INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-03-29 18:32 UTC

CVE-2026-21385, Qualcomm Android Kernel Privilege Escalation [SCC-2026-0002]

CVE VULNERABILITY | HIGH | CVSS 7.8

SCC Item ID	SCC-CVE-2026-0002
Type	CVE Vulnerability
CVE ID	CVE-2026-21385
Severity	HIGH
CVSS Base Score	7.8
Affected Products	Android devices with Qualcomm chipsets — unpatched against SPL 2026-03-05
Published	20260305

Executive Summary

A privilege escalation flaw in the Qualcomm Android kernel allows a local attacker with basic app execution capability to gain full kernel-level control of the device. Included in Google's March 2026 Android Security Bulletin. Patch is available via the March 2026 Android Security Patch Level (SPL 2026-03-05).

Technical Analysis

CVE-2026-21385 is a privilege escalation vulnerability in the Qualcomm kernel component affecting Android devices. An attacker with local code execution can escalate to kernel privileges, enabling full device compromise including access to all stored credentials, communications, and sensor data. The vulnerability is present across a range of Qualcomm chipset families used in Android devices from multiple OEMs. Google included this CVE in the March 2026 Android Security Bulletin. The fix is delivered via SPL 2026-03-05. Devices that cannot receive the March patch (end-of-life or delayed OEM update) should be treated as unpatched until confirmed otherwise. Note: EPSS score pending, check api.first.org/data/1.0/epss?cve=CVE-2026-21385 before publishing. CVSS base score: 7.8 (HIGH). CISA KEV status: not listed as of 2026-03-04 pipeline close.

Action Checklist

1. 1. Confirm whether your MDM policy has deployed March 2026 Android Security Patch (SPL 2026-03-05) to all managed Android devices
2. 2. Run MDM compliance report, flag any devices on SPL < 2026-03-05

3. 3. For devices that cannot receive the patch (EOL devices, delayed OEM update): consider restricting access to corporate email and sensitive applications until patched

4. 4. For BYOD environments: issue employee advisory requiring March patch before accessing corporate resources

5. 5. Check EPSS score at api.first.org before publishing, update _scc_severity.epss_score field

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate to CISO and external IR firm if any unpatched Qualcomm device has been subject to a privilege escalation attack attempt detected in kernel logs (SELinux audit denials for kernel module load, unexpected syscalls, or memory access violations), or if EOL device population exceeds 5% of managed fleet.
Recovery Notes	Post-patch deployment, verify integrity via MDM re-scan and correlate patch success logs with device enrollment records. For any device that rejected the patch, investigate OEM update blockers (storage space, bootloader lock status, carrier locks) and escalate to vendor support or retire from sensitive access. Conduct a post-incident review (NIST 800-61r3 §3.4.2) to document time-to-detection and time-to-remediation for this CVE class.
Forensic Artifacts	/data/anr/traces.txt (Android native crash dumps; kernel privilege escalation attempts may appear as unexpected kernel module loading or memory protection violations) | SELinux audit logs: /var/log/audit/audit.log or dumpsys selinux_policy (privilege escalation attempts trigger SELinux policy violations) | Kernel dmesg buffer (dmesg | grep -i 'qualcomm|kernel|oops|panic' — captures kernel-level exploitation indicators) | MDM enrollment and patch deployment logs with timestamps and failure codes for unpatched devices | Authentication and VPN gateway access logs for baseline usage and post-restriction compliance (event IDs 4624, 4625 for Windows; syslog auth.log for Linux/Unix gateways)

Per-Action IR Details

1. Confirm whether your MDM policy has deployed March 2026 Android Security Patch (SPL 2026-03-05) to all managed Android devices

NIST Phase: Preparation

Reference: NIST 800-61r3 §2.1 (Preparation phase: tools and processes); NIST 800-61r3 §3.2.4 (vulnerability management integration)

Controls: NIST SI-2 (Flaw Remediation), NIST SI-12 (Information Handling and Retention), CIS 4.1 (Inventory and Control of Hardware Assets), CIS 7.3 (Address Unauthorized Software)

Compensating: If no native MDM reporting: manually query enrolled devices via ADB (Android Debug Bridge) across all connected devices using `adb shell getprop ro.build.version.security_patch` in batch mode, log results to CSV with device IMEI, enrollment date, current SPL. Cross-reference against enrollment manifest.

Evidence: Capture baseline MDM enrollment manifest (device count, model, current SPL) before initiating compliance scan. Export unpatched device roster with enrollment timestamps. Preserve MDM command history logs showing patch deployment attempts and failure reasons for any device that rejected the update.

2. Run MDM compliance report, flag any devices on SPL < 2026-03-05

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 (Detection and Analysis); NIST 800-61r3 §3.2.1 (analysis goals)

Controls: NIST CA-7 (Continuous Monitoring), NIST SI-4 (Information System Monitoring), CIS 8.1 (Unified Endpoint Management), CIS 8.2 (Enforcing Mobile Device Management)

Compensating: For teams without MDM: deploy a lightweight asset discovery script using `adb` to poll all paired devices hourly; parse `ro.build.version.security_patch` property and compare against baseline using `grep` and `awk`. Store results in timestamped logs for trend analysis. Flag any device not reporting within 24 hours as offline risk.

Evidence: Before running report: export current MDM database schema and row count. Capture compliance report generation timestamp, filtering criteria applied (SPL date threshold), and total devices in scope. Preserve the raw unpatched device list with enrollment status, last-seen timestamp, and MDM policy assignment. Document any devices that fail to report compliance status.

3. For devices that cannot receive the patch (EOL devices, delayed OEM update): consider restricting access to corporate email and sensitive applications until patched

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 (Containment); NIST 800-61r3 §3.3.1 (short-term containment options)

Controls: NIST AC-3 (Access Control Enforcement), NIST AC-6 (Least Privilege), NIST IA-2 (Authentication), CIS 3.3 (Configure Data Access Control Lists), CIS 6.2 (Ensure Necessary Ports, Protocols, and Services Are Restricted)

Compensating: Configure MDM-equivalent access control via network-level enforcement: (1) Deploy a conditional access policy on the email gateway (e.g., Postfix with dovecot LDAP filter) that queries device SPL metadata and blocks IMAP/POP3 authentication if SPL < 2026-03-05. (2) For SaaS applications (O365, Salesforce): implement API-driven conditional access using OAuth scopes tied to device posture queries. (3) For on-prem apps: use reverse proxy (nginx/Apache) with mod_auth_oidc to inject device posture checks before granting credentials.

Evidence: Before implementing access restrictions: capture baseline of which EOL devices exist in inventory, their current access permissions (email, VPN, cloud app grants), and user roles associated with each. Log all access restriction policy changes with timestamp and justification. Preserve authentication attempt logs from email and app gateways for 90 days to detect bypass attempts post-restriction.

4. For BYOD environments: issue employee advisory requiring March patch before accessing corporate resources

NIST Phase: Preparation

Reference: NIST 800-61r3 §2.1 (Preparation: awareness and training); NIST 800-61r3 §3.2 (Detection coordination with end users)

Controls: NIST AT-1 (Security Awareness and Training), NIST AT-2 (Security Awareness Training), NIST AC-2 (Account Management), CIS 17.7 (User Training on Recognizing and Reporting Security Incidents)

Compensating: If no automated BYOD enforcement: issue advisory via email with enforcement deadline (48–72 hours). Require users to self-attest patch status via a simple web form (no privileged access required). Cross-reference self-attestations against optional ADB device info submissions. For non-compliant users after deadline, disable VPN and corporate email access via Active Directory group policy or equivalent (revoke group membership). Notify manager and employee simultaneously with 24-hour grace period.

Evidence: Preserve the advisory announcement (date, distribution list, content). Log all user attestations with submission timestamp. Capture VPN gateway and email server access logs for 7 days pre-advisory and 30 days post-deadline to establish baseline usage patterns and identify access attempts post-restriction.

5. Check EPSS score at api.first.org before publishing, update _scc_severity.epss_score field

NIST Phase: Post Incident

Reference: NIST 800-61r3 §3.4 (Post-Incident Activities); NIST 800-61r3 §3.4.2 (lessons learned)

Controls: NIST IR-6 (Incident Reporting), NIST SI-4 (Information System Monitoring), CIS 18.1 (Define Roles & Responsibilities), CIS 18.2 (Assign a Lead Incident Response Organizer)

Compensating: For air-gapped networks: manually fetch the EPSS API response from an external workstation (curl -s 'https://api.first.org/data/v1/epss?cve=CVE-2026-21385' | jq '.data[0].epss') and record the score in your documentation system. If API is unreachable, use CVSS 7.8 as default severity proxy. Document the exact timestamp of EPSS retrieval and whether the score differs materially from CVSS (>0.5 difference warrants re-risk assessment).

Evidence: Capture the EPSS API response JSON with full timestamp. Compare EPSS score against initial CVSS 7.8 rating and document any discrepancy. Preserve the advisory publish timestamp and the EPSS query timestamp to establish recency of threat intelligence.

Detection Guidance

This vulnerability requires local code execution as a precondition, it cannot be exploited remotely without chaining with another vulnerability. Detection focus: look for unusual privilege escalation patterns on Android endpoints in EDR/MDM telemetry. Monitor for root detection bypass attempts on enrolled devices.

Indicators of Compromise

Type	Value	Context	Confidence
FILE_PATH	`/proc/self/mem`	Access to /proc/self/mem is suspicious when initiated by untrusted applications or non-system processes attempting to read/write kernel memory space without proper SELinux contexts, particularly when combined with capability escalation syscalls (capset, prctl) or execution of unsigned kernel modules; legitimate use is restricted to debuggers (gdb, strace) with explicit user permission and proper ptrace capabilities, whereas exploitation attempts typically bypass permission checks through memory mapping techniques or exploit kernel race conditions without user interaction.	MEDIUM
FILE_PATH	`/dev/kgsl-3d0`	Suspicious when accessed by unprivileged processes without GPU driver initialization or when followed by capability escalation syscalls (prctl, execve with elevated privileges); legitimate use occurs only during GPU driver setup by system services (surfaceflinger, hwcomposer) or graphics applications with proper file descriptor inheritance, whereas exploitation attempts typically show direct device node access from user-space exploits or memory mapping operations followed by ioctl calls attempting to trigger out-of-bounds writes in kernel memory.	MEDIUM

Type	Value	Context	Confidence
FILE_PATH	`/dev/qseecom`	Suspicious when unprivileged processes or third-party applications attempt direct ioctl calls to /dev/qseecom without corresponding qseecomd daemon initialization or when accessed outside the TEE communication stack, as legitimate use is restricted to system daemons (qseecomd, hwcomposer, vendor.qti.hardware.perf) running with CAP_SYS_ADMIN that communicate through authenticated kernel module handlers, whereas exploitation attempts originate from user-space apps, shell contexts, or processes with insufficient SELinux domain transitions to qseecom_device. Detection should flag open() or ioctl() syscalls to this device node from non-whitelisted processes, missing parent process chain validation (qseecomd ancestry), or direct device manipulation without intermediate kernel TEE driver validation.	MEDIUM
FILE_PATH	`/data/local/tmp/*.so`	Suspicious when .so files are written to /data/local/tmp/ by non-system processes and subsequently loaded via dlopen() or LD_PRELOAD, as this world-writable directory is not used by legitimate Android applications for library staging; legitimate shared libraries are packaged within APKs or installed to system-protected directories (/system/lib*, /vendor/lib*). Detection should trigger on file creation in this path followed by process execution with environment variables or syscalls indicating dynamic library loading, particularly when the loading process lacks proper signature verification or runs with elevated privileges.	MEDIUM

Type	Value	Context	Confidence
FILE_PATH	`/data/local/tmp/poc`	This is suspicious when a non-system process writes executable files to /data/local/tmp/poc and subsequently executes them with elevated privileges, as legitimate applications do not stage kernel exploit binaries in this world-writable directory. Look for file creation events followed by execve() calls with uid/gid changes or SELinux policy violations in kernel logs, which differs from normal app behavior where /data/local/tmp is used only for temporary caches that are never directly executed.	MEDIUM
FILE_PATH	`/system/bin/su`	Presence of su binary at /system/bin/su is suspicious when spawned by unprivileged processes or non-interactive system daemons, as legitimate su access on Android is restricted to authorized applications and typically requires explicit user interaction; detection should flag execution without corresponding user consent prompts, unexpected parent processes, or su invocations from third-party applications in logs and EDR tools, distinguishing this from legitimate privileged operations initiated through Android's official permission framework.	MEDIUM
FILE_PATH	`/sbin/su`	Root binary installation path indicating successful Android kernel privilege escalation	MEDIUM

Type	Value	Context	Confidence
FILE_PATH	`/data/local/tmp/libexploit.so`	This file is suspicious when found in /data/local/tmp/ because world-writable staging directories are used to bypass SELinux restrictions and signature verification that would block loading from system paths, allowing unprivileged processes to dynamically load and execute kernel exploitation payloads - legitimate shared libraries are always installed in /system/lib64/, /vendor/lib64/, or app-specific directories with enforced SELinux contexts and certificate chains, never in world-writable temporary locations. Detection should focus on libexploit.so being written by non-system processes, followed by dlopen() calls from unprivileged processes, absence of valid signature chains, and SELinux policy violations attempting to load from /data/local/tmp/.	MEDIUM
FILE_PATH	`/proc/kallsyms`	Reading /proc/kallsyms from unprivileged processes or non-standard applications (e.g., third-party apps, scripts, or processes without kernel development context) is suspicious because legitimate access is restricted to privileged debugging tools and kernel developers, whereas exploit chains targeting CVE-2026-21385 enumerate this file to bypass KASLR and locate kernel function addresses for privilege escalation. Look for EDR alerts on file reads of /proc/kallsyms originating from user-space processes, mobile apps, or unusual process parents, combined with subsequent attempts to write to kernel memory or execute code at elevated privileges.	MEDIUM

Type	Value	Context	Confidence
FILE_PATH	`/proc/iomem`	Access to /proc/iomem by unprivileged processes is suspicious because it enables reconnaissance of physical memory layout required for kernel exploitation - legitimate access is restricted to privileged system utilities (e.g., kdump, systemd-boot) during boot/diagnostics, whereas CVE-2026-21385 exploitation typically involves user-space processes reading this file prior to privilege escalation attempts, detectable via EDR monitoring for non-root processes opening /proc/iomem followed by syscalls associated with memory mapping or capability escalation (capset, prctl), especially when originating from writable directories (/tmp, Downloads) or spawned by interpreters (python, perl, bash).	MEDIUM
FILE_PATH	`/dev/ion`	/dev/ion access is suspicious when initiated by unprivileged processes, non-system applications, or processes spawning unexpected child processes with ioctl calls, as legitimate use is restricted to system services and hardware-accelerated media frameworks; look for CVE-2026-21385 exploitation attempts via abnormal ioctl commands (ION_IOC_ALLOC, ION_IOC_FREE) from userland processes, unusual heap allocation patterns, or failed privilege escalation attempts followed by elevated code execution, which differs from normal legitimate access patterns of system daemons (mediaserver, hwcomposer) performing routine memory allocation through expected kernel interfaces.	MEDIUM
FILE_PATH	`/sys/kernel/debug/ion`	ION debugfs path that may be accessed during heap grooming for kernel exploit staging	MEDIUM
REGISTRY_KEY	`N/A - Android/Linux kernel exploit; no Windows registry keys applicable`	This CVE affects Android/Linux on Qualcomm hardware; registry keys are not a relevant artifact	MEDIUM
FILE_PATH	`/data/data//lib/libpwn.so`	Malicious application native library executing kernel exploit payload delivered via side-loaded or trojanized APK	MEDIUM

Type	Value	Context	Confidence
FILE_PATH	`/proc/self/status`	This is suspicious when accessed by unprivileged processes spawning exploitation code that repeatedly reads /proc/self/status to verify UID transitions from non-zero to uid=0, differing from legitimate system monitoring tools which typically query this file once per sampling interval rather than in tight loops following suspicious syscalls like ioctl to Qualcomm kernel drivers. Detection should focus on sequences of rapid /proc/self/status reads preceded by CVE-2026-21385 vulnerable driver calls or followed by privilege-level execution changes within the same process context.	MEDIUM
FILE_PATH	`/data/local/tmp/kpatch`	Binary artifact created in world-writable directory by unprivileged process following CVE-2026-21385 kernel exploit execution is suspicious because /data/local/tmp should never host kernel patches - legitimate patching occurs exclusively via signed OTA updates or MDM deployment. In EDR/logs, detect this by correlating file writes to /data/local/tmp/kpatch from processes lacking system certificates with subsequent process creation events executing kpatch under root uid; legitimate kernel tools are never ad-hoc staged or executed from temporary directories by user-space daemons or unprivileged processes, dropped by unauthorized or anomalous process execution.	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1404** — Exploitation for Privilege Escalation

CIS-V8

- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management
- **5.4** — Restrict Administrator Privileges to Dedicated Administrator Accounts

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities

NIST-800-53R5

- **AC-6** — Least Privilege

SOC2-TSC

- **CC6.3** — Authorizes, modifies, or removes access

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1404		

Sources

Source	URL	Tier
Google Android Security Bulletin March 2026	https://source.android.com/docs/security/bulletin/2026-03-01	T3
Qualcomm Security Advisories	https://www.qualcomm.com/company/product-security/advisories	T3
NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-21385	T1

Generated 2026-03-29 18:32 UTC by TJS Security Command Center