SoK: Insecurity of Cellular Basebands

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 cellular protocols make them a prime target for attacks. Although billions of devices rely on these components, baseband security remains poorly explored, despite many papers published in the past few years. 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, or modem processors. 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 [73]. Moreover, most basebands still support 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 [69]. Such complexity hinders the implementation of cellular logic in baseband firmware and can lead to undefined behaviors [50] potentially introducing vulnerabilities into the system [70].

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

This protocol complexity results in large and sophisticated firmware, making analysis challenging for several reasons. The first reason is that baseband firmware size is important, a firmware for Samsung's Shannon baseband chipset can contain more than 90k functions [46]. Additionally, baseband firmwares are proprietary, with little information about their inner workings being publicly available. Most available information comes from reverse engineering of these firmware. Also, as debug information is removed from the baseband firmware, for example, in Qualcomm's baseband firmware [30], this reverse engineering is challenging to do.

These firmware run on dedicated, undocumented hardware and use custom *Real-Time Operating System (RTOS)* which needs to be reversed in order to understand the inner workings of the firmware. For example, Samsung and Qualcomm basebands use their own proprietary RTOS [62, 74].

In order to investigate the security of those basebands, researchers initially applied manual reverse engineering [19, 27, 34]. However, as external information is scarce, and as the firmware size is significant, reverse engineering is difficult and does not scale. Researchers have developed automated tools to ease this analysis [41, 45, 46, 50, 61, 70, 83].

The analysis of baseband's firmware is also made harder because debugging interfaces are generally disabled on production devices [30]. They additionally suffer from the classical difficulty of detecting crashes in embedded systems [67].

However, most implementations are written in memory unsafe languages such as C and C++, raising the risk of memory vulnerabilities (such as buffer overflows, integer overflows, use after free or other vulnerabilities [81]).

These chips run independently of the application processor, and are less hardened than application processors [86]. For example, Samsung basebands do not support ASLR [15]. Some researchers [57] recommend hardening the basebands firmware or using memory-safe languages for critical compo-

nents, such as parsers that handle untrusted data.

Vulnerabilities in baseband are critical as they can lead to privacy issues, the baseband has access to internal components such as the microphone [86]. Furthermore, vulnerabilities in basebands leading to crashes can endanger people relying on those devices as they can prevent accessing emergency services. But those vulnerabilities can also be exploited silently [56, 90] as the baseband is independent of the main OS processor, it is therefore difficult to monitor the baseband activity, and it is not possible to run antivirus on it.

Moreover, those vulnerabilities can be exploited remotely as basebands are able to communicate over the air with other 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 often present in multiple smartphones because fewer than a dozen of companies build those chips. Therefore, one vulnerability in one baseband can affect a whole span of smartphones from different vendors [90].

Therefore, researchers have investigated ways to detect flaws in the baseband implementations, by means of static analysis, *Over-the-air (OTA)* testing, emulation. Those tools report both logical bugs and inconsistencies due to incorrect implementation of the cellular specification, but they also report crashes due to invalid memory handling or invalid assertions. Those bugs 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.

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;
- outline future research areas.

The remainder of this paper is organized as follows. Section 2 provides background on cellular protocol generations and baseband architecture, the attack models and vulnerabilities taxonomy are introduced in section 3. Section 4 reviews existing analysis methodologies and discusses their respective strengths and limitations, and section 5 explores additional attack surfaces, such as SIM interactions and application processor interfaces. Finally, section 6 examines future work and section 7 concludes the paper.

2 Background

2.1 Protocol generations

Cellular networks evolved from 2G (GSM), which lacked base station and core network mutual authentication. Because of this, it is not possible to prevent fake base station attacks. This was fixed in 3G (UMTS). Additionally, some network operators are starting to disable it [9, 12, 17], Google added an option to disable its usage in Android 14 [43]. However, 2G support in iPhones can only be disabled by enabling the Lockdown Mode [1] 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.

5G strengthened L3 security with mechanisms like *Subscriber Concealed Identifier (SUCI)*. These mechanisms were intended to prevent device deanonymization by attackers. However, the new lower layers capabilities of the 5G introduce a new attack surface that can be exploited [59].

2.2 Baseband architecture

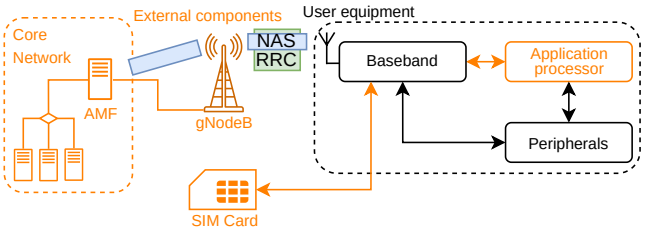

Figure 1: Overview of the baseband architecture and its interaction with both peripheral components and network entities. Elements highlighted in orange represent adversarial components considered in this study.

Basebands interact with a wide variety of devices, but also air interfaces with many layers and protocols. Four peripherals are central in the attack surface analysis: the base station, the core network, the SIM Card and the application processor (Figure 1).

Base Station. The baseband and the base station communicate over the air interface to transmit user data and phone calls. In 4G networks, the base station is referred to as the *eNodeB*, while in 5G it is called the *gNodeB*. The base station plays a role in the cellular connectivity security, as it authenticates the user and ensures integrity protection and encryption of this communication channel.

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 3.1) 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.

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 [13]. This functionality introduces an additional attack surface.

SIM cards exist in multiple 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 communicate through AT commands [44, 68, 82]. Some recent basebands use proprietary communication protocols instead of AT commands [6]. On the Android platform, the *Radio Interface Layer (RIL)* daemon of the application processor is responsible for handling this communication [53] (Section 5.2).

3 Attack model

3.1 Over-the-air attack surface

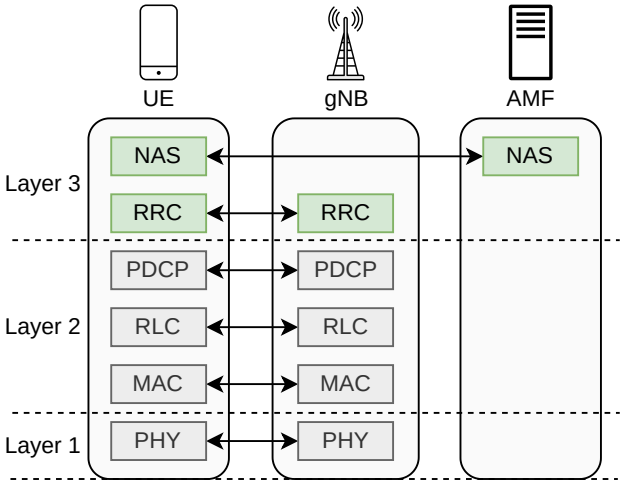

Figure 2: Organization of the cellular protocols for the 5G control plane. In green, the integrity protected and encrypted protocols.

The link between the base station and the baseband constitutes a broad attack surface. Cellular connectivity is complex and spans multiple generations of complex protocol stack, spanning from 2G to 5G. This broad attack surface can be divided into the 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, 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)* and the base station and 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 outlined here corresponds to 5G standard, but its underlying design principles remain consistent with those of earlier generations.

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

Layer 2 is rarely the focus of previous work, however, it constitutes a significant attack surface. Protocols at this layer are neither integrity protected nor encrypted, yet they expose a bread range of functionalities that can be exploited to trigger crashes or memory vulnerabilities in the basebands leading to potential remote code execution [15, 35, 38]. This layer comprises several protocols, including the *Packet Data Convergence Protocol (PDCP)*, the *Radio Link Control (RLC)*, and the *Medium Access Control (MAC)*. The *PDCP* protocol is responsible for providing integrity protection and encryption to both the control plane and user plane data, ensuring confidentiality and preventing tampering during communication between the UE and the base station. The *RLC* protocol manages packet segmentation, sequencing, and, reassembly. Finally, the *MAC* protocol coordinates access to the physical transmission channel, ensuring optimal use of the radio spectrum.

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

The RRC layers controls the communication between the UE and the base station and performs mutual authentication. NAS and RRC messages rely on distinct security contexts, resulting in different unauthenticated attack surface.

The NAS context is established first by the AMF through a *NAS Security Mode Command*, which specifies the ciphering and integrity algorithms. This enables integrity protection for NAS messages, while RRC messages remain unprotected. The base station then sends an *RRC Security Mode Command* to configure ciphering and integrity algorithms for PDCP, completing the RRC security setup.

For testing purposes, specifications allow null ciphering and null integrity protection, but these should never be permitted in production. Nevertheless, some basebands accept them [70].

3.2 Attack scenario

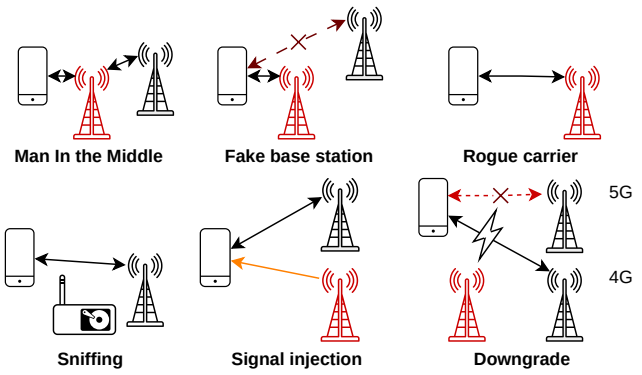

Figure 3: Attack scenarios.

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.

Fake Base Station (FBS). In the fake base station attack [47, 54, 66, 78], a fake base station is set up to impersonate a valid base station. In this model, the attacker does not have access to the cryptographic keys of the valid network. Therefore, if the baseband implementation is secure, the attacker will not be able to send messages after the authentication to the baseband. To be able to conduct this attack, the attacker only requires a laptop and a *Software Defined Radio (SDR)*, such as bladeRF 2.0 micro xA9 costing ~900$. This attack technique is therefore not limited to state actors but can also be executed by adversaries with comparatively limited resources.

Man-In-the-Middle (MitM). In a Man-In-the-Middle attack [76], the attacker will setup a fake base station with the name of a legitimate base station. When a UE connects to it, the rogue base station cannot prove to be the legitimate base station as the attacker does not have access to the cryptographic keys (except in 2G where those are not needed). However, the attacker can forward messages to the legitimate base station. This allows reading, modifying, dropping, injecting or replying messages sent between the UE and the legitimate base station. Once the security context is set up, integrity and the encryption of messages should prevent the attacker from reading and modifying the messages, however, some attacks have been discovered to bypass this protection (Section 3.4.1).

Rogue Carrier. In the case of a rogue carrier [71], 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 unauthenticated and authenticated section of the protocol. This attack surface also needs to be inspected as it could be used by a malicious or compromised network operator. Also, with the introduction of *Non-Public Networks (NPN)* [4], where cellular networks are operated under the control of private entities, the number of deployed cellular networks will increase, thereby amplifying the risk of vulnerabilities in individual deployments.

Signal injection. Signal injection attacks [29, 60, 89], involve crafting radio signals to inject messages into the ongoing communication between the victim UE and a legitimate base station. Unlike fake base station attacks, this technique cannot guarantee full success because the injected signal competes with a legitimate one. To improve reliability, attackers can jam the legitimate network and repeat the injected messages multiple times [60]. Additionally, distance can reduce the effectiveness of this method. Experiments in SNI5GECT [60], 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 respond an invalid response which can be rejected by the legitimate base station.

Sniffing. Sniffing represents a comparatively less powerful attack vector [58, 60], yet it merits consideration as inadequate mitigation can enable adversaries to capture messages or derive device specific fingerprints. In certain scenarios, vulnerabilities allow traffic to be transmitted without encryption, leaving the user unaware of its vulnerability to sniffing attacks [75].

Downgrade. A downgrade is not an attack scenario in itself but rather an outcome of other attack techniques. Certain protocol or baseband vulnerabilities can be leveraged to trigger downgrades. However, it enables adversaries to exploit additional attack surface by forcing a UE to reconnect using a less secure cellular generation. This typically involves disrupting the communication between the UE and its serving base station, compelling the UE to attempt reconnecting to a different base station operating on an older standard. Such legacy often contains well-known vulnerabilities. For example, downgrading to 2G is particularly critical due to the absence of integrity protection (Section 3.4.3).

[1] Although the author indicate using a rogue base station attacker model, they consider that the attacker does not have access to the cryptographic keys.

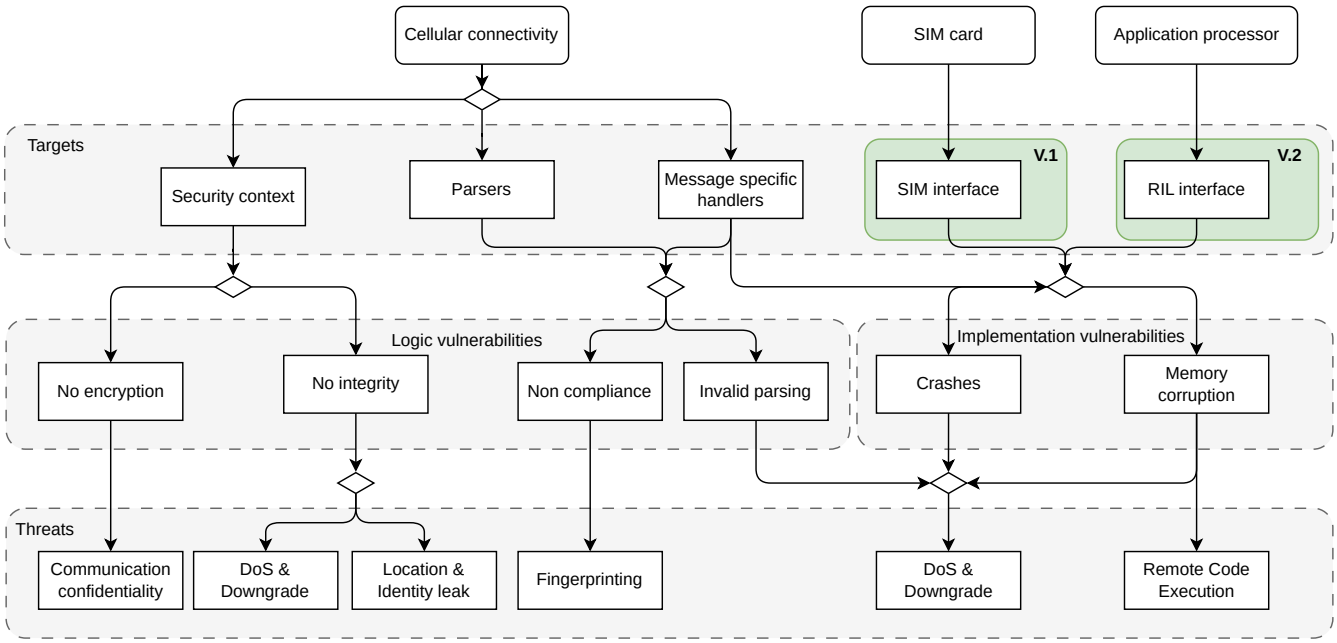

Figure 4: Taxonomy of basebands attacks surface.

3.3 Attacker model

Multiple attacker models can be considered, each exploiting different parts of the attack surface.

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.

Compromised Mobile Network Operator. This attacker has access to cryptographic keys, representing the rogue carrier model [71]. It constitutes a highly powerful attacker because it can exploit both pre-authentication and post-authentication attack surfaces. Such scenarios are relevant when considering untrusted network operators, especially in regions where states actors target civilians [39]. This model also applies in cases where the SIM card itself is malicious [55], potentially enabling the UE to connect to rogue networks.

Adversarial UE. An attacker can send messages to the phone without requiring network-level access. For example, sending multiple silent SMS messages with a recognizable pattern can be used to track a user's location by detecting this pattern when sniffing cellular communications in a specific area [65].

We consider this to be equivalent to the fake base station attack model.

3.4 Vulnerabilities

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 because they affect all devices and require a new version of the protocol to fix. Such flaws can lead to deanonymization, fingerprinting, impersonation. Some work [18, 23, 24, 40, 73, 76] has analyzed the protocol to identify vulnerabilities in the specification. Although protocol vulnerabilities in the specification are important, they are not really the focus of this paper which focuses on the implementation errors in the baseband itself. This said, baseband testing can reveal vulnerabilities from the specification too.

Logical bugs. Errors can occur during protocol implementation, including invalid integrity checks and incorrect message parsing. Logical bugs are difficult to detect and often require custom oracles specially targeting specific protocol components. Designing these oracles may demand specialized protocol knowledge. Few off-the-shelf detectors exist, unlike memory sanitizers. In DoLTEst [70] paper, the authors identified three categories of logical bugs when handling cellular messages: issues in messages' integrity and encryption handling, flaws in generic parser or flaws in specific message

Paper	FBS	M	I	R	Vulns
BaseSpec [46]	Not specified				</>
BASECOMP [45]	●				</>
DIKUEUE [41]	●	●			</>
5GBaseChecker [83]	●	●			</>
DoLTEst [70]	●	●	●		</>
Instruction unclear [50]			●		</>
UFuzz [79]		●			
UE Security Reloaded [21]	Not specified				</>
SNI5GECT [60]			●		</>
5GHoul [33]	●		●		
OTABase [71]	●	●	●	●	
LLFuzz [38]	●	●	●		
BVFinder [56]	●	●	●	●	
BaseSAFE [61]	Not specified				
FIRMWIRE [37]	● [1]				
FIRMSTATE [42]	Not specified				
BASEBRIDGE [49]	Not specified				
LORIS [74]	●	●	●		
Hexagon fuzz [72]	Not specified				
SIMurai [55]	Hostile SIM				
BaseMirror [53]	Application Processor				

Table 1: Summary of the attack models used in each paper and the vulnerabilities analyzed.
F: Fake Base station, M: Man-In-the-Middle, I: Injection, R: Rogue base station. Attacker model considered in each paper, : memory vulnerabilities, </>: logic vulnerabilities.

handlers.

Memory vulnerabilities. Improper message handling can trigger invalid memory accesses, integer overflows, or use-after-free errors [81]. Such vulnerabilities have been found in basebands from MediaTek [33, 51, 71], UNISOC [63], Huawei [7], Qualcomm [8, 32] and Samsung [34] basebands. In addition to being programmed in memory-unsafe languages, basebands often lack the memory protection mechanisms present on desktop and server operating systems. To detect these vulnerabilities, researchers can build memory sanitizers using hooks available in emulators [61], an approach that is broadly applicable to other embedded systems.

Protocol vulnerabilities have already been reviewed in earlier literature [20,28]. Core network security has been covered in prior work [31, 48].

In this paper, we focus our analysis on logical and memory vulnerabilities affecting basebands (Table 1). These two classes of vulnerabilities can arise independently, but they can also interact and produce broader security consequences for the communication stack or the device as a whole.

Both classes can be further divided into subcategories. Logical vulnerabilities typically affect the integrity of the protocol state machine or the correctness of higher-level procedures. Memory vulnerabilities, on the other hand, can undermine the confidentiality, integrity, or availability of the baseband through issues such as buffer overflows, use-after-free flaws, or improper memory isolation.

3.4.1 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 protocol should be both integrity protected and encrypted. 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. BASECOMP [45] investigates the correctness of integrity enforcement in baseband firmware. Using symbolic analysis, the authors compute the constraints applied to incoming messages and compare those constraints against the specification. This approach enables the detection of messages that are accepted without the required integrity protection.

When integrity checks are missing or incorrectly implemented on the received packets, the attackers may be able to exploit those vulnerabilities by injecting malicious messages, allowing to attack the baseband where it should normally only accept authenticated messages.

This can also lead to privacy issues. For example, this may allow sending SMS phishing messages, sending invalid network identity and time elements. Other messages can be sent, and those will have an impact on the confidentiality or the availability of the communication.

In 4G/5G, SMS messages can be sent through the user plane over IP to the IMS server [11] or over NAS in the control plane [10]. DoLTEst [70] 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 [45] authors and 5GBaseChecker [83] authors managed to reproduce this attack using other vulnerabilities allowing to bypass the integrity verification. Those vulnerabilities enable an attacker to send fishing 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 [45, 70].

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 [15, 36, 86, 88] and a public report [14].

3.4.2 Confidentiality

Users care about their privacy, mobile protocols have been traditionally weak in this respect, with broadcasting identifiers in the clear (e.g., IMSI) or transmitting data in clear (SMS in 2G). However, as those protocol design vulnerabilities are addressed, attackers will target implementation vulnerabilities to track devices identities and access users data.

User confidentiality.

As stated in 3.4.1 integrity vulnerabilities can allow sensitive RRC or NAS messages, normally requiring a valid security context. For example, an attacker could send a *RRCConnectionReconfiguration* with the *measConfig* IE which would leak signal information and therefore help to fingerprint or locate the UE [70]. However, in 4G, this could also be used to deanonymize the UE by sending an *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 exposures of permanent identifiers over the air. Nevertheless, certain attack [25] can still infer the presence of arbitrary devices within a given area, despite the use of SUCI.

Additionally, device's capabilities or implementation specificities can be used to fingerprint the device [70], without requiring the integrity keys.

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 like the control plane. However, DoLTEst [70] authors identified a 4G vulnerability where the baseband accepts the EIAO integrity mechanism. This forces the use of null integrity, which in turn results in null ciphering. Exploiting this flaw, an attacker operating a fake base station can intercept unencrypted traffic. This issue is especially critical when the UE does not alert the user that the connection lacks encryption [75]. Likewise, the authors of 5GBaseChecker [83] demonstrated that exploiting the integrity bypass presented in (Section 3.4.1) 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, D. Klischies et al. [50] discovered a memory vulnerability in the *Public Warning System (PWS)* that permits out-of-bound reads. Under the right conditions, this flaw could expose sensitive data from memory, including PDCP session encryption keys.

3.4.3 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 [50]. However, this recovery process introduces a delay, which can be exploited for persistent disruption. Such crashes often result from invalid memory accesses or reachable assertions [2] in the firmware implementation [5, 33, 50, 70].

Denial-of-service can also occur without crashes. For example, the DIKUEUE [41] 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 the exposure of the IMSI. 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 GUTI, 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 [21, 33]. In such attacks, the adversary will deliberately jam one network forcing the device to revert to a legacy standard that may contain well documented vulnerabilities.

Existing research has demonstrated several downgrade vectors. For instance 5GHoul [33] describes an attack that flood the UE with *Deregistration Accept* messages exhausting the UE handling resources and leading to a downgrade. This technique requires modifications to the base station software to enable sending arbitrarily retry messages. Another vulnerability identified by the same tool 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 injecting an *RRC Release* message can force the UE to reconnect using another protocol version [83], effectively triggering a downgrade.

4 Analysis methods

To systematically explore the extensive attack surface of cellular basebands, researchers have developed specialized analysis tools that fall into three primary analysis categories: *static analysis*, *Over-The-Air (OTA) analysis* and *emulation*.

4.1 Over The Air 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 [33, 34, 38, 80, 85–87]. Nevertheless, OTA testing suffers from practical limitations: tests are slow, costly to scale as one physical UE is required per testing process.

For each test case, the testing system must wait a few seconds when the device silently ignores a response to confirm that the response was truly ignored [70].

When a crash happens, the baseband state needs to be reset. This requires to turn the airplane mode on and off [50] or even, in some cases, to reboot the device or removing and reinserting the SIM card [71]. This can take up to a dozen of seconds and causes an important slowdown in the fuzzing process.

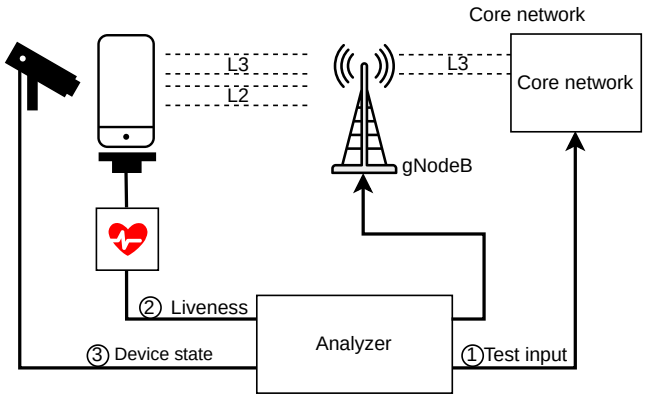

Figure 5: Example Over The Air (OTA) baseband testing platform setup.

A typical OTA baseband testing platform (Figure 5) operates in three stages: ① the platform creates test inputs to be transmitted over-the-air to the device under test. ② the platform evaluates the effect of the test input using security oracles, such as detecting baseband crashes or improper security context. ③ in the context of fuzzing, additional observers may be introduced to guide the mutators and improve coverage.

4.1.1 Input generation

Fuzzing target. OTA testing serves multiple purposes. It can uncover crashes or timeouts caused by memory vulnerabilities or reachable assertions [33, 38, 71, 79, 80]. It can also reveal logic vulnerabilities [50, 70], and even help infer the finite state machines implemented in certain basebands [41, 83]. Table 3 provides an overview of the protocols covered by the various works analyzed in this paper.

Generating the fuzzing input. OTA testing is inherently slow, so inputs must be sufficiently valid to avoid early rejection, yet not overly constrained, as mutations are necessary to uncover vulnerabilities. Valid messages enable deeper state exploration in the firmware, while mutated inputs test robustness against malformed data. This creates a trade-off between fidelity, which impacts exploration depth, and mutation or non-compliant input, which increases the likelihood of triggering bugs.

Two main approaches are commonly used: (i) manually crafting protocol messages [50, 70], or (ii) relying on an existing network stack to generate messages and applying mutation strategies to produce malformed or adversarial inputs [33, 38, 71, 80].

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 in inserting mutation hooks within the mobile network implementation, enabling selective modification of messages before they are transmitted to the UE. When mutating messages that are subject to integrity protection, these hooks must be placed prior to the application of authentication and integrity mechanisms; otherwise, any modification will cause the messages to be rejected due to integrity check failures.

To generate messages or inject messages, researchers commonly rely on open-source cellular network implementations, such as *OpenAirInterface (OAI)*[2] [33], *OpenBTS*[3], or *srsRAN*[4] [38, 71, 80].

However, the baseband is highly stateful, requiring the UE to be in a specific state before accepting certain messages. Some researchers use those cellular network implementations to establish valid communication between the baseband and the core network, ensuring that the device is in the correct state before injection test inputs [50, 70].

Also, both commercial and open-source stacks have limitations, as they do not implement all the features defined in the standard [71]. Consequently, researchers must devise strategies to test unimplemented messages and modify the software to allow sending custom, invalid packets [71].

5GHoul [33], mutates inputs generated by *OpenAirInter-*

[2] https://openairinterface.org/oai-5g-ran-project/
[3] https://github.com/PentHertz/OpenBTS
[4] https://www.srslte.com/

face, like CovFUZZ [80] they support both packet mutation and replay. To improve the fuzzing process LLFuzz [38] uses the cellular specification to create new standard compliant packets and to target mutations to security sensitive fields.

Finally, fuzzing lower layers (e.g., MAC) is particularly challenging because they cannot tolerate delays greater than the slot transmission time (~1 ms) [33].

4.1.2 Impact analysis

In order to assess the robustness of the baseband, it is critical to be able to detect crashes of the baseband. Common strategies include monitoring system logs via *Android Debug Bridge (ADB)* [38] or performing periodic liveness checks on the RF link or both [50]. However, detecting memory corruption remains challenging in embedded systems as they may not trigger an instantaneous crash [67]. Unlike traditional applications, basebands often lack comprehensive debugging interfaces, making automated crash detection and recovery a non-trivial task. Some devices, such as Apple's phones, are lacking such a debug mechanism or requires jailbroken devices [80]. This currently prevent using this technique to analyze them.

4.1.3 Guiding mutators

Beyond detecting crashes, effective fuzzing benefits from observing how input mutations influence the device behavior. Instrumentation can provide valuable feedback, such as code coverage metrics in emulated environments or the number of state transitions achieved for one input. These observers guide mutators toward unexplored states, improving the fuzzing efficiency, compared to purely random fuzzing.

In order to optimize the fuzzing process, 5GHoul [33] constructs a state machine by recording standard interactions between the gNodeB and the UE. This is a one-time effort. The objective of the 5GHoul fuzzer is to maximize the transitions of the protocol finite state machine. UFuzz [79] also aims to maximize state transitions, but additionally considers the time elapsed before receiving a response.

4.2 Static analysis

Static analysis examines the firmware without executing it. Some researchers search vulnerabilities manually in a disassembler, they would start by identifying important parsers and functions (e.g., handling Type Length Value (TLV) packets) and manually identify vulnerable code [86]. The effectiveness of static analysis can greatly be improved by using advanced tools. Techniques such as taint analysis or symbolic analysis tracks data flows to help identify potential vulnerabilities. Other methods such as formal verification leverage mathematical models to reason about the binary code and verify security properties.

One advantage of static analysis is that it does not suffer from the slow testing speed issues of the over-the-air testing approach. However, the firmware needs to be obtained (unpacked, decrypted, or dumped, etc.) and the tools need to support the firmware CPU architecture. Furthermore, classic static analysis challenges apply such as resolving dynamic state (indirect calls, functions pointers, state variables) and the large number of false positives.

4.2.1 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 [56] 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. Indeed, an 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, similar to the others static analysis tools, this approach requires manual implementation of mechanisms to resolve indirect function calls.

4.2.2 Parsers functions analysis

Baseband firmware relies on dedicated functions to decode binary-encoded messages and forward their content to the appropriate message handlers. BaseSpec [46] relies on the fact that implementations of L3 messages parsers adhere closely to the 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.

4.2.3 Negative testing

Similarly to the integrity analysis where some messages should be rejected when not properly protected or encrypted,

some messages shouldn't be accepted if the device is not in the correct state. In some cases, such as implementation errors, some messages are, nevertheless, accepted. Those implementation errors can also be due to the fact that the specification is not clear or even does not clearly specify the expected behavior.

The negative testing approach is explored in DoLTEst [70], which primarily focuses on the security of messages with respect to authentication. 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. This method is furthermore investigated by D. Klischies et al. [50]. They describe the Public Warning System (PWS), SMS and RRC specification in a custom format. This document is then fed to a model checker to detect undefined state of the specification and generate test cases to explore those. This approach allows them to model the state more precisely than in DoLTEst and to test invalid messages ordering.

4.2.4 Deviant behavior

Due to the complexity of the cellular specification, implementing it correctly can be a daunting task. Therefore, it is possible that implementation mismatch exists. Also, no reference implementation exists, and no official *Finite State Machine* (FSM) reference exists [83]. Therefore, researchers cannot compare implementations with a ground truth and have to rely on other methods. Researchers proposed two main methods to detect logic bugs, first, comparing the implementation with the specification, or, second, compare multiple implementations and look for deviant behavior.

As the cellular specification's size is important, manual analysis to extract security oracles can be impractical. Another solution is to consider that all the basebands should implement the same specification. Therefore, two different basebands should behave similarly when exposed to the same set of inputs. Based on this insight, DIKEUE [41] and 5GBaseChecker [83] extract the protocol's FSM for multiple devices and compare them to identify discrepancies. Such discrepancies can be due to actual protocol implementation error or imprecise protocol specification.

FSM extraction. Extracting the protocol FSM from a baseband poses significant challenges. For this, passive analysis can be done. By this, the communication between the base station and a device is recorded. However, during the messages exchanged in this recording might not be exhaustive. In order to increase the coverage of the different states, DIKEUE rely on active FSM learning. In this mode, messages are actively sent to test unexplored part of the FSM. This active method can involve a high amount of messages which may take a significant amount of time. To speed the process up, 5GBaseChecker reuse information used when exploring previous devices to reduce the amount of messages to send.

Identification of deviation in FSM. Comparing FSMs to identify differences in the implementation is difficult. Having loops in it can make the process more complex [41]. DIKEUE uses a custom method to identify those deviant behaviors between the extracted FSM's of two different basebands. Instead, 5GBaseChecker uses information from previous FSM construction to verify them on different basebands.

4.3 Emulation based fuzzing

To overcome the challenges of limited speed and lack of introspection in OTA testing, researchers explored [37, 42, 49, 61, 74] how baseband firmware could be fuzzed in an emulator. This approach provides a controlled environment that enables significantly higher fuzzing speed (e.g., 100x to 10,000x speedup compared to OTA testing [49]). This acceleration is due to the higher execution speed of desktop or server processors compared to the embedded microcontroller units (MCU). But this speedup is also due to the fact that emulation allows restoring the baseband's state after a crash from a snapshot. It does not require toggling the airplane mode on and off or rebooting the device. This method makes it also possible to control more precisely the execution, pause the emulation, modify some registers or do some snapshots to restore the state of the firmware after having tested one input. Additionally, as everything is being simulated, this approach is easily scalable as we can parallelize the fuzzing on multiple powerful servers. Moreover, emulating the firmware allows adding more easily some instrumentation to collect, for example, the coverage. This helps to get more information to improve the mutators efficiency. It also enables researchers to use more precise observers to detect crashes more accurately and detect vulnerabilities using sanitizers [61] (provided that they support binary firmware or the emulator). However, emulation requires significant engineering effort to mimic hardware behavior and peripheral interactions accurately enough to let the emulation proceed further in the firmware. It is therefore likely impossible to achieve the same fidelity as with over the air testing, in particular as the documentation for those baseband processors is not available to third parties.

Supporting new architectures can represent an important amount of manual work. First the analyzed architecture needs to be supported by the emulator. Recent research has primarily focused on MediaTek and Samsung's Shannon basebands. In contrast, Qualcomm, despite holding one of the largest shares in the baseband's market, has remained largely unexplored. Indeed, Qualcomm's baseband relies on their custom architecture, Hexagon. No emulator currently supports full system emulation for this architecture. However, Qualcomm

has recently introduced support for full-system emulation for the Hexagon instruction set in its QEMU fork [16, 22] By integrating this fork with the QEMU LibAFL fork [64], B. Produit et al. developed the first fuzzer targeting for the Hexagon architecture [72].

Paper	Target	Layers
FIRMWIRE [37]	S, M	4G: RRC, SM, 2G: CC
FIRMSTATE [42]	S	4G: RRC, NAS
BASEBRIDGE [49]	S, M	4G: RRC, NAS
LORIS [74]	S, M	4G & 5G: NAS
Hexagon Fuzz [72]	Hexagon	

Table 2: Summary of the full system emulation works. S: Shannon, M: MediaTek

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 [61] 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 (Table 2). Certain variables, memory layouts, or function pointers are initialized only during the boot process. For example, the scatter loading performed in the Shannon firmware [46] occurs during firmware initialization. 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.

One difficulty with fuzzing emulated firmware is to find how to send the messages to the parsers. Indeed, the RF frontend is not emulated. For this, FirmWire [37] inject some tasks into the RTOS in order to use its internal functions to send some inputs directly to the correct parser task.

The fuzzing of the parsers is therefore done in a state where the baseband is not connected to the network. This leads to some state variables not being set. To resolve this issue BASEBRIDGE [49] and FIRMSTATE [42] try to extract the state of a running baseband in order to restore it in the emulated firmware. However, this method requires having access to the physical device, and being able to dump the firmware out of it. Loris [74] removes the dependency on the physical device by synthesizing some potential state variables by the mean of symbolic execution.

5 Other attack surface

5.1 SIM protocol attack surface

SIM cards are important components that allow the baseband to authenticate itself to the network. It's an important component when analyzing the smartphone's security. Without it the phone cannot authenticate with the network.

Previous vulnerabilities demonstrated that SIM handling could have important security impact on the security of the UE such as a vulnerability leading to a lock pin bypass due to invalid SIM code handling [77].

Emulating the interaction between the sim card and the baseband will allow deeper states when doing full system fuzzing.

However, some work has been done to support the emulation of some of the baseband peripherals such as the SIM card. SIMurai [55] showed that it is possible to emulate the interaction with a complex device such as the SIM.

5.2 Application processor attack surface

When considering the baseband's attack surface, the *Application Processor* (AP) needs to be taken into account. Indeed, the baseband and the AP communicate through inter-process communication (IPC) and shared memory. IPC replaced the AT commands used in legacy android devices.

The AT interface is an important attack surface as it is accessible through a Bluetooth or USB interface. Also, this interface allows resetting the baseband, enabling the flashing mode, bypassing the lock screen on some devices [82]. Some of those commands can lead to network disruptions or even leave the baseband in a corrupted state requiring a new flash of the firmware to recover.

C. Mulliner et al. [68] showed that it is possible to fuzz this interface by injecting a layer between the modem driver and the telephony stack to monitor and inject command sent to the baseband processor.

I. Karim al. [44] applied fuzzing but using a custom fuzzer to generate valid AT commands using a handcrafted protocol grammar and execution time as a fuzzer feedback. Using fuzzing allows avoiding manual work to extract the relevant binary contained in the firmware images. As each vendor has its own packaging method, this can be a tedious process. In this work, the baseband's AT interface was fuzzed through the Bluetooth and the USB connection. This work highlighted that one device would accept and execute syntactically invalid command consisting of a combination of two valid but unauthorized AT command.

Some baseband constructors implement custom AT commands. They can enable the user to do privileged operations, enable debug modes, or other sensitive actions. This adds a new attack surface that needs to be analyzed. Tian et al. [82] extracted AT commands supported by some basebands. Using regular expressions and some heuristics, they identified AT commands contained in the firmware. They combined those commands with commands specified in the specification. To assess the security implication of those commands, they sent individually all the commands identified, and they manually observed their impact.

In BaseMirror, W. Li et al. [53] 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 application.

6 Future work

Baseband firmware and mobile protocols are complex and there are therefore large unexplored areas. Lower layers, such as the PDCP layer, responsible for ensuring the message confidentiality haven't been studied. Even though the PDCP has been fuzzed by LLFuzz [38] to find memory vulnerabilities, logical bugs could be interesting to explore.

Combining Over The Air testing approach with a fully emulated system would provide a very complete and flexible testing environment. However, as mentioned in the full emulation section, fully rehosting a firmware for more than analyzing some parsers is difficult, this will require interaction with peripherals that are complex, not emulated and not documented (interaction with the physical layer radio, DSPs, etc).

All the works covered in this state-of-the-art analysis are considering the cryptographic primitives to be correct. One research area could be to investigate if those cryptographic primitives are correctly used [84] and implemented. For example, if the message signature and encryption is correct, and more importantly if the message signature is properly checked before the message is decrypted.

6.1 Support for Apple and less studied devices

Market studies report that 99% of basebands are manufactured by, from the most to the least populars: MediaTek, Qualcomm, Apple, UNISOC, Samsung and HiSilicon (Huawei) [26]. So far research largely focused on MediaTek and Samsung.

Limited previous work [41, 52, 86] have targeted the analysis of baseband used in iPhones. [86] focuses on very old iPhone baseband (Intel basebands used in iPhone 4, released in 2010). [52] targets the communication interface between iOS CommCenter and Intel's baseband. Qualcomm's baseband is using another interface, not supported in this work. Also, they require a jailbroken iOS devices. [41] on the other hand analyses iPhones (XS, from 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 [3]).

Some basebands' manufacturers are not extensively looked into such as Unisoc [63] or Huawei (HiSilicon) basebands.

Firmware analysis and emulation of Qualcomm basebands have been rendered difficult by the use of their Hexagon processor architecture [87], and researchers focused on dynamic testing. Recent progress have been made on system level Hexagon emulation [22] and used for preliminary baseband emulation [72].

7 Conclusion

In this paper, we investigated the security of the baseband, 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 and memory vulnerabilities. Additionally, we stressed the importance of considering the rogue network attack model.

We analyzed three dominant research paradigms: Over-the-Air testing, static analysis, 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, including fuzzing of lower-layer protocols, analysis of Apple basebands, and advancing full-system emulation, particularly for architectures such as Qualcomm's Hexagon.

Paper	Testing	Bugs	Target
BaseSpec [46]	🔍📄	</>	Messages decoder (L3)
BaseComp [45]	🔍📄	</>	Integrity functions
BVFinder [56]	🔍	🧠	Messages specific handlers
DIKUEUE [41]	📶	</>	4G control-plane
5GBaseChecker [83]	📶	</>	5G control-plane
DoLTEst [70]	📶	</>	4G control-plane
Instruction unclear [50]	📶	</>	4G (RRC, PWS, SMS)
U-Fuzz [79]	📶		
UE Security Reloaded [21]	📶	</>	5G (RRC, NAS)
5ghoul [33]	📶	🧠	5G (MAC, RLC, PDCP, RRC, NAS)
OTABase [71]	📶	🧠	4G&5G (RRC, NAS)
LLFuzz [38]	📶	🧠	4G (PHY, MAC, RLC, PDCP)
BaseSafe [61]	🖥	🧠	
FirmWire [37]	🖥	🧠	4G (RRC, NAS), 2G (CC)
FirmState [42]	🖥	🧠	4G (RRC, NAS)
BaseBridge [49]	🖥	🧠	4G (RRC, NAS)
Loris [74]	🖥	🧠	4G&5G (RRC, NAS)
Hexagon fuzz [72]	🖥	🧠	
SIMurai [55]	🖥	🧠	SIM interface
BaseMirror [53]	🖥	🧠	RIL

Table 3: Overview of the approaches and target covered by the different works. 🧠: memory vulnerabilities, </>: logic bugs, 🔍: static analysis, 📄: comparison with specification, 📶: Over-the-air testing, 🖥: emulation

References

[1] About Lockdown Mode. https://support.apple.com/en-us/105120.

[2] CWE - CWE-617: Reachable Assertion (4.18). https://cwe.mitre.org/data/definitions/617.html.

[3] libimobiledevice · A cross-platform FOSS library written in C to communicate with iOS devices natively. https://libimobiledevice.org/.

[4] Non-Public Networks (NPN). https://www.3gpp.org/technologies/npn.

[5] NVD - CVE-2025-20750. https://nvd.nist.gov/vuln/detail/CVE-2025-20750.

[6] QMI-RIL - Replicant. https://redmine.replicant.us/projects/replicant/wiki/QMI-RIL.

[7] 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.

[8] Security Bulletins | Qualcomm Documentation | CVE-2025-21483. https://docs.qualcomm.com/product/publicresources/securitybulletin/september-2025-bulletin.html#_cve-2025-21483.

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

[10] 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).

[11] 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).

[12] WE'RE SWITCHING OFF 2G. https://ee.co.uk/2g-3g-switch-off.

[13] TS 102 226 - V17.0.0 - Smart Cards; Remote APDU structure for UICC based applications (Release 17), 2022.

[14] CVE-2023-21517: Samsung Baseband LTE ESM TFT Heap Buffer Overflow. https://labs.taszk.io/blog/post/85_ss_esm_bof/, November 2023.

[15] 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.

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

[17] 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.

[18] 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.

[19] David Berard and Vincent Fargues. How to design a baseband debugger. In *SSTIC*, 2020.

[20] Elisa Bertino, Syed Rafiul Hussain, and Omar Chowdhury. 5G Security and Privacy - A Research Roadmap.

[21] 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, New York, NY, USA, 2023. Association for Computing Machinery. https://doi.org/10.1145/3558482.3590194.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] Counterpoint Research. Global smartphone soc market share (q1 2024 - q2 2025), September 2025. https://counterpointresearch.com/en/insights/global-smartphone-apsoc-market-share-quarterly.

[27] Guillaume Delugré. Rétroconception et débogage d'un baseband Qualcomm. In *SSTIC*, 2012.

[28] 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.

[29] 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.

[30] Alisa Esage. Advanced Hexagon DIAG and getting started with baseband vulnerability research. https://zerodayengineering.com/research/slides/CCC2020_AdvancedHexagonDiag.pdf, 2020.

[31] 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. http://arxiv.org/abs/1708.04027, August 2017.

[32] 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.

[33] 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.

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

[35] Dyon Goos and Marius Muench. Overcoming State: Finding Baseband Vulnerabilities by Fuzzing Layer-2, August 2024.

[36] Marco Grassi and Xingyu Chen. Over The Air Baseband Exploit: Gaining Remote Code Execution on 5G Smartphones.

[37] 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.

[38] 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.

[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] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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/.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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/.

[55] 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.

[56] 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.

[57] 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.

[58] Norbert Ludant, Pieter Robyns, and Guevara Noubir. From 5G Sniffing to Harvesting Leakages from Privacy-Preserving Messengers.

[59] 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.

[60] 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.

[61] 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.

[62] Slava Makkaveev. Pwn2Own Qualcomm DSP. https://research.checkpoint.com/2021/pwn2own-qualcomm-dsp/, May 2021.

[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] 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.

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

[66] Kazi Samin Mubasshir, Imtiaz Karim, and Elisa Bertino. Gotta Detect 'Em All: Fake Base Station and Multi-Step Attack Detection in Cellular Networks.

[67] 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.

[68] 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.

[69] 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.

[70] 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.

[71] 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.

[72] 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.

[73] 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.

[74] 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.

[75] 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.

[76] 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/.

[77] David Schütz. Accidental $70k Google Pixel Lock Screen Bypass - bugs.xdavidhu.me. https://bugs.xdavidhu.me/google/2022/11/10/accidental-70k-google-pixel-lock-screen-bypass/, November 2022.

[78] Altaf Shaik, Ravishankar Borgaonkar, N. Asokan, Valtteri Niemi, and Jean-Pierre Seifert. Practical Attacks Against Privacy and Availability in 4G/LTE Mobile Communication Systems. http://arxiv.org/abs/1510.07563, August 2017.

[79] 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.

[80] Ilja Siroš, Dave Singelée, and Bart Preneel. CovFUZZ: Coverage-based fuzzer for 4G&5G protocols. http://arxiv.org/abs/2410.20958, October 2024.

[81] 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/.

[82] 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.

[83] 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.

[84] Mathy Vanhoef and Frank Piessens. Symbolic Execution of Security Protocol Implementations: Handling Cryptographic Primitives. In *12th USENIX Workshop on Offensive Technologies (WOOT 18)*, Baltimore, MD, August 2018. USENIX Association. https://www.usenix.org/conference/woot18/presentation/vanhoef.

[85] Ralf-Philipp Weinmann. All Your Baseband Are Belong To Us. https://archive.hack.lu/2010/Weinmann-All-Your-Baseband-Are-Belong-To-Us-slides.pdf, 2010.

[86] 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.

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

[88] Xuang Xing, Eugene Rodionov, Xiling Gong, and Farzan Karimi. Over the Air, Under the Radar Attacking and Securing the Pixel, 2023.

[89] 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.

[90] 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.