GOOGLE ANDROID 12 SECURITY TECHNICAL IMPLEMENTATION GUIDE (STIG) SUPPLEMENTAL PROCEDURES

Version 1, Release 1

14 September 2021

Developed by Google and DISA for the DoD

Trademark Information

Names, products, and services referenced within this document may be the trade names, trademarks, or service marks of their respective owners. References to commercial vendors and their products or services are provided strictly as a convenience to our users, and do not constitute or imply endorsement by DISA of any non-Federal entity, event, product, service, or enterprise.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

1. ANDROID ENTERPRISE

An Android Enterprise solution is a combination of three components: The Enterprise Mobility Management/Mobile Device Management (EMM/MDM) console, a device policy controller (DPC), which is the EMM/MDM agent, and managed Google Play.

Figure 1-1: Components of an Android Enterprise Solution
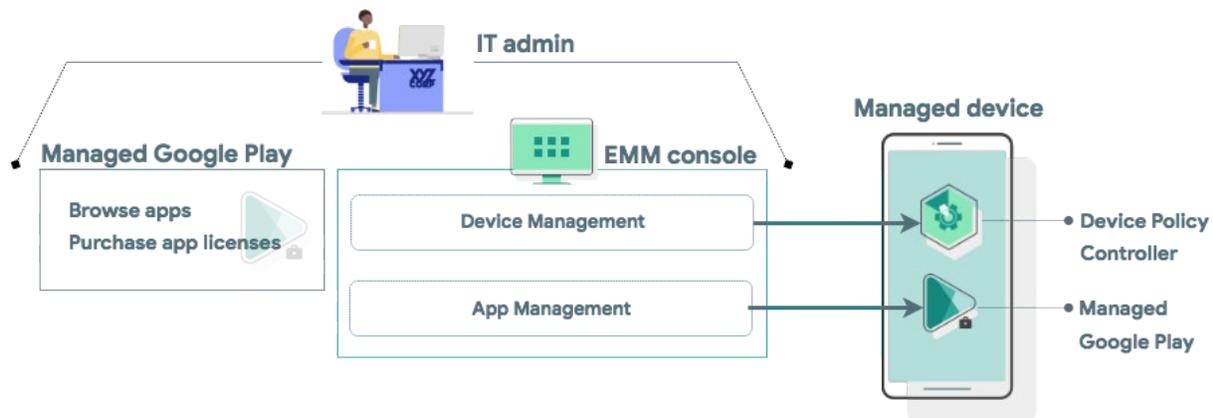
1.1 EMM/MDM Console

EMM solutions typically take the form of an EMM console—a web application that allows IT administrators to manage their organization, devices, and apps. To support these functions for Android, the console must be integrated with the Application Programming Interfaces (APIs) and User Interface (UI) components provided by Android Enterprise.

1.2 DPC (Device Policy Controller)

All Android devices managed by an organization through an EMM console must install a DPC (Device Policy Controller) app during setup. A DPC is an agent that applies the management policies set in the EMM console to devices. Depending on which development option is chosen, the EMM solution can be coupled with Android's DPC or with a custom user-developed DPC.

End users can provision a fully managed or dedicated device using a DPC identifier (e.g., "afw#") or by scanning a QR code created by the EMM according to the implementation guidelines defined in the Play EMM API developer documentation.

- The EMM's DPC must be publicly available on Google Play, and the end user must be able to install the DPC from the device setup wizard by entering a DPC-specific identifier or by scanning a QR code generated by the EMM.
- Once installed, the EMM's DPC must guide the user through the process of provisioning a fully managed or dedicated device.

1.3   Managed Google Play

Managed Google Play is an enterprise app platform based on Google Play that is free to Android Enterprise customers and available to integrate into an EMM solution. It combines the familiar user experience and app store features of Google Play with a set of management capabilities designed specifically for enterprises.

IT administrators can use managed Google Play to discover apps, view app details, purchase app licenses, and configure granular app permissions. Typically, an IT administrator curates, manages, and distributes apps through an EMM console.

Using Android Enterprise APIs, an EMM console can distribute apps to managed devices. Apps can be remotely installed on a device or added to the device's managed Google Play store.

On managed devices, managed Google Play is the user's enterprise app store. The interface is similar to Google Play: users can browse apps, view app details, and install them. Unlike the public version of Google Play, users can only install apps from managed Google Play that are allowlisted for them.

Figure 1-2: Managed Google Play
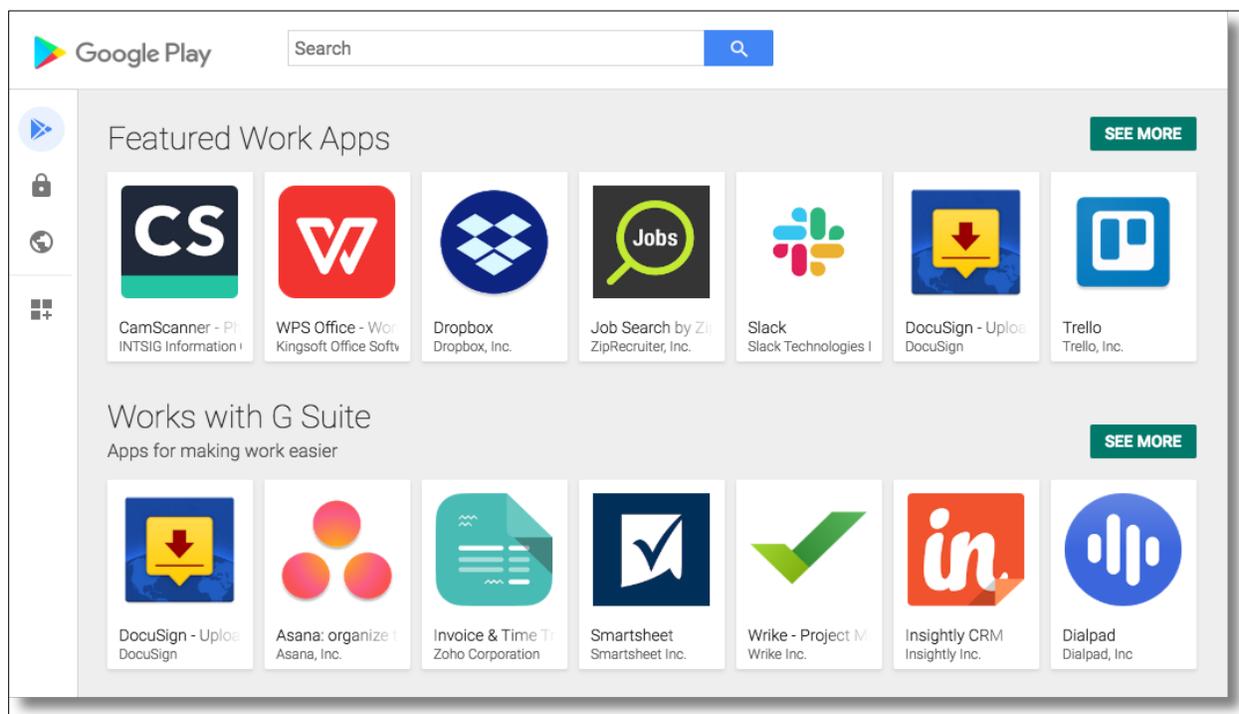
2. ANDROID SECURITY OVERVIEW

2.1 Android Operating System

Android is an open source operating system (OS) built on the Linux kernel[1] that provides an environment for multiple apps to run simultaneously. These apps are signed and isolated into application sandboxes associated with their application signature. The application sandbox defines the privileges available to the application. Apps are generally built using Android Runtime and interact with the OS through a framework that describes system services, platform APIs, and message formats. Other high-level and lower-level languages, such as C/C++, are allowed and operate within the same application sandbox.

2.2 Trusted Execution Environment

Android devices that support a lock screen and ship with Android 7.0 Nougat and above have a secondary, isolated environment called a Trusted Execution Environment (TEE). This enables further separation from any untrusted code. The capability is typically implemented using secure hardware such as ARM TrustZone technology.

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

- **Lock screen passcode verification**: Available on devices that support a secure lock screen and ship with Android 7.0 or newer; lock screen verification is provided by TEE unless an even more secure environment, such as tamper-resistant hardware, is available.
- **Fingerprint template matching**: Available on devices that have a fingerprint sensor and ship with Android Marshmallow 6.0 or newer.
- **Protection and management of KeyStore keys**: Available on devices that support a secure lock screen that ship with Android 7.0 or newer.

2.3 Tamper-Resistant Hardware

Google Android 11 and higher devices are required to ship with tamper-resistant hardware to perform security-critical operations. This hardware is built with additional protections against physical tampering and shares only very limited resources with the main application processor, significantly reducing its attack surface and the potential of side channel attacks. Devices shipped with Android 8.0 and above can use tamper-resistant hardware to verify a device's lock screen passcode.

---

[1] SELinux (Security Enhanced Linux) kernel

Figure 2-1: Tamper-Resistant Hardware Provides Numerous Protections on the Device
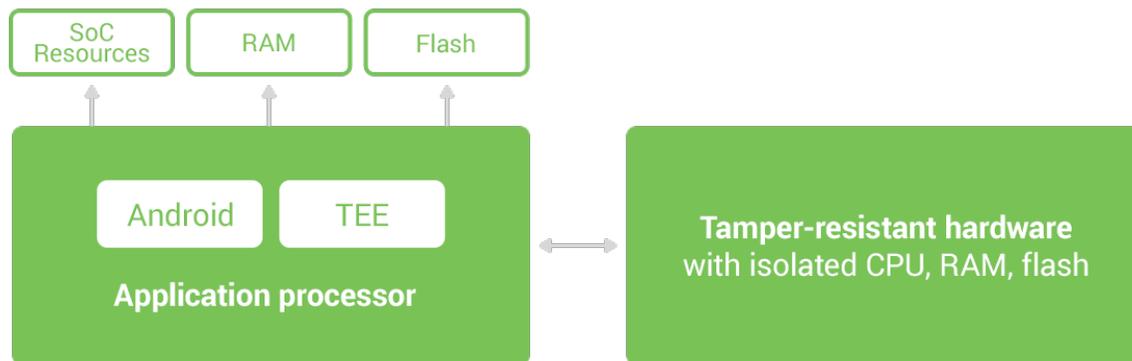
2.4   Device Integrity

Device integrity features protect the mobile device from running a tampered or compromised OS. With companies using mobile devices for essential communication and core productivity tasks, keeping the OS secure is essential. Without device integrity, few security properties can be assured. Android adopts several measures to guarantee device integrity at all times.

Google's SafetyNet Attestation service is an anti-abuse API that allows app developers and EMM partners to assess the state of the Android devices their app is running on. The API, which is integrated into EMM solutions, should be used as part of an abuse detection system to help determine an Android device's integrity.

The SafetyNet Attestation API provides a cryptographically signed attestation, assessing the device's integrity. To create the attestation, the API examines the device's software and hardware environment, looking for integrity issues and comparing with the reference data for approved Android devices. The generated attestation is bound to the nonce that the caller app (the EMM DPC) provides. The attestation also contains a generation timestamp and metadata about the requesting app.

2.4.1   Verified Boot

Verified Boot is Android's secure boot process that verifies system software before running it. This makes it more difficult for software attacks to persist across reboots and provides users with a safe state at boot time. Each Verified Boot stage is cryptographically signed, and each phase of the boot process verifies the integrity of the subsequent phase prior to executing that code. Full boot of a compatible device with a locked bootloader proceeds only if the OS satisfies integrity checks. Verification algorithms used must be as strong as current recommendations from NIST for hashing algorithms (SHA-256) and public key sizes (RSA-2048).

Figure 2-2: Verified Boot
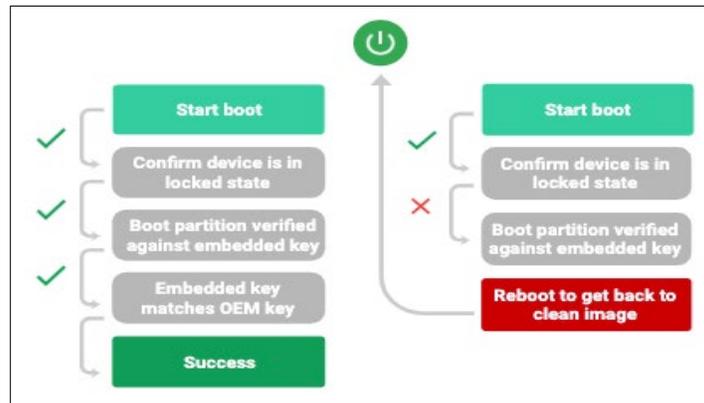
The Verified Boot state is used as an input in the process to derive disk encryption keys. If the Verified Boot state changes (e.g., the user unlocks the bootloader), then the secure hardware prevents access to data used to derive the disk encryption keys used when the bootloader was locked. Verified Boot on compatible devices running Android 9.0 and above requires rollback protection. This means that a kernel compromise (or physical attack) cannot put an older, more vulnerable, version of the OS on the system and boot it. Rollback protection state is also stored in tamper-evident storage.

Enterprises can check the state of Verified Boot using KeyStore key attestation. This retrieves a statement signed by the secure hardware attesting to many attributes of Verified Boot along with other information about the state of the device.

2.4.2   Version Binding

To further harden and safeguard the boot process, Android devices are required to use Version Binding. This ensures that keys are bound to the operating system and the patch level of the system image installed on an Android device. An attacker who discovers a weakness in an old version of the system or TEE software cannot roll a device back to the vulnerable version and use keys created with the newer version.

In addition, when a key with a given version and patch level is used on a device that has been upgraded to a newer version or patch level, the key is upgraded before it can be used and the previous version of the key invalidated. In this way, as the device is upgraded, the keys will "ratchet" forward along with the device, but any reversion of the device to a previous release will cause the keys to be unusable.

2.5   Sandboxing

Android runs all apps inside sandboxes to prevent malicious or buggy app code from compromising other apps or the rest of the system. Because the application sandbox is enforced in the kernel, this enforcement extends to the entire app regardless of the specific development environment, APIs used, or programming language. A memory corruption error in an app allows

only arbitrary code execution in the context of that particular app, with the permissions enforced by the OS.

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

Android employs several sandboxing techniques, including Security-Enhanced Linux (SELinux), seccomp, and file-system permissions.

2.5.1   SELinux

Android uses SELinux to enforce mandatory access control (MAC) over all processes and apps, including processes running with root and superuser privileges. SELinux provides a centralized auditable security policy that can be used to strongly separate processes from one another. Android devices implement SELinux policy on a per-domain basis in enforcing mode—no permissive mode domains are allowed. Illegitimate actions that violate policy are blocked and all violations (denials) are logged by the kernel. They are then readable using the dmesg and logcat command-line tools.

As of Android 8.0, with Project Treble, SELinux is used to enforce a separation between the framework and the device-specific vendor components. They run in different processes and communicate with each other via a set of standard vendor interfaces implemented as Hardware Abstraction Layers (HALs). Device original equipment manufacturers (OEMs) can create a HAL implementation that runs in its own sandbox and is only permitted to access the hardware driver it controls, with permissions granted to the process limited to only those required to do its job. On the framework side, the client runs in a sandbox that does not allow it access to hardware drivers and other permissions and capabilities needed by the HAL implementations.

2.5.2   Filesystem Sandboxing

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

2.6   Enhanced Exploit Protection

Android 10 and above continues the effort to offer exploit protection such as Kernel/Control Flow Integrity and Integer Overflow Sanitization. New compiler-based mitigations have been added to make bugs harder to exploit and prevent certain types of bugs from becoming vulnerabilities. Android 10 and above expands existing compiler mitigations, which directs the runtime operations to safely abort when undefined behavior occurs. Android 11 and higher implements error detection with GWP-Asan, which is a native memory allocator.

Some vulnerabilities are exploited by attackers changing the normal control flow of an application to perform arbitrary malicious activities with all the privileges of the exploited application. Android now requires Control flow integrity (CFI), a security mechanism that disallows changes to the original control flow graph of a compiled binary, making it significantly harder to perform such attacks.

2.7   Data Protection

Android uses industry-leading security features to protect user data. The platform creates an application environment that protects the confidentiality, integrity, and availability of user data.

2.7.1   File-Based Encryption

Encryption is the process of encoding user data on an Android device using an encryption key to prevent an unauthorized party from accessing data. Android 10 and above smartphones and tablets use file-based encryption (FBE), which allows different files to be encrypted with different keys that can be unlocked independently.

Direct Boot enables encrypted devices to boot straight to the lock screen, alarms to operate, accessibility services to be available, and phones to receive calls before a user has provided their credentials.

With file-based encryption and APIs to make apps aware of encryption, it is possible for apps to operate within a limited context before users have provided their credentials while still protecting private user information.

On an FEB-enabled device, each device user has two storage locations available to apps: Credential Encrypted (CE) storage and Device Encrypted (DE) storage.

CE, the default storage location, is only available after the user has unlocked the device. CE keys are derived from a combination of user credentials and a hardware secret. It is available after the user has successfully unlocked the device the first time after boot and remains available for active users until the device shuts down, regardless of whether the screen is subsequently locked.

DE is a storage location available during Direct Boot mode and after the user has unlocked the device. DE keys are derived from a hardware secret available only after the device has performed a successful Verified Boot.

By default, apps do not run during Direct Boot mode. If an app needs to take action during Direct Boot mode, such as an accessibility service (e.g., Talkback) or an alarm clock app, the app can register components to run during this mode.

DE and CE keys are unique and distinct—no user's CE or DE key will match another. File-based encryption allows files to be encrypted with different keys, which can be unlocked independently. All encryption is based on AES-256 in XTS mode. Due to the way XTS is defined, it needs two 256-bit keys. In effect, both CE and DE keys are 512-bit keys.

By taking advantage of CE, file-based encryption ensures a user cannot decrypt another user's data. This is an improvement on full-disk encryption where there is only one encryption key, so all users must know the primary user's passcode to decrypt data. Once decrypted, all data is decrypted.

2.7.2   Metadata Encryption

In addition to file-based encryption, Android 9 and above smartphones can use metadata encryption. With metadata encryption, a single key present at boot time encrypts file system metadata that is not otherwise encrypted by file-based encryption (FBE). This key is protected by Keymaster, which in turn is protected by Verified Boot.

More information about FBE and its related set of features can be found here.

2.7.3   Lock Screen

Both fingerprint template matching and passcode verification can only take place on secure hardware with rate limiting (exponentially increasing timeouts) enforced. Android's GateKeeper throttling is also used to prevent brute-force attacks. After a user enters an incorrect password, GateKeeper APIs return a value in milliseconds in which the caller must wait before attempting to validate another password. GateKeeper will ignore any attempts before the defined amount of time has passed. Gatekeeper also keeps a count of the number of failed validation attempts since the last successful attempt. These two values together are used to prevent brute-force attacks of the Target of Evaluations' (TOE's) password.

For biometric fingerprint authentication, the user can attempt five failed fingerprint unlocks before fingerprint is locked for 30 seconds. After the 20th cumulative attempt, the device locks the fingerprint until the password is entered.

Android offers APIs that allow apps to use fingerprints for authentication and users to authenticate by using their fingerprint scans on supported devices. These APIs are used in conjunction with the Android KeyStore system. Android 11 and higher uses the Biometric Manager Authenticator interface, which defines the types of authentication that applications can require: Biometric Strong, Biometric Weak, and Device Credential.

2.7.4   Additional Authentication Methods

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

However, note that Smart Lock does not meet the same level of assurance as other unlock methods on Android and is not allowed to unlock auth-bound KeyStore keys. Organizations can disable this using the KEYGUARD_DISABLE_TRUST_AGENTS flag. In this STIG, Smart Lock is not currently approved for use.

Android 11 and higher include additional support for auth-per-use keys that require the user to present a biometric credential or device credential each time the app needs to access data protected by the key.

2.8 Hardware-Backed KeyStore and KeyChain

2.8.1 KeyStore

The Android KeyStore class allows a user to manage private keys in secure hardware to make them more difficult to extract from the device. It was introduced in Android 4.3 and focuses on apps storing credentials used for authentication, encryption, or signing.

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

Additionally, version binding binds keys to an operating system and patch level version. This ensures that an attacker who discovers a weakness in an old version of system or TEE software cannot roll a device back to the vulnerable version and use keys created with the newer version. The KeyStore is implemented in secure hardware. This guarantees that even in the event of a kernel compromise, KeyStore keys are not extractable from the secure hardware.

2.8.2 KeyStore Key Attestation

Android 7 and above smartphones support Key Attestation, which empowers a server to gain assurance about the properties of keys. Devices that support Google Play are provisioned at the factory with an attestation key generated by Google. The secure hardware on such devices can sign statements with the provisioned key, which attests to properties of keys protected by the secure hardware (for example, the key was generated and cannot leave the secure hardware). Attestation fields include purpose, padding, activate DateTime, and authTimeout. Key attestation better enables the location of important properties about the device, such as the OS version, patch level, and whether it passed Verified Boot.

More information about verifying hardware-backed keys with Key Attestation is located here.

2.8.3 KeyChain

The KeyChain class allows apps to use the system credential storage for private keys and certificate chains. KeyChain is often used by Chrome, Virtual Private Network (VPN) apps, and many enterprise apps to access keys imported by the user or by the mobile device management app.

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

2.9   Work Profile Security

Work profile mode is initiated when the DPC initiates a managed provisioning flow. The work profile is based on the Android multi-user concept, where the work profile functions as a separate Android user segregated from the primary profile. The work profile shares common UI real estate with the primary profile. Apps, notifications, and widgets from the work profile show up next to their counterparts from the primary profile and are always badged to indicate the type of app.

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

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

A DPC running in profile owner mode can require users to specify a security challenge for apps running in the work profile. The system shows the security challenge when the user attempts to open any work apps. If the user successfully completes the security challenge, the system unlocks the work profile and decrypts it if necessary.

2.9.1   Separate Work Challenge

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

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

The work challenge is verified within secure hardware to protect against brute-force attacks. The passcode, mixed in with a secret from the secure hardware, is used to derive the disk encryption

key for the work profile to ensure an attacker cannot derive the encryption key without either knowing the passcode or breaking the secure hardware.

The STIG does not require the use of separate work challenge.

2.9.2   COPE Deployments and User Privacy

Google Android 12 devices deployed using COPE will have enhanced privacy for users. Personal apps in the personal profile cannot be configured, monitored, or enumerated by an MDM. Allowlists and blocklists should be used to permit or block specific applications from use in the personal profile. It is highly recommended that DoD mobile service providers deploy/redeploy all Android phones using Zero-Touch with personal app allowlists/blocklists to provide visibility on which personal apps are installed on managed devices.

2.10   Network Security

In addition to data-at-rest security (protecting information stored on the device), Android provides network security for data in transit to protect data sent to and from Android devices. Android provides secure communications over the internet for web browsing, email, instant messaging, and other internet apps by supporting Transport Layer Security (TLS), including TLS v1.0, TLS v1.1, TLS v1.2, and TLS v1.3. Android 10 and 11 default to TLS v1.3.

2.11   DNS over TLS

Android includes built-in support for DNS over TLS in Android 10 and higher. Devices will automatically upgrade to DNS over TLS if a network's DNS server supports it. Android then sends all DNS queries over a secure channel to this server or marks the network as "No internet access" if it cannot reach the server. DNS over TLS mode automatically secures the DNS queries from all apps on the system.
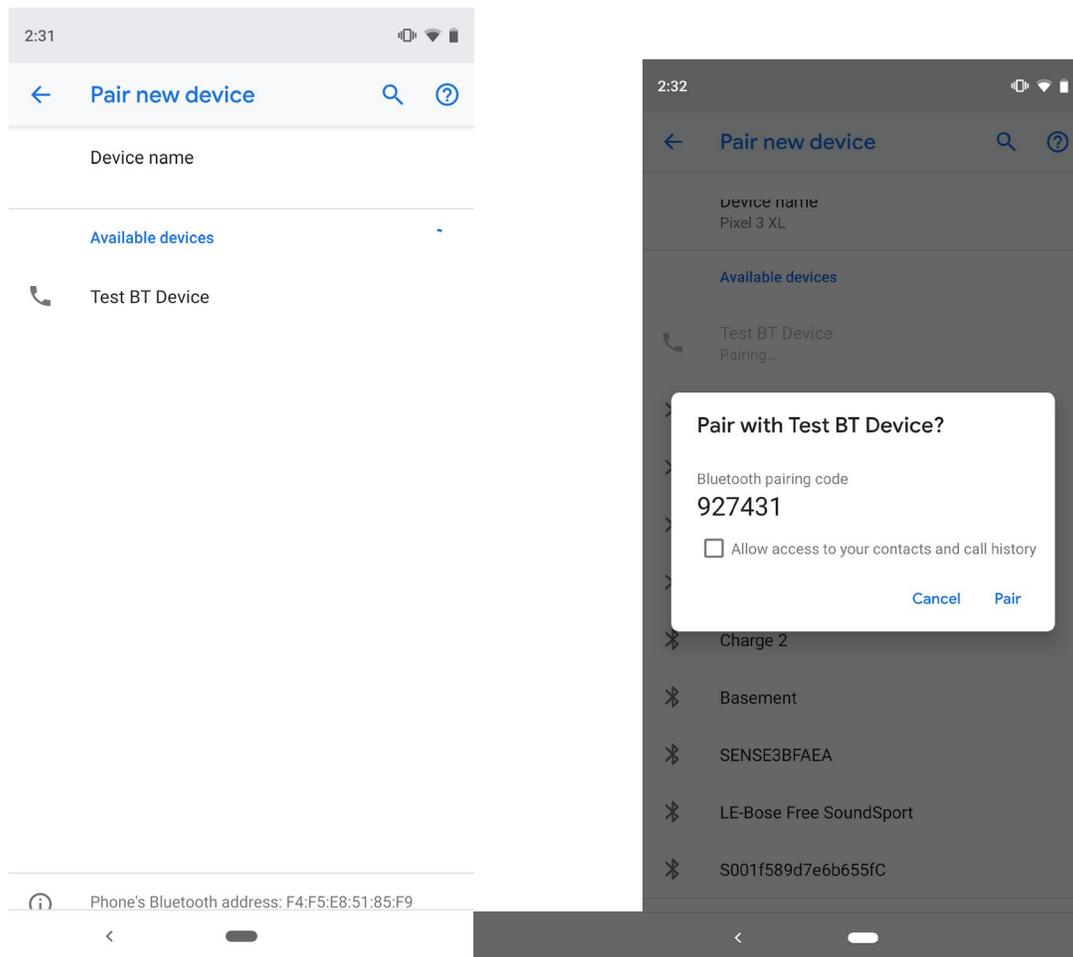
2.11.1 Bluetooth

Follow the steps below to pair and connect using Bluetooth:

Pair:

1. Open the phone or tablet's Settings app .
2. Tap Connected devices >> Connection preferences >> Bluetooth.
3. Ensure Bluetooth is turned on.
4. Tap Pair new device.
5. Tap the name of the Bluetooth device to be paired with the phone or tablet.
6. Follow any on-screen steps.

Figure 2-3: Bluetooth Pairing

Connect:

1. Open the phone or tablet's Settings app ⚙.
2. Tap Connected devices >> Connection preferences >> Bluetooth.
3. Ensure Bluetooth is turned on.
4. In the list of paired devices, tap a paired but unconnected device.

Note: When the phone/tablet and the Bluetooth device are connected, the device will show as "Connected" in the list.

Note: If the phone is connected to another device through Bluetooth, at the top of the screen, a Bluetooth icon ✳ will be displayed.

Remove Previously Paired Device:

1. Open the phone or tablet's Settings app ⚙.
2. Tap Connected devices >> Previously connected devices.
3. Tap the gear icon to the right of the device to be unpaired.

4. Tap "Forget" and then confirm in the pop-up window by tapping "Forget device".

Figure 2-4: Remove Previously Paired Device
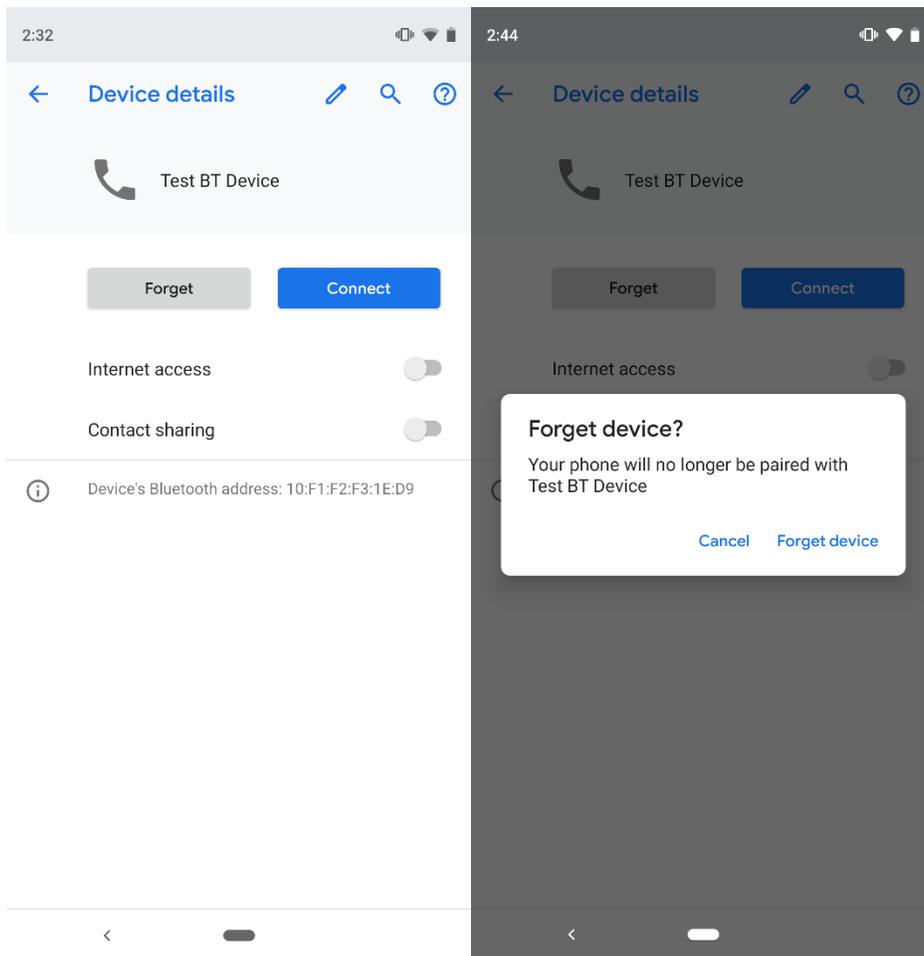
For additional support information around Bluetooth, refer to this support link.

2.11.2 Wi-Fi

Android supports the WPA2-Enterprise (802.11i) protocol, which is specifically designed for enterprise networks and can be integrated into a broad range of Remote Authentication Dial-In User Service (RADIUS) authentication servers. Android 10 and above support WPA3.

IT admins can silently provision enterprise Wi-Fi configurations on managed devices, including:

- SSID, via the EMM's DPC
- Password, via the EMM's DPC
- Identity, via the EMM's DPC
- Certificate for clients' authorization, via the EMM's DPC
- CA certificate(s), via the EMM's DPC

UNCLASSIFIED

IT administrators can lock down Wi-Fi configurations on managed devices to prevent users from creating new configurations or modifying corporate configurations.

IT administrators can lock down corporate Wi-Fi configurations in either of the following configurations:

- Users cannot modify any Wi-Fi configurations provisioned by the EMM but may add and modify their own user-configurable networks (for example, personal networks).
- Users cannot add or modify any Wi-Fi network on the device, limiting Wi-Fi connectivity to networks provisioned by the EMM.

When the device tries to connect to a Wi-Fi network, it performs a standard captive portal check that bypasses the full tunnel VPN configuration. The administrator must turn off the captive portal check physically on the device before enrolling it into the MDM by completing the steps below:

1. To enable Developer Options, tap Settings >> About and then tap "Build number" five times until the Developer options have been enabled.
2. To enable Android Debug Bridge (ADB) over USB, tap Settings >> System >> Advanced >> Developer options and then scroll down to USB debugging and enable the toggle to "On".
3. To connect the device to a workstation with ADB installed, type "adb shell settings put global captive_portal_mode 0" and then tap "enter".
4. To verify the change, type "adb shell settings get global captive_portal_mode". Ensure the return value is "0".
5. To turn off Developer options, tap Settings >> System >> Advanced >> Developer options, and then toggle the "On" option to "Off" at the top.

If a Wi-Fi connection unintentionally terminates, the end user must reconnect to reestablish the session.

2.11.3 VPN

Android supports securely connecting to an enterprise network using VPN:

- **Always-On VPN** – The VPN can be configured so that apps cannot access the network until a VPN connection is established to prevent apps from sending data across other networks.
  - ○ Always-On VPN supports VPN clients that implement VpnService. The system automatically starts that VPN after the device boots. Device owners and profile owners can direct work apps to always connect through a specified VPN.
  - ○ Additionally, users can manually set Always-On VPN clients that implement VpnService methods using Settings >> More >> VPN. Always-On VPN can also be enabled manually from the settings menu.
- **Per User VPN** – On multi-user devices, VPNs are applied per Android user; all network traffic is routed through a VPN without affecting other users on the device. VPNs are applied per work profile, which allows an IT administrator to specify that only their

enterprise network traffic goes through the enterprise-work profile VPN and not the
user's personal network traffic.

- **Per Application VPN** – Enables VPN connections on allowed apps and prevents VPN
  connections on disallowed apps.

2.12 Over-the-Air Updates

Over the Air (OTA) updates, which include baseband processor updates, use public key chaining
to the Root Public Key, a hardware protected key whose SHA-256 hash resides inside the
application processor. If this verification fails, the software update will fail and the update will
not be installed. Devices with Android 8 and above also provide roll-back protection for OTA
updates to prevent a user from installing a prior/previous version of software by check.

Android devices with Android 9 and above leverage A/B system updates, also known as
seamless updates. This approach ensures that a workable booting system remains on the disk
during an OTA update and reduces the likelihood of an inactive device after an update, resulting
in fewer device replacements and device reflashes at repair and warranty centers. Other
commercial-grade operating systems, such as ChromeOS, also use A/B updates successfully.

Figure 2-5: Over-the-Air Updates
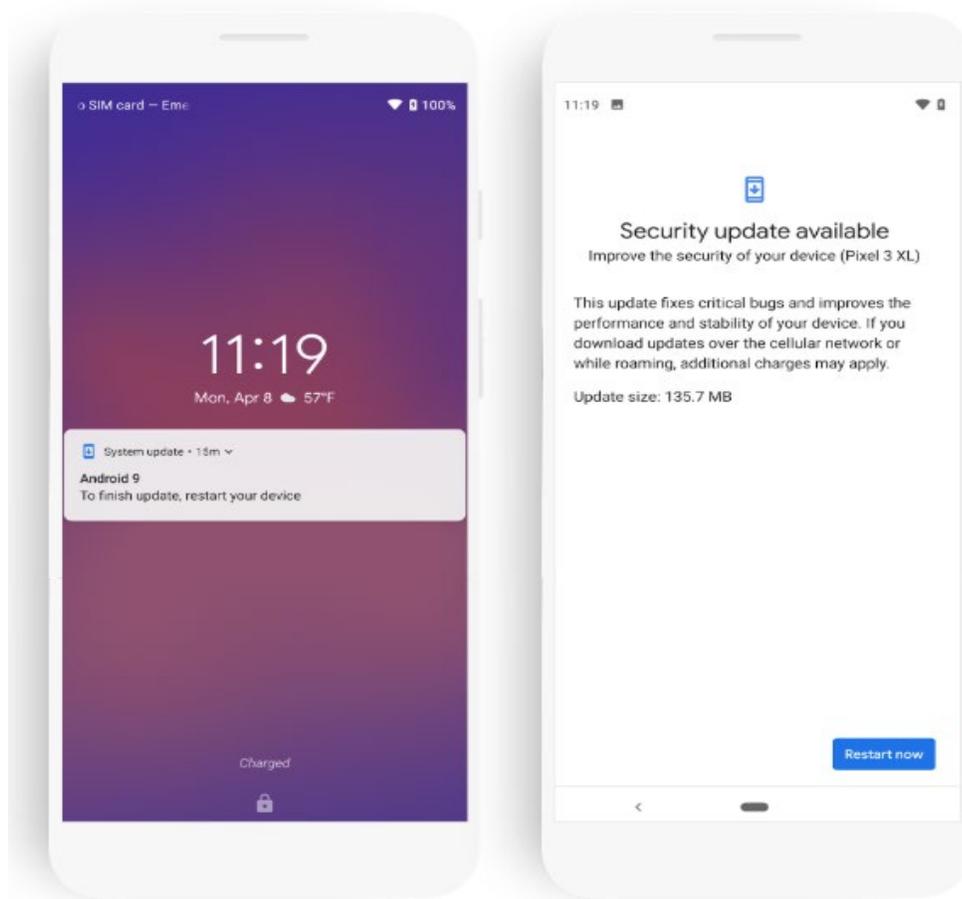
The user will receive a notification when an update is made available. No special configuration will be required to ensure a secure update process.

In Android 10 and above, administrators of fully managed devices can install system updates via a system update file. Manual system updates allow IT administrators to do the following:

- Test an update on a small number of devices before installing them widely.
- Avoid duplicate downloads on bandwidth-limited networks.

Stagger installations or update devices only when not being used.

3. GOOGLE SECURITY SERVICES

3.1 Google Play Protect

Google Play Protect is a powerful threat detection service that actively monitors a device to protect it, its data, and its apps from malware. The always-on service is built into any device that has Google Play, protecting more than 3 billion devices.

Google Play Protect regularly scans all the apps on a device, including any not installed from the Play Store, for harmful behavior or security risks. If it detects an app containing malware, it notifies the user, who can then uninstall the application. Google Play Protect can also remove malicious apps automatically as part of its prevention initiative and use the information it gathers to improve the detection of potentially harmful applications (PHAs). In addition, the user can opt to have unknown apps sent to Google for better detection information.

Google Play Protect is available on devices enabled with Google Mobile Services. On devices running Android 4.2 or higher, users can opt out of Google Play Protect, although keeping it on is recommended.

An enterprise can further minimize the potential for malware by using the DISALLOW_INSTALL_APPS user restriction to prevent users from installing any apps to their device when fully managed. DISALLOW_INSTALL_UNKNOWN_SOURCES allows an organization to restrict users to only install apps from system sources such as the Play Store. ENSURE_VERIFY_APPS can disable the ability to turn off app verification through Google Play Protect for fully managed devices or the work profile.

3.2 SafetyNet

SafetyNet is a set of Google Play Protect APIs that protects apps against security threats. This series of APIs can mitigate against device tampering, bad URLs, PHAs, and fake users.

The SafetyNet Attestation API provides several tools to determine the security of the Android environment for apps. These APIs analyze the devices that have installed the application. The service attests if the device is known to Google as CTS compatible. The return value indicates to the calling application (for example, an EMM Agent or other enterprise application) whether the device is a known device running a known build. Additionally, the service provides a third-party API in Google Play services, using GoogleApiClient, which returns a value indicating whether the device is in the claimed state.

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

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

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

3.3   Google Safe Browsing

Google Safe Browsing helps protect more than four billion devices every day by showing warnings to users when they attempt to navigate to dangerous sites or download dangerous files. Safe Browsing also notifies webmasters when their websites are compromised by malicious actors and helps them diagnose and resolve the problem so that their visitors stay safer. Safe Browsing protections work across Google products and power safer browsing experiences across the internet.

4. DEVICE CONFIGURATION

Work profiles on company-owned devices are for Government-furnished devices used for both work and personal purposes. The organization still manages the entire device; however, the separation of work data and apps into a work profile allows organizations to enforce two separate sets of policies. For example:

- A stronger set of policies for the work profile that applies to all work apps and data
- A more lightweight set of policies for the personal profile that applies to the user's personal apps and data

Figure 4-1: Personal Profile and Work Profile
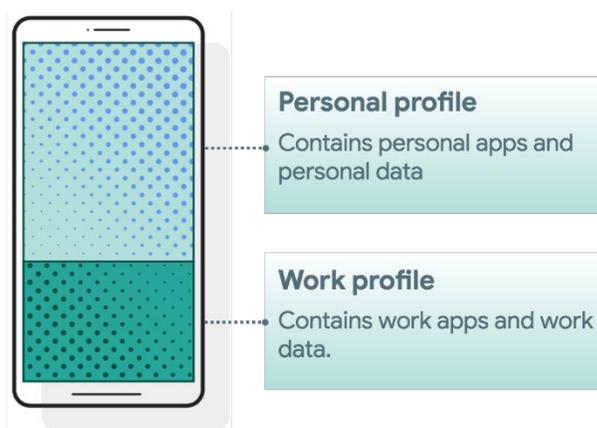
5. PROCEDURES

5.1 Device Wipe

Google Android 10 and above devices can be wiped by a factory data reset, EMM, or when the failed authentication limit is reached. Pre-installed apps in the Data partition will be wiped from the device after a device wipe. If any of those apps are configured in the application disable list, the policy will no longer be effective, and the user will not be prevented from installing them. The only solution is to both uninstall/disable the unwanted apps and use either application installation allowlisting or blocklisting.

- For application installation allowlisting, the unwanted apps will be implicitly blocklisted (all apps blocklisted), and the unwanted apps will not be allowlisted.
- For application installation blocklisting, the unwanted apps will be explicitly blocklisted.

Application installation blocklisting must only be used if the Authorizing Official (AO) has not approved unrestricted use of personal apps where a personal and work profile exists.

6. SPECIAL GUIDANCE

6.1 Google Android Device Disposal

Follow the procedure below prior to disposing of (or transferring to another user) Android devices that have never been exposed to classified data using site property disposal procedures for mobile devices.

Follow the device manufacturer's instructions for wiping all user data and installed applications from the device memory.

6.2 Configuration of the Personal Space

DoD mobile service providers may allow users full access to the Google Play app store for the personal space, including downloading and installing Google Play apps and syncing personal data on the device with personal cloud data storage accounts when ALL of the following conditions have been met:

- The site AO has approved full access to the Google Play app store for the personal space, including downloading and installing Google Play apps into the personal space and syncing personal data on the device with personal cloud data storage accounts; written approval must be available for any system compliance review.
- The site AO has provided guidance on acceptable use and restrictions, if any, on downloading and installing personal apps and data (music, photos, etc.) on the Google Android device personal space (guidance can be added to user training or the User Agreement).
- Site mobile devices are configured with a technology used for data separation between work apps and data and personal apps and data that is NIAP certified.
  - Currently, Android Enterprise (AE) is the only NIAP-certified technology for application separation for Google Android mobile devices.
- The site EMM is configured to restrict the download of apps from all third-party app stores.
- The EMM or user restricts the use of DoD VPN profiles within the personal space.
- Site mobile device users receive training on known Google Play application risks and required STIG controls that must be enabled by the user (User-Based Enforcement).
  - See STIG requirement GOOG-12-009800 for more information.

21

7. DOD PKI PUREBRED

Purebred is a key management server and set of apps for mobile devices that provides a secure, scalable method of distributing software certificates for DoD PKI subscribers' use on commercial mobile devices.

Requirements for Google Android devices credentialed using DoD PKI Purebred are as follows:

- Users are responsible for maintaining positive control of their credentialed devices; the DoD PKI certificate policy requires subscribers to maintain positive control of the devices that contain private keys and to report any loss of control so the credentials can be revoked.
- Upon device retirement, turn in, or reassignment, ensure a factory data reset is performed prior to device handoff; follow mobility service provider decommissioning procedures as applicable.

Additional information is available at https://cyber.mil/pki-pke/purebred/.

8.   ADDITIONAL CONSIDERATIONS

8.1   Wearables

The use of virtual reality (VR) wearables with a DoD-owned Google Android 12 device is prohibited. VR wearables are considered a personal use product with no DoD mission requirement.

8.2   Google Location Tracking

DoD policy memorandum "Use of Geolocation-Capable Devices, Applications, and Services," 03 August 2018, prohibits the use of geolocation-capable devices, applications, and services on DoD mobile devices in designated operational areas (OAs). Independent researchers and DISA analysis has determined that even when "Location History" is disabled, Google continues to store location data on the mobile device[2]. Therefore, AOs should consider additional actions to limit Google tracking mobile devices when these devices are operated in OAs.

---

[2] A copy of DISA's "Google Location Tracking on Samsung Devices" white paper can be requested by sending an email to disa.stig_spt@mail.mil.