eric Biggers
Eric Lynch
Eugene Liderman
Güliz Seray Tuncay
Irene Ang
James Nugent
Jeffrey Vander Stoep
Jui Tamhane
Luke Haviland
Melanie Aley
Michael Groover
Mike Burr
Nithan Sannappa
Oli Gaymond
Paul Crowley
Pete Bentley
Raz Lev
Rishika Hooda
Ryan Kim
Sandy Leung
Shailesh Saini
Shannon Morales
Steve Kafka
Vivek Bhavsar
Zhou Zhou

Android

Learn more about Android security