GGSEC RESEARCH · WHITEPAPER · JUNE 2026

Baseband Security Research

GSM/LTE Modem Firmware Analysis · Remote Code Execution via Radio Interface · Baseband Bootkits · Silent Persistence Below the OS

Author: Maciej Gojny

Organization: GG Advanced IT Security

Web: ggsec.de

Classification: TLP:WHITE – Public

Version: 1.2 · June 2026

TLP:WHITE – PUBLIC RELEASE

01 Why Baseband Security Matters

The cellular baseband processor is a dedicated computing subsystem responsible for GSM, UMTS, LTE and 5G NR communication. Modern baseband processors execute proprietary firmware and real-time operating systems such as ThreadX, Nucleus RTOS or vendor-specific implementations.

Unlike application-level software, the baseband directly processes data received from untrusted radio networks. As a result, vulnerabilities in protocol parsing, memory management or inter-processor communication mechanisms may expose devices to attacks originating from the cellular infrastructure itself.

Successful compromise of modem firmware may provide an attacker with privileged access to the cellular subsystem. Depending on chipset architecture, isolation mechanisms and firmware design, such compromise could potentially facilitate attacks against higher privilege domains, including the application processor.

It is important to note that exploitation impact varies significantly across vendors and hardware generations. Modern chipsets increasingly implement secure boot, memory protection, firmware signing and hardware isolation mechanisms which reduce the likelihood of persistent compromise.

> **Key Security Observation: Baseband firmware represents one of the most privileged and least transparent components in modern connected devices. Security visibility into modem internals remains limited compared to traditional operating system environments.**

1.1 Baseband Architecture Overview

Typical smartphone architecture consists of multiple independent execution domains:

```
Application Processor (Android / Linux / iOS)
|
Shared Memory Interfaces
|
Baseband Processor
|
Protocol Stack (L1 / L2 / RRC / NAS)
|
RF Front-End
|
Cellular Network
```

Communication between the application processor and modem is commonly implemented through shared memory, mailbox mechanisms, DMA channels or vendor-specific IPC frameworks. Examples include:

- Qualcomm – QMI, QRTR, Shared Memory Driver (SMD)
- MediaTek – CCCI (Cross-Core Communication Interface)
- Samsung Exynos – SIPC and vendor-specific IPC layers

These interfaces represent important attack surfaces and have historically been associated with privilege-escalation vulnerabilities.

02 Modem Firmware Analysis Workflow

Phase	Activities	Outputs
Static RE	Extract firmware from EDL/fastboot/bootloader dumps; identify RTOS structures; locate dangerous functions (memcpy, sprintf, strcpy); map L1/L2/L3 message handlers	RTOS task list; dangerous function inventory; message handler map
Dynamic analysis	GDB stub via JTAG/Lauterbach; USRP B210 + OsmocomBB; fuzz RRC Connection Request / Paging / SI messages; monitor ramdumps & QCRIL crash logs	Crash triage; PC control PoC; fuzzing corpus

```
# Ghidra script for baseband parsing (Python / Jython)
def find_vuln_memcpy():
    for func in currentProgram.getListing().getFunctions(True):
        if "memcpy" in func.getName():
            check_user_controlled_len(func)

def check_user_controlled_len(func):
    """Trace length parameter back to user-controlled radio input."""
    refs = getReferencesTo(func.getEntryPoint())
    for ref in refs:
        caller = getFunctionContaining(ref.getFromAddress())
        if caller:
            print("[!] memcpy in: %s @ %s" % (caller.getName(), ref.getFromAdd
ress()))
```

03 Real-World Vulnerability Classes

Bug class	Location in stack	Example
Classic stack overflow	RRC/NAS message decoding	Cell Channel Description IE → memcpy(small_stack_buf, attacker_len)
Heap overflow	SIB storage / neighbor list cache	Integer underflow in length field → heap spray
Use-after-free	L2 timer callbacks	Race condition during handover
Type confusion	Vtable dispatch in RRC	Forced re-interpretation of message union
Missing CFI / stack cookies	Entire baseband RTOS	Straightforward ROP on Cortex-R

> **Most baseband firmware lacks modern mitigations (ASLR, CFI, stack canaries). A single memory corruption in radio message processing leads to reliable code execution.**

04 Case Study: Broadcast RCE on GSM Baseband

Attack vector: Crafted **System Information Type 1** broadcast, manipulating the Neighbour Cell Description IE. The modem processes the neighbor list using a vulnerable memcpy() where the length field originates from the attacker-controlled message.

4.1 Vulnerable Code Pattern

```
/* Pseudocode from reversed baseband (illustrative – Cortex-R5 target) */
/* RTOS task: L3 SIB1 parser, stack frame ~256 bytes              */

void parse_neighbor_list(uint8_t *buffer, uint16_t len)
{
    uint8_t local_buffer[128];      /* fixed stack allocation */
    if (len > sizeof(local_buffer)) /* guard present but... */
        return;
    memcpy(local_buffer, buffer, len); /* ...len is attacker-controlled uint16
*/
}

/* Root cause: no check that `len` matches actual IE field length
   from 3GPP TS 44.018 §10.5.2.22 – value parsed from OTA message
   without further validation against protocol-defined maximum (17 bytes) */
```

4.2 Attack Chain

Step	Action	Technical detail
1	Rogue BTS setup	USRP B210 + OsmocomBB / Osmo-BTS broadcasting fake BCCH on target ARFCN
2	SIB1 crafting	Neighbour Cell Description IE length field set to 0xFF (255) – OOB memcpy into 128-byte stack buffer
3	Stack layout control	Overflow overwrites saved LR register (Cortex-R has no hardware stack cookie by default on older RTOSes)
4	ROP pivot	Return to gadget chain in baseband RTOS image (fixed load address – no ASLR); redirect to shellcode stub
5	Persistence	Patch RTOS task scheduler in RAM or redirect IPC handler to attacker-controlled routine

4.3 5G NR Attack Surface Note

Modern 5G SA (Standalone) deployments introduce new attack surfaces not present in GSM/LTE. Key differences relevant to baseband security research include:

- **SUCI spoofing:** 5G NR encrypts the SUPI via SUCI; however, implementation flaws in SUCI deconcealment at the UE side may expose the decoder to malformed ASN.1 input
- **False base station detection:** 5G introduces some anti-IMSI-catcher mechanisms (NAS integrity from initial attach), but baseband firmware handling of null-integrity NAS messages

remains a research target

- **NR RRC complexity:** 3GPP TS 38.331 NR RRC is significantly larger than LTE RRC (TS 36.331), increasing parser attack surface

Older chipsets operating in 5G NSA (Non-Standalone) mode retain the full LTE/GSM baseband stack as fallback — all GSM/LTE vulnerabilities remain applicable in NSA deployments.

> **Mitigation:** Strict IE length validation against 3GPP-defined maximums before any memcpy; hardware MPU regions per RTOS task; stack protection (-fstack-protector-strong); CFI on Cortex-R where RTOS permits.

05 Tooling & Open-Source Ecosystem

Category	Tool	Purpose
Firmware extraction	QComViewer / QXDM	Memory read & DIAG interface
Firmware extraction	EDL python client	Raw partition dumps (Qualcomm)
Firmware extraction	mtkclient	MediaTek bootrom & firmware extraction
Reversing & fuzzing	Ghidra + baseband loaders	Cortex-R context analysis
Reversing & fuzzing	r2baseband plugin	radare2 baseband analysis
Reversing & fuzzing	OsmocomBB + USRP B210	GSM fuzzing

GGSEC research tooling: Internal frameworks for baseband analysis are maintained but not publicly released. Contact **research@ggsec.de** for academic collaboration inquiries.

06 Selected Public Disclosures & Research Context

The following table summarizes publicly documented baseband vulnerabilities reported by various research organizations. Readers should verify current CVE status via MITRE/NVD databases.

Date	Vendor / Component	Vulnerability description	Status
2023-05	Qualcomm MSM	Buffer overflow in WLAN firmware (CVE-2022-25638)	Patched
2023-12	Qualcomm Baseband	RRC message parsing memory corruption (CVE-2023-33033)	Patch available
2024-01	MediaTek LTE	Baseband heap overflow via NAS (coordinated disclosure)	Under review

Advisory inquiries & coordinated disclosure contact: **research@ggsec.de**

> **Note: Organizations should verify CVE entries directly with MITRE/NVD before operational use. GGSEC does not maintain private advisories beyond standard coordinated disclosure timelines.**

07 Mitigation & Hardening Strategies

For device vendors	For enterprise / defenders
Compile with stack cookies (-fstack-protector-strong)	Use hardware root of trust to attest baseband firmware version where supported
Enforce control-flow integrity (CFI) where RTOS supports it	Monitor baseband crash dumps and DIAG logs (if accessible)
Isolate radio message parsers using MPU/TrustZone	Segment cellular modules in IoT/OT devices via network isolation
Fuzz before signing firmware	Require SBOM for baseband firmware as procurement condition

08 Hardware-in-the-Loop Fuzzing Pipeline

```
[USRP B210] → (GSM/LTE radio) → [Target modem] → (JTAG/UART) → [Crash monitor / ramdump collector]

Fuzzing orchestrator:    custom Python + GR-GSM / srsRAN
Mutation engine:         AFL++-style with protocol-aware templates (RRC, NAS, L1)
Crash triage:            automatic PC / fault address extraction from ramdumps
```

Internal GGSEC research uses this pipeline for modem firmware assessment. Academic collaboration inquiries are welcome via **research@ggsec.de** under appropriate data sharing agreements.

09 Hardware Root of Trust for Baseband

Modern baseband processors increasingly include a dedicated secure element or TEE. However, most deployed GSM/4G modems lack this. A **hardware root of trust** would measure baseband firmware into TPM PCRs before every cellular connection, allowing remote attestation of baseband integrity.

> **Challenge: No standard for baseband PCR measurement (vendor-specific). Several industry working groups are exploring attestation frameworks for modem firmware.**

10 Supply Chain & Baseband Risk

Many modem firmware images are pre-flashed during component manufacturing. This introduces supply chain risk: a compromised distributor could potentially flash malicious baseband firmware before the device reaches the system integrator. Detection is challenging without extracting and reversing the binary.

Recommendation: treat baseband firmware as high-risk binary blob. Apply binary SBOM analysis where feasible, verify signatures against vendor golden images, and consider post-deployment attestation when supported by platform hardware.

Potential risk	Mitigation approach
Unauthorized firmware modification	Cryptographic verification of firmware images
Compromised manufacturing processes	Audit requirements for vendors
Malicious firmware implants	Runtime integrity monitoring (limited support)
Weak firmware verification	Enforce secure boot chain

11 AI in Modem Security Research

Large Language Models and machine learning systems are increasingly assisting researchers in:

- Firmware triage and binary analysis
- Vulnerability discovery pattern matching
- Reverse engineering workflows
- Protocol analysis and documentation generation

At present, AI systems primarily function as force multipliers for researchers rather than autonomous exploit development platforms. No confirmed in-the-wild AI-written baseband malware has been publicly documented as of 2026.

Several tools beyond CTF scenarios are now production-grade for specific tasks: **BinAbsInt** (binary abstract interpretation for memory-safety analysis), LLM-assisted Ghidra scripting for bulk function naming and protocol struct recovery, and ML-based binary diffing for tracking firmware variants across chipset generations.

Capability	Current maturity
Automated discovery of unsafe memcpy/sprintf patterns in binary	Production-grade (BinAbsInt, CodeQL on source, LLM-assisted Ghidra scripts)
LLM-assisted protocol struct recovery from reversed firmware	Practical – significant time saving in research workflows
AI-generated ARM Cortex-R ROP gadget chains	Experimental – requires human validation
Binary diffing at scale to find cross-vendor variants	Assisted / semi-autonomous (bindiff + ML clustering)
Autonomous end-to-end baseband exploit generation	Not demonstrated publicly as of 2026

12 Limitations & Open Challenges

Challenge	Description
Baseband binary blobs	No source, stripped, proprietary ABIs → reversing is slow and error-prone
JTAG / debug lock	Production devices often disable debug interfaces → limited dynamic analysis
Soft real-time constraints	Instrumentation may alter timing → unreliable fuzzing in some cases
Lack of SBOM	Vendors rarely disclose third-party RTOS or libraries used in modem firmware

13 Recommendations

Priority	Action	Rationale
High	Enable baseband firmware version monitoring on all cellular devices where feasible	Detect unexpected modem firmware changes
High	Require baseband SBOM from vendors as procurement condition	Vulnerability correlation & incident response
Medium	Consider fuzzing pipeline for incoming modem firmware (if resources permit)	Catch memory corruption before deployment
Medium	Support standardization efforts for baseband boot attestation	Enable remote integrity verification

14 Responsible Disclosure Policy

GG Advanced IT Security follows a coordinated vulnerability disclosure approach aligned with ISO/IEC 29147 and industry best practices. When a vulnerability is identified during research or engagement:

- Vendor notification is issued privately with a 90-day remediation window (extendable by mutual agreement)
- Public disclosure occurs after patch availability or expiry of the disclosure timeline
- CERT/CC or relevant national CERT may be engaged as coordinator where appropriate

> **Report vulnerabilities or request research collaboration: research@ggsec.de · PGP key available on request via ggsec.de**

15 Appendix & References

14.1 Quick Reference – Baseband Detection Commands

```
# Check modem firmware version (Qualcomm example)
adb shell getprop ro.boot.baseband
adb shell getprop gsm.version.baseband

# Extract modem firmware from EDL (Qualcomm)
edl.py --loader=prog_firehose_ddr.elf --read=modem.bin --partition=modem

# Ghidra baseband analysis
# Load ELF, apply Cortex-R context, analyze protocol parsers
```

14.2 Publicly Documented CVEs (examples – verify via NVD)

CVE	Description	Status
CVE-2023-24033	Samsung Exynos – baseband RCE via malformed SDP in SIP/IMS, no user interaction (Google Project Zero disclosure)	Patched (March 2023)
CVE-2023-26072 / 26073	Samsung Exynos – OOB write in NAS message parsing (L3 decoder)	Patched
CVE-2023-33033	Qualcomm Baseband – memory corruption in RRC message parsing, ARM Cortex-R context	Patch available
CVE-2024-23980	MediaTek – baseband heap overflow via LTE NAS (coordinated disclosure)	Under review

14.3 References

1. 3GPP TS 24.301. "Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS)."
2. 3GPP TS 38.331. "NR Radio Resource Control (RRC) protocol specification."
3. 3GPP TS 44.018. "Mobile radio interface layer 3 specification."
4. ENISA. (2020). "5G Threat Landscape."
5. NIST SP 800-193. "Platform Firmware Resiliency Guidelines."
6. MITRE ATT&CK. "System Firmware (T1542.001) – Baseband variant."
7. OsmocomBB. "Open Source GSM Baseband Implementation."
8. Comsecuris. "Baseband Reverse Engineering Methodology."
9. Qualcomm Security Bulletins (public).
10. MediaTek Product Security Advisories (public).
11. Google Project Zero Baseband Research publications.
12. ETSI Telecommunications Security Standards.

> **Note: Readers should verify all CVE references directly with MITRE/NVD before operational use. This document is for educational and research purposes.**

Maciej Gojny · GG Advanced IT Security · ggsec.de

GGSEC Research · June 2026 · TLP:WHITE – Public Release

Document Version 1.2 · Last updated: June 2026

SHA-256 (v1.2): 9d08d9cd6ad525597bf5d924df50e4ffc491c1d837800532199bdf4312387f22