USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

SoK: Insecurity of Cellular Basebands

Henri Carnot, *Avantix and EURECOM;* Aurélien Francillon, *EURECOM*

https://www.usenix.org/conference/woot26/presentation/carnot

This paper is included in the Proceedings of the
20th USENIX WOOT Conference on Offensive Technologies.

August 10–11, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-57-1

Open access to the Proceedings of the
20th USENIX WOOT Conference on Offensive Technologies
is sponsored by USENIX.

SoK: Insecurity of Cellular Basebands

Henri Carnot
Avantix, EURECOM

Aurélien Francillon
EURECOM

Abstract

Cellular basebands are critical components in mobile devices, enabling connectivity with cellular networks. Their notorious complexity, proprietary design, insufficient security hardening, and support for multiple generations of complex cellular protocols make them a prime target for attacks. Although billions of devices rely on these components, and despite the many papers published in recent years, baseband security remains poorly explored. This paper provides a systematization of knowledge on baseband security, analyzing prior work on vulnerability discovery and attack surfaces. We present a taxonomy of vulnerabilities, review state-of-the-art analysis techniques, including static analysis, over-the-air testing, and emulation, and discuss their respective strengths and limitations. Finally, we outline promising research directions to address gaps in current methodologies. Our goal is to provide a comprehensive framework that guides future work and strengthens the security of cellular ecosystems.

1 Introduction

Smartphones contain dedicated chips that handle communication with cellular networks. These chips are commonly referred to as *basebands*, cellular processors, modem processors, or even chipsets. They enable the phone to connect to the mobile network, send and receive phone calls and SMS, but also handle data connectivity and manage handovers.

To enable these interactions, they support multiple generations of complex protocols. These protocols are standardized in multiple documents, written in natural language, which are sometimes imprecise or subject to interpretations [1,2]. Moreover, network standards remain in use for years after their conception, therefore, vulnerabilities can persist for years as well. For instance, most basebands still support the 2G standard, first deployed in 1991. Although 2G is being phased out, and can be manually deactivated in some devices, it is still needed. The UK, for example, scheduled phasing out 2G in 2033 [3]. Such complexity hinders the implementation of cellular logic in baseband firmware and can lead to undefined behaviors [2] potentially introducing vulnerabilities into the system [4].

Moreover, cellular protocols support numerous capabilities, resulting in a large attack surface. Part of this attack surface is accessible even before the phone authenticates with the cellular network.

This protocol complexity leads to large and sophisticated firmware, making analysis challenging. A primary difficulty arises from the substantial size of baseband firmware, for example, a firmware for Samsung's Shannon baseband chipsets can contain more than 90k functions [5]. Moreover, baseband firmware images are proprietary and lack public documentation, meaning that most insights depend on reverse engineering. This process is further impeded by the removal of debug information, as seen in Qualcomm's firmware [6].

Baseband firmware images also run on dedicated, undocumented hardware and most often rely on proprietary *Real-Time Operating System (RTOS)* that must be reversed to understand the inner workings of the firmware. Notably, both Samsung and Qualcomm basebands use their own proprietary RTOS [7, 8].

In order to investigate the security of those basebands, researchers initially applied manual reverse engineering [9–11]. However, the scarcity of external information and the sheer size of firmware images make reverse engineering both difficult and impractical at scale. Researchers have developed automated tools to ease this analysis [2,4,5,12–15].

The analysis of baseband firmware is also made harder because debugging interfaces are generally disabled on production devices [6].

However, most implementations are written in memory-unsafe languages such as C and C++, increasing the risk of memory vulnerabilities (such as buffer overflows, integer overflows, use after free or other vulnerabilities [16]).

These chips run independently of the application processor, and are less hardened than application processors [9], earlier work [17] showed lack of memory-corruption mitigations such as ASLR. Some researchers [18] recommend harden-

ing the basebands firmware or using memory-safe languages for critical components that handle untrusted data, such as parsers.

Vulnerabilities in basebands are critical as they can lead to privacy issues, the baseband having access to internal components such as the microphone [9]. Furthermore, vulnerabilities in basebands leading to crashes can endanger people relying on those devices as they can prevent accessing emergency services. Those devices can also be used in critical machines where human life is at stake, such as cars [19].

Moreover, those vulnerabilities can be exploited silently [20, 21]. Since the baseband operates independently of the main OS processor, monitoring its activity is difficult, and, to the best of our knowledge, no antivirus or intrusion detection system has been designed for basebands.

Moreover, those vulnerabilities can be exploited remotely as basebands are able to communicate Over-the-Air with base stations, thereby exposing part of their internal code to external users. Critically, part of this protocol logic is accessible without authentication. Moreover, basebands are present in many smartphones, cars, and IoT devices, yet few manufacturers produce these components. Therefore, one vulnerability in one baseband can affect a whole span of smartphones from different vendors [20, 22].

Therefore, researchers have investigated ways to detect flaws in the baseband implementations, by means of static analysis, emulation, *Over-The-Air (OTA)* testing or fuzzing. These flaws can lead to vulnerabilities such as identity leaks, *Denial-of-Service (DoS)*, message eavesdropping or *Remote Code Execution (RCE)*.

This paper does not aim to exhaustively cover all research on cellular connectivity security. Instead, our goal is to illustrate the primary approaches focused on analyzing baseband security, discuss their strengths and limitations, and outline promising directions for future work.

To conduct this research, we reviewed both academic and non-academic work. We searched for terms such as “baseband security” and “baseband fuzzing,” as well as related variants such as “cellular modem security.” We then manually examined the references and citations of the resulting papers and retained works directly relevant to baseband security analysis. To prioritize our exploration of the literature, we ranked papers according to the number of citations they received from works already selected in our corpus.

In this systematization of knowledge, we make the following contributions:

- present the fundamental challenges of analyzing cellular basebands;
- describe the baseband attack surface, describe its associated threat model and provide a taxonomy of vulnerabilities;
- review and classify the methods used to analyze this attack surface, considering both academic and non-academic publications.
- outline future research areas.

The remainder of this paper is organized as follows. Section 3 provides background on cellular protocol generations and baseband architecture, a description of the baseband’s attack surface is given in Section 4. Section 5 presents the Over-The-Air attack scenario and Section 6 presents the threat model related to this attack surface. Section 7 reviews existing analysis methodologies under the prism of test cases generation while Section 8 reviews the oracle used. Finally, Section 9 examines future work and Section 10 concludes the paper.

2 Research questions

RQ1. What defines the baseband attack surface, and which potential attacks can be derived from it?

RQ2. What threat model arises from these adversary capabilities, and what classes of vulnerabilities are known to affect basebands?

RQ3. How can security-relevant test cases be systematically generated to explore the vast input space of cellular protocols?

RQ4. How can basebands be effectively tested while accounting for the stateful nature of cellular protocols and ensuring independence across test executions?

RQ5. Given that basebands operate largely as black-box devices, how can crashes or anomalous behaviors be reliably detected during testing?

3 Background

3.1 Protocol generations

Cellular networks have evolved through several generations, from 2G to 5G. Each of the new generations provides specific security services (authentication, confidentiality, integrity and privacy). However, basebands support both old and new protocol generations. Therefore, vulnerabilities persist in old protocol components that remain reachable through downgrade or fallback mechanisms, even when operators are progressively shutting down legacy networks.

GSM (2G) Second-generation systems introduced digital modulation. Authentication in GSM is unilateral: the network authenticates the subscriber, but the *User Equipment (UE)* does not authenticate the base station. This design fails to prevent fake base station attacks. While many *Mobile Network Operators (MNO)* are announcing 2G sunsetting [23–25], 2G remains enabled in numerous networks for coverage in rural areas or for support for legacy Machine-to-Machine deployments. Although Google added an option to disable its usage in Android 14 [26], it is not supported by all phones. iPhones require Lockdown Mode [27] to disable it, which disables many other functionalities. Unfortunately, basebands will likely need to support 2G for many years to be able to be

used in rural areas or in degraded mode.

UMTS (3G) was designed to address several of GSM's security limitations. It introduced mutual authentication between the UE and the core network.

LTE (4G) moved to an all-IP packet-switched architecture with a new architecture for the core network. Voice calls and SMS are supported through *IP Multimedia System (IMS)*.

5G introduced new security mechanisms, such as downgrade prevention [28] or protection against device deanonymization. However, 5G's lower-layer capabilities introduce a new attack surface [29].

3.2 Cellular network architecture

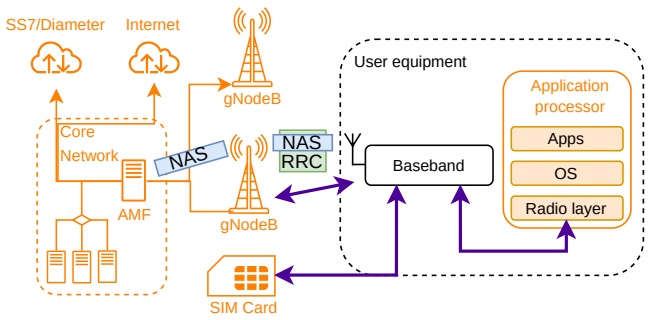

Figure 1: Overview of the baseband interactions with the SIM card, application processor and the cellular network.
In orange are the adversarial components considered in this study and in blue the interfaces that constitute the baseband's attack surface.

Basebands are embedded in a larger architecture that comprises the base station, the core network but also the SIM card and the application processor (Figure 1).

Baseband. The baseband is the target of our study, it is the phone component of a smartphone, it handles the communication with the base station and the core network to provide cellular connectivity to the Application Processor.

Base Station. The base station respectively eNodeB and gNodeB in 4G and 5G, provides the wireless link between the UE and the Core Network. It is also responsible for handling the encryption and integrity protection of the communication between the base station and the UE.

Core Network. The core network provides mandatory services such as user authentication and mobility management. In 5G, the gNodeB forwards the *Non Access Stratum (NAS)* messages (Section 4.1) to the *Access and Mobility Management Function (AMF)* server. It is a proxy between the base station and the other components of the core network. The AMF handles similar services as the *Mobility Management Entity (MME)* in 4G. The Core Network also provides the gateways to the internet and to the telephony network.

SIM card. The *Subscriber Identity Module (SIM)* allows the user to be authenticated on the cellular network. It stores sensitive information such as the authentication key or the *International Subscriber Mobile Identity (IMSI)*, making it a highly sensitive device.

Beyond its primary role, SIM cards are capable of running applications and can be updated remotely [30]. This functionality introduces an additional attack surface.

SIM cards exist in multiple form factors. They can be external physical devices inserted into the phone, or they can be embedded within the phone as the eSIM or iSIM. The eSIM is a dedicated chip on the phone board, while the iSIM is integrated directly into the SoC.

The Application Processor (AP) runs the phone's operating system. The baseband and the application processor generally communicate through AT commands [31–33]. Some basebands extend this communication with proprietary communication protocols [34]. On the Android platform, the *Radio Interface Layer (RIL)* daemon of the application processor is responsible for handling this communication [35] (Section 4.2).

4 Attack surface

RQ1: The baseband attack surface can be subdivided into three main categories, the Over-The-Air attack surface, the SIM attack surface and the application processor attack surface.

4.1 Over-The-Air attack surface

The link between the base station and the baseband constitutes a broad attack surface as it encompasses multiple generations of complex protocols. This broad attack surface can be divided into the user plane and the control plane.

User Plane. The user plane is responsible for transmitting user data, such as Internet traffic, voice calls, and SMS messages. Vulnerabilities in the baseband can lead to improper integrity protection and encryption of the user plane data, allowing adversaries to intercept or modify the traffic between the base station and the user equipment.

Control Plane. The control plane transmits the signaling information between the *User Equipment (UE)*, the base station and the core network. It is an important area of the cellular connectivity as most security aspects of the communication are handled in it such as the authentication, the configuration of the security context, handovers. For this reason, most works are focused on the security of the control plane.

For a single generation of the cellular standard, multiple protocols organized into three distinct layers work together to provide cellular connectivity (Figure 2). The architecture

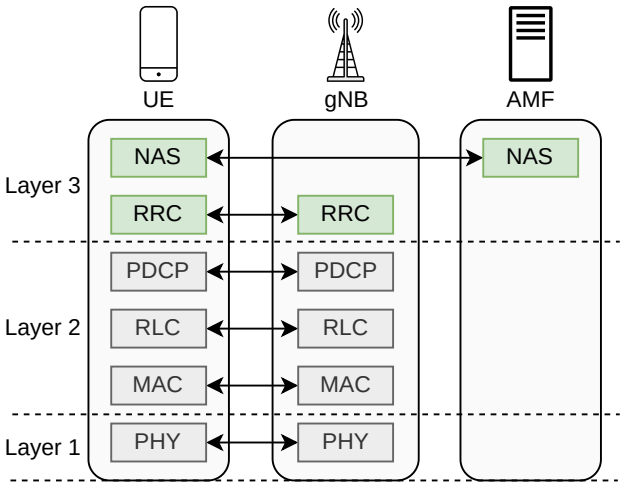

Figure 2: Organization of the cellular protocols for the 5G control plane. In green, are the layers that can be integrity protected or encrypted.

outlined here corresponds to 5G standard, but its underlying design principles remain consistent with those of earlier generations.

The first layer is the physical layer; it performs the wireless signal processing.

Layer 2 comprises *Packet Data Convergence Protocol (PDCP)*, the *Radio Link Control (RLC)*, and *Medium Access Control (MAC)* sublayers. PDCP provides encryption and integrity protection to both the control-plane and user-plane data, ensuring confidentiality and preventing tampering during communication between the UE and the base station. RLC performs packet segmentation, reassembly, retransmission, and sequence control. MAC coordinates access to the physical transmission channel, ensuring optimal use of the radio spectrum. Those lower layers are neither protected nor encrypted and expose a broad range of functionalities that can be exploited to trigger crashes or memory vulnerabilities in the basebands leading to potential remote code execution while communicating with a non-authenticated peer [17, 36, 37].

Layer 3 consists of the *Non Access Stratum (NAS)* and *Radio Resource Control (RRC)* layers. The NAS sublayer manages mobility and session control, acting as a high-level signaling interface between the UE and the core network. It provides mutual authentication between the UE and the core network. It maintains its own security context and enforces integrity protection and encryption for NAS messages.

The communication between the UE and the base station happens through the RRC sublayer. It enables control of the radio resources, and establishes its own security context after the authentication and establishment of a secure connection between the UE and the core network through the NAS layer.

For testing purposes, specifications allow null ciphering and null integrity protection. However, null integrity can't be used except when unauthenticated emergency calls are required [28] or for some specific RRC commands [38].

Moreover, the DoLTEst paper [4] defines several authentication states which leads to separate attack surfaces as depending on the authentication, the baseband will accept or not unprotected messages.

Cellular connectivity exposes a broad attack surface, enabling adversaries to operate at various proximities, from nearby radio attackers to remote Internet-based ones. In this work, we exclude Internet-based adversaries, as IP-layer traffic is handled by the application processor rather than the baseband.

Passive Radio Attacker. The passive radio attacker listens to messages exchanged Over-the-Air to extract information without transmitting any signals.

Unauthenticated Radio Attacker. This is the most common attacker model. The adversary does not possess the encryption keys and, assuming cryptographic primitives are correctly implemented, cannot impersonate a legitimate base station to exploit vulnerabilities in the authenticated attack surface of the protocol. Nevertheless, such attackers can still exploit vulnerabilities in the pre-authenticated attack surface. This model encompasses the fake base station, man-in-the-middle, and signal injection attacks (Section 5).

Adversarial Mobile Network Operator (MNO). In the case where the attacker knows the cryptographic keys, it can exploit both pre-authentication and post-authentication attack surfaces. Such scenarios are relevant when considering untrusted network operators, when states actors target civilians [39], or when the operator has been compromised.

Adversarial UE. An attacker can use another UE to send messages to a victim UE, for example, sending multiple silent SMS messages [40]. Those SMSes won't trigger a notification on the targeted device and can be used to track a user's location by detecting a specific pattern when sniffing cellular communications in an area [41].

4.2 Application processor attack surface

When considering the baseband's attack surface, the *Application Processor* (AP) needs to be taken into account. AT commands are used to communicate with the baseband. Some vendors also implement custom protocol, such as Qualcomm QMI [42].

On some devices, the AT interface has been accessible via USB in the past. Previous work showed that this interface could be abused to enable USB ADB debug mode despite the lock screen [31].

Malicious Application Processor. Another adversary model to consider is that of a compromised or malicious application processor, in which a malicious or exploited application attempts to interact with and abuse the baseband interface.

Such an attacker may seek to trigger unintended behavior or escalate privileges within the baseband.

4.3 SIM protocol attack surface

SIM cards are important components, usually considered as trusted, that allow the baseband to authenticate itself to the network. Previous works demonstrated that SIM vulnerabilities could have important impact on the security of the UE such as a lock pin bypass due to invalid SIM code handling [**?**].

Adversarial SIM card. Emulating the interaction between the SIM card and the baseband will allow deeper states when doing full system fuzzing. SIMurai [43] showed that this emulation is possible, and that vulnerabilities found by doing fuzzing on the emulated firmware can be confirmed by using a SIM interposer on a real device. These findings highlight the need to consider an attacker model in which the SIM card itself is malicious.

Moreover, eSIM expands the attack surface by introducing a remote SIM provisioning stack, including the *Embedded Universal Integrated Circuit Card (eUICC)* and profile-management infrastructure [44]. For example, recent work showed that vulnerabilities in eUICC implementations can enable eSIM cloning in practice, allowing calls, SMS messages, and one-time codes to be redirected to a cloned device [45].

5 Over-the-Air Attack scenario

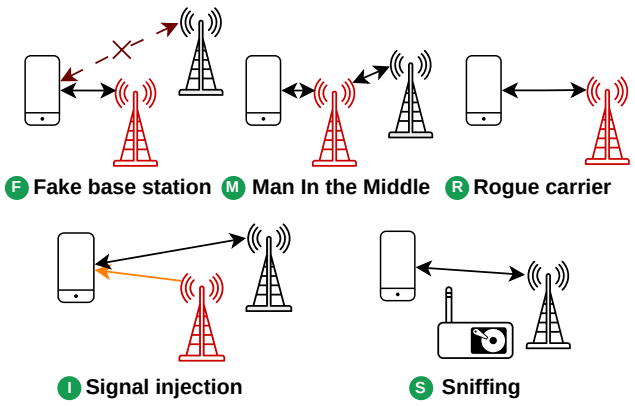

Figure 3: Attack scenarios.

Several attack scenarios exist when analyzing the security connection between the baseband and the base station (Figure 3): Man-In-the-Middle, Fake base station, rogue carrier, signal injection, sniffing and downgrade attacks.

F **Fake Base Station (FBS).** In the fake base station attack [46–49], the adversary deploys a rogue base station that impersonates a legitimate one. Because the attacker does not possess the cryptographic keys of the genuine network, a correctly implemented baseband should prevent the adversary from accessing or influencing any post-authentication protocol surface. Executing such an attack requires only commodity equipment, typically a laptop and a *Software Defined Radio (SDR)*, making this threat model accessible not only to state-level actors but also to adversaries with comparatively limited resources.

M **Man-In-the-Middle (MitM).** In a Man-In-the-Middle attack [50], the attacker will use a fake base station with the name of a legitimate base station. In this model, the attacker does not possess the cryptographic keys required to authenticate the messages as coming from a legitimate base station or core network. However, the attacker can forward messages to the legitimate base station. In this model, the attacker can read, modify, drop, inject or replay messages sent between the UE and the legitimate base station. However, integrity protection should allow detecting modifications of the protected messages. However, some attacks have been discovered to bypass this protection (Section 6.1).

R **Rogue Carrier.** In the case of a rogue carrier [51], the attacker has access to cryptographic keys allowing to create a valid base station and core network. This scenario exposes the broadest attack surface as the baseband will accept messages associated to both the pre-authentication and post-authentication protocol surface.

I **Signal injection.** Signal injection attacks [52–54], involve crafting precisely timed and modulated radio signals that are injected into the ongoing communication between the victim UE and a legitimate base station. Unlike fake base station attacks, which require deploying an impersonating base station that can often be detected through network-side monitoring, signal injection attacks are significantly harder to detect as they do not require establishing a rogue cell. Instead, the attacker exploits a legitimate base station, however, this technique cannot guarantee full success as the adversarial signal competes with the legitimate transmission. To improve the probability of successful injection, the attackers can jam the legitimate network and repeat the injected messages multiple times [54]. Additionally, distance can reduce the effectiveness of this method. Experiments in SNI5GECT [54], achieved an 83% success rate at a distance of 20 meters from the UE. Finally, those attacks might be complex to conduct as after the injection the victim UE might return an invalid response which can be rejected by the legitimate base station.

S **Sniffing.** Sniffing represents a comparatively less powerful attack vector [54, 55], yet it merits consideration as inadequate mitigation can enable adversaries to capture messages or derive device or message specific fingerprints. In certain scenarios, vulnerabilities allow traffic to be transmitted without encryption, leaving the baseband vulnerable to sniffing attacks [56].

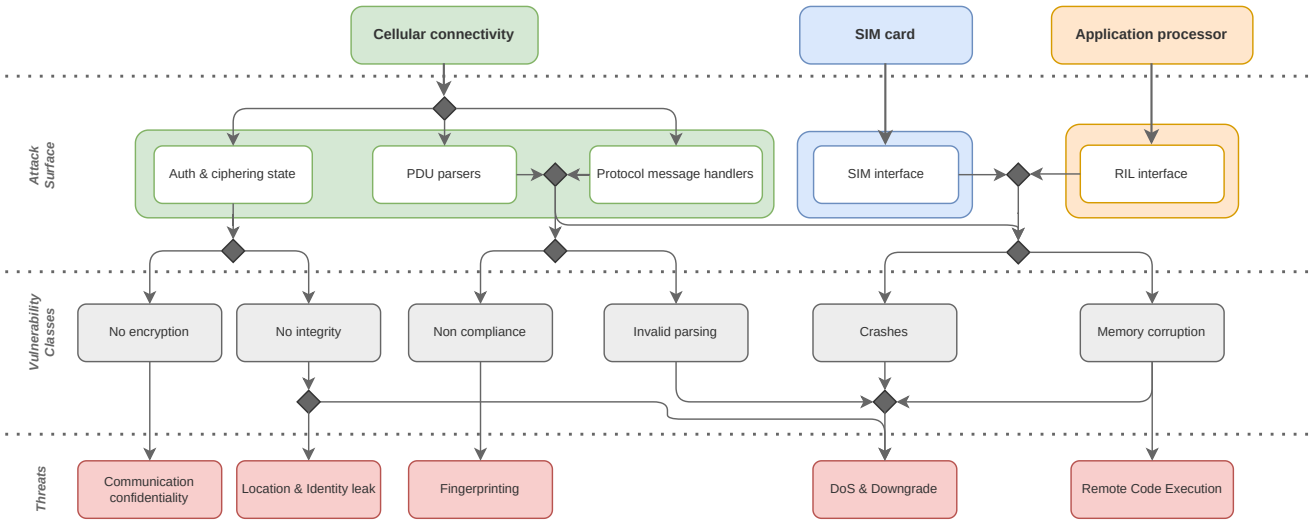

Figure 4: Taxonomy of basebands attack surface.

6 Threat model

RQ2: Vulnerabilities in cellular communication can be divided between protocol vulnerabilities contained in the specification and implementation vulnerabilities in network devices such as the baseband, the base station, or the core network (Figure 4).

Protocol vulnerabilities. Vulnerabilities in the specification are the most critical as they affect all devices and require a new version of the protocol to fix. Such flaws can lead to deanonymization, fingerprinting, and impersonation. Some work [1, 50, 57–60] has analyzed the protocol to identify vulnerabilities in the specification. Although vulnerabilities arising from flaws in the specification are significant, they are not the focus of this work which concentrates instead on implementation errors in the baseband itself. Nevertheless, the analysis of such implementation defects can occasionally uncover previously unrecognized vulnerabilities or ambiguities in the specification [2].

Implementation errors. Errors can occur when implementing the specification. They can lead to (i) vulnerabilities or non-conformance with the specification (ii) crashes due to memory corruptions or reachable assertions.

(i) Detecting protocol non-conformance is hard as unlike memory vulnerabilities, detection oracles need to be especially crafted to verify a specific type of non-conformance. Logical bugs related to cellular message handling can be split into three categories [4], issues in the integrity and encryption handling of cellular messages, flaws in generic parsers or flaws in specific message handlers.

(ii) Crashes or memory vulnerabilities have been found in basebands from MediaTek [51, 61, 62], UNISOC [63], Huawei [64], Qualcomm [65, 66] and Samsung [11] basebands.

Protocol vulnerabilities have already been surveyed in prior work [67, 68]. Similarly, the security of core network has been examined in earlier studies [69, 70], and broader analysis of mobile network security has been provided by Rupprecht et al. [71].

In contrast, this work focuses specifically on implementation-level vulnerabilities within baseband firmware.

6.1 Integrity

When analyzing the baseband, the integrity can apply to both the communication integrity or the baseband integrity.

Communication integrity. Once the security context of RRC and NAS layers is complete, both RRC and NAS signaling should be integrity protected. Integrity protection ensures that the baseband can verify the origin of messages and reject any that do not originate from a legitimate base station and legitimate core network, thereby mitigating man-in-the-middle and signal injection attacks.

When integrity checks are absent or incorrectly enforced on received packets, an attacker may exploit this weakness by injecting crafted messages, thereby inducing the baseband to process inputs that should be accepted only under a valid security context.

This can also lead to privacy issues. For example, this may allow sending SMS phishing messages.

In 4G/5G, SMS messages can be sent through the user plane over IP to the IMS server [72] or over NAS in the control plane [73]. DoLTEst [4] found one vulnerability in the integrity protection in the RRC layer and one in the NAS layer which can be exploited to inject an SMS message over NAS

signaling which will be accepted despite the invalid NAS and RRC security context. By exploiting a vulnerability in the NAS *Authentication and Key Agreement (AKA)* procedure BASECOMP [13] authors and 5GBaseChecker [15] authors managed to reproduce this attack using other vulnerabilities allowing to bypass the integrity verification. Those vulnerabilities enable an attacker to send phishing SMS using a false base station, MitM or signal injection attacks.

Additionally, invalid integrity can allow an attacker to control information displayed to the user. For example, by modifying the *EMM Information* message, the attacker can change the UE time by sending invalid time or the attacker can also control the displayed network name [4, 13].

Baseband's integrity. Memory corruption vulnerabilities in baseband firmware can escalate beyond simple crashes to enable *Remote Code Execution (RCE)*. This grants attackers arbitrary control over the modem processor. This threat has been demonstrated in several research works [9, 17, 74, 75] and a public report [76].

6.2 Confidentiality

User confidentiality. As stated in 6.1 integrity vulnerabilities can allow a device to send messages belonging to the authenticated attack surface despite not being authenticated. For example, an attacker could send a *RRCConnectionReconfiguration* with the *MeasConfig* IE which would leak signal measurement information and therefore help to fingerprint or locate the UE [4]. Moreover, in 4G, such a vulnerability could be used to deanonymize the UE by sending a *NAS Identity Request* which would leak the *International Mobile Subscriber Identity (IMSI)* and *International Mobile Equipment Identity (IMEI)* revealing both the permanent subscriber identity stored in the SIM and the phone identifier.

To mitigate user tracking, 5G introduces a new identity management scheme that replaces the IMSI. The *Subscriber Permanent Identifier (SUPI)* is never transmitted in clear, instead, it is used to derive a dynamic, encrypted identifier known as the *Subscriber Concealed Identifier (SUCI)*. This mechanism significantly reduces the exposure of permanent identifiers over the air. Nevertheless, research has shown that encrypted SUCIs do not fully eliminate presence or linkability-based tracking attacks. Chlosta et al. [77] demonstrate that so-called 5G SUCI-catcher can still infer whether a particular subscriber is present in a given area by exploiting linkability through the 5G AKA procedure.

Additionally, device's capabilities or implementation characteristics can be used to fingerprint the device [4], without requiring access to the post-authentication attack surface.

Data confidentiality. The confidentiality of the user plane packets between the UE and the base station such as the SMS over IP or Internet traffic is ensured by the PDCP protocol. Previous work showed that some vulnerabilities could allow an attacker to disable the integrity and ciphering protections [4], this issue is especially critical when the UE does not alert the user that the connection lacks encryption [56]. Likewise, the authors of 5GBaseChecker [15] demonstrated that exploiting the integrity bypass (Section 6.1) enables attackers to read Internet traffic exchanged between the UE and the base station.

Baseband confidentiality. Certain vulnerabilities in the baseband can lead to unintended information leakage. For example, Klischies et al. [2] discovered a memory vulnerability in the *Public Warning System (PWS)* that permits out-of-bound reads. The authors argue that, under the right conditions, this flaw could expose sensitive data from memory, including PDCP session encryption keys.

6.3 Availability

Denial-of-Service. Previous security analyses have shown that basebands can contain NAS or RRC logic flaws that cause crashes when processing malformed messages. When this occurs, the device loses its ability to communicate with the base station. In production environments, basebands typically reboot after a crash [2]. However, this recovery process introduces a delay, which can be exploited for persistent disruption. Such crashes are often the result of invalid memory accesses or reachable assertions [78] in the firmware implementation [2, 4, 62, 79].

Denial-of-service can also occur without crashes. For example, the DIKEUE [14] tool identified a vulnerability in the 4G protocol implementation for some devices related to the *Globally Unique Temporary Identifier (GUTI)* relocation leading to a denial-of-service. The GUTI, introduced in 4G, is a temporary identifier to replace the IMSI to reduce its exposure. By doing a MitM relay, an attacker could trick the UE into not changing its GUTI by dropping a *GUTI_reallocation* message from the MME. The device then will not respond to the network sending messages with the new GUTI as the device will have kept the old identifier, effectively disrupting the communication.

Inter-generation downgrade. When a baseband detects a base station but fails to establish a connection, it may attempt to connect to another available base station operating on an older protocol version. This fallback behavior can be exploited to perform downgrade attacks [62, 80]. In such attacks, the adversary will deliberately jam a protocol, or exploit a vulnerability in the baseband firmware, to force the device to revert to a legacy standard that may contain well known vulnerabilities.

Existing research has demonstrated several downgrade vectors. For instance 5GHoul [62] describes an attack that floods the UE with *Deregistration Accept* messages exhausting the UE handling resources and leading to a downgrade. This technique requires modifying the base station software to send arbitrary retry messages. Another vulnerability identified by the same work involves crashing the baseband's 5G handler,

which coerces the device into downgrading to 4G or even 3G networks.

Additionally, integrity failures can also facilitate downgrades. For example, an attacker sending an *RRC Release* message can force the UE to reconnect using another protocol version [15], effectively triggering a downgrade.

7 Test case generation, exploration strategy

RQ3, RQ4: To analyze the security of baseband firmware, multiple methods have been used: (i) use the specification to extract security assertions and use static analysis to verify them [5, 13, 21] (Section 7.1) (ii) generating test cases and validating OTA [4, 14, 54, 80–82] (Section 7.2.1) (iii) relying on existing network stack to generate messages and apply mutations to produce malformed or adversarial inputs [36, 46, 51, 62, 83, 84] (Section 7.2.3) (iv) letting the fuzzer generate the test cases [7, 12, 85–87] (Section 7.2.4).

For complex, multi-layered protocols such as those used in cellular networks, manual message crafting is particularly challenging due to the large number of message types and intricate state dependencies. An alternative approach consists of inserting mutation hooks within the mobile network implementation, enabling selective modification of messages before they are transmitted to the UE. Figure 5 provides an overview of the available analysis methods, and Table 1 maps each surveyed paper to the techniques it employs.

7.1 Static analysis

Static analysis examines the baseband firmware without executing it. In its simplest form, researchers manually triage binary images in a disassembler to locate security-critical components such as message parsers and common decoding routines, and then audit these code paths for memory-safety vulnerabilities and logic flaws [9]. This approach requires access to the firmware binary and adequate tooling for the target CPU architecture. Moreover, static analyzers often need additional preprocessing to make the firmware analyzable, because some vendors implement non-standard boot-time loading behaviors. For example, Samsung's Shannon basebands perform a scatter-loading, in which several custom loading steps are done during the firmware initialization. Those steps include copying code and data into their designated memory regions and handling compressed sections before execution can begin. These behaviors must be reconstructed by static analysis tools to obtain a correct memory layout and enable accurate disassembly [5]. Static analysis has several well-known limitations. It often struggles with dynamic control flow related to indirect calls, function pointers, and global state. Furthermore, it may generate false positives that necessitate manual verification.

Taint-based identification of sensitive functions. *Taint analysis* tracks values originating from specific inputs as they propagate through the program to functions of interest. This approach enables the identification of cases in which data derived from input messages is used in memory-sensitive operations. For instance, BVFinder [21] employs taint analysis to detect memory vulnerabilities in message-specific handlers. The critical memory operations detected include functions such as *memcpy*, where the size parameter is controlled by the input. A second category of sensitive operations involves loops whose iteration count is influenced by the input. Invalid iteration count can result in out-of-bounds access. However, these handcrafted detectors cannot perform more granular checks on variables and may therefore produce false positives. Consequently, manual review of the generated reports is necessary. Moreover, similarly to the other static analysis tools, this approach requires manual implementation of mechanisms to resolve indirect function calls.

Parsers functions analysis. Baseband firmware relies on dedicated functions to decode binary-encoded messages and forward their content to the appropriate message handlers. BaseSpec [5] is predicated on the premise that implementations of Layer 3 message parsers adhere closely to the protocol specification. Leveraging this insight, the authors systematically identify decoding functions and extract their mapping logic to verify that all expected message types are present. They also search for decoders that handle message types not defined in the specification, as they could indicate potential security vulnerability. The extraction of the message structures from the binary is performed manually, which is labor-intensive. However, because the decoding logic tends to remain consistent across different baseband models from the same vendor, this approach should be largely reusable with minimal modifications within the same vendor. Despite this advantage, this approach has only been used for a single vendor because porting this implementation to a different baseband vendor requires a significant amount of work.

Integrity function analysis. BASECOMP [13] examines the correctness of integrity enforcement mechanisms within the baseband firmware. Leveraging symbolic analysis, the authors derive the constraints that the firmware applies to incoming messages and systematically compare these constraints against the specification. This methodology enables the identification of cases in which messages are erroneously accepted without the required integrity protection.

Functions identification. To apply static analysis techniques across multiple baseband images, analysts must first identify the functions relevant to message parsing and security-critical logic. BVFinder [21] treats this challenge as out of scope assuming that the addresses of message-specific handlers are already provided through external means. BaseSpec [5], in contrast, locates functions by matching function signatures associated with scatter-loading behaviors and by applying rules to identify decoder functions while BASECOMP [13] uses a custom method to identify integrity-protection functions.

Application processor interface. In BaseMirror, Li et al. [35]

Technique	Year	Paper	Attack surface	Non compliance detection	Crash				
					Log	Passive probing	Active probing	Emulator crash	Sanitizer
Static analysis	2021	BaseSpec [5]	Messages decoder (L3)	●	○	○	○	○	○
	2023	BASECOMP [13]	Integrity functions NAS (4G)	●	○	○	○	○	○
	2024	BVFinder [21]	Messages specific handlers	●	○	○	○	○	○
OTA testing	2021	DIKEUE [14]	4G control-plane	●	○	○	○	○	○
	2024	5GBaseChecker [15]	5G control-plane	●	○	○	○	○	○
	2022	DoLTEst [4]	4G control-plane	●	○	○	○	○	○
	2023	Instruction unclear [2]	RRC, PWS, SMS (4G)	●	●	○	○	○	○
	2022	Garbelini et al. [88]	MAC, RLC, PDCP, RRC, NAS (4G&5G)	●	○	●	○	○	○
	2023	UE Security Reloaded [80]	RRC, NAS (5G)	●	○	○	○	○	○
	2023	Contester [82]	NAS (4G)	●	○	○	○	○	○
	2024	ASTRA-5G [81]	NAS (5G)	●	○	○	○	○	○
	2025	SNI5GECT [54]	RRC, NAS (5G)	●	●	○	○	○	○
OTA Fuzzing	2019	LTEFuzz [46]	RRC, NAS (4G)	●	○	○	○	○	○
	2024	UFuzz [83]	RRC, NAS (5G)	○	●	●	○	○	○
	2024	CovFUZZ [84]	RRC, NAS (4G&5G)	○	○	●	○	○	○
	2025	5GHoul [62]	MAC, RLC, PDCP, RRC, NAS (5G)	○	●	○	○	○	○
	2025	OTABase [51]	RRC, NAS (4G&5G)	○	●	●	●	○	○
	2025	LLFUZZ [36]	PHY, MAC, RLC, PDCP (4G)	○	●	○	○	○	○
Emulation fuzzing	2020	BaseSAFE [12]	NAS, RRC (4G)	○	○	○	○	●	●
	2022	FIRMWIRE [85]	RRC, NAS (4G), SM, CC (2G)	○	○	○	○	●	○
	2025	FIRMSTATE [86]	RRC, NAS (4G)	○	○	○	○	●	○
	2025	BASEBRIDGE [87]	RRC, NAS (4G)	○	○	○	○	●	○
	2025	LORIS [7]	NAS (4G&5G)	○	○	○	○	●	●
	2025	Hexagon fuzz [89]		○	○	○	○	●	○
	2024	SIMURAI [43]	SIM interface	○	○	○	○	●	○
	2024	BaseMirror [35]	RIL	○	○	○	●	○	○

Table 1: Overview of Analysis Approaches and Targets in the Surveyed Literature.

used taint analysis to uncover undocumented proprietary commands in the *Radio Interface Layer (RIL)*. The RIL is an abstraction layer used by Android to provide a unified interface to communicate with the baseband. Some commands can be sent even by non-root applications.

7.2 Dynamic analysis

The baseband attack surface is composed of multiple protocols, such as the cellular protocol, some of which expose vast input surface. Covering this vast attack surface is challenging. Also, the large number of states in the cellular state machine makes it difficult to reach deep states and detect improper state transitions. Inputs must be sufficiently valid to avoid early rejection, yet sufficiently invalid to trigger vulnerabilities, while avoiding requiring a reset of the baseband state, which slows the testing process. However, avoiding resets increases the risk to obtain non-reproducible crashes. To mitigate the state reset issue, many researchers take advantage of emulation. By emulating the firmware, they can snapshot and restore it to the state prior to input injection.

7.2.1 OTA testing

OTA testing evaluates UE by interacting with them through standard-compliant cellular protocols. This method involves sending crafted messages over the air interface to probe protocol behavior and identify flaws. OTA analysis is highly generalizable, enabling the assessment of diverse devices regardless of baseband implementation [9, 11, 36, 51, 54, 62, 84, 90]. Nevertheless, OTA testing suffers from practical limitations: tests are slow, costly to scale as one physical UE is required per

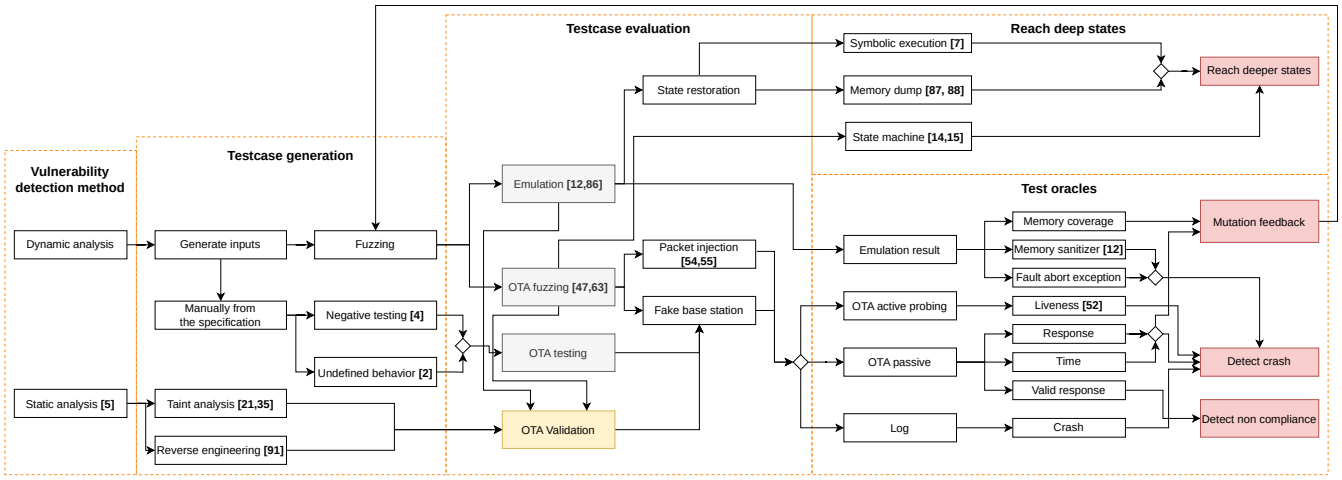

Figure 5: Taxonomy of baseband vulnerability analysis techniques.

testing process. Thus, each test case should be optimized to maximize the likelihood of uncovering a bug and some papers make it a requirement when building their tools [4].

Specification based test. The 3GPP defines conformance test suites for UE behavior and protocol compliance, but these are not designed to extensively test UE security. To solve this issue Contester [82] uses natural language processing to analyze the specification in order to build a set of test cases targeted to verify security assertions.

Negative testing or undefined behavior. Conformance testing typically focuses on positive testing, i.e., verifying that a device responds correctly to well-formed input and expected states. Negative testing complements this by checking that the device correctly rejects messages it should never accept: messages sent out of sequence, without the proper security context. When such messages are nevertheless accepted, this indicates an implementation error or, in some cases an underspecification in the standard itself. DoLTEst [4] explores this approach for cellular protocols targeting the authentication boundary for 4G. Specifically, the study tests whether messages that should be integrity protected are rejected when sent without authentication or integrity protection. To achieve this, the specification is manually analyzed, and for each message, the conditions under which it should not be accepted are described. This collection of rules is then used to generate test cases. To model the connection state, only the authentication state is considered, approximated using four discrete states. Klischies et al. [2] further examine this method by modeling the *Public Warning System (PWS)*, along with the SMS and RRC specifications. This model is then provided to a model checker to identify undefined states within the specification and to generate test cases aimed at exploring those states.

Sending the inputs. Both OTA testing and OTA fuzzing require a controllable base station capable of injecting arbitrary test messages. Most works rely on open-source base station and core network software for this purpose [36, 36, 46, 51, 62, 80–82, 84, 91], which additionally allows placing the UE in the protocol state required to receive target messages.

Moreover, the OTA testing process is slow because for each test case, the testing system must wait a few seconds to verify whether the test case was accepted or silently ignored [4].

Run independence. To make sure that the runs are independent, the UE state needs to be reset between the runs. For example, Klischies et al. [2] toggle the airplane mode on and off to reset the RRC state after each run. However, this method slows down the OTA testing process significantly. In contrast, DoLTEst [4] tracks the exchanged messages affecting the device's security state, allowing for the connection state to be maintained without the need to toggle airplane mode.

7.2.2 Deviant behavior

Instead of manual parsing the cellular specifications to derive a protocol model, researchers have leveraged the observation that baseband firmware implementations must ultimately conform to the same underlying standards when providing cellular functionality. Consequently, two different basebands are expected to exhibit similar behavior when subject to the same sequence of inputs. Building on this insight, DIKEUE [14] and 5GBaseChecker [15] automatically extract protocol state machines from multiple devices and compare them to identify behavioral discrepancies. Such discrepancies may indicate implementation errors or uncover ambiguities in the protocol specification.

FSM extraction. Extracting the protocol FSM from a baseband is challenging. A passive approach can be used, where the communication between a device and a base station is recorded. However, such traces are typically incomplete,

since only a subset of the reachable protocol paths is explored during normal operation. To overcome this limitation, DIKEUE [14] employs active FSM learning in which crafted queries are actively sent to explore previously unobserved transitions. This leads to better coverage but may require many queries and therefore substantial execution time. To mitigate this cost, 5GBaseChecker [15] introduces reuse of previously discovered knowledge. The tool leverages information extracted from one device's FSM to reduce the number of learning queries required on subsequent devices.

7.2.3 OTA fuzzing

An alternative approach to assessing firmware security is through fuzzing. This technique involves generating a variety of inputs and transmitting them to a device, continuing the process until a crash is detected. When applied to cellular protocols, researchers typically modify inputs originating from cellular stacks to create the test cases. Those alterations can be both message mutations and message replay [62, 84]. To improve the fuzzing process LLFuzz [36] uses the cellular specification to create standard compliant packets and focus mutations on security sensitive fields.

To enhance the efficiency of fuzzing, feedback mechanisms like code coverage can guide the fuzzer toward valuable inputs aligned with specific objectives. However, in the context of Over-The-Air fuzzing for basebands, coverage information is often unavailable. As a result, researchers must rely on alternative metrics. To overcome this challenge, 5GHoul [62] and UFuzz [83] build a state machine from live or recorded interaction between the UE and the network. This state machine is used to guide the fuzzing process towards maximizing the device transitions. UFuzz additionally considers the time elapsed before receiving a response as a feedback metric.

Diversity of Mutation Targets. Another challenge is ensuring that mutations are distributed across various layers and fields, thereby facilitating comprehensive testing. To address this concern, LLFuzz [36] enhances 5GHoul by promoting greater diversity in test cases at lower layers.

Generating Unsupported Messages. Both commercial and open-source base station stacks have inherent limitations, as they do not encompass all the features detailed in the standards. Consequently, researchers must devise strategies to handle unimplemented messages and protocol states [36], and modify base station software to transmit custom, invalid packets [51]. LLFuzz [36] enhances the RRC configuration message by analyzing specific lower-layer configurations that are not supported by srsRAN.

Mutations strategies. To avoid test cases being rejected early, mutations should be applied carefully to avoid producing messages that parsers will not accept. For example, LLFuzz [36] preserves essential fields such as LCID or RLC SN when fuzzing lower layers, ensuring that mutated messages remain associated with valid communication channels, unlike 5GHoul [62]. OTABase [51] tries to optimize the exploration process by generating messages that respect the structure. It also modifies security headers to avoid the early rejection problem.

Finally, fuzzing lower layers (e.g., MAC) is particularly challenging because they cannot tolerate delays greater than the slot transmission time (~1 ms) [62, 88].

Research recommendation: When doing fuzzing, mutation should be applied on fields which won't lead to the message being early rejected.

Research suggestion: How can protocol features that are not implemented in existing open-source cellular network platforms be systematically evaluated?

Application processor interface. Fuzzing has also been applied to assess the security of the application processor interface. A first study [33] injected a layer between the telephony stack on the application processor and the baseband, using this they analyzed the SMS handling of the application processor.

The baseband can also be fuzzed from the application processor side, Karim et al. [32] fuzzed the baseband's AT interface by connecting to it through USB or BlueTooth connection. For this, they created a custom grammar by combining AT command from the specification with device specific AT commands extracted from the firmware by the help of heuristics and regular expressions. To assess the security implication of those commands, they sent individually all the commands identified, and manually observed their impact.

7.2.4 Emulation fuzzing

To overcome the challenges of limited speed and lack of introspection in OTA testing, researchers explored how baseband firmware could be fuzzed in an emulator [7, 12, 85–87]. This approach provides a controlled environment that enables significantly higher fuzzing speed (e.g., 100x to 10,000x speed up compared to OTA testing [87]). Such a speed up is possible thanks to the parallelization of the execution but also thanks to the possibility to snapshot the emulation state before sending the input and restoring it afterward. In addition to the snapshotting capability, this method grants higher control over the execution allowing to pause the emulation or modify some registers.

Moreover, firmware emulation enables powerful instrumentation to collect fine-grained behavioral feedback (e.g., coverage and crash signals) for each input which can be used to guide mutators towards unexplored code paths. It also enables researchers to use more precise observers to detect crashes more accurately and detect vulnerabilities using crash oracles such as sanitizers [12].

Supporting additional architectures often requires substantial amount of manual work. Beyond obtaining and unpacking

the firmware, the emulator must implement the target *Instruction Set Architecture (ISA)* and mimic hardware behavior and peripheral interactions accurately enough to let the emulation proceed further in the firmware. Emulating the peripherals [92], requires significant engineering effort and it may be impossible to perfectly rehost the firmware. It is therefore likely impossible to achieve the same fidelity as with Over-the-Air testing, in particular as those baseband's processors documentation is not available to third parties.

Even when (or if) full-system emulation is possible, vulnerability validation and triage typically still require a physical device to confirm that the issue manifests under real hardware and peripheral conditions (i.e., that the vulnerability is reproducible and not an emulation artifact).

Depending on the level of emulation fidelity required, firmware can be emulated using two approaches: *partial emulation* or *full-system emulation.*

Partial emulation. In partial emulation, only a subset of the firmware is executed. For example, BaseSAFE [12] focuses exclusively on fuzzing the NAS and RRC parsers of the LTE protocol. This approach is suitable when the function under test does not rely on external state or peripherals, or when such interactions can be bypassed. By inserting hooks, functions interacting with these peripherals can be skipped. However, skipping peripheral interactions may hinder access to deeper parts of the function under test, only reachable with proper peripheral emulation.

Full-system emulation. To support fuzzing of tasks that involve interactions with global state or peripherals, full-system emulation can be employed. Certain variables, memory layouts, or function pointers are initialized during the boot process. In the Shannon firmware, some memory zones are uncompressed at boot during the scatter loading [5]. This requirement introduces additional complexity for partial emulation or static analysis approaches, as resolving these dependencies must be done manually. Starting emulation at the reset vector mitigates this issue; however, it requires handling the interactions with the rest of the peripherals.

Recent emulation based research has concentrated on platforms well-supported by emulators notably MediaTek and Samsung's Shannon basebands while Qualcomm modems have long remained harder to study in practice. Qualcomm's basebands rely on the proprietary Hexagon ISA [90], and mainstream QEMU only used to support user mode emulation. However, Qualcomm recently introduced support for full-system emulation for the Hexagon instruction set in its QEMU fork [93, 94]. By integrating this fork with the QEMU LibAFL fork [95], Produit et al. developed the first fuzzer targeting the Hexagon architecture [89] (although not rehosting an actual baseband SoC).

A key challenge in fuzzing emulated firmware is delivering test messages to the intended protocol parsers. To address this, FirmWire [85] injects a dedicated fuzzing task into the baseband RTOS that obtains inputs from the fuzzer and forwards them to the target parser task via the firmware's native task communication mechanism.

However, because test inputs are injected via internal RTOS entry points rather than through an emulated RF stack, the emulated baseband typically corresponds to a freshly booted device that has not yet attached to any cellular network. Therefore, connection-dependent protocol state remains uninitialized, many messages are silently discarded which limits a deep exploration of the protocol. To recover realistic protocol state, researchers can extract it from physical devices and transplant it into the emulator. FIRMSTATE [86] semi-automatically extracts state information from real phones during actual network communication by using vendor-specific tools to trigger a crash dump and using Shannon's *Back Trace Log(BTL)* to recover control flow and locate state-relevant data structures inside the dump, which are loaded into the FirmWire snapshot to enable state-aware emulation. BASEBRIDGE [87] generalizes state transfer to both Shannon and Mediatek by restoring selected states from a real device. To do so, a clean emulation snapshot is used to trace the firmware's memory accesses. Memory chunks are restored iteratively if they allow reaching new basic blocks, lead to new log messages or prevent a crash.

LORIS [7] performs automatic synthesis of state variables using symbolic execution. Unlike FIRMSTATE and BASEBRIDGE, it does not require a crash dump from a real device. However, symbolic execution does not scale to full protocol state coverage.

Research suggestion: Full-system emulation is complex, new research should support new architectures such as Hexagon and reduce the complexity of supporting new phone models and peripherals emulation.

8 Crash detection, impact analysis, response validation

RQ5: The current baseband research mainly focuses on two outcomes for the test case output: whether the input triggers a non-compliant response or whether the input triggers a crash.

Non-compliance detection. To detect non-compliance Garbelini et al. [88] builds a state machine during normal interaction between the UE and the network. This state machine is then used during the fuzzing stage to detect any deviation indicating a corruption of the baseband state. DoLTEst [4] iteratively builds oracles to analyze the baseband response to the test inputs. Since the messages are constructed for negative testing, no messages are expected, while positive responses indicate either an implementation error or a specification exception. In addition to detecting crashes from the specification, Klischies et al. [2] manually inspect traffic logs to identify problematic behavior.

Crash detection. Device log can be used to extract crash-related information by targeting crash related strings [2,62,83]. However, crash logs can vary from one manufacturer to another, making a first initialization step necessary [36]. Some devices, such as Apple's phones, lack such a debug mechanism or require jailbroken devices [84]. This limitation currently hinders the use of this technique for analysis. According to the authors of OTABase [51], this method does not capture all baseband crashes. Therefore, in addition to analyzing ADB logs, they rely on passive listening to the device's RLC responses to confirm that the test case was acknowledged. Furthermore, they implement active probing every ten messages by sending RRC or NAS messages. Additionally, utilizing the manufacturer debug mode can help validate crashes.

Crash detection can also be performed passively, by monitoring device responsiveness, prolonged timeouts or a lack of expected replies can indicate that the baseband has entered a hung or stalled state [88]. However, detecting memory corruptions in baseband firmware suffer from the classical issue that a memory corruption in embedded systems can be silent [96].

Crash detection in emulation. Emulation based fuzzing finds vulnerability by relying on reachable assertions already present in the firmware or fault exception vector being run due to an emulation fault [7, 85–87]. In order to detect memory corruptions more accurately, memory sanitizer can be used [7, 12]. Instrumentation of the execution is also possible to achieve better corruption detection.

Research suggestion: How can an oracle be built to detect protocol implementation vulnerabilities such as protocol downgrade when doing emulation?

Crash validation. Validation of the bug requires OTA testing to make sure that the vulnerability does affect a real device or not. However, not all papers do it [5]. Most often for Over-the-Air fuzzing work, as each test case comprises multiple mutations, a refinement stage is needed where each mutation of the crashing input is being selectively applied to identify the responsible mutation [51].

9 Future work

Combining Over-the-Air testing approach with a fully emulated system would provide a very complete and flexible testing environment. However, fully emulating a firmware and its physical environment is difficult, this requires interaction with peripherals that are complex, not emulated and not documented (interaction with the physical layer radio, DSPs, etc.).

Support for Apple and less studied devices. Market studies report that 99% of basebands are manufactured by, from the most to the least popular: Qualcomm, MediaTek, Samsung, Apple, UNISOC and HiSilicon (Huawei) [97]. So far research largely focused on MediaTek and Samsung.

Only a limited number of works have analyzed basebands used in iPhones [9, 14, 98]. Weinmann [9] focuses on early iPhone Intel basebands used in the iPhone 4 (2010). Kröll et al. [98] target the communication interface between iOS CommCenter and the Intel baseband; they do not support Qualcomm's baseband, which uses a different interface, and require a jailbroken device. Hussain et al. [14], in contrast, analyze the iPhone XS (2018) purely as a black box.

The current iPhone's baseband technology is an in house baseband, which originates from Apple's 2019 acquisition of Intel's mobile modem division. This makes it critical to find new ways to investigate the security of baseband on iPhones as those baseband are now only used on Apple devices. Indeed, it is not possible anymore to benefit from vulnerabilities found on Android devices using the same baseband chip as iPhones. For example, the baseband log can often be accessed on Android devices, but not on iPhones. Even Over-the-Air analysis is hard to do on iPhones as there is limited visibility on the baseband status (e.g., like ADB), and limited monitoring of the baseband for crash and reset (libimobiledevice was used for this in DIKEUE [99]).

Some baseband manufacturers remain understudied, including Unisoc [63] and Huawei (HiSilicon).

10 Conclusion

In this paper, we investigated the security of basebands, emphasizing their complexity, proprietary design, and reliance on memory-unsafe languages, which collectively introduce vulnerabilities exploitable by remote adversaries. We presented the Over-the-Air attack surface and highlighted the necessity of considering additional attack vectors such as the malicious application processor and SIM cards. Our taxonomy distinguishes between logic vulnerabilities and crashes. Additionally, we stressed the importance of considering the rogue network attack model.

We analyzed three dominant research paradigms: static analysis, Over-The-Air testing, and emulation-based fuzzing, highlighting their respective strengths and limitations. While OTA testing ensures realism, it suffers from scalability constraints; static analysis allows precise security analysis of the protocol implementation but is difficult to adapt to other baseband vendors, whereas emulation offers introspection and speed but demands substantial engineering effort.

Finally, we outlined future research directions, analysis of Apple basebands, and advancing full-system emulation, particularly for architectures such as Qualcomm's Hexagon.

Acknowledgments

We thank the anonymous reviewers for their feedback and suggestions. This work is supported in part by a government

grant managed by the National Research Agency under France 2030 with reference “ANR-22-PECY-0009”.

References

[1] Mirza Masfiqur Rahman, Imtiaz Karim, and Elisa Bertino. Cellularlint: a systematic approach to identify inconsistent behavior in cellular network specifications. In *33rd USENIX security symposium (USENIX security 24)*, pages 5215–5232, Philadelphia, PA, August 2024. USENIX Association. https://www.usenix.org/conference/usenixsecurity24/presentation/rahman.

[2] Daniel Klischies, Moritz Schloegel, Tobias Scharnowski, Mikhail Bogodukhov, David Rupprecht, and Veelasha Moonsamy. Instructions unclear: Undefined behaviour in cellular network specifications. In *32nd USENIX security symposium (USENIX security 23)*, pages 3475–3492, Anaheim, CA, August 2023. USENIX Association. https://www.usenix.org/conference/usenixsecurity23/presentation/klischies.

[3] Ofcom. Switching off the uk’s 2g and 3g mobile networks: what you need to know. https://www.ofcom.org.uk/phones-and-broadband/coverage-and-speeds/3g-switch-off, October 2025.

[4] CheolJun Park, Sangwook Bae, BeomSeok Oh, Jiho Lee, Eunkyu Lee, Insu Yun, and Yongdae Kim. Doltest: In-depth downlink negative testing framework for lte devices. In *31st USENIX security symposium (USENIX security 22)*, pages 1325–1342, Boston, MA, August 2022. USENIX Association. https://www.usenix.org/conference/usenixsecurity22/presentation/park-cheoljun.

[5] Eunsoo Kim, Dongkwan Kim, CheolJun Park, Insu Yun, and Yongdae Kim. Basespec: Comparative analysis of baseband software and cellular specifications for l3 protocols. In *Proceedings 2021 Network and Distributed System Security Symposium*, Virtual, 2021. Internet Society. https://www.ndss-symposium.org/wp-content/uploads/ndss2021_6B-4_24365_paper.pdf.

[6] Alisa Esage. Advanced hexagon diag and getting started with baseband vulnerability research. https://zerodayengineering.com/research/slides/CCC2020_AdvancedHexagonDiag.pdf, 2020.

[7] Ali Ranjbar, Tianchang Yang, Kai Tu, Saaman Khalilollahi, and Syed Rafiul Hussain. Stateful analysis and fuzzing of commercial baseband firmware. In *2025 IEEE symposium on security and privacy (SP)*, pages 1120–1139, Los Alamitos, CA, USA, May 2025. IEEE Computer Society. https://doi.ieeecomputersociety.org/10.1109/SP61157.2025.00143.

[8] Slava Makkaveev. Pwn2own qualcomm dsp. https://research.checkpoint.com/2021/pwn2own-qualcomm-dsp/, May 2021.

[9] Ralf-Philipp Weinmann. Baseband attacks: Remote exploitation of memory corruptions in cellular protocol stacks. In *6th USENIX Workshop on Offensive Technologies (WOOT 12)*, Bellevue, WA, August 2012. USENIX Association. https://www.usenix.org/conference/woot12/workshop-program/presentation/Weinmann.

[10] David Berard and Vincent Fargues. How to design a baseband debugger. In *SSTIC*, 2020. https://www.sstic.org/media/SSTIC2020/SSTIC-actes/how_to_design_a_baseband_debugger/SSTIC2020-Article-how_to_design_a_baseband_debugger-berard_fargues.pdf.

[11] Nico Golde and Daniel Komaromy. Breaking band - reverse engineering and exploiting the shannon baseband. https://comsecuris.com/slides/recon2016-breaking_band.pdf, 2016.

[12] Dominik Maier, Lukas Seidel, and Shinjo Park. Basesafe: baseband sanitized fuzzing through emulation. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 122–132, Linz Austria, July 2020. ACM. https://dl.acm.org/doi/10.1145/3395351.3399360.

[13] Eunsoo Kim, Min Woo Baek, CheolJun Park, Dongkwan Kim, Yongdae Kim, and Insu Yun. Basecomp: A comparative analysis for integrity protection in cellular baseband software. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 3547–3563, Anaheim, CA, August 2023. USENIX Association. https://www.usenix.org/conference/usenixsecurity23/presentation/kim-eunsoo.

[14] Syed Rafiul Hussain, Imtiaz Karim, Abdullah Al Ishtiaq, Omar Chowdhury, and Elisa Bertino. Noncompliance as deviant behavior: An automated black-box noncompliance checker for 4g lte cellular devices. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1082–1099, Virtual Event Republic of Korea, November 2021. ACM. https://dl.acm.org/doi/10.1145/3460120.3485388.

[15] Kai Tu, Abdullah Al Ishtiaq, Syed Md Mukit Rashid, Yilu Dong, Weixuan Wang, Tianwei Wu, and Syed Rafiul Hussain. Logic gone astray: A security analysis framework for the control plane protocols of 5g basebands. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 3063–3080, Philadelphia, PA, August 2024. USENIX Association. https://www.usenix.org/conference/usenixsecurity24/presentation/tu.

[16] L. Szekeres, M. Payer, Tao Wei, and Dawn Song. Sok: Eternal war in memory. In *2013 IEEE Symposium on Security and Privacy*, pages 48–62, Berkeley, CA, May 2013. IEEE. http://ieeexplore.ieee.org/document/6547101/.

[17] Unburdened by what has been: Exploiting new attack surfaces in radio layer 2 for baseband rce on samsung exynos. https://labs.taszk.io/articles/post/there_will_be_bugs/, July 2024.

[18] Ivan Lozano and Roger Piqueras Jover. Hardening cellular basebands in android. https://security.googleblog.com/2023/12/hardening-cellular-basebands-in-android.html, December 2023.

[19] Evangelos Bitsikas, Jason Veara, and Aanjhan Ranganathan. Security analysis of lte connectivity in connected cars: A case study of tesla. http://arxiv.org/abs/2510.22024, October 2025.

[20] Google Project Zero. Project zero: Multiple internet to baseband remote code execution vulnerabilities in exynos modems. https://googleprojectzero.blogspot.com/2023/03/multiple-internet-to-baseband-remote-rce.html, March 2023.

[21] Yiming Liu, Cen Zhang, Feng Li, Yeting Li, Jianhua Zhou, Jian Wang, Lanlan Zhan, Yang Liu, and Wei Huo. Semantic-enhanced static vulnerability detection in baseband firmware. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pages 1–12, Lisbon Portugal, April 2024. ACM. https://dl.acm.org/doi/10.1145/3597503.3639158.

[22] Daniel Klischies, Philipp Mackensen, and Veelasha Moonsamy. Vulnerability, where art thou? an investigation of vulnerability management in android smartphone chipsets. In *Proceedings 2025 Network and Distributed System Security Symposium*, 2025. http://arxiv.org/abs/2412.06556.

[23] Shutdown of 2g and 3g technologies in france - wholesale. https://wholesale.orange.com/france/en/news/shutdown-of-2g-and-3g-technologies-in-france/.

[24] We're switching off 2g. https://ee.co.uk/2g-3g-switch-off.

[25] Deutsche Telekom AG. More speed on old frequencies: 2g switch-off ensures better network. https://www.telekom.com/en/media/media-information/archive/more-speed-on-old-frequencies-2g-switch-off-ensures-better-network-1081866, October 2024.

[26] Roger Piqueras Jover, Yomna Nasser, and Sudhi Herle. Android 14 introduces first-of-its-kind cellular connectivity security features. https://security.googleblog.com/2023/08/android-14-introduces-first-of-its-kind.html.

[27] About lockdown mode. https://support.apple.com/en-us/105120.

[28] Ts 133 501 - v15.2.0 - 5g; security architecture and procedures for 5g system (3gpp ts 33.501 version 15.2.0 release 15).

[29] Norbert Ludant, Marinos Vomvas, Stavros Dimou, and Guevara Noubir. Low-layer attacks against 4g/5g networks. In *18th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 248–255, Arlington VA USA, June 2025. ACM. https://dl.acm.org/doi/10.1145/3734477.3734725.

[30] Ts 102 226 - v17.0.0 - smart cards; remote apdu structure for uicc based applications (release 17), 2022.

[31] Dave (Jing) Tian, Grant Hernandez, Joseph I. Choi, Vanessa Frost, Christie Raules, Patrick Traynor, Hayawardh Vijayakumar, Lee Harrison, Amir Rahmati, Michael Grace, and Kevin R. B. Butler. {attention} spanned: Comprehensive vulnerability analysis of {at} commands within the android ecosystem. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 273–290, 2018. https://www.usenix.org/conference/usenixsecurity18/presentation/tian.

[32] Imtiaz Karim, Fabrizio Cicala, Syed Rafiul Hussain, Omar Chowdhury, and Elisa Bertino. Opening pandora's box through atfuzzer: dynamic analysis of at interface for android smartphones. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 529–543, San Juan Puerto Rico USA, December 2019. ACM. https://dl.acm.org/doi/10.1145/3359789.3359833.

[33] Collin Mulliner and Charlie Miller. Injecting sms messages into smart phones for security analysis. In *Proceedings of the 3rd USENIX Conference on Offensive Technologies*, page 5, USA, 2009. USENIX Association.

[34] Qmi-ril - replicant. https://redmine.replicant.us/projects/replicant/wiki/QMI-RIL.

[35] Wenqiang Li, Haohuang Wen, and Zhiqiang Lin. Basemirror: Automatic reverse engineering of baseband commands from android's radio interface layer. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, pages 2311–2325, New York, NY, USA, December 2024. Association for Computing Machinery. https://doi.org/10.1145/3658644.3690254.

[36] Tuan Dinh Hoang, Taekkyung Oh, CheolJun Park, Insu Yun, and Yongdae Kim. Llfuzz: An over-the-air dynamic testing framework for cellular baseband lower layers. In *34rd USENIX Security Symposium (USENIX Security 25)*. USENIX Association, August 2025. https://www.usenix.org/conference/usenixsecurity25/presentation/hoang.

[37] Dyon Goos and Marius Muench. Overcoming state: Finding baseband vulnerabilities by fuzzing layer-2. https://i.blackhat.com/BH-US-24/Presentations/REVISED-US24-Goos-Overcoming-State-Finding-Baseband-Vulnerabilities-Thursday.pdf, August 2024.

[38] Ts 138 331 - v17.1.0; - 5g; nr; radio resource control (rrc); protocol specification (3gpp ts 38.331 version 17.1.0 release 17). https://www.etsi.org/deliver/etsi_ts/138300_138399/138331/17.01.00_60/ts_138331v170100p.pdf.

[39] Sam Biddle Hussain, Murtaza. Hacked documents: How iran can track and control protesters' phones. https://theintercept.com/2022/10/28/iran-protests-phone-surveillance/, October 2022.

[40] Haohuang Wen, Phillip Porras, Vinod Yegneswaran, and Zhiqiang Lin. Thwarting smartphone sms attacks at the radio interface layer. In *Proceedings 2023 Network and Distributed System Security Symposium*, San Diego, CA, USA, 2023. Internet Society. https://www.ndss-symposium.org/wp-content/uploads/2023/02/ndss2023_f432_paper.pdf.

[41] Matteo Mandolini. Using silent sms to localize lte users. https://mandomat.github.io/2023-09-21-localization-with-silent-SMS/, September 2023.

[42] slavam. Security probe of qualcomm msm data services. https://research.checkpoint.com/2021/security-probe-of-qualcomm-msm/, May 2021.

[43] Tomasz Piotr Lisowski, Merlin Chlosta, Jinjin Wang, and Marius Muench. Simurai: Slicing through the complexity of sim card security research. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 4481–4498. USENIX Association, 2024. https://www.usenix.org/conference/usenixsecurity24/presentation/lisowski.

[44] European Union Agency for Cybersecurity. *Embedded SIM ecosystem, security risks and measures – March 2023*. European Union Agency for Cybersecurity, 2023.

[45] Shalk Zakeer. Overcoming cloning and surveillance risks from esim vulnerability | ibm. https://www.ibm.com/think/insights/critical-wake-up-call-e-sim-vulnerability, November 2025.

[46] Hongil Kim, Jiho Lee, Eunkyu Lee, and Yongdae Kim. Touching the untouchables: Dynamic security analysis of the lte control plane. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1153–1168, San Francisco, CA, USA, May 2019. IEEE. https://ieeexplore.ieee.org/document/8835363/.

[47] Altaf Shaik, Ravishankar Borgaonkar, N. Asokan, Valtteri Niemi, and Jean-Pierre Seifert. Practical attacks against privacy and availability in 4g/lte mobile communication systems. 2017. http://dx.doi.org/10.14722/ndss.2016.23236.

[48] Zhenhua Li, Weiwei Wang, Christo Wilson, Jian Chen, Chen Qian, Taeho Jung, Lan Zhang, Kebin Liu, Xiangyang Li, and Yunhao Liu. Fbs-radar: Uncovering fake base stations at scale in the wild. In *Proceedings 2017 Network and Distributed System Security Symposium*, San Diego, CA, 2017. Internet Society. https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/fbs-radar-uncovering-fake-base-stations-scale-wild/.

[49] Kazi Samin Mubasshir, Imtiaz Karim, and Elisa Bertino. Gotta detect 'em all: Fake base station and multi-step attack detection in cellular networks. In *USENIX security symposium*, 2024.

[50] David Rupprecht, Katharina Kohls, Thorsten Holz, and Christina Popper. Breaking lte on layer two. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1121–1136, San Francisco, CA, USA, May 2019. IEEE. https://ieeexplore.ieee.org/document/8835335/.

[51] CheolJun Park, Marc Egli, BeomSeok Oh, Tuan Dinh Hoang, Suhwan Jeong, Martin Crettol, Insu Yun, Mathias Payer, and Yongdae Kim. Otabase: Enhancing over-the-air testing to detect memory crashes in cellular basebands. In *Annual Computer Security Applications Conference*. ACSAC, December 2025.

[52] Simon Erni, Martin Kotuliak, Patrick Leu, Marc Roeschlin, and Srdjan Capkun. Adaptover: Adaptive overshadowing attacks in cellular networks. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 743–755, October 2022. http://arxiv.org/abs/2106.05039.

[53] Hojoon Yang, Sangwook Bae, Mincheol Son, Hongil Kim, Song Min Kim, and Yongdae Kim. Hiding in plain signal: Physical signal overshadowing attack on lte. In *28th USENIX security symposium (USENIX security 19)*, pages 55–72, Santa Clara, CA, August 2019. USENIX Association. https://www.usenix.org/conference/usenixsecurity19/presentation/yang-hojoon.

[54] Shijie Luo, Matheus Garbelini, Sudipta Chattopadhyay, and Jianying Zhou. Sni5gect: A practical approach to inject anrchy into 5g nr. In *34rd USENIX Security Symposium (USENIX Security 25)*. USENIX Association, August 2025. https://www.usenix.org/conference/usenixsecurity25/presentation/luo-shijie.

[55] Norbert Ludant, Pieter Robyns, and Guevara Noubir. From 5g sniffing to harvesting leakages from privacy-preserving messengers. *2023 IEEE Symposium on Security and Privacy (SP)*, pages 3146–3161, 2023.

[56] David Rupprecht, Kai Jansen, and Christina Pöpper. Putting lte security functions to the test: a framework to evaluate implementation correctness. In *10th USENIX workshop on offensive technologies (WOOT 16)*, Austin, TX, August 2016. USENIX Association. https://www.usenix.org/conference/woot16/workshop-program/presentation/rupprecht.

[57] Syed Rafiul Hussain, Mitziu Echeverria, Imtiaz Karim, Omar Chowdhury, and Elisa Bertino. 5greasoner: A property-directed security and privacy analysis framework for 5g cellular network protocol. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 669–684, London United Kingdom, November 2019. ACM. https://dl.acm.org/doi/10.1145/3319535.3354263.

[58] Yi Chen, Yepeng Yao, XiaoFeng Wang, Dandan Xu, Chang Yue, Xiaozhong Liu, Kai Chen, Haixu Tang, and Baoxu Liu. Bookworm game: Automatic discovery of lte vulnerabilities through documentation analysis. In *2021 IEEE symposium on security and privacy (SP)*, pages 1197–1214, 2021.

[59] Yi Chen, Di Tang, Yepeng Yao, Mingming Zha, XiaoFeng Wang, Xiaozhong Liu, Haixu Tang, and Dongfang Zhao. Seeing the forest for the trees: Understanding security hazards in the 3gpp ecosystem through intelligent analysis on change requests. In *31st USENIX security symposium (USENIX security 22)*, pages 17–34, Boston, MA, August 2022. USENIX Association. https://www.usenix.org/conference/usenixsecurity22/presentation/chen-yi.

[60] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. A formal analysis of 5g authentication. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1383–1396, Toronto Canada, October 2018. ACM. https://dl.acm.org/doi/10.1145/3243734.3243846.

[61] Daniel Komaromy. Cve-2021-32487: Heap buffer overflow in gsm rrm channel release, cell selection indicator. https://labs.taszk.io/blog/post/68_mtk_hof_4/, March 2022.

[62] Matheus E Garbelini, Zewen Shang, Shijie Luo, Sudipta Chattopadhyay, Sumei Sun, and Ernest Kurniawan. 5ghoul: Unleashing chaos on 5g edge devices via stateful multi-layer fuzzing. *IEEE Transactions on Dependable and Secure Computing*, June 2025.

[63] Slava Makkaveev. Vulnerability within the unisoc baseband opens mobile phones communications to remote hacker attacks. https://research.checkpoint.com/2022/vulnerability-within-the-unisoc-baseband/, June 2022.

[64] Security advisory - stack overflow vulnerability in baseband module of some huawei smart phones. https://www.huawei.com/en/psirt/security-advisories/2017/huawei-sa-20171125-01-baseband-en.

[65] Security bulletins | qualcomm documentation | cve-2025-21483. https://docs.qualcomm.com/product/publicresources/securitybulletin/september-2025-bulletin.html#_cve-2025-21483.

[66] Laura French. Qualcomm chip vulnerability enables remote attack by voice call. https://www.scworld.com/news/qualcomm-chip-vulnerability-enables-remote-attack-by-voice-call, January 2024.

[67] Stavros Eleftherakis, Domenico Giustiniano, and Nicolas Kourtellis. Sok: Evaluating 5g-advanced protocols against legacy and emerging privacy and security attacks. In *18th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 196–210, Arlington VA USA, June 2025. ACM. https://dl.acm.org/doi/10.1145/3734477.3734716.

[68] Elisa Bertino, Syed Rafiul Hussain, and Omar Chowdhury. 5g security and privacy - a research roadmap. https://arxiv.org/abs/2003.13604.

[69] Hwankuk Kim. 5g core network security issues and attack classification from network protocol perspective. *Journal of Internet Services and Information Security*, 10(2):1–15, May 2020. https://doi.org/10.22667/JISIS.2020.05.31.001.

[70] Mohamed Amine Ferrag, Leandros Maglaras, Antonios Argyriou, Dimitrios Kosmanos, and Helge Janicke. Security for 4g and 5g cellular networks: A survey of existing authentication and privacy-preserving schemes. 101:55–82, August 2017.

[71] David Rupprecht, Adrian Dabrowski, Thorsten Holz, Edgar Weippl, and Christina Pöpper. On security research towards future mobile network generations. *IEEE Communications Surveys & Tutorials*, March 2018. https://ieeexplore.ieee.org/document/8329226.

[72] Ts 124 341 - v16.0.0 - digital cellular telecommunications system (phase 2+) (gsm); universal mobile telecommunications system (umts); lte; 5g; support of sms over ip networks; stage 3 (3gpp ts 24.341 version 16.0.0 release 16).

[73] Ts 124 301 - v16.6.0 - universal mobile telecommunications system (umts); lte; 5g; non-access-stratum (nas) protocol for evolved packet system (eps); stage 3 (3gpp ts 24.301 version 16.6.0 release 16).

[74] Marco Grassi and Xingyu Chen. Over the air baseband exploit: Gaining remote code execution on 5g smartphones. In *Black Hat*, USA, 2021. https://keenlab.tencent.com/zh/whitepapers/us-21-Over-The-Air-Baseband-Exploit-Gaining-Remote-Code-Execution-on-5G-Smartphones-wp.pdf.

[75] Xuang Xing, Eugene Rodionov, Xiling Gong, and Farzan Karimi. Over the air, under the radar attacking and securing the pixel modem. https://i.blackhat.com/BH-US-23/Presentations/US-23-Karimi-Over-the-Air-Under-the-Radar.pdf, 2023.

[76] Cve-2023-21517: Samsung baseband lte esm tft heap buffer overflow. https://labs.taszk.io/blog/post/85_ss_esm_bof/, November 2023.

[77] Merlin Chlosta, David Rupprecht, Christina Pöpper, and Thorsten Holz. 5g suci-catchers: still catching them all? In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 359–364, Abu Dhabi United Arab Emirates, June 2021. ACM. https://dl.acm.org/doi/10.1145/3448300.3467826.

[78] Cwe - cwe-617: Reachable assertion (4.18). https://cwe.mitre.org/data/definitions/617.html.

[79] Nvd - cve-2025-20750. https://nvd.nist.gov/vuln/detail/CVE-2025-20750.

[80] Evangelos Bitsikas, Syed Khandker, Ahmad Salous, Aanjhan Ranganathan, Roger Piqueras Jover, and Christina Pöpper. Ue security reloaded: Developing a 5g standalone user-side security testing framework. In *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '23, pages 121–132, Guildford, United Kingdom, 2023. Association for Computing Machinery. https://doi.org/10.1145/3558482.3590194.

[81] Syed Khandker, Michele Guerra, Evangelos Bitsikas, Roger Piqueras Jover, Aanjhan Ranganathan, and Christina Pöpper. Astra-5g: Automated over-the-air security testing and research architecture for 5g sa devices. In *Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 89–100, Seoul Republic of Korea, May 2024. ACM. https://dl.acm.org/doi/10.1145/3643833.3656141.

[82] Yi Chen, Di Tang, Yepeng Yao, Mingming Zha, XiaoFeng Wang, Xiaozhong Liu, Haixu Tang, and Baoxu Liu. Sherlock on specs: Building lte conformance tests through automated reasoning. In *32nd USENIX security symposium (USENIX security 23)*, pages 3529–3545, Anaheim, CA, August 2023. USENIX Association. https://www.usenix.org/conference/usenixsecurity23/presentation/chen-yi.

[83] Zewen Shang, Matheus E. Garbelini, and Sudipta Chattopadhyay. U-fuzz: Stateful fuzzing of iot protocols on cots devices. In *2024 IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 209–220, May 2024. https://ieeexplore.ieee.org/document/10638624.

[84] Ilja Siros, Dave Singelee, and Bart Preneel. Covfuzz: Coverage-based fuzzer for 4g&5g protocols. In *2025 IEEE 10th european symposium on security and privacy (EuroS&P)*, pages 737–754, Los Alamitos, CA, USA, July 2025. IEEE Computer Society. https://doi.ieeecomputersociety.org/10.1109/EuroSP63326.2025.00047.

[85] Grant Hernandez, Marius Muench, Dominik Maier, Alyssa Milburn, Shinjo Park, Tobias Scharnowski, Tyler Tucker, Patrick Traynor, and Kevin R. B. Butler. Firmwire: Transparent dynamic analysis for cellular baseband firmware. In *Symposium on Network and Distributed System Security (NDSS)*, 2022.

[86] Suhwan Jeong, Beomseok Oh, Kwangmin Kim, Insu Yun, Yongdae Kim, and CheolJun Park. Firmstate: Bringing cellular protocol states to shannon baseband emulation. In *18th ACM Conference on Security and*

Privacy in Wireless and Mobile Networks, WiSec 2025, pages 242–247, New York, NY, USA, June 2025. Association for Computing Machinery. https://dl.acm.org/doi/10.1145/3734477.3734726.

[87] Daniel Klischies, Dyon Goos, David Hirsch, Alyssa Milburn, Marius Muench, and Veelasha Moonsamy. Basebridge: Bridging the gap between over-the-air and emulation testing for cellular baseband firmware. In *2025 IEEE symposium on security and privacy (SP)*, pages 1101–1119, Los Alamitos, CA, USA, May 2025. IEEE Computer Society. https://doi.ieeecomputersociety.org/10.1109/SP61157.2025.00142.

[88] Matheus E. Garbelini, Zewen Shang, Sudipta Chattopadhyay, Sumei Sun, and Ernest Kurniawan. Towards automated fuzzing of 4g/5g protocol implementations over the air. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pages 86–92, Rio de Janeiro, Brazil, December 2022. IEEE. https://ieeexplore.ieee.org/document/10001673/.

[89] Bruno Produit, Luca Glockow, and Rachna Shriwas. Securing the airwaves emulation, fuzzing, and reverse engineering of iphone baseband firmware. https://troopers.de/troopers25/talks/8p3ft8/, June 2025.

[90] Ralf-Philipp Weinmann. Baseband exploitation in 2013: Hexagon challenges. https://comsecuris.com/slides/rpw-pacsec2013-hexagon.pdf, November 2013.

[91] Grant Hernandez and Kevin R. B. Butler. Basebads: Automated security analysis of baseband firmware: poster. In *Proceedings of the 12th conference on security and privacy in wireless and mobile networks*, WiSec '19, pages 318–319, Miami, Florida, 2019. Association for Computing Machinery. https://doi.org/10.1145/3317549.3326310.

[92] Andrew Fasano, Tiemoko Ballo, Marius Muench, Tim Leek, Alexander Bulekov, Brendan Dolan-Gavitt, Manuel Egele, Aurélien Francillon, Long Lu, Nick Gregory, Davide Balzarotti, and William Robertson. Sok: Enabling security analyses of embedded systems via rehosting. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pages 687–701, Virtual Event Hong Kong, May 2021. ACM. https://dl.acm.org/doi/10.1145/3433210.3453093.

[93] Github - quic/qemu at hex-next. Qualcomm Innovation Center, Inc., October 2025. https://github.com/quic/qemu/tree/hex-next.

[94] Brian Cain. Kvm forum 2025: Hexagon system emulation. https://gitlab.com/qemu-project/kvm-forum/-/raw/main/_attachments/2025/KVM_Forum_2_9J9ZVp8.pdf, September 2025.

[95] Romain Malmain, Andrea Fioraldi, and Aurélien Francillon. Libafl qemu: a library for fuzzing-oriented emulation. In *Workshop on binary analysis research (colocated with NDSS symposium)*, Bar 24, San Diego (USA), March 2024.

[96] Marius Muench, Jan Stijohann, Frank Kargl, Aurélien Francillon, and Davide Balzarotti. What you corrupt is not what you crash: Challenges in fuzzing embedded devices. In ISOC, editor, *NDSS 2018, Network and Distributed Systems Security Symposium, 18-21 February 2018, San Diego, CA, USA*, San Diego, 2018.

[97] Qualcomm gains share in smartphone ap/soc shipments in q4 2021; mediatek continues to lead. https://counterpointresearch.com/en/insights/ap-soc-shipments-q4-2021.

[98] Tobias Kröll, Stephan Kleber, Frank Kargl, Matthias Hollick, and Jiska Classen. Aristoteles – dissecting apple's baseband interface. In Elisa Bertino, Haya Shulman, and Michael Waidner, editors, *Computer Security – ESORICS 2021*, pages 133–151, Cham, 2021. Springer International Publishing.

[99] libimobiledevice · a cross-platform foss library written in c to communicate with ios devices natively. https://libimobiledevice.org/.