MOBILE TELEPHONY FORENSICS · NQ-MTF-2026-001 · August 2026

Reconstructing Telephony Faults from the Radio Log: Per-Slot Event Timelining Under an Adversarial Input Model

A method for reducing Android radio logcat to a scored registration and service timeline, with subscriber-identifier redaction by default, grounded in field diagnosis of a secondary-profile IMS fault.

0xPersist · NorthQuinn Inc.

ABSTRACT

Telephony failures on Android are diagnosed from a high-volume radio log in which the diagnostically significant events—service loss, radio access technology downgrades, IMS deregistration, registration flapping, and explicit reject causes—are obscured by routine noise and complicated by multi-SIM interleaving. This paper presents a method that extracts these events into a normalized, **per-slot** timeline using a ranked RAT transition model, classifies five anomaly types by severity, and does so under an explicit **adversarial input model** in which log content is treated as untrusted. Design commitments include subscriber-identifier redaction by default via hashed tokens that preserve correlation semantics, terminal escape-injection defense, data minimization, crafted-input resource guards, and SHA-256 integrity stamping of the analyzed bytes. Downgrades toward the 2G family are weighted HIGH as a cell-site-simulator indicator, with the stated boundary that host-side logs cannot confirm radio-frequency conclusions. The method is grounded in field diagnosis of a secondary-profile IMS registration fault on GrapheneOS and is implemented as the open-source tool **radio-triage**.

1. The Diagnostic Problem in Mobile Telephony

Telephony failures on modern Android devices are difficult to diagnose because the evidence is buried. The radio logcat buffer records baseband and telephony-framework events in volume, interleaving registration state changes, radio access technology transitions, IMS session lifecycle events, and carrier configuration activity with a large quantity of routine noise. An investigator confronted with a device that has lost service, cannot place calls, or has silently degraded to a legacy network must reconstruct what occurred from this stream, typically by hand.

This reconstruction is both tedious and error-prone. The events that matter, a transition to out-of-service, a downgrade from LTE to a 2G-family technology, an IMS deregistration that disables voice-over-LTE, are individually legible but collectively obscured. Two additional complications compound the difficulty. First, radio log content is influenced by network-side behavior and must be treated as untrusted input rather than as a neutral record. Second, multi-SIM devices interleave events across physical and logical slots, so a naive reading can misattribute a recovery on one slot as an outage, or fabricate a cross-slot downgrade that never occurred.

This paper describes a method for extracting the registration and service events that carry diagnostic weight, scoring the transitions by severity, and doing so under an explicit adversarial input model. The method is grounded in field diagnosis of a real telephony fault on a hardened Android platform, summarized in Section 5.

2. Event Extraction and the Transition Model

The core of the method is a state model over telephony registration. The radio access technology (RAT) is ranked from most to least capable—NR, then LTE, then the 3G family, then the 2G family (GSM, GPRS, EDGE)—so that any change in registration can be classified as an upgrade, a downgrade, or a lateral move. Transition lines are parsed on rightmost-token-wins semantics, reflecting the “old to new” convention of the underlying log format, and all state is tracked per phone and per slot using the phoneId and slotId hints present in the log.

This per-slot tracking is not a refinement but a correctness requirement. Because registration state is maintained independently for each slot, a recovery on one slot is never misread as an outage, and the interleaved events of a multi-SIM device never produce a spurious cross-slot downgrade. The output is a normalized event timeline rather than a reproduction of raw message bodies, which both improves legibility and serves the data-minimization posture described in Section 4.

3. Anomaly Classification

Extracted transitions are classified into five anomaly types, each assigned a severity that reflects both its operational impact and its potential security significance.

Anomaly	Severity	Trigger condition
service_loss	HIGH	Transition to OUT_OF_SERVICE or EMERGENCY_ONLY.
rat_downgrade	HIGH if to 2G, else LOW	RAT rank decrease (NR > LTE > 3G > 2G).
ims_deregistration	MEDIUM	IMS deregistered; VoLTE / VoNR voice impact.
registration_flapping	MEDIUM	Repeated service transitions within a short window.
registration_reject	MEDIUM	Reject or denial cause present in registration messages.

Table 1. Anomaly classes, severity assignment, and trigger conditions.

The severity weighting of the RAT downgrade class warrants explanation. A forced downgrade toward the 2G family is weighted HIGH because coerced downgrade to a legacy, weakly-authenticated technology is a classic indicator associated with cell-site simulators. This assignment is made with a deliberate and stated epistemic boundary, addressed directly in the following section: host-side logs record what the operating system observed, not what the network did, and a HIGH downgrade finding is an indicator to be corroborated, not a conclusion.

4. Security Model

Because network-side behavior influences what the baseband records, log content is treated as untrusted input throughout. This adversarial input model produces several concrete design commitments that distinguish the

method from a conventional log parser.

Subscriber-identifier redaction by default. Phone numbers, IMSI, ICCID, and IMEI-shaped digit runs, labeled identifiers, and cell identity fields are redacted in all output as a hashed token of the form [REDACTED:<hash8>]. The hashing preserves same-versus-different semantics, so an analyst retains the ability to correlate repeated identifiers across the timeline without the identifiers themselves ever appearing in shareable output. Redaction is disabled only by explicit flag for strictly local analysis.

Terminal escape-injection defense. All control characters are rendered as visible escapes, preventing a crafted log from injecting terminal control sequences into an analyst's session.

Data minimization. Raw log excerpts are printed only for flagged anomalies; the timeline itself shows normalized events rather than message bodies. The investigator sees the raw material precisely where it is evidentially relevant and nowhere else.

Crafted-input guards and integrity. Input is capped at 128 MiB and 500,000 events to bound resource consumption against a hostile file, and a SHA-256 hash of the exact analyzed bytes is stamped on every output, establishing a verifiable chain between the artifact examined and the conclusions drawn. Capture tooling validates labels against path traversal and writes captures with owner-only permissions. The implementation performs no network access and no subprocess execution and depends only on the standard library.

One limitation is stated plainly in the tool itself and is repeated here as a matter of intellectual honesty: pattern-based redaction cannot guarantee that every unrecognized identifier format is caught, and output should be reviewed before sharing regardless. A self-contained regression suite of seventeen checks exercises the detection logic, redaction, escape handling, resource guards, hash integrity, and the minimum Python version.

5. Field Grounding: A Secondary-Profile IMS Fault on GrapheneOS

The method is not purely synthetic. Its design was informed by field diagnosis of a real telephony fault on a GrapheneOS device using a mobile virtual network operator eSIM, in which the owner profile could place calls but a secondary user profile could not. The registration timeline localized the fault to IMS: voice in the secondary profile had no path because IMS never registered, forcing any call attempt onto legacy circuit-switched fallback or failing outright.

Root-cause analysis established that the carrier configuration enabling IMS is delivered by a carrier application that installs into the owner profile but is not propagated to secondary user profiles. In the absence of that application, the secondary profile fell back to a default carrier configuration lacking IMS enablement for the operator. The corrective action was to install the existing carrier application into the affected user and cycle airplane mode to force re-registration. The finding was reported upstream through the GrapheneOS issue tracker. This case illustrates the method's intent: the value is not in observing that calls failed, which the user already knew, but in reconstructing the registration sequence that explains why, at which point the fix follows directly.

6. Representative Output

```
$ ./capture-radio.sh baseline           # -> baseline-radio.txt
$ python3 radio_triage.py --log baseline-radio.txt --timeline

# adb logcat -b radio -v threadtime -d > radio.txt   (manual capture)
```

Figure 1. Capture and analysis invocation. The timeline view renders normalized events per slot.

Output is available as a human-readable timeline, a per-anomaly view with the corresponding redacted raw excerpt, and a machine-readable JSON form for pipeline integration. The header reports a lines-matched count that functions as a drift indicator: because radio log tag and message formats vary across original equipment manufacturers and Android versions, a low match count signals that the parser should be validated against the specific device and version under examination.

7. Stated Limitations

The method is presented with its boundaries explicit, consistent with the principle that a forensic tool which overstates its reach is worse than one which states its reach precisely. Detection rules are validated against synthetic logcat fixtures, and format drift across vendors and Android versions is a real and disclosed constraint, mitigated by the drift indicator. Minor known limitations include a flap window measured in event count rather than elapsed time, a benign cause value that can match the reject rule, and the absence of a year in logcat timestamps, which makes event ordering across a year boundary unreliable. Most importantly, host-side logs establish what the operating system observed, not what the network did; any radio-frequency conclusion, including the presence of a cell-site simulator, requires radio-frequency evidence and cannot be drawn from host logs alone. A HIGH downgrade finding is therefore an instruction to corroborate with RF-side capture, not a determination.

8. Conclusion

Mobile telephony diagnosis exemplifies a broader forensic principle: the raw record contains the answer, but the answer is inaccessible until the record is reduced to the events that carry evidential weight and those events are placed in a defensible temporal and per-slot model. Doing so under an adversarial input model, with subscriber identifiers redacted by default, control characters neutralized, and the analyzed bytes hash-stamped, allows the resulting timeline to be shared and relied upon. The method described here reduces a wall of radio log noise to a scored sequence of registration and service events, grounded in a real field fault, and bounded by a candid statement of what host-side logs can and cannot establish.

Availability. The method is implemented as **radio-triage**, an open-source tool that parses the Android radio logcat buffer into a per-slot telephony event timeline and classifies five anomaly types with subscriber-identifier redaction by default. It is self-contained, standard-library only, requires Python 3.8 or later, performs no network access or subprocess execution, and ships a seventeen-check regression suite. It is the companion to **carrier-diff**, which answers what telephony state differs between profiles where radio-triage answers what changed over time. MIT licensed. Source: github.com/0xPersist/radio-triage