Bullet Signaling: Precise Message Spoofing Against 4G/5G Access Security

Andrea Paci[1,2,*], Matteo Chiacchia[1,2], Riccardo Marizilli[1] and Giuseppe Bianchi[1,2]

[1]CNIT National Network Assurance and Monitoring (NAM) Lab, Rome, Italy
[2]University of Rome Tor Vergata, Rome, Italy

Abstract

Among the multiple strategies to assess the robustness of cellular access networks, a key approach consists in observing how a live deployment reacts when confronted with corner-case, malformed, or spoofed signaling messages. Unlike passive auditing, this method requires precise temporal alignment and context coherence with the mobility and security establishment procedures: when a message is injected mid-exchange, it must be delivered at the exact protocol step and with a coherent prior state to avoid immediate rejection. In this work, we present a programmable framework that enables precise, state-consistent spoofing of 4G/5G signaling messages from the user side, operating entirely outside the network infrastructure. Built on top of a real User Equipment stack and Software Defined Radio, the framework exposes a high-level API that allows testers to intercept, mutate, and re-inject RRC and NAS messages at specific protocol steps, while transparently managing timing, encapsulation, retransmissions, and cryptographic protection. We demonstrate the capabilities of the framework through two new attacks. First, we identify a NAS parsing vulnerability affecting open-source core network implementations, resulting in control-plane crashes under malformed inputs. Second, and most significantly, we uncover UE Detach, a targeted spoofing primitive that forcibly disconnects a legitimate subscriber by injecting a counterfeit registration signal with the victim TMSI/IMSI. Unlike previously reported detach mechanisms, this attack requires no vicinity to the victim, no rogue base station and no downgrade, and has been experimentally evaluated across the real-world networks of four major mobile operators, finding that one of them is effectively vulnerable to immediate subscriber detachment.

Keywords

4G, 5G, Security, Attack, Vulnerability, User-testing

1. Introduction

Security testing in cellular radio networks remains a cornerstone of mobile infrastructure resilience. Over the past decade, operators and vendors have increasingly adopted formal verification frameworks, conformance suites, fuzzing campaigns, and standards-driven auditing methodologies such as 3GPP Security Assurance Specifications (SCAS), accessible via the 3GPP portal [1], to evaluate access security compliance and robustness. While these approaches have proven valuable, they primarily validate correctness under *normative and well-structured* signaling conditions.

A complementary and more adversarial perspective consists in observing how a deployed network behaves when confronted with *carefully crafted signaling corner cases.* Beyond malformed payloads, these include messages that are syntactically valid but contextually unexpected, such as an unsolicited `Identity Response`, an out-of-sequence `Attach Request` that reuses a stale security context, or a `Tracking Area Update Request` carrying inconsistent mobility or subscription parameters. Unlike bulk fuzzing, which floods the radio interface with randomized structures, targeted corner-case probing exposes latent semantic flaws that emerge only when a message is formally compliant with the specification yet deviates from the behavior the implementation implicitly expects.

Joint National Conference on Cybersecurity (ITASEC & SERICS 2026), February 09-13, 2026, Cagliari, IT
*Corresponding author.
andrea.paci@cnit.it (A. Paci); matteo.chiacchia@cnit.it (M. Chiacchia); riccardo.marzilli@cnit.it (R. Marizilli); giuseppe.bianchi@uniroma2.it (G. Bianchi)
0009-0004-6034-219X (A. Paci); 0009-0002-9368-4378 (M. Chiacchia); 0009-0003-7047-0614 (R. Marizilli); 0000-0001-7277-7423 (G. Bianchi)

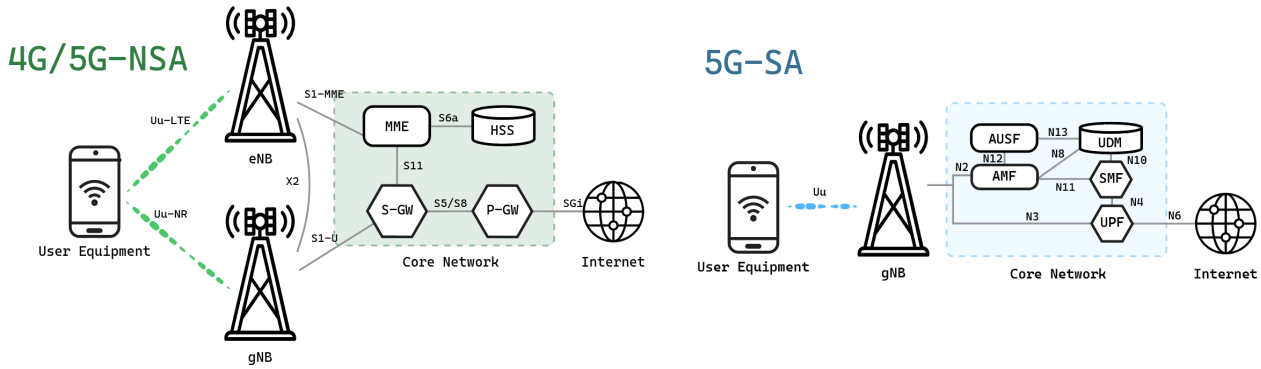

Figure 1: 4G, 5G NSA and 5G SA Network Architecture

Constructing and delivering such messages in real-time is non-trivial. The attach and mobility procedures in 4G/5G systems are inherently stateful: message validity depends not only on payload format, but also on *when* it is transmitted and with which previously negotiated context. When a spoofed message is injected mid-handshake, it must be released with correct sequence numbers, security context, cipher selection, and UE history; otherwise, the network resets or discards the exchange. Timing precision is further constrained by vendor-specific transmission windows and tolerances, and the difficulty is amplified when aiming to make such testing accessible beyond baseband specialists.

In this work, we present a *framework that enables controlled spoofing of cellular protocol messages* during active attach, handover and other mobility procedures. The system provides a high-level API exposing injection primitives while abstracting radio timing, NAS/RRC encapsulation, integrity computation, and retransmission handling, thus enabling reproducible adversarial message delivery even by non-expert testers. Our implementation relies on *Software Defined Radio* (SDR), granting full transparency and programmability at the signaling layer without requiring UE baseband chipset modifications.

Using this framework, we uncover two distinct attack classes effective in laboratory settings. The first, *PoisoNAS*, is based on a crafted NAS signaling element that consistently causes execution crashes in open-source implementations (srsRAN, Open5GS and Free5GC). While we expect commercial stacks to exhibit greater robustness, for obvious ethical and legal reasons we have *not* tested malformed NAS signalling messages with the potential to cause crashes in live operator environments.

The second attack, *UE Detach*, demonstrates full practical relevance. By spoofing the UE Identity, an attacker can forcibly detach a legitimate subscriber from the network without being in the vicinity of the victim, base station impersonation, or downgrade preconditions. While three operators rejected the forged mutation as invalid, one national operator accepted the crafted message, leading to immediate subscriber detachment. This confirms that UE Detach constitutes not merely a lab artifact, but an operationally actionable signaling-layer vulnerability in at least one live network.

Although the literature reports techniques for manipulating control-plane traffic in cellular systems, most approaches are limited to RAN-side testing and emphasize UE implementation assessment. We instead target the RAN implementation via OTA testing and substantially simplify the user interaction needed to generate or manipulate control traffic under edge conditions.

The rest of the paper is organized as follows:
Section 2 introduces the LTE and 5G architecture, protocol stack, and core signaling mechanisms relevant to this study. Section 3 presents the proposed signaling spoofing framework. Section 4 discusses the attacks identified and their exploitation via targeted signaling injection. Section 5 evaluates the framework and attacks in both controlled and real-world deployments.

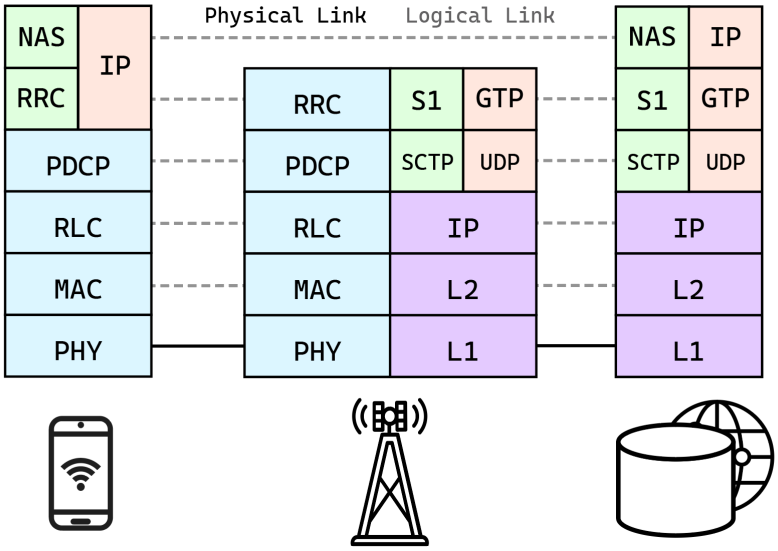

Figure 2: 4G and 5G Protocol Stack

2. Background

The focus of this paper is on standard 4G and 5G cellular architectures, composed of a User Equipment (UE), a radio access node (eNB/gNB), and a Core Network (CN). As the architectural and protocol differences between LTE and 5G are not relevant to the attacks discussed in this work, we treat both systems jointly. The CN handles mobility management, authentication, and security context establishment, while the radio access node provides RRC signaling and connectivity between the UE and the core. Figure 1 summarizes the reference architecture and signaling interfaces relevant to this study. We next provide a minimal overview of the protocol stack and registration procedures relevant to the threats discussed in this work.

2.1. LTE/5G radio protocol stack

LTE and 5G networks follow a layered protocol stack (Figure 2). Since security configuration and control-plane signaling are handled at the upper layers, this work focuses exclusively on those components, while lower layers (PHY, MAC, RLC) are not discussed further.

Packet Data Convergence Protocol (PDCP). The PDCP sublayer implements the main Access Stratum (AS) security mechanisms, providing encryption and integrity protection for control plane and, where applicable, user plane traffic. In addition to security services, PDCP supports functionalities such as packet ordering, handover assistance, and data rate optimization through techniques like Robust Header Compression (ROHC) [2].

Radio Resource Control (RRC). The RRC layer governs radio connection management and radio bearer control in LTE and 5G networks. Its responsibilities include the establishment, maintenance, and release of radio sessions, management of UE measurements, negotiation of AS security parameters and keys, and the transparent transport of Non-Access Stratum (NAS) messages [3]. Although RRC signaling is generally protected by encryption and integrity mechanisms, messages exchanged prior to AS security activation are transmitted in cleartext.

Non-Access Stratum (NAS). The NAS layer terminates control plane signaling between the UE and the Core Network and is responsible for mobility and session management. Its functions include user identification, authentication, and security control via the Authentication and Key Agreement (AKA) procedure, as well as tracking area updates, and the allocation of temporary identifiers and IP addresses.

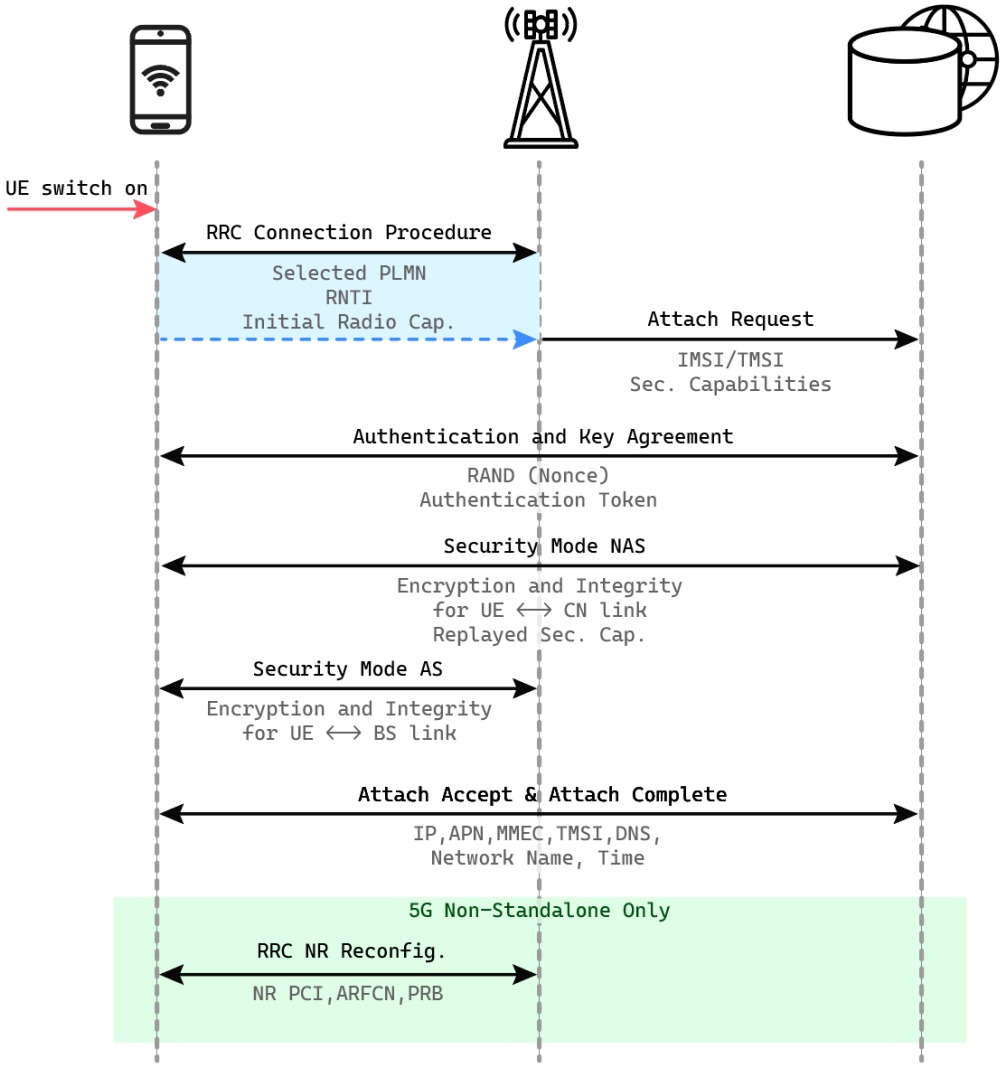

Figure 3: Registration Procedure in 4G/5G NSA

NAS signaling messages are subject to mandatory integrity protection and optional encryption [4, 5].

2.2. Attach Procedure

When a UE powers on or enters coverage, it initiates an attach (LTE) or registration (5G) procedure (Figure 3) toward the core network, managed by the MME or AMF, respectively, to obtain connectivity and mobility services. During this process, the UE provides either its permanent identity (IMSI) or a previously assigned temporary identifier (TMSI), triggering authentication, security activation, and the delivery of user-plane configuration parameters.

To limit IMSI exposure over the radio interface, the Core Network allocates a Temporary Mobile Subscriber Identity (TMSI), which is generated by the MME or AMF and conveyed to the UE in the Attach Accept or Registration Accept message. The TMSI is subsequently used for mobility-related signaling, such as paging and tracking area updates, and is periodically refreshed to reduce long-term subscriber linkability.

3. Motivation and Approach

The goal of our framework is to provide an *accessible interface for controlled manipulation of control-plane signaling*, enabling systematic assessment of network behavior under edge-case and security-relevant conditions. While in principle spoofing signaling messages for testing is enabled by the availability of open-source UE implementations such as srsRAN and OpenAirInterface, achieving this in practice typically requires invasive code changes and deep interaction with the full protocol stack. Even when the goal is limited to manipulating higher-layer signaling, all lower-layer procedures must remain correctly implemented and orchestrated for messages to be successfully delivered. As a result, directly modifying UE source code is impractical for systematic testing, as it demands detailed knowledge of low-level protocol logic and repeated code changes for each new test case.

With our framework, we make such signaling manipulations accessible to testers through a **programmable interface** that operates on top of a standard-compliant UE, enabling precise modification of selected control-plane messages while abstracting away lower-layer complexity and preserving baseline protocol behavior. Unlike approaches based on stateless fuzzing or invasive UE code modifications, our framework does *not* reimplement protocol state machines or require changes to lower-layer logic, but instead operates by selectively intercepting and modifying messages generated by an otherwise standard-compliant UE.

In the remaining subsections, we outline the framework's architectural workflow in Section 3.1 and detail the message manipulation and injection API exposed to the test programmer in Section 3.2.

3.1. Overall system workflow

Although open-source UE implementations technically allow control-plane modifications, directly altering UE behavior requires invasive changes to the codebase and careful orchestration of the full protocol stack, even when only higher-layer signaling is targeted. This makes experimentation cumbersome, as each test case demands low-level modifications while preserving correct lower-layer operation.

The proposed framework overcomes this limitation by operating on top of a standard-compliant software-radio-based UE, intercepting control-plane messages during live procedures and selectively modifying only specific data elements. By leveraging the native UE protocol execution instead of reimplementing state machines or crafting custom procedures, the framework enables precise, state-consistent signaling manipulation at defined protocol stages. This separation of message manipulation from lower-layer protocol handling preserves timing and protocol correctness, allowing testers to focus on security-relevant mutations without interacting with UE internals.

To extract and programmatically manipulate 4G and 5G control-plane data, as shown in Figure 4, the following three steps are required:

- **Extraction**: control-plane messages are intercepted at a selected protocol layer during normal UE execution. Intercepting at higher layers simplifies interaction but limits manipulation granularity, while lower-layer interception offers finer control at the cost of increased complexity.
- **Manipulation**: extracted messages are modified according to user-defined test logic, enabling targeted changes such as altering specific information elements or adjusting message sequencing.
- **Injection**: modified messages are reinserted into the UE protocol stack and transmitted along the standard execution path, thus preserving protocol state and consistency so that the network processes them as valid signaling traffic.

In our framework, the above process is implemented via two main components, which form a modified UE. As shown in Figure 4, we developed suitable *protocol-level* hooks inside a **Host UE**, namely an open source UE implementation that supports all (or most) 4G and 5G procedures and capabilities. It is used both to generate the "baseline" protocol messages for a given procedure and to deliver the crafted messages to the target network. In our case, the chosen Host UE is `srsRAN`, which provides flexible

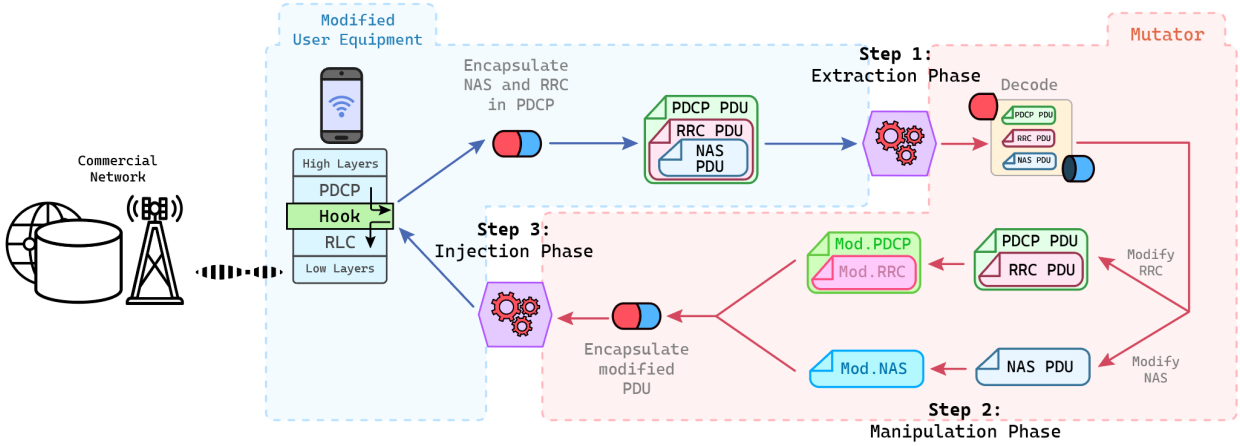

Figure 4: Framework workflow and architecture

configuration options and a clear code structure, thereby facilitating extraction of outgoing messages and injection of mutated ones.

We specifically developed message interception hooks at the **PDCP** control-plane interface[1], where signaling messages are available in cleartext and protocol state has already been established. At this point, lower-layer procedures (PHY, MAC, RLC) have been correctly executed by the host UE, while higher-layer messages (RRC and NAS) can be safely extracted, inspected, and modified without disrupting timing, retransmissions, or security context handling.

The signaling messages extracted from the Host UE are then sent to a **Mutator**, which receives and modify them according to the user-defined input. Once the mutation is applied, it re-injects the newly crafted packets into the Host UE protocol stack. This component has been implemented in Python, allowing rapid prototyping of new tests and easy integration with existing tooling.

3.2. Message manipulation API

Since this tool aims to give the user the ability to inject and/or modify existing signaling data with custom payloads, the system must accept a precise and expressive description of the desired mutation. In particular, the user must be able to identify which messages are of interest and how they should be altered, while the framework takes care of the technical details of extraction, encoding, and reinsertion into the protocol stack. This separation of concerns allows security researchers to focus on test logic instead of low-level implementation details. Formally, the system takes as input the following information:

- **What message to modify**: This specifies the selection criteria for the target message. It includes, for example, at which protocol level the message must be intercepted (e.g., RRC, NAS), which mandatory keys or Information Elements must be present, and which values or patterns must appear within the message. This allows the user to precisely target specific procedures or message types (e.g., `RRCConnectionReconfiguration`, `AttachAccept`) instead of blindly mutating all traffic.
- **How to modify it**: This defines the mutation logic to be applied once a matching message is detected. The user indicates at which layer the field should be altered, which specific field or Information Element should be changed, and what new value (or generation rule) should be used. This can range from simple constant replacement to more complex transformations (e.g., boundary values, malformed encodings, or fuzzed payloads) that are designed to trigger edge cases in the target implementation.

[1]We have preferred to operate at such relatively low layer, as the alternative of operating at the higher NAS level would exclude possibility to test the RRC layer. Rather, by working at the PDCP/RRC boundary, the framework retains the ability to exercise both RRC and NAS procedures while avoiding the additional complexity introduced by lower-layer radio mechanisms.

By structuring the input in this way, the framework can be extended to support increasingly sophisticated mutation strategies while maintaining a clear, high-level interface for the user.

The message manipulation process must also consider the multiple layers of security applied within the mobile communication stack. Since the main objective of the system is to enable the manipulation of RRC and NAS packets, both of which are independently encrypted and integrity-protected, message interception and manipulation must explicitly take this aspect into account.

The extraction process is structured as follows:

- **Outgoing Packets**: Intercepted immediately before the Host UE applies encryption and integrity protection on PDCP Packet Data Unit. This ensures that the mutator operates on unencrypted, plain-text RRC payloads that can be safely modified before transmission.
- **Incoming Packets**: Intercepted right after the Host UE deciphers the PDCP Packet Data Unit payload and removes integrity protection. At this point, the packets are again in plain-text form and can be analyzed and logged as needed.

This approach ensures that **RRC packets are always delivered to the mutator in a decrypted and integrity-free form**. Consequently, the mutator does not need to implement any RRC-level security mechanisms, as all cryptographic operations are transparently handled by the Host UE.

Handling of NAS packets. In contrast to RRC, NAS messages introduce additional complexity. Although NAS messages are encapsulated within RRC packets, they implement *independent encryption and integrity protection.* As a result, once the NAS security context is established, any direct modification of NAS payloads would invalidate the message unless the associated security fields are correctly recomputed. To address this, the Host UE forwards NAS PDUs to the mutator together with the *corresponding encryption and integrity keys.* After applying the requested mutations, the mutator recomputes the message authentication code and re-encrypts the NAS packet before reinjecting it into the UE protocol stack. This design preserves protocol correctness while remaining fully compliant with standardized NAS security procedures.

Packet representation and decoding. All intercepted packets are provided to the mutator as **raw byte sequences**. These byte streams are decoded, modified, and re-encoded by the mutator using the open-source PyCrate library [6]. PyCrate provides ASN.1 and CSN.1 runtimes along with precompiled 3GPP specifications, enabling:

- ASN.1-based decoding of RRC messages,
- Information Element–based decoding of NAS messages.

This allows the mutator to operate on structured representations rather than opaque buffers.

JSON abstraction and field selection. Once decoded, each packet is converted into a *JSON-like hierarchical structure* composed of nested dictionaries and lists that mirror the underlying ASN.1 and NAS information element trees. Directly addressing fields in such structures would normally require full knowledge of the exact JSON path, which is often complex and variable across different protocol executions. To mitigate this, the framework introduces a *JSON-handling abstraction layer* that automatically generates *virtual paths* over the decoded structure. These paths allow users to select fields using pattern-based expressions rather than fixed, fully qualified JSON paths. For example:

`RRC.*CellAccessRelatedInfo.*cellIdentity`

matches any occurrence of the `cellIdentity` field within an `RRC` message subtree containing `CellAccessRelatedInfo`. The concrete JSON path is resolved dynamically at runtime, even if the nesting structure varies across messages. In most cases, the user only needs to specify the target key to be mutated. When ambiguities arise due to multiple occurrences of the same key, selection can be refined using partial sub-paths or more restrictive patterns. This approach balances expressiveness and usability, even for complex and evolving protocol structures.

Raw byte-level injection. Beyond structured field manipulation, the injector also supports direct operation on *raw byte sequences*. This mode enables the generation of intentionally malformed packets that violate encoding rules, length constraints, or field dependencies, potentially triggering decoding failures in the target implementation. Such unstructured injection is particularly effective for exposing low-level robustness issues, including memory corruption and buffer management vulnerabilities in baseband implementations, core network components, or intermediary protocol parsers.

Injection capabilities. By providing precise control over packet contents, the injector supports:

- **Semi-structured injection**, where selected fields are mutated while preserving the remaining byte sequence;
- **Fully unstructured injection**, where arbitrary byte-level modifications are applied.

This flexibility significantly broadens the range of security tests that can be performed against cellular network components.

4. Vulnerabilities and Exploitation

This section describes the **vulnerabilities uncovered** using the proposed signaling-spoofing framework and illustrates how precise, state-consistent manipulation of control-plane messages can expose security weaknesses that remain invisible to conventional testing approaches. Rather than aiming for exhaustive vulnerability enumeration or exploit weaponization, our focus is on demonstrating how semantically inconsistent yet syntactically valid signaling messages, when injected at specific points within live procedures, can trigger unintended and security-relevant network behavior.

All vulnerabilities reported in this section were identified through *over-the-air* interactions with the target systems, treating both open-source and commercial deployments as **black boxes**. Laboratory environments were used to safely explore malformed inputs and implementation faults, while real-world testing was deliberately restricted to attacks with limited and controlled impact on subscriber availability[2].

The vulnerabilities found are the following:

The first, **PoisoNAS**, demonstrates how malformed NAS payloads can trigger *decoder failures* in open-source core network implementations, highlighting the framework's ability to exercise deep protocol corner cases.

The second attack, termed **UE Detach**, constitutes a practically exploitable signaling-layer vulnerability that enables the *selective disconnection of a legitimate subscriber* by injecting the UE network identifier of the victim during an active registration procedure. The feasibility of this attack has been validated on *live commercial networks*. The attack can use as UE network identifiers both temporary identifiers (i.e. TMSI) and permanent identifiers (i.e. IMSI/SUPI). This attack is further facilitated by weaknesses in the randomness of TMSI allocation. In particular, in some real world deployments, TMSIs are assigned in a predictable manner, or even not updated for a significant amount of time, effectively reducing the entropy of temporary identifiers. Patterns of predictable TMSI allocation have been observed and documented in [7]. Moreover, acquiring such information from a victim UE is feasible even with COTS hardware, as demonstrated by publicly available researches on IMSI Catcher [8].

4.1. PoisoNAS

Attack description. PoisoNAS is a vulnerability that emerges during the decoding phase of NAS messages, which is based on the following general observation: by injecting a specifically crafted NAS payload, it is possible to construct a *deeply nested NAS PDU composed of multiple encapsulated information elements*, where the internal structure remains syntactically valid but violates implicit assumptions made

[2] Attacks that could plausibly cause broader service disruption or infrastructure instability were *not executed on commercial networks*

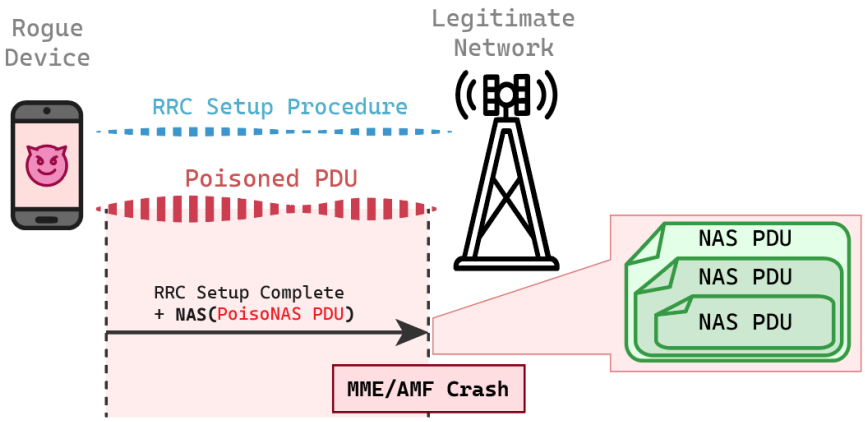

Figure 5: PoisoNAS attack description

by the decoder regarding nesting depth, container composition, or length consistency. In particular, our specifically injected message[3] embeds multiple layers of NAS containers whose declared presence, ordering, or size relationships are mutually inconsistent, yet still pass preliminary validation checks. When the Core Network performs decoding, the message triggers a potentially unhandled parsing condition that may crash the NAS decoder. Note that such vulnerability does not rely on malformed headers or invalid encoding at the byte level, but rather on semantically inconsistent combinations of otherwise standard-compliant information elements, which are difficult to reach through stateless fuzzing. The attack high-level procedure can be seen in Figure 5.

Threat model. A key characteristic of PoisoNAS is that the attack can be mounted from virtually any protocol state in which NAS messages are accepted, including both pre-authentication and post-authentication phases. When launched in a pre-authentication state, no valid SIM or subscriber credentials are required, which significantly lowers the bar for an attacker to reach the vulnerable code path. In the evaluation setup, the standard Attach Request or Registration Request—the first NAS message in the registration procedure—was overwritten with the maliciously crafted payload; upon reception and decoding, this single message was sufficient to crash the network entity responsible for NAS handling.

Framework-Based Attack Realization. The input API given for our framework, in the instance of 4G, is shown in the Appendix.

In this case, the injector selects the NAS message identified by the key `"Attach Request"`, and for that message it directly manipulates the NAS byte sequence (accessed by intercepting at the RRC level) with the crafted payload. This approach allows precise control over the NAS content while relying on the existing RRC procedure to transport the modified bytes toward the core network.

Impact. PoisoNAS was evaluated exclusively against open-source core network implementations, as exercising malformed NAS payloads on commercial infrastructure would pose unacceptable ethical and operational risks. As shown in Section 5.3, all tested open-source cores consistently crashed upon processing the crafted message, indicating that the potential impact of this vulnerability class is non-negligible. While we make no claims regarding the presence of PoisoNAS in commercial deployments, a comparable decoder robustness issue in operational cores would represent a significant risk, as a single NAS decoding failure could affect mobility management services for multiple base stations served by the same control-plane function.

[3]For responsible disclosure reasons, the exact message layout and byte sequence are omitted. Nevertheless, even if the structural description provided is intentionally abstracted, it appears sufficient to characterize the vulnerability class, as the observed behavior is consistent with decoder robustness issues arising from insufficient validation of nested NAS container structures.

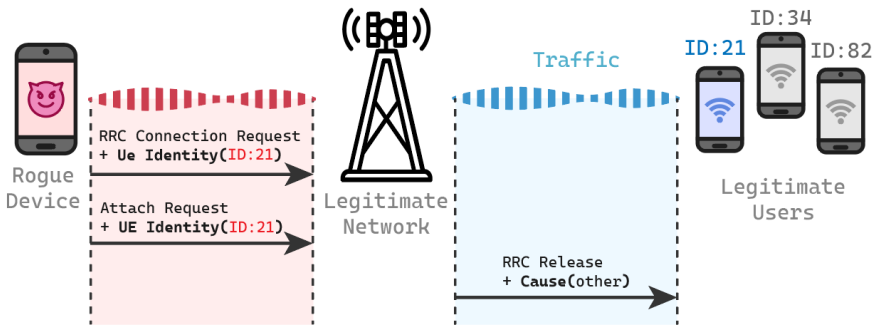

Figure 6: UE Detach attack description

Potential mitigations. This is not a protocol-level issue, but an implementation flaw in NAS decoding logic, and should therefore be addressed through hardened decoder implementations. Practical countermeasures include validating nested container structures, enforcing consistency of length and presence constraints, and ensuring that unexpected parsing conditions are handled through controlled error paths, allowing invalid messages to be safely rejected without impacting control-plane stability.

4.2. UE Detach

Attack description. UE Detach is a vulnerability that enables the selective deregistration of a chosen subscriber by knowing only its TMSI or IMSI/SUPI. In a scenario where multiple users are attached to a public-land mobile network, an attacker can impersonate the victim by initiating a new registration using the victim's identifier, causing the network to tear down the legitimate user's session. The impact of such an attack is significant, as it allows a single crafted interaction to disrupt the communication of one or more targeted users without causing collateral damage to unintended subscribers and without generating the conspicuous interference associated with techniques such as *jamming*. In the evaluated setup, the attack uses two messages, typically sent in a pre-authentication phase. The first is the RRC Connection Request, where the victim's TMSI is used as the contention resolution identity during the random access procedure[4]; the second is the Attach/Registration Request NAS message, which embeds one of the victim's identifiers, which can be the IMSI or the TMSI of the victim, for network authentication. By carefully selecting the identifier, the attacker can induce the core network to associate subsequent signaling with the spoofed context and ultimately force the legitimate UE to be detached. The attack high-level procedure can be seen in Figure 6.

Threat model. The threat model is equivalent to PoisoNAS. Since spoofing takes place before the authentication phase, any attacker who can attach to a network operated by the same provider as the victim and who can obtain one of the victim's identifiers, such as TMSI or IMSI, can launch the attack. Consequently, no valid SIM bound to the victim is required; only the ability to transmit pre-authentication signaling and knowledge of a usable identifier are needed. In the evaluation setup, the RRC Connection Request and the standard Attach Request or Registration Request—the first NAS message in the registration procedure—were mutated so that it carried the victim's identifier instead of the one generated by the open-source UE stack. Upon reception and processing by the core network, this spoofed message led to the immediate detachment of the legitimate subscriber associated with that identifier.

Framework-Based Attack Realization. The input API given for our framework, in the instance of 4G and by using TMSI as victim identifier, is shown in the Appendix.

[4]Extensive testing showed that the UE identifier used during Random Access has no observable impact on whether the victim is released. Nevertheless, this additional mutation was retained to preserve consistency in the attack procedure.

In this case, two messages are selected—RRC Connection Request and Attach/Registration Request—and their relevant fields are overwritten with the victim's identifier (e.g., TMSI or IMSI/SUPI). By consistently injecting this identifier across both messages, the attacker aligns the radio access and NAS layers with the victim's context, causing the network to associate the new signaling session with the targeted subscriber and thereby enabling selective deregistration.

Impact. UE Detach was evaluated against both open-source network implementations and commercial networks. In the latter case, the victim UE was always a device under our direct control, in order to avoid disconnecting unintended users. As detailed in Section 5.3, two open-source core networks were found to be vulnerable to this scenario, whereas among the commercial operators only a single network exhibited the vulnerability. In all vulnerable configurations, the observable outcome is the immediate disconnection of the targeted UE under our control, confirming the feasibility of selective deregistration in realistic deployment settings. The *disconnection interval* is not directly controllable by the attacker. It depends on how the victim's baseband firmware handles the reported release cause and on vendor-specific recovery behavior. In practice, some smartphones promptly reconnected to the legitimate network, while others required a noticeably longer time before restoring service. Because this attack targets the NAS layer, and therefore the Core Network, it can disconnect users attached to the same core instance and potentially to other cores that share the same vulnerable component. In practice, this gives the attack a wide geographical reach, since a single core typically serves many cells spread across different areas. As a consequence, the attacker and the victim do not need to be connected to the same eNB/gNB for the victim's session to be disrupted.

Potential mitigations. A practical mitigation for this attack, which corresponds to the behavior observed on networks not vulnerable to it, is to verify the legitimacy of the UE before disconnecting the existing subscriber bound to a given identifier. Rather than immediately tearing down the old context when a new connection appears with the same TMSI or IMSI/SUPI, the network should first confirm that the newly connecting UE is actually the legitimate owner of that identifier. There are legitimate situations in which a UE does not properly notify the network of a disconnection (for example, due to coverage loss or an abrupt reboot), so when it later reconnects, the network may momentarily see two sessions associated with the same identifier. This scenario can arise in normal operation and is not, by itself, proof of an attack. However, the network must still verify the authenticity of the new connection bound to that identifier, for instance by enforcing a fresh Authentication procedure. Successfully completing authentication demonstrates that the connecting UE legitimately owns the identifier, allowing the network to safely drop the stale context associated with the same identifier while preserving service for the real user.

5. Evaluation

To evaluate the capabilities and practicality of the proposed framework, a series of experiments were conducted on both commercial and open-source network deployments. This dual setting made it possible to validate the tool's effectiveness in triggering edge-case scenarios under controlled laboratory conditions and then verify whether similar behaviors appear in real-world infrastructures. Within the testing pipeline, the open-source RAN implementations served as the primary targets during development and initial experimentation, in both 4G and 5G configurations. The framework was iteratively refined using these environments until vulnerable or anomalous behaviors were identified, at which point the corresponding test cases were ported to commercial networks to assess their impact in operationally realistic settings. An overall representation of the experimental setup can be seen in Figure 7.

5.1. Laboratory experiments

Laboratory experiments were primarily used to validate the correctness and reliability of the signaling interception and re-injection mechanisms, and to safely explore malformed and semantically inconsistent

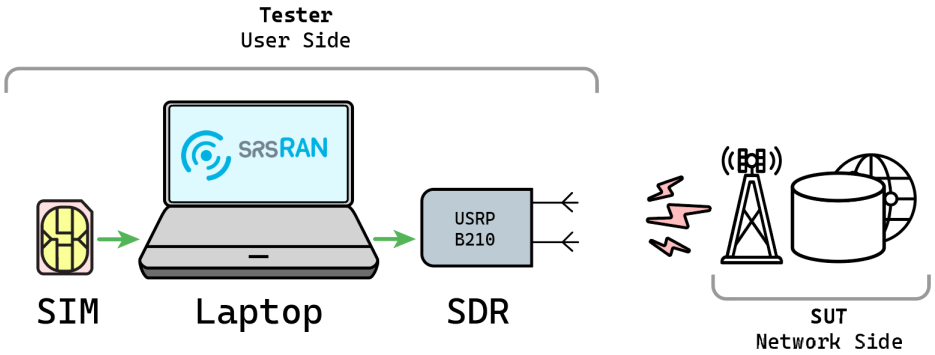

Figure 7: Experimentation setup used for both Open Source Network assessment and Commercial Network assessment

inputs that would be inappropriate to test on live networks.

These experiments were set up as follows. One host PC ran the injector framework, consisting of a modified `srsUE` instance and the injector module responsible for manipulating the extracted messages. An Ettus SDR B210[9] was attached to this host to provide the radio interface towards the target system. The target, an open-source RAN, was deployed on a second host PC equipped with its own SDR. In the 4G scenario, both the RAN and the core network were implemented using `srsRAN 4G` [10]. In the 5G scenario, the RAN was based on the `srsRAN Project` gNB [11], while the core network was alternatively provided by `Open5GS 2.2.7`[12] and `free5GC v3.4.0`[13]. Since this was a controlled lab environment, no physical SIM cards were required; instead, a soft-SIM approach was used. By controlling both the network and the UE, it was possible to provision subscriber credentials and perform all SIM-related operations purely in software. In the 5G setup, an OctoClock[14] was used as a common timing reference for the UE and RAN SDRs to improve synchronization stability and reduce radio-layer issues during long-running experiments.

5.2. Real world experiments

Testing on real commercial networks posed additional challenges, primarily because the target infrastructure was outside our control. For each of the four major Italian operators, it was first necessary to perform a frequency scan to identify which carriers were present in the area and which cells offered sufficient signal quality to run the experiments. This scanning phase was carried out using a tool from the `srsRAN` suite capable of sweeping a given band and reporting detected cells, along with their basic parameters. Another constraint was SIM handling. Since the commercial cores are not under our control, the internal cryptographic material is not directly accessible, so the experiments relied on real SIM cards from the target operators. The SIMs were interfaced via a standard PC/SC-compatible SIM card reader[15] by Sysmocom, which allowed issuing APDU commands and performing the operations required by the UE stack to authenticate and establish secure sessions[5].
Working with commercial networks also introduced two major limitations.

- **No 5G standalone availability**: In the tested area, only 4G and 5G Non-Standalone were available, as full 5G SA coverage remains unavailable in Italy. Consequently, the evaluation on public infrastructure focused on LTE and EN-DC scenarios.
- **Restricted test space**: Attacks targeting other users were only performed in a controlled way, using victim devices and SIMs under our direct control, so as not to affect uninvolved subscribers. In contrast, tests that could plausibly trigger logic faults or memory-related crashes in operator

[5]Even if the tool is configured to accept a SIM card via a SIM reader, the specific two attacks demonstrated—PoisoNAS and UE Detach—do not require a SIM card. A SIM could prove useful in testing scenarios subsequent to authentication, such as post-authentication message flows or stateful interactions.

equipment were deliberately not executed on commercial networks, to avoid risking service disruption or damage to production infrastructure.

Finally, we emphasize that all experiments affecting subscriber availability were conducted in strict adherence to ethical principles and were performed exclusively on devices and SIMs under the authors' direct control.

5.3. Results

These tests covered a range of experimental setups, separated into 4G and 5G network scenarios. Due to the lack of 5G standalone coverage in the country where the experiments were conducted, the evaluation of commercial operators was limited to 4G networks only. In total, four commercial Italian mobile network operators were assessed (anonymized as MNO1–MNO4), together with one open-source 4G deployment based on srsRAN. For 5G, the analysis focused on open-source environments, using srsRAN Project as the RAN and Open5GS and Free5GC as 5G Core implementations. Results can be seen in Table 1.

Table 1
Results obtained from the two main attacks found with the Framework

	4G					5G	
	srsRAN	**MNO1**	**MNO2**	**MNO3**	**MNO4**	**Open5GS**	**Free5GC**
PoisoNAS	✓	?	?	?	?	✓	✓
UE Detach	✓	X	X	✓	X	✓	X

✓Vulnerable, **X** Not vulnerable, **?** Not tested

The results indicate that **all** open-source deployments under test (srsRAN, Open5GS, and Free5GC) are vulnerable to the PoisoNAS attack. This indicates that the vulnerability class is not limited to a single implementation; however, no claims are made regarding its presence in commercial cores since PoisoNAS was not tested against any real world operational infrastructure. The erroneous code that triggers the exception in the Open5GS instance when the PoisoNAS attack is launched is shown in the Appendix, which highlights the exact location responsible for crashing the AMF component. For the UE Detach attack, only srsRAN and Open5GS were found to be vulnerable, whereas Free5GC correctly enforced the relevant countermeasures and resisted the selective deregistration attempts by authenticating the attacker before deregistering the victim user. In the commercial ecosystem, only one operator (MNO3) exhibited vulnerability to this attack, while the others implemented behavior consistent with previously explained mitigation.

In Figure 8, a trace captured from a real COTS UE (specifically, a Samsung S23) illustrates the `RRC Release` message sent by the network to the victim device. The capture was obtained using SCAT, the Signaling Collection and Analysis Tool, which parses Qualcomm- and Samsung-based baseband diagnostics and exports them as GSMTAP PCAPs for offline analysis in Wireshark[16].

```
▾ LTE Radio Resource Control (RRC) protocol
  ▾ DL-DCCH-Message
    ▾ message: c1 (0)
      ▾ c1: rrcConnectionRelease (5)
        ▾ rrcConnectionRelease
            rrc-TransactionIdentifier: 3
          ▾ criticalExtensions: c1 (0)
            ▾ c1: rrcConnectionRelease-r8 (0)
              ▾ rrcConnectionRelease-r8
                  releaseCause: other (1)
```

Figure 8: Wireshark trace captured via SCAT of the resulting connection release of the victim after the attack

6. Related Works

Protocol Fuzzing and Vulnerability Discovery. Protocol fuzzing is widely adopted to identify implementation vulnerabilities in cellular networks. OTABase [17] applies specification-guided mutations to security-sensitive NAS and RRC fields to test LTE baseband implementations without requiring access to internal details. Garbelini et al. [18] propose an automated fuzzing framework for 4G and 5G signaling that leverages protocol state-machine awareness to explore vulnerable execution paths, while CovFUZZ [19] enhances this approach through coverage-based feedback to improve fuzzing effectiveness during the attach procedure. Recent studies highlight the importance of stateful fuzzing for protocol-driven systems such as the 5G Core. 5GC-Fuzz [20] employs a grammar-aware, state-machine-guided approach with feedback mechanisms to prioritize underexplored signaling transitions, enabling the discovery of vulnerabilities that require specific message sequences. Similarly, 5GHoul [21] allows fine-grained manipulation of 5G protocol data units but is limited to 5G systems and relies on low-level C implementations, resulting in a higher barrier to entry compared to more abstract and scriptable frameworks.

Over-the-Air Testing and Black-Box Assessment. Over-the-air (OTA) testing has proven crucial for evaluating real-world deployments where direct access to infrastructure is unavailable. LLFUZZ [22] provides a dynamic testing framework for detecting memory corruption vulnerabilities in lower-layer cellular baseband implementations through OTA interactions, addressing the challenge of assessing black-box devices without instrumenting target code. The emphasis on OTA methodologies reflects a broader trend in mobile security research, where researchers must evaluate commercial equipment without privileged internal access, requiring careful design of test oracles and crash detection mechanisms.

Baseband fuzzing. Recent work has shifted from stateless fuzzing toward more holistic analysis of cellular systems. FirmWire [23] demonstrated emulation of monolithic baseband firmware with introspection support for Exynos and MediaTek platforms. Addressing the stateless nature of such emulation, Basebridge [24] introduces a state-transfer mechanism that synchronizes a physical device with an emulator, enabling high-fidelity exploration of late-stage protocol interactions. Despite these advances, extending emulation-based testing to other chipsets remains challenging: Glockow et al. [25] showed that firmware encryption and the proprietary Hexagon DSP hinder full-stack emulation of Qualcomm-based basebands, leaving them largely inaccessible to existing frameworks. While these works significantly advance UE-side vulnerability analysis, they focus primarily on user equipment. In contrast, our approach targets RAN and core network components, enabling the assessment of vulnerabilities in network infrastructure rather than in the UE.

Disconnection-type attacks. A few works works describe attacks that achieve similar effects by disconnecting a victim using one of its identifiers[26, 27, 28]. While the overall goal is comparable, these approaches differ from ours in that they primarily target the RRC layer rather than NAS, and their impact is confined to victims attached to the same radio cell or local radio network. As a consequence, their geographical scope is significantly narrower than an attack that operates at the Core Network level and can affect users across multiple eNBs/gNBs served by the same core, or even across different cores sharing vulnerable components. Furthermore, existing solutions rely only on spoofing the victim's TMSI, whereas the vulnerability presented in this work can exploit both TMSI and IMSI as spoofable identities. This broader set of exploitable identifiers increases the flexibility of the attack and may enable execution even in scenarios where temporary identifiers are frequently refreshed or partially protected.

7. Conclusion

In this paper, we presented a framework that enables precise spoofing of 4G and 5G control-plane signaling from the user side, exposing a programmable interface that makes state-consistent security testing simple and usable. By operating on top of a standard UE implementation, the framework allows crafted RRC and NAS messages to be injected under protocol-compliant conditions without modifying lower-layer logic or network infrastructure.

Using this approach, we identified two vulnerability classes. PoisoNAS exposes robustness issues in NAS parsing within open-source core networks, while UE Detach demonstrates that a legitimate subscriber can be forcibly disconnected by spoofing its identity during registration, without rogue base stations, proximity to the victim, or protocol downgrades. Tests on commercial networks show that this behavior can occur in real deployments.

Acknowledgements

This work was partially supported by the EU under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnerships "SEcurity and RIghts In the CyberSpace" (PE00000014 - program "SERICS") and “Telecommunications of the Future” (PE00000001 - program “RESTART”).

Responsible disclosure

We followed a responsible disclosure process and notified the affected and/or relevant parties/MNOs prior to publication. At the time of writing, it is unknown if the vulnerabilities persist. In the article, we have omitted or redacted sensitive details that could enable misuse.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT in order to: Grammar and spelling check, Paraphrase and reword. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication’s content.

References

[1] 3rd Generation Partnership Project (3GPP), 3GPP Portal, 2026. URL: https://www.3gpp.org.

[2] 3GPP, Packet Data Convergence Protocol (PDCP) specification, Release 15, 2018. URL: https://www.etsi.org/deliver/etsi_ts/138300_138399/138323/15.02.00_60/ts_138323v150200p.pdf.

[3] 3GPP, Radio Resource Control (RRC) Protocol specification, Release 15, 2019. URL: https://www.etsi.org/deliver/etsi_ts/138300_138399/138331/15.07.00_60/ts_138331v150700p.pdf.

[4] 3GPP, Evolved Universal Terrestrial Radio Access Network (E-UTRAN), S1 Application Protocol (S1AP), 2014. URL: https://www.etsi.org/deliver/etsi_ts/136400_136499/136413/12.03.00_60/ts_136413v120300p.pdf.

[5] 3GPP, Non-Access-Stratum (NAS) protocol for 5G System, 2018. URL: https://www.etsi.org/deliver/etsi_ts/124500_124599/124501/15.00.00_60/ts_124501v150000p.pdf.

[6] P1SEC, Pycrate, 2024. URL: https://github.com/P1sec/pycrate.

[7] A. Paci, M. Chiacchia, G. Bianchi, 5gmap: Enabling external audits of access security and attach procedures in real-world cellular deployments, Computer Communications 234 (2025) 108091. URL: https://www.sciencedirect.com/science/article/pii/S0140366425000489. doi:https://doi.org/10.1016/j.comcom.2025.108091.

[8] A. Paci, G. Bologna, I. Palamà, G. Bianchi, Flashcatch: Minimizing disruption in imsi catcher operations, in: 18th ACM Conference on Security and Privacy in Wireless and Mobile Networks,

WiSec 2025, Association for Computing Machinery, New York, NY, USA, 2025, p. 124–135. URL: https://doi.org/10.1145/3734477.3734705. doi:10.1145/3734477.3734705.

[9] National Instruments, USRP B210 (Board Only), 2022. URL: https://www.ettus.com/all-products/ub210-kit/.

[10] SRS, srsRAN_4G: An Open Source 4G LTE and 5G NR Software Radio Implementation, 2024. URL: https://github.com/srsran/.

[11] SRS, srsRAN Project: open-source 5G CU/DU from SRS, 2026. URL: https://github.com/srsran/srsRAN_Project.

[12] Open5GS: Open Source implementation for 5G Core and EPC, 2025. URL: https://open5gs.org/.

[13] free5GC: Linux Foundation project dedicated to implementing 3GPP Release 15 and beyond core networks, 2025. URL: https://www.free5gc.org/.

[14] National Instruments, OctoClock CDA-2990, 2024. URL: https://www.ettus.com/all-products/octoclock/.

[15] Sysmocom, Omnikey CardMan 6121 USB CCID interface, 2025. URL: https://shop.sysmocom.de/Omnikey-CardMan-6121-USB-CCID-interface-2FF-sized/cm6121.

[16] fgsect, SCAT: Signaling Collection and Analysis Tool, 2025. URL: https://github.com/fgsect/scat.

[17] C. Park, M. Egli, B. Oh, T. D. Hoang, S. Jeong, M. Crettol, I. Yun, M. Payer, Y. Kim, OTABase: Enhancing Over-the-Air Testing to Detect Memory Crashes in Cellular Basebands, in: Proceedings of the Annual Computer Security Applications Conference (ACSAC), ACM, 2025.

[18] M. E. Garbelini, Z. Shang, S. Chattopadhyay, S. Sun, E. Kurniawan, Towards automated fuzzing of 4g/5g protocol implementations over the air, in: GLOBECOM 2022-2022 IEEE Global Communications Conference, IEEE, 2022, pp. 86–92.

[19] I. Siroš, D. Singelée, B. Preneel, Covfuzz: Coverage-based fuzzer for 4g&5g protocols, in: 2025 IEEE 10th European Symposium on Security and Privacy (EuroS&P), IEEE, 2025, pp. 737–754.

[20] Y. Sun, X. Liu, Q. Sun, J. Wang, L. Tian, J. Liu, 5gc-fuzz: Finding deep stateful vulnerabilities in 5g core network with black-box fuzzing, in: IEEE INFOCOM 2025-IEEE Conference on Computer Communications, IEEE, 2025, pp. 1–10.

[21] M. E. Garbelini, Z. Shang, S. Luo, S. Chattopadhyay, S. Sun, E. Kurniawan, 5Ghoul: Unleashing Chaos on 5G Edge Devices via Stateful Multi-Layer Fuzzing, IEEE Transactions on Dependable and Secure Computing 22 (2025) 6230–6247. doi:10.1109/TDSC.2025.3582093.

[22] T. D. Hoang, T. Oh, C. Park, I. Yun, Y. Kim, LLFUZZ: an over-the-air dynamic testing framework for cellular baseband lower layers, in: Proceedings of the 34th USENIX Conference on Security Symposium, SEC '25, USENIX Association, USA, 2025.

[23] G. Hernandez, M. Muench, D. Maier, A. Milburn, S. Park, T. Scharnowski, T. Tucker, P. Traynor, K. R. B. Butler, FirmWire: Transparent Dynamic Analysis for Cellular Baseband Firmware, in: Symposium on Network and Distributed System Security (NDSS) , 2022.

[24] D. Klischies, D. Goos, D. Hirsch, A. Milburn, M. Muench, V. 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), 2025, pp. 1101–1119. doi:10.1109/SP61157.2025.00142.

[25] L. Glockow, R. Shriwas, B. Produit, Securing the airwaves: Emulation, fuzzing, and reverse engineering of iphone baseband firmware, in: TROOPERS25 Conference, 2025. Talk presented at TROOPERS25.

[26] T. Dayaratne, N. D. Pham, V. Vo, S. Lai, S. Abuadbba, H. Suzuki, X. Yuan, C. Rudolph, From description to detection: Llm based extendable o-ran compliant blind dos detection in 5g and beyond, 2025. URL: https://arxiv.org/abs/2510.06530. arXiv:2510.06530.

[27] S. R. Hussain, M. Echeverria, I. Karim, O. Chowdhury, E. 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, CCS '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 669–684. URL: https://doi.org/10.1145/3319535.3354263. doi:10.1145/3319535.3354263.

[28] H. Kim, J. Lee, E. Lee, Y. Kim, Touching the untouchables: Dynamic security analysis of the lte control plane, in: Proceedings - 2019 IEEE Symposium on Security and Privacy, SP 2019,

Proceedings - IEEE Symposium on Security and Privacy, Institute of Electrical and Electronics Engineers Inc., 2019, pp. 1153–1168. doi:10.1109/SP.2019.00038, publisher Copyright: © 2019 IEEE.; 40th IEEE Symposium on Security and Privacy, SP 2019 ; Conference date: 19-05-2019 Through 23-05-2019.

Appendix A: Injection input for PoisoNAS attack

The following JSON snippet represents the input supplied to the injector to trigger the PoisoNAS attack in a 4G setup. To prevent misuse, the exact payload bytes have been redacted, while preserving the overall structure and the fields used by the framework to locate and mutate the target NAS message.

poisoNAS.json

```
{
    "RRC": [
        {
            "message-field-nas": "attachrequest",
            "message-key": "dedicatedinfonas",
            "message-val": "XXYYZZ" # exact bytes redacted
        }
    ]
}
```

In this instance, the `RRC` key signals that the key-value pair we intend to modify is at RRC level. `message-field-nas` indicates that we want to select the message that contains that specific NAS key. `message-key` and `message-val` represent the key we want to manipulate and the new corresponding value to inject.

In this case, we are modifying the RRC message which contains a NAS message with the key `attachrequest`. The framework is capable of selecting a message using a key from a different level (e.g., NAS) than the one we intend to manipulate (e.g., RRC).

Appendix B: Injection input for UE Detach attack

The following JSON snippet represents the input provided to the injector to trigger the UE Detach attack in a 4G setup. In this configuration, the attack follows the TMSI-based variant, where the victim's temporary identifier is supplied as the spoofed value used to overwrite the relevant fields in the selected RRC and NAS messages. The TMSI value has been set as a *placeholder* since it depends on the actual TMSI of the victim and is not a fixed value.

UE_Detach.json

```
{
    "RRC": [
        {
            "message-field-rrc": "rrcconnectionrequest",
            "message-key": "m-tmsi",
            "message-val": "AABBCCDD"
        }
    ]
},
{
    "NAS": [
        {
            "message-field-nas": "attachrequest",
            "message-key": "m-tmsi",
            "message-val": "AABBCCDD"
        }
    ]
}
```

In this instance, the RRC key signals that the key-value pair we intend to modify is at RRC level, with the JSON specifying an overwrite of the m-tmsi field in the rrcconnectionrequest message to the value AABBCCDD. The NAS entry similarly targets the NAS level, using m-tmsi in the attachrequest message with the same injected value.

message-field-rrc and message-field-nas selects the messages containing the respective keys. message-key and message-val represent the key to manipulate and the new value to inject.

In this case, we are manipulating two distinct message in a single "execution".

Appendix C: Open5GS Code snippet

The following code snippet shows the portion of the Open5GS AMF implementation that crashes after the PoisoNAS attack. This vulnerable path is reached when the AMF processes the specially crafted NAS message sent by the injector.

lib/core/ogs-pkbuf.h snippet

```
148 static ogs_inline void *ogs_pkbuf_pull_inline(
149         ogs_pkbuf_t *pkbuf, unsigned int len)
150 {
151     pkbuf->len -= len;
152     return pkbuf->data += len;
153 }
154
155 static ogs_inline void *ogs_pkbuf_pull(ogs_pkbuf_t *pkbuf, unsigned int len)
156 {
157     return ogs_unlikely(len > pkbuf->len) ?
158         NULL : ogs_pkbuf_pull_inline(pkbuf, len);
159 }
```