RESEARCH-ARTICLE

CITesting: Systematic Testing of Context Integrity Violations in LTE Core Networks

MINCHEOL SON, Korea Advanced Institute of Science and Technology, Daejeon, South Korea

KWANGMIN KIM, Korea Advanced Institute of Science and Technology, Daejeon, South Korea

BEOMSEOK OH, Korea Advanced Institute of Science and Technology, Daejeon, South Korea

CHEOL-JUN PARK, Kyung Hee University, Seoul, South Korea

YONGDAE KIM, Korea Advanced Institute of Science and Technology, Daejeon, South Korea

Open Access Support provided by:

Korea Advanced Institute of Science and Technology

Kyung Hee University

Published: 19 November 2025

Citation in BibTeX format

CCS '25: ACM SIGSAC Conference on Computer and Communications Security
October 13 - 17, 2025
Taipei, Taiwan

Conference Sponsors:
SIGSAC

https://doi.org/10.1145/3719027.3765230
ISBN: 9798400715259

CITesting: Systematic Testing of Context Integrity Violations in LTE Core Networks

Mincheol Son*
KAIST
Daejeon, Republic of Korea
mcson@kaist.ac.kr

Kwangmin Kim*
KAIST
Daejeon, Republic of Korea
kwangmin.kim@kaist.ac.kr

Beomseok Oh
KAIST
Daejeon, Republic of Korea
beomseoko@kaist.ac.kr

CheolJun Park†
Kyung Hee University
Yongin, Republic of Korea
cheoljunp@khu.ac.kr

Yongdae Kim†
KAIST
Daejeon, Republic of Korea
yongdaek@kaist.ac.kr

Abstract

Cellular networks increasingly support critical infrastructure, yet their security remains an ongoing concern. While prior research has focused mainly on downlink vulnerabilities, uplink security—how user equipment (UE) affects the core network—has received limited attention. We study a class of uplink vulnerabilities, which we define as context integrity violations (CIVs), where an unauthenticated or improperly authenticated UE modifies the internal state of other subscribers. Prior work identified a few instances of CIVs, but the broader attack surface remains unexplored.

We present CITesting, the first framework for systematically detecting CIVs in LTE core networks. CITesting explores diverse procedure chains, tests a broad range of Information Elements (IEs), and validates behavior across UE connection states. It introduces stateful dual-UE control testing to manage victim UE state and employs a behavioral oracle to detect context modifications in black-box networks. We evaluated CITesting on two open-source (Open5GS, srsRAN) and two commercial (Amarisoft, Nokia) LTE core network implementations, identifying 29, 22, 16, and 59 distinct CIVs after post-analysis. These findings enable remote attacks including UE detachment, IMSI exposure, and presence detection attacks. Note that traditional attack models such as fake base station and active SigOver require the active attacker to be co-located in the same cell. In contrast, our attacks require the active attacker to be in the same MME region (significantly broader than a cell) as the victim UE. All findings were responsibly disclosed, and patches were contributed to Amarisoft and Open5GS.

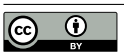
CCS Concepts

• **Security and privacy → Mobile and wireless security**.

Keywords

Cellular Security; LTE; Protocol; User Context; Security Testing

*Both authors contributed equally to this research.
†They are co-corresponding authors.



CCS '25, Taipei, Taiwan

ACM ISBN 979-8-4007-1525-9/2025/10
https://doi.org/10.1145/3719027.3765230

ACM Reference Format:
Mincheol Son, Kwangmin Kim, Beomseok Oh, CheolJun Park, and Yongdae Kim. 2025. CITesting: Systematic Testing of Context Integrity Violations in LTE Core Networks. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), October 13–17, 2025, Taipei, Taiwan.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3719027.3765230

1 Introduction

Cellular networks are foundational to modern infrastructure, connecting billions of users and enabling services such as industrial automation, autonomous systems, and smart cities. This evolution has been made possible by the massive deployment of long-term evolution (LTE) infrastructure over the past decade—with over 5 billion mobile subscriptions and 835 commercial operators [21, 28]. As these networks expand into critical domains, the need to secure mobile infrastructure becomes increasingly vital.

While security research in cellular networks has made significant progress, it has primarily focused on the downlink vulnerabilities—how network-initiated messages affect user equipment (UE). In contrast, the uplink attack surface, where UE-originated messages impact the core network, has received far less attention. This gap is not due to a lack of importance but rather stems from the practical difficulty of testing core network behavior, the limited diversity of deployed core implementations, and the regulatory and operational restrictions that hinder dynamic testing.

Among uplink vulnerabilities, a particularly dangerous and underexplored class involves violations of context integrity—situations where an unauthenticated or improperly authenticated UE can modify the internal state of other subscribers in the core network. These vulnerabilities violate a fundamental security principle: unauthenticated or improperly authenticated messages must not alter internal system state. In cellular networks, failing to enforce this principle can compromise the integrity of shared network state.

Although early versions of the 3GPP specifications did not explicitly forbid such behavior, it was long assumed to be prohibited by general security design principles. However, implementation failures in real-world systems revealed otherwise. In response, 3GPP later clarified its stance: Release 17 [4] mandates that networks must reject failed authentications cleanly, and Release 18 [5] explicitly requires that *"the network shall maintain, if any, the EMM-context and EPS security context unchanged"* upon authentication failure.

Table 1: Prior Approaches to Core Network Testing

Approach	Bug	Generation	Target	IM	PE	Dual-UE	CIV
RANsacked [10]	M	LTE, 5G	Core	-	-	-	-
CORECRISIS [20]	M, L	5G	Core	-	✓	-	-
Chlosta *et al.* [16]	L	LTE	Core	-	✓	-	-
LTEFuzz [39]	L	LTE	RAN, Core	✓	-	-	△
CITesting	L	LTE	Core	✓	✓	✓	✓

△: Partial, M: Memory Bug, L: Logical Bug, IM: All Types of Initial Messages, PE: Procedure Exploration

However, this definition remains narrow: it addresses only explicit authentication failures, whereas a broader principle is needed–no unauthenticated message, including those that do not require authentication (*e.g.*, emergency attach), should ever modify another subscriber's state. Moreover, context integrity must be preserved across entire procedure chains, not just individual messages. An attacker may initiate a procedure with a spoofed identifier, but the network must ensure that all subsequent steps leave the legitimate subscriber's state unaffected.

Definition 1. We define a **context integrity violation (CIV)** as *any instance where the core network, in response to an unauthenticated or improperly authenticated interaction—including an entire procedure chain—modifies the internal state associated with a subscriber identity that is unrelated to and not authorized to be controlled by the originating UE.*

Several prior studies (Table 1) have conducted dynamic testing of LTE and 5G core network implementations. RANsacked [10] focuses on detecting memory corruption bugs, while CORECRISIS [20] expands the scope to include logical flaws in 5G core. Both CORECRISIS and Chlosta et al. [16] employ multi-step procedure exploration to uncover logical bugs. However, none of these approaches target CIVs, and their input coverage remains limited to a small number of initial NAS message types. As a result, they do not explore broader procedure chains that may expose subtle flaws.

To systematically uncover and evaluate such CIVs, we present **CITesting**, a framework designed to detect CIVs in LTE core networks. The only prior work in this space, LTEFuzz [39], found a few CIVs but explored only a narrow and incomplete attack surface, focusing only on a limited set of individual messages without considering sequences of procedures, ignoring UE connection state variations, and lacking mechanisms to detect subtle context corruptions. In contrast, CITesting systematically explores diverse procedure chains, varies a wide range of Information Elements (IEs), validates behavior across all UE connection states, and leverages a behavioral oracle to detect subtle context inconsistencies. Table 2 summarizes the key differences between LTEFuzz and CITesting. Overall, LTEFuzz generates only 31 test cases, whereas CITesting covers between 2,802 and 4,626 test cases, depending on the core network implementation, such as support for emergency attach.

To realize these capabilities and systematically uncover CIVs, CITesting addresses three key challenges in testing core networks. First, to test extensive procedure chains while preventing combinatorial explosion, we analyze cellular standard documents to select test messages and apply semantic IE value pruning. This specification-guided approach reduces the test space while maintaining broad coverage. We then dynamically explore procedure chains by sending appropriate response messages based on network behavior, with continuous pruning based on procedure equivalence and subsequent network actions. Second, to address connection

Table 2: Comparison of LTEFuzz and CITesting

Approach	Test Cases	Test IEs				Procedure Exploration	UE States	Stateful Oracle
		SHT	MAC	SQN	Other IEs			
LTEFuzz	31	✓	✓	✓	-	-	-	-
CITesting	2,802–4,262	✓	✓	✓	✓	✓	✓	✓

SHT: Security Header Type, MAC: Message Authentication Code, SQN: Sequence Number

state-dependent behavior, we employ stateful dual-UE control testing. While DoLTEst [47] and LTEFuzz [39] considered aspects of state, they either focused on downlink messages to a single UE (DoLTEst) or performed single-message testing without controlling victim UE state (LTEFuzz). In contrast, CITesting explicitly manages the victim UE's state across procedure chains, enabling systematic evaluation under varied connection states. Third, to detect context modifications in black-box networks, where internal states are not observable, we leverage a behavioral validation approach that observes victim UE interactions as an oracle. More concretely, after each test, the victim UE sends `Ping`, `Service Request`, and `Attach Request` corresponding to its original state (**CONN.** (Connected), **IDLE**, or **DEREGI.** (Deregistered)) to verify whether the network still maintains the correct context. A mismatch between the expected and actual responses reveals CIVs. While our testing methodology is applicable to both LTE and 5G networks, we focus this study exclusively on LTE due to its widespread deployment and the persistence of unpatched vulnerabilities.

Through evaluation on two open-source (Open5GS [43], srsRAN [25]) and two commercial (Amarisoft [8], Nokia [46]) LTE core network implementations, we identified 2,354, 2,604, 672, and 2,523 CIVs, respectively. Through post-analysis, we grouped potentially overlapping vulnerabilities and reduced them to 29, 22, 16, and 59 distinct violations. We show that these vulnerabilities allow a man-on-the-side unauthorized UE to remotely disconnect victims, capture IMSIs, and detect presence. Unlike previous attacks that required Fake Base Stations (FBS) [19, 44] or signal overshadowing (SigOver) [22, 57], our attacks operate directly through the core network without requiring physical proximity, enabling attacks with expanded range beyond local cells.

We responsibly disclosed all findings to the vendors, resulting in patch by the vendor in the case of Amarisoft. Additionally, for the open-source implementations, we performed code analysis to locate the root causes, leading to patches being integrated into the official Open5GS repository.

In summary, we provide the following contributions:

- We present CITesting, the first systematic testing framework for CIVs in LTE core networks, covering diverse procedure chains rather than isolated messages.
- We introduce stateful dual-UE control testing, enabling precise victim UE state management and behavioral validation across different connection states.
- We evaluate two open-source (Open5GS, srsRAN) and two commercial (Amarisoft, Nokia) LTE core network implementations, identifying 29, 22, 16, and 59 distinct CIVs after post-analysis.
- We responsibly disclosed all identified vulnerabilities, leading to vendor-side patches in Amarisoft and code-level contributions to the Open5GS project.
- We release the CITesting framework to support further research at https://github.com/SysSec-KAIST/CITesting.

2 Background

2.1 LTE EMM Procedures

In LTE, the core network is responsible for critical operations such as authentication, key establishment, and mobility handling. These operations are defined as part of the Evolved Packet System (EPS), which consists of the radio access and core network components of LTE. In particular, these are carried out through EPS Mobility Management (EMM) procedures [5], which operate over the Non-Access Stratum (NAS) protocol to enable signaling between the core network and UEs.

Among EMM procedures, the attach procedure is fundamental for a UE to connect to the network. To initiate, the UE transmits `Attach Request` message to the core network's Mobility Management Entity (MME), including its identity. The UE can use either international mobile subscriber identity (IMSI), a permanent identity unique to each user, or global unique temporary identifier (GUTI), a temporary identity that network allocates to each UE after the attach. If the UE sends a GUTI but the network fails to find the corresponding user, it can initiate the identification procedure to request identity (usually IMSI) of the UE. Upon successful user identification, the network then initiates authentication procedure to authenticate UE and proceed with further procedures. Once authentication succeeds, the network then initiates security mode control procedure to establish security contexts for secure communication. After NAS security establishment, the network proceeds to set up bearers for the UE, and the attach procedure is completed. In emergency scenarios, UEs perform an attach procedure using the emergency attach type. This attach type enables the UE to attach the network even without a subscriber identity module (SIM).

Beyond attach procedure, UEs can attach to the network by initiating different procedures. One of them is tracking area update (TAU) procedure, which is used to inform the network of the UE's current location when moving across tracking areas. A UE may initiate service request procedure if it has no radio connection but still registered to the network. Also, either the network or the UE may initiate a detach procedure to release the UE from the network.

2.2 UE Connection States

To efficiently manage radio resources, mobility, and session, the core network assigns dedicated states to UEs. Specifically, these states—EMM, EPS Connection Management (ECM), and EPS Session Management (ESM)—collectively define the UE's connection status within the LTE network. While EMM and ECM states are essential for describing UE registration and signaling connectivity, ESM primarily relates to data session states and is relevant only when the UE is registered. In this paper, we focus on the EMM and ECM aspects of UE connection states.

The ECM state represents the signaling connectivity. If the UE has an active connection, it is in the **ECM-Connected** state; if the connection is released, the UE is in the **ECM-Idle** state. Notably, UEs in the ECM-Connected state also maintain a radio connection with the base station, whereas UEs in the ECM-Idle state do not.

The EMM state indicates the UE's registration status. A UE in the **EMM-Registered** state has successfully completed the attach procedure and is considered registered. In contrast, the **EMM-Deregistered** state indicates that the UE is not registered—either because it has not completed an attach procedure or because it has been deregistered due to inactivity or a detach procedure.

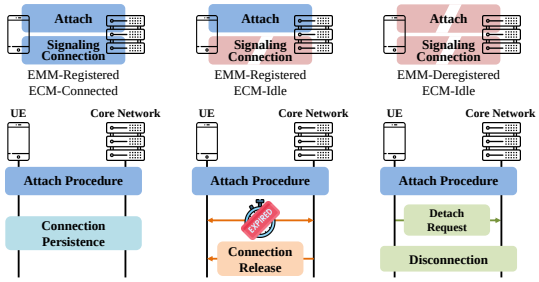


Figure 1: EMM and ECM States

UEs operate as state machines based on these EMM and ECM states, and their behavior varies accordingly. The combination of EMM and ECM states yields the following representative states:

- **EMM-Registered/ECM-Connected (CONN.)**: The UE is registered to the core network with active signaling and radio connections, representing the fully connected operational state.
- **EMM-Registered/ECM-Idle (IDLE)**: The UE remains registered but without active signaling or radio connection. It can resume communication via a `Service Request`.
- **EMM-Deregistered/ECM-Idle (DEREGI.)**: The UE is neither registered to the network nor maintains any radio connection.

2.3 UE Contexts in Cellular Networks

The UE context refers to per-user state information maintained by the core network. It includes identifiers (*e.g.*, IMSI, GUTI), connection states (EMM/ECM), security parameters (keys, algorithms), and location information. These contexts are critical for managing authentication, secure communications, paging, handover, and overall mobility management.

The UE context consists of two main components:

EPS security context: The EPS security context is formally defined in the 3GPP standards [5]. It contains security-related information maintained both at the UE and the serving network domain. The EPS security context includes:

- NAS security parameters: encryption and integrity protection algorithms and keys derived from the NAS root key (K_{ASME}).
- AS security parameters: keys and counters used at the Access Stratum (AS) for securing radio communications. Together, these parameters enable mutual authentication, secure signaling, and replay protection.

EMM context: Unlike the EPS security context, the EMM context is not formally defined in a single specification but is described through associated procedures. It includes:

- Identifiers: permanent (IMSI) and temporary (GUTI) identifiers.
- Connection states: the EMM and ECM states indicating the UE's registration and signaling status.
- Registration and subscription information used for mobility and session management. The EMM context enables the core network to manage UE connectivity, track UE location, and ensure continuity of services.

MME should maintain the integrity of these contexts. As they are crucial for providing security and reliable services to the users, their misalignment can lead to security issues and service disruptions.

3 Overview

3.1 Context Integrity: Specification and Scope

Context integrity refers to the property that a mobile core network's internal state for each subscriber is modified solely by authenticated and authorized signaling messages. A context integrity violation (CIV) occurs when an unauthenticated or improperly authenticated message alters a legitimate subscriber's context, enabling denial of service, IMSI exposure, or presence detection.

Although early 3GPP standards implicitly assumed context integrity as a security principle, they did not explicitly enforce it. Real-world deployment failures showed implicit assumptions were insufficient. Consequently, Release 17 introduced requirements for clean rejection of failed authentications, and Release 18 strengthened the language: Section 4.4.4.3 of the LTE NAS specification [5] states: *"When the authentication procedure is not successful, the network shall maintain, if any, the EMM-context and EPS security context unchanged."* Release 18 further clarified this rule across specific procedures (`Attach Request`, `TAU Request`, `Service Request`) and added explicit constraints on message origin (genuine UE). Similar updates were applied to the 5G NAS specification. However, current standards address only explicit authentication failures, leaving broader unauthenticated procedures and multi-step procedure chains vulnerable to CIVs. Thus, we define CIV (Definition 1) as the unauthorized modification of a subscriber's internal state by an unauthenticated or improperly authenticated interaction, including across entire procedure chains. Based on this definition, we argue that Section 4.4.4.3 should be revised as follows: *"Upon receipt of an unauthenticated or improperly authenticated message, or at any step of an incomplete or unsuccessful procedure that lacks proper authentication, the network shall maintain, if any, the EMM-context and EPS security context unchanged."*

Our study focuses on LTE core networks, as discussed in §1. While the CITesting approach (§4.2) and design (§5) are applicable to 5G networks, evaluating 5G deployments is beyond the scope of this study and left as future work. Our scope includes LTE procedures lacking proper authentication (*e.g.*, emergency attach and pre-authentication procedures), where an attacker can spoof a legitimate user's identity, as well as scenarios where an attacker sends messages with invalid message authentication codes (MACs) while impersonating an authenticated user. To identify such cases, we explore various procedure paths across representative EMM/ECM state combinations. We focus on protocol-compliant messages, excluding malformed messages causing memory-related issues.

3.2 Threat Model

We consider an adversary that operates as a malicious UE, aiming to manipulate the core network's internal user context associated with other subscriber identities. The adversary is capable of sending crafted NAS messages to the core network but does not possess the cryptographic keys of legitimate subscribers. As such, the attacker is limited to unauthenticated or improperly authenticated interactions—either in plaintext or with invalid integrity protection, which is commonly assumed in cellular security research [11, 32, 39, 47, 52, 53]. We assume the adversary can obtain legitimate subscriber's identifiers such as IMSI or GUTI through known passive attacks [30, 31, 52]. Unlike prior attack based on FBS [19, 44], Man-in-the-Middle (MitM) [50, 51], or SigOver [22, 42, 57], our adversary does not require proximity to the victim or control of the radio channel. Instead, the attacker directly interacts with the core network over a legitimate radio access network, enabling remote attacks without being physically near the victim. In LTE architecture, UE context is managed by the MME. Since each MME typically serves a large geographic area spanning multiple cells [1], an attacker only needs to be located within the same MME region as the victim UE. This relaxed proximity requirement enables remote attacks that target any UE served by the same MME, significantly expanding the attack surface beyond cell-level constraints.

4 Challenges & Approaches

Our goal is to develop CITesting, a CIV testing framework that systematically explores diverse procedure chains, identifies CIVs across distinct UE connection states, and employs an oracle to detect subtle context modifications. In this section, we present the key technical challenges and our approaches to address them.

4.1 Challenges in Context Integrity Testing

C1: Combinatorial explosion in procedure chaining. Exploring procedure chains in cellular networks presents two key challenges. First, initial messages contain various IEs with various possible values, each potentially causing the network to initiate different procedures. For example, an `Attach Request` with different identity types (IMSI vs. GUTI) or attach types may trigger completely different network responses and procedure sequences. Second, for each network-initiated procedure, the UE can respond in multiple ways, and each response type leads the network to execute different subsequent procedures. For instance, after receiving an `Authentication Request`, the UE might respond with an `Authentication Response` or send an `Authentication Failure` with various cause values—each triggering distinct network behaviors and subsequent procedure paths.

These factors create a massive test space: numerous initial messages with different IEs lead to different procedures; each procedure can receive multiple UE responses leading to different subsequent procedures. As these interactions compound across rounds, exhaustive testing becomes computationally infeasible, requiring a strategic exploration approach.

C2: Connection state dependent behavior in UE-network interactions. Our testing faces a unique challenge: we must consider not only the test messages sent from our malicious UE, but also the state of the victim UE that is the target of the attack. The core network processes incoming messages differently depending on the UE's EMM/ECM states. As a UE's state determines the network's procedure to initiate or proceed after receiving the incoming message, it heavily influences the maintenance of context integrity. Thus, CITesting must account for UE states to uncover state-dependent vulnerabilities—security issues that may only manifest under specific EMM/ECM states. A vulnerability that is exploitable in one victim UE state may not appear in others, or may manifest differently under varying state conditions. Moreover, because an adversary can launch attacks irrespective of the victim UE's state, it is essential to verify vulnerabilities under varying

[1] For example, CISCO ASR5500-DPC2 supports up to 128K eNodeB connections [18].

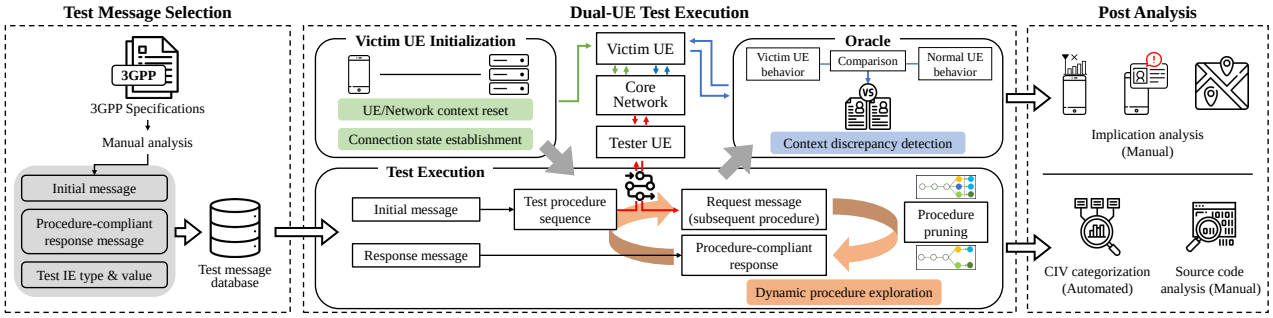


Figure 2: CITesting Design

states. During CITesting, it is necessary to reliably control the victim UE's state by establishing and maintaining specific EMM/ECM states and automating state transitions.

C3: Black-box detecting of internal context changes. Detecting unauthorized modifications to a subscriber's internal context is challenging in core networks without internal visibility, where testers cannot directly observe how malicious interactions alter the UE context stored in the core network. This creates a fundamental detection challenge: regardless of whether a network accepts, rejects, or ignores a malicious message, it may still modify the user context in ways that are undetectable through external observation. The inability to observe internal context changes directly makes detecting CIVs fundamentally difficult.

4.2 Approach

A1: Dynamic procedure exploration guided by specification analysis. To address the combinatorial explosion in procedure chaining (C1), we propose a dynamic procedure exploration guided by specification analysis. Our approach begins with a careful study of 3GPP specifications [5] to identify uplink messages and IEs that influence network behavior. For each initial message type (*e.g.*, `Attach Request`), we select a set of IEs that determine how the network handles subsequent procedures (*e.g.*, EPS mobile identity, key set identifier (KSI), and security header type) and apply semantic IE value pruning (see §5.1) to reduce redundant value combinations. As the network responds to each initial message, CITesting dynamically explores valid response sequences by transmitting procedure-compliant replies (*e.g.*, `Identity Response`, `Authentication Failure`). To avoid redundant paths and control test depth, we implement path pruning (see §5.2.2) based on two criteria: (1) semantic equivalence of the ongoing procedure flow and (2) the type of subsequent network behavior. This enables efficient exploration of deep procedure chains while maintaining broad coverage.

A2: Stateful dual-UE control testing. To uncover state dependent vulnerabilities (C2), CITesting employs a stateful dual-UE framework that precisely controls both victim and tester UEs throughout the testing process. For each test, the system transitions the victim UE to a desired EMM/ECM state and monitors the state throughout testing (see §5.2.1). Once the victim's state is established, the tester UE attempts to modify the victim's context by sending a sequence of test messages following specific procedure chains (see §5.2.2). This setup enables us to compare the impact of identical attack sequences across different UE states, revealing CIVs that only manifest under specific state conditions. Furthermore, by reproducing realistic attacker and victim roles within the network, CITesting generates attack scenarios that reflect actual deployment risks.

A3: Context discrepancy detection through victim UE behavior. To address the challenge of detecting context modifications without internal visibility (C3), we rely on the externally observable behavior of the victim UE as an indirect indicator of internal context. The key insight is that improper context modifications cause a mismatch between the victim UE's local context and the network's internal state, which manifests as abnormal behavior during service attempts—for example, the network may reject valid requests or initiate redundant procedures. Specifically, we apply distinct validation methods (§5.2.3) considering each connection state, systematically exercising the victim UE's network interactions to detect anomalies such as unexpected service rejections, connection failures, or deviations from the expected procedure sequence.

5 Design

CITesting consists of three main phases (Figure 2): 1) test message selection, 2) dual-UE test execution, and 3) post-analysis. In the first phase, we select initial and response messages to reduce the test space through 3GPP specification analysis. In the next phase, we test the target core network using a dual-UE framework (victim and tester UE). For each test, the victim UE is initialized to a specific connection state, the tester UE executes tests, and context discrepancies are detected. Starting with initial messages, we extend procedure chains by responding to network requests, systematically exploring potential vulnerabilities across different UE states. In the last phase, we analyze CIVs to identify root causes and security impacts. By examining both the conditions under which violations occur and the resulting behaviors of the victim UE, we categorize the discovered violations according to their impact, such as IMSI exposure, user presence detection, and denial of service.

5.1 Test Message Selection

We strategically select messages for testing to reduce unnecessary test cases based on the specification analysis. To explore diverse procedure chains effectively, CITesting must determine which initial message to send to trigger a procedure (§5.1.1) and which response message to return based on the network's reply to continue that procedure (§5.1.2). To this end, we manually analyze 3GPP specifications to identify representative messages for both categories—initial and response—and determine which IEs in each message influence network behavior. This specification-guided selection reduces redundant and semantically equivalent test cases while preserving

Table 3: Selected IEs of Initial Messages

Message Type (# of Messages)	Selected IE(s)
Attach Request (96)	SHT, MAC, SQN, NAS KSI, EPS mobile identity, EPS attach type, UE network capability
Detach Request (96)	SHT, MAC, SQN, NAS KSI, EPS mobile identity, Detach type
Tracking Area Update Request (96)	SHT, MAC, SQN, NAS KSI, Old GUTI, EPS update type
Service Request (6)	SHT, MAC (short), KSI and sequence number

behavioral diversity. Since the selection is driven solely by protocol specifications, and not by any specific implementation, the resulting test cases are implementation-agnostic and can be uniformly applied across different core network deployments.

5.1.1 Initial Message Selection. Initial message selection is important because different initial messages trigger distinct network behaviors. However, exhaustively testing all message variations results in a combinatorial explosion. To address this, we analyze 3GPP specifications [5] to identify initial message types and the key IEs within those messages that influence subsequent network-side behavior. We then apply semantic pruning to those IEs to eliminate value combinations that lead to the same procedural outcome.

Initial message type selection. We select the uplink NAS message types that a UE can send to initiate procedures with the core network. This includes `Attach Request`, `Detach Request`, `Service Request`, and `TAU Request`. These messages represent the main entry points for UE-initiated communication. By testing these initial message types, we can examine the network's behavior across the range of procedures that can be triggered by UE-initiated messages.

Procedure behavior-related IEs. We manually select procedure-relevant IEs by analyzing 3GPP specifications, as exhaustively testing all IEs is infeasible. For example, an `Attach Request` has 35 IEs, and each IE has at least two value sets, making it practically impossible to test all IE combinations. Therefore, we identify IEs influencing the execution and operation of network-initiated procedures through 3GPP specification analysis. The standards clearly define how specific IEs affect network behavior. For instance, the "EPS mobile identity" IE, containing UE identifier values such as IMSI or GUTI, determines whether the network performs the identification procedure. Similarly, IEs related to security parameters, such as "KSI" or "UE network capability," influence whether the network initiates the authentication and security mode control procedures. By focusing on these procedure-related IEs, we identified three to seven key IEs for each initial message type, as detailed in Table 3.

Semantic IE value pruning. We eliminate redundant test values by grouping semantically equivalent values among the selected IEs. For example, KSI values can be mutated in eight different ways (*i.e.*, 0 to 7) depending on the value agreed upon between the UE and the network [5], but many have the same semantic meaning. Therefore, we group semantically equivalent KSIs and finally consider only three cases for testing: '7' (indicating no selected key), valid (the agreed value between UE and network), and invalid (all other values). Additionally, we specifically target the victim's identifiers in our tests to determine context integrity vulnerabilities. By using these identifiers, we can directly test whether the network modifies a victim's context when receiving malicious messages from an unauthenticated source that is impersonating the legitimate user.

By carefully selecting IE types and values, we consider combinations of the selected values. This approach results in 294 distinct initial messages that serve as entry points for various procedure paths between the UE and the network, providing broad coverage while avoiding combinatorial explosion.

5.1.2 Procedure-Compliant Response Message Selection. After sending an initial message, the network may respond with a request message to perform subsequent procedures. At this point, we can either respond with a procedure-compliant response message, send another initial message, or send a procedure-incompliant response message. For efficient testing, we only consider procedure-compliant response messages for two key reasons.

First, sending another initial message during an ongoing procedure is ineffective. According to the standard [5], when the network receives a new initial message during an ongoing procedure, it either ignores the message or aborts the current procedure to process the new one. If the network ignores the message, no new behavior is observed. If it aborts the current procedure and starts processing the new initial message, this scenario becomes identical to directly starting a test with that initial message, which we already cover in our testing approach. This would lead to duplicate test cases.

Second, sending procedure-incompliant response messages (responses that do not match the expected type for the current request) is also ineffective. The network always expects a specific response message type corresponding to its request. In our experiments, all target core networks consistently ignore such mismatched responses. For example, when the network sends an `Identity Request`, it only processes an `Identity Response` and ignores other responses.

Therefore, we analyze standard documents to select procedure-compliant messages appropriate for the request messages of each procedure. For example, when encountering an `Authentication Request`, we test both `Authentication Response` with invalid values and `Authentication Failure` messages with various failure causes. Unlike our selective approach with initial messages, we test all IEs and their combinations in response messages, as these directly determine the network's next actions. This approach minimizes the number of test cases while thoroughly exploring the range of procedures that can progress from our selected initial messages.

5.2 Dual-UE Test Execution

We employ a dual-UE framework to simultaneously control the victim UE and the tester UE while varying the victim's state during testing. The tester UE operates as an attacker: an unauthenticated UE attempting to modify the context of the victim by sending NAS messages. The victim UE represents a legitimate device that can connect to the network and receive services. This dual-UE framework systematically evaluates network behavior when malicious UEs attempt context manipulation across different victim UE states. The framework performs each test in three stages: 1) victim UE initialization, 2) test execution, and 3) victim UE behavior validation. To support large-scale testing, this entire process is fully automated,

enabling CITesting to evaluate thousands of test cases across different UE states without manual intervention. We now detail these three steps and the testing process.

5.2.1 Victim UE Initialization. Before executing tests, we first move the victim UE to the desired connection state. We consider three possible connection states, each reached through different methods.
EMM-Registered/ECM-Connected (CONN.): To transition the UE to the **CONN.** state, we initiate an `Attach Request` from the UE to the network, complete the authentication and security procedures, and exchange `Attach Accept/Complete` messages. After completing this process, the UE maintains an active signaling connection with the network.
EMM-Registered/ECM-Idle (IDLE): To move the UE to the **IDLE** state, we complete the attach procedure and wait for a certain period (*Inactivity Timer*). We first perform the attachment procedure to reach the **CONN.** state, then cease all data transmission and wait for the network's inactivity timer to expire. Since UEs cannot autonomously release their signaling connections, we must wait for the network-initiated release. Consequently, testing in this state requires additional time for each test case, equivalent to the duration of the inactivity timer (*e.g.*, 15 s or 30 s). If administrative access to network parameters is available, modifying the inactivity timer value can significantly reduce the overall testing time.
EMM-Deregistered/ECM-Idle (DEREGI.): To achieve the **DEREGI.** state, we first attach the UE to the network, then send a `Detach Request` message to explicitly terminate the connection. This detachment process deregisters the UE from the network and releases the signaling connection simultaneously.

For all state initializations, the victim UE performs an IMSI attach procedure to refresh the network's UE context and generate victim UE-related values (*e.g.*, GUTI, KSI, etc.) required for testing.

5.2.2 Test Execution. After initializing the victim UE, the tester UE starts the test by sending an initial message and monitoring the network's response. The goal of the tester UE is to test procedures derivable from each initial message selected in §5.1. When the network receives a message (including both initial and response messages), it typically responds in one of three ways: sending a request to continue a subsequent procedure, ignoring the message, or rejecting the message with an explicit reject message. If the network ignores or rejects our message, that test path terminates. Additionally, the test is also terminated if the procedure completes successfully (*e.g.*, in the case of emergency attach). On the other hand, if the network responds with a request, we send procedure-compliant response messages to trigger subsequent procedures, thus extending the test procedure path. The process iteratively continues, enabling exploration of deeper procedure chains.

As we explore deeper into these procedure paths, we encounter a practical challenge: managing the growth of test sequences. For each network request message, we test multiple response message variants, causing the number of test cases to multiply with each additional step in the procedure chain. As the depth of testing increases, the number of test sequences grows considerably, making it challenging to test all 294 initial messages and their derived procedure paths. To address this growth, we implement procedure path pruning based on two primary criteria:

- **Current procedure equivalence**: Whether the message type sent in the current procedure is the same or not.
- **Subsequent network procedure**: What procedure the network executes next in response to our message.

Applying these pruning criteria at each procedure step significantly reduces redundant test paths while maintaining broad coverage. This dynamic exploration strategy adapts to each network's specific implementation characteristics, resulting in customized test sequences that carefully evaluate each equipment vendor's implementation behavior.

5.2.3 Victim UE Behavior Validation as an Oracle. To detect context modifications in black-box core networks, we validate victim UE behavior after each test step using standard-compliant network access procedures. For each UE state and core network implementation, we first establish a baseline response by executing the validation action under normal (non-adversarial) conditions. During testing, we trigger the same validation behavior and compare the observed response to the baseline; any deviation is interpreted as a CIV. The validation action and expected network behavior for each UE state are as follows.
CONN.: A ping request is sent to confirm connectivity. The network normally responds successfully, indicating that the UE remains connected. If the context is modified, the network may drop the connection or fail to respond.
IDLE: The UE issues a `Service Request` to resume communication. The network normally accepts the request. If the context is modified, the network may reject or ignore the request.
DEREGI.: The UE initiates an `Attach Request` using a stored GUTI. The network normally accepts the request directly or initiates identification or authentication procedures, depending on its implementation of GUTI-based attachment. If the context is modified, the request may be rejected, ignored, or result in behavior that deviates from the previously observed baseline.

This behavioral oracle, based on external observation of victim UE interactions, enables CIV detection without requiring visibility into internal network state.

5.3 Post-Analysis

After completing all tests, we conduct post-analysis to categorize CIVs, identify their root causes, and assess their implications. This phase involves three components: (1) merging CIVs to identify distinct conditions for context modifications (§6.3), (2) examining the source code of open-source implementations to pinpoint fundamental implementation flaws (§6.4), and (3) studying victim UE behavior to discover potential attacks (§7).
CIV categorization. CITesting explores a large test space by varying UE connection states, message sequences, and IE values. As a result, thousands of test cases can trigger CIVs, many of which reflect equivalent behaviors due to shared procedure chains and overlapping IE values. To reduce such redundancy and extract insights into the conditions that trigger CIVs, we categorize results per UE state using a two-stage process. First, we group test cases based on the observed procedure chain and the network response captured by the state-specific behavioral oracle. Next, we refine each group by identifying the subset of IEs that influence CIV. An IE is classified as a wildcard if all of its tested values consistently

lead to CIVs across all combinations of other IE values in the group. In contrast, selective IEs are those for which only specific values lead to violations. By merging test cases with wildcard IEs and preserving value-specific combinations for selective IEs, we eliminate redundancy and expose input conditions that meaningfully contribute to CIVs. Since CITesting traces all transmitted and received messages throughout testing, this process can be automated.

Source code analysis. For open-source implementations, we perform code analysis to understand the root causes of CIVs. Although our black-box testing identifies numerous CIVs across each implementation, we find that many of them originate from a small number of flawed handling routines—such as incorrect state checks or incomplete context updates. This demonstrates that a few subtle bugs can expose a broad attack surface and lead to widespread inconsistencies across different procedures and input conditions.

Implication analysis. Finally, we study the potential impact of CIVs on victim UEs. By examining observable UE behaviors following each violation, we identify several classes of attacks enabled by context corruption. These include denial of service, IMSI exposure, and presence detection.

6 Evaluation

6.1 Implementation

We implement CITesting on top of srsUE [25], an open-source LTE UE protocol stack, modifying approximately 5,700 lines of code to support the coordinated execution of two concurrently running UE instances. These instances, designated as the victim and tester UEs, interact with the core network over the air. Both run on a single Ubuntu 20.04 LTS machine, each connected to a separate USRP B210 [56] device to ensure independent radio communication.

Development and testing environment. We develop a Python-based controller (~ 450 LoC) that automates the testing workflow, including victim UE initialization, procedure exploration, and response validation. To reduce the overhead of repeated USRP initialization, the controller performs a one-time setup and executes multiple test cases. To support automated exploration, we extend the tester UE to generate arbitrary NAS messages with customizable IEs and log responses. We also implement support for the TAU procedure, not available in srsUE. For the victim UE, we add controlled state transitions and implement state-specific validation routines to assess network behavior. We also develop a script to automatically merge detected CIVs across multiple test cases.

We evaluated CITesting across four LTE core network implementations: two open-source (Open5GS [43], srsRAN [25]) and two commercial-grade systems (Amarisoft [8], Nokia [46]). Table 4 summarizes the tested versions. When testing the open-source core networks, we used the srsENB base station from the srsRAN project to interface with CITesting, and conducted experiments inside a shield box to prevent interference with commercial networks. For the commercial LTE implementations, testing was conducted using the base station integrated into each system, requiring over-the-air (OTA) transmission. The Amarisoft setup was evaluated inside a shield box under controlled OTA conditions, while the Nokia was tested in a shielded lab environment at a national research facility.

Testing process. For each initial message, CITesting performs an exploration-validation cycle comprising three steps. First, the controller initializes the victim UE by attaching it to the testing core network and transitioning it to one of the three target states (**CONN.**, **IDLE**, or **DEREGI.**), thereby establishing a baseline context (§5.2.1). Next, the controller directs the tester UE to send the selected initial message to the core network. If the network responds with a request, multiple procedure-compliant responses may be valid. The controller generates all applicable response candidates and extends the procedure chain accordingly. To manage the combinatorial growth of these chains, CITesting prunes paths based on the network's response messages and their impact on the victim UE. This process continues until one of the following termination conditions is met: the tester UE completes an attachment procedure, receives an explicit reject message, or the network ceases to respond to further messages (§5.2.2). Lastly, the controller instructs the victim UE to perform state-specific service attempt behavior, recording all responses and monitoring its behavior to detect context modifications (§5.2.3).

CITesting repeats this process for all 294 initial messages across the three victim UE states. Once all sequences have been tested, post-analysis is conducted to identify and categorize CIVs.

Table 4: CIVs Detected across Core Networks by UE State

Core Network (version)	CONN.	IDLE	DEREGI.	Total
Open5GS (v2.7.2)	822(9)/1062	966(11)/1062	566(9)/678	2354(29)/2802
srsRAN (Rel. 23.11)	870(5)/1062	870(5)/1062	864(12)/1062	2604(22)/3186
Amarisoft (2023-03-17)	240(6)/1542	192(4)/1542	240(6)/1542	672(16)/4626
Nokia (CMM 22.0)	1104(22)/1206	948(21)/1206	471(16)/630	2523(59)/3042

X(Y)/Z: *X*: # of procedure chains with a CIV. *Y*: # of merged cases after post-analysis (§6.3). *Z*: Total # of explored procedure chains (§5.2.2).

6.2 Result

The evaluation results across all four core networks are summarized in Table 4. For each implementation (Open5GS, srsRAN, Amarisoft, and Nokia) and victim UE state, the table shows the number of detected CIVs, distinct CIVs grouped through post-analysis (§6.3), and the total procedure chains explored during testing (§5.2.2).

Explored procedure chains. Beginning with 294 initial messages, CITesting generated 2,802–4,626 procedure chains across the evaluated core networks. Although the same initial set was used, chain counts varied due to differences in state-dependent logic and supported features. Notably, Amarisoft supported emergency attach—allowing unauthenticated procedure execution—which led to 1,542 chains, far more than others. Both Amarisoft and srsRAN exhibited uniform behavior across UE states, maintaining consistent chain counts regardless of state. Open5GS and Nokia showed strong state dependence: 678 and 630 chains in **DEREGI.**, versus 1,062 and 1,206 in **IDLE** and **CONN.** states. These variations demonstrate state-dependent behavior in core network implementations, where an initial message such as a `TAU Request` may lead to different procedure handling depending on the UE's state.

Table 5: Representative Examples of CIVs

Index	Core Network	Victim State	Tester UE Init. Msg. (Key IEs‡)	Procedure Chain (Tester UE ⇋ Network)	Oracle		Implication (§7)
					Victim Behavior	Network Resp.	
A1	Amarisoft	DEREGI.	Attach Req. (Emergency AT/IMSI)	↓SM Comm. » ↑SM Comp. » ↓Attach Accept » ↑Attach Comp. *(SHT #0)*	GUTI Attach	Identification	P
A2	Amarisoft	DEREGI.	Attach Req. (Emergency AT/IMSI)	↓SM Comm. » ↑SM Comp. » ↓Attach Accept » ↑Attach Comp. *(SHT #1)* » ↓EMM Info.	GUTI Attach	Ignore†	D
A3	Amarisoft	DEREGI.	Attach Req. (Emergency AT/GUTI)	↓ID Req. » ↑ID Resp. » ↓SM Comm. » ↑SM Comp. » ↓Attach Accept » ↑Attach Comp. *(SHT #1)* » ↓EMM Info.	GUTI Attach	Ignore†	D
A4	Amarisoft	CONN.	Attach Req. (Emergency AT/GUTI)	↓ID Req. » ↑ID Resp. » ↓SM Comm. » ↑SM Comp. » ↓Attach Accept » ↑Attach Comp. *(SHT #1)* » ↓EMM Info.	Ping	Ping Fail	D
O1	Open5GS	DEREGI.	Detach Req. (SHT #0/GUTI/KSI 7)	↓Service Reject	GUTI Attach	Identification	E
O2	Open5GS	DEREGI.	Attach Req. (NC 0,1,2,3)	↓Auth. Req.	GUTI Attach	Authentication	-
O3	Open5GS	DEREGI.	Attach Req. (NC 0,1,2,3)	↓Auth. Req. » ↑Auth. Fail *(Cause #21)* » ↓Attach Reject	GUTI Attach	Identification	P, E
S1	srsRAN	CONN.	Service Req. (*)	↓Service Reject	Ping	Ping Fail	D
S2	srsRAN	IDLE	Service Req. (*)	↓Service Reject	Service Request	Service Reject	D
N1	Nokia	DEREGI.	Attach Req. (Combined AT/IMSI/NC 0,1,2,3)	↓Auth. Req. » ↑Auth. Fail *(Cause #20)* » ↓Auth Reject	GUTI Attach	Identification	P

Illustrative Walkthrough (A1): The victim UE is initially in the *DEREGI.* state. (third column, §5.2.1) The tester UE sends an *Attach Request* of type *emergency attach*, embedding the *victim UE's IMSI.* (fourth column, §5.2.2) The network responds with a *Security Mode Command*, to which the tester UE replies with a *Security Mode Complete.* The network then sends an *Attach Accept*, and the tester UE completes the procedure with an *Attach Complete.* (fifth column, §5.2.2) Subsequently, the victim UE performs a *GUTI Attach.* In response, the network initiates an identification procedure to request the victim UE's identity, thereby indicating a user presence detection (P). (Oracle column, §5.2.3)
Terms: ↓: Downlink message ↑: Uplink message †: Network simply ignores the victim's uplink message. ‡: Key IEs denote the fields that influence the CIV; other IEs have no effect on the outcome. *: all combinations of tested IEs. ID: Identity, Auth.: Authentication, SM: Security Mode, Req.: Request, Resp.: Response, Comm.: Command, Comp.: Complete, AT: Attach Type, SHT: Security Header Type NC: UE Network Capability Info: Information D: Denial-of-Service, E: IMSI Exposure, P: User Presence Detection
"NC 0,1,2,3" indicates that the UE supports EPS Integrity Algorithms 0–3 (EIA0–EIA3) and EPS Encryption Algorithms 0–3 (EEA0–EEA3).

CIV detection. To detect CIVs, CITesting applies the state-specific behavioral oracle (§5.2.3). For each UE state, we established baseline behavior by observing the network's response under normal conditions and detected CIVs by identifying deviations from this baseline. In the **CONN.** state, all tested networks consistently responded to ping requests, confirming active connectivity. CIVs were detected when the network failed to respond or dropped the connection, resulting in ping failures. In the **IDLE** state, `Service Request` was normally accepted across all implementations; CIVs were identified when the network either rejected the request with EMM cause #9 or #10, or ignored it. For the **DEREGI.** state, while handling of GUTI-based `Attach Request` can vary across implementations due to the lack of standardization in 3GPP [5], all four networks in our evaluation responded immediately with an `Attach Accept` under baseline conditions, without additional message exchanges. Thus, CIVs were identified when the network instead triggered identification or authentication procedures, or failed to respond entirely.

As shown in Table 4, CITesting identified 2,354 CIVs in Open5GS, 2,604 in srsRAN, 672 in Amarisoft, and 2,523 in Nokia. These results were obtained by exploring combinations of initial message types, IE values, and procedure chains. Since many cases manifest the same root cause, we perform post-analysis (§6.3) to group violations by procedural equivalence and extract distinct CIVs.

Comparison with LTEFuzz. We compare CITesting with LTEFuzz [39], the only prior system for detecting CIVs in cellular networks. All violations reported by LTEFuzz are reproduced by CITesting; a detailed summary is provided on our project website [1]. LTEFuzz evaluates only 31 static test cases, each involving a single message in **CONN.** In contrast, CITesting systematically explores multi-step procedure chains across all three UE connection states and applies state-specific validation to uncover subtle and state-dependent violations. This yields significantly broader coverage, with 2,802–4,626 test cases per implementation.

6.3 Categorization of CIVs

Applying our categorization method from §5.3, we grouped the detected CIVs into distinct categories through an automated two-stage procedure. As summarized in Table 4, this process yielded 29 distinct CIVs in Open5GS, 22 in srsRAN, 16 in Amarisoft, and 59 in Nokia. These results demonstrate that the three key factors used in test case generation—UE connection state, procedure chain, and IE values—each meaningfully affect the manifestation of CIVs. Table 5 highlights a representative subset selected to illustrate key patterns (see Appendix B and website [1] for the full list). We describe below how each factor contributes to distinct CIV behavior.

6.3.1 Impact of UE Connection State. The occurrence of CIVs is strongly influenced by the victim UE's connection state. We found that identical message sequences could either trigger or fail to trigger a CIV depending on the victim UE's state, highlighting state-dependent differences in procedure handling. In Amarisoft, a CIV was triggered when the test UE replied to the network's `Attach Accept` with `Attach Complete` containing SHT #1, but only in the **DEREGI.** (Table 5, A3) and **CONN.** (Table 5, A4) states—not in the **IDLE**. In srsRAN, initiating a `Service Request` led to a CIV in the **CONN.** (Table 5, S1) and **IDLE** (Table 5, S2) states, but not in the

DEREGI. These observations indicate that core network behavior varies across UE states, even under identical input conditions.

6.3.2 Impact of Procedure Chains. CITesting detected violations resulting from multiple procedures that could not be triggered by a single message. For example, Nokia's core network did not exhibit a CIV when processing a single `Attach Request`—with the attach type set to "combined," the identity type set to IMSI, and supported security algorithms set to "0, 1, 2, 3"—in the absence of subsequent procedures. However, executing the authentication procedure after this message resulted in a CIV (Table 5, N1). In other cases, although the execution of additional procedures did not affect whether a CIV occurred, it influenced how the core network processed context during subsequent interactions. In Open5GS, the same `Attach Request` configuration triggered a CIV regardless of whether the authentication procedure was executed (Table 5, O2-O3). However, the network initiated an identification procedure during re-attachment only if authentication had been included.

6.3.3 Impact of IE Values. The occurrence of CIVs is often sensitive to specific IE configurations. Even when the message type remains fixed, changes in fields such as the SHT, identity type, or KSI can alter how procedures are handled or triggered. In some cases, a change in a single field is sufficient to trigger or suppress a violation. In other cases, CIVs manifest only when multiple fields are set to particular combinations. For example, in Amarisoft, the network either accepted or ignored a GUTI-based attach depending solely on the SHT value in the `Attach Complete` message (Table 5, A1–A2). In Open5GS, a `Detach Request` triggered context modification only when SHT was set to "plain," the identity type was GUTI, and KSI was set to 7 (Table 5, O1). These findings highlight that both individual IEs and their interactions can play a critical role in triggering CIVs.

6.4 Open Source Analysis

To investigate the root causes of CIVs, we performed source code analysis on two open-source implementations: Open5GS [43] and srsRAN [25]. Although our black-box testing identified 29 and 22 distinct CIVs in Open5GS and srsRAN respectively (as shown in Table 4), our source code analysis revealed these violations stem from only 2 and 5 underlying root causes in each core. Due to space constraints, we present one representative case from each implementation; remaining cases are detailed in our website [1].

RC1: Context deletion triggered by reject message transmission. In Open5GS, UE context is deleted upon transmission of any reject message, regardless of type. This includes removal of critical IMSI–GUTI mappings. Consequently, an adversary can spoof messages using the victim's identifier, inducing the network to send a reject and delete the associated context. Subsequent GUTI-based attach attempts by the victim lead to IMSI exposure, as the network, no longer recognizing the GUTI, initiates an identification procedure. We disclosed this vulnerability to Open5GS developers, who acknowledged and patched the issue in a later release.

RC2: Improper EMM state transition during authentication. In srsRAN, a spoofed Attach Request causes the network to transition the victim UE's EMM state to 'deregistered,' even before authentication completes. If authentication fails, the state is not restored, leading to a persistent mismatch between the network and the victim UE (still in **CONN.** or **IDLE**). This inconsistency results in connection failures or rejected `Service Request`. We reported the issue to the srsRAN team via responsible disclosure.

Table 6: Security Implications of Context Mismatches

Victim State	Network-side Mismatch	Victim Experience	Potential Implication
CONN.	Case 1	Ping Fail (Release)	D
	Case 2	Ping Fail (Release)	D
	Case 3	Ping Fail (Release)	D
IDLE	Case 1	Service Reject (#10)	D
	Case 2	Service Reject (#9), Ignore	D →E, P
	Case 3	Service Reject (#9)	D →E, P
DEREGI.	Case 2	Authentication, Attach Reject, ignore	D
	Case 3	Identification	E, P

Illustrative Walkthrough (First Row): The victim UE is initially in the CONN. state (Victim State). A CIV occurs that corrupts the network's internal context, causing it to consider the victim as DEREGI. (Network-side Mismatch). As a result, the network releases the connection, disconnecting the victim UE (Victim Experience), leading to a denial of service (D) (Potential Implication).
Case 1: Transition to DEREGI., Case 2: Security key mismatch, Case 3: Loss of IMSI-GUTI mapping
D: Denial-of-Service (§7.1), E: IMSI Exposure (§7.2), P: User Presence Detection (§7.3)

7 Implication

We analyze the security implications of discovered CIVs by determining whether behavioral deviations in the victim UE lead to exploitable conditions. Recall that a CIV occurs when the core network modifies internal information associated with the victim UE in response to an unauthenticated or improperly authenticated message. This creates context inconsistencies between the UE and the network—such as security key mismatches, desynchronized temporary identifiers (*e.g.*, GUTI), or divergent EMM states—as confirmed in our open-source analysis (see §6.4). These inconsistencies manifest as observable deviations in UE behavior, detected by our validation oracle (§5.2.3).

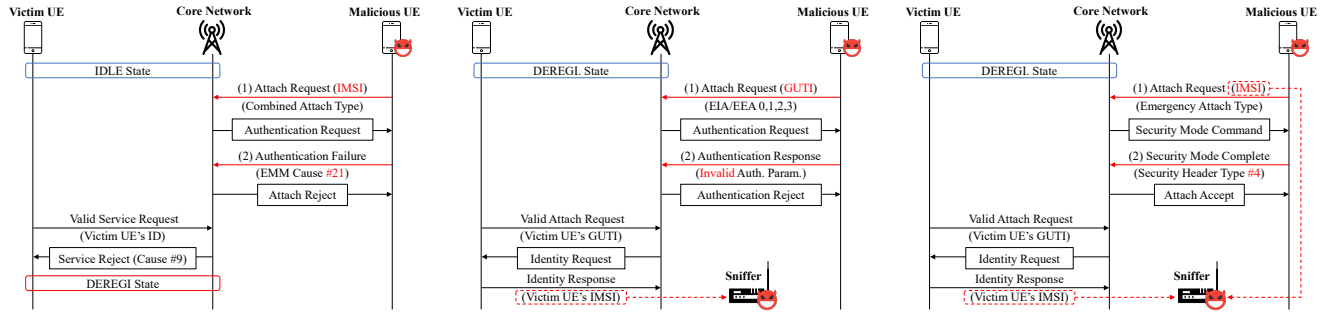
Table 6 categorizes the security implications into three types: (1) denial of service (DoS), (2) exposure of permanent identifiers (IMSI), and (3) user presence detection. These are derived by analyzing state-specific victim UE behaviors following validation across different connection states. Some cases yield a single implication, while others lead to multiple consequences depending on recovery behavior specified in [5]. Such chains are marked with a right arrow (→) to indicate implication progression.

DoS (1) can be exploited remotely, as long as the attacker resides in the same MME region—unlike prior attacks such as FBS [19] and SigOver [56], which require physical proximity for signaling injection. In contrast, implications (2) and (3) require a stronger adversary than our attack model, specifically by passively monitoring OTA signaling near the victim UE (*e.g.*, using LTESniffer [30]).

7.1 Denial of Service

Description. We identified CIVs that lead to DoS conditions, in which the victim UE is unable to initiate or maintain network connectivity. In this context, DoS refers to the failure of a UE to complete standard service procedures—such as re-attachment or session establishment—due to corrupted or desynchronized context.

CIVs leading to attack. All four tested core network implementations—Open5GS, srsRAN, Nokia, and Amarisoft—exhibited CIVs resulting in DoS conditions, as summarized in our website [1]. Notably, the impact on UE behavior varied by connection state. Following a CIV, when the victim UE in the **IDLE** state attempts to



(a) Denial-of-Service (b) IMSI Exposure (c) User Presence Detection

Figure 3: Attack Scenario

reconnect using a `Service Request`, the message was either rejected (with EMM cause #9 or #10) or silently ignored by the core network. Similar failures occur in other states. In the **DEREGI.** state, re-attachment attempts were either silently dropped or explicitly rejected during subsequent procedures such as security mode control. In the **CONN.** state, the core network immediately terminated the victim UE's connection, preventing access to data services. These observations confirm that CIV-induced context desynchronization across all UE states can result in denial of service. This behavior is exemplified by a representative case observed in the Nokia core network (Figure 3(a)). In this scenario, an adversary impersonates a victim UE in the **IDLE** state and sends an `Attach Request` containing the victim's IMSI. The core network responds with an `Authentication Request`, to which the adversary replies with an `Authentication Failure` with EMM cause #21, which is followed by an `Attach Reject` from the network. Subsequently, when the victim UE attempts to reconnect by sending a `Service Request`, the core network responds with a `Service Reject`, blocking the UE's access to network service.

Security impact. Unlike prior attacks such as FBS and SigOver, which require proximity to the victim and interference with the radio channel, our threat model does not interact with the victim's wireless channel. Instead, it sends unauthenticated uplink messages directly to the core network, enabling remote DoS from any location within the same MME region. While victim UEs typically attempt to reconnect after a reject, even brief disruption can be critical in mission-critical networks such as public safety LTE (PS-LTE) or remote healthcare systems. Furthermore, when the attack targets a permanent identifier (*i.e.*, IMSI), it can be repeated indefinitely, enabling sustained denial of service with minimal attacker effort.

7.2 IMSI Exposure

Description. Certain CIVs cause the core network to discard the GUTI-to-IMSI mapping associated with the victim UE. As a result, when the victim later attempts to re-attachment using its previously assigned GUTI, the network fails to resolve it to an existing context and initiates the identification procedure, prompting the UE to transmit its IMSI in cleartext over the air. When the attacker triggers a CIV using the victim's GUTI and monitors the air interface using a sniffer, the resulting identification procedure becomes an instance of IMSI exposure. We also observe that some DoS-related CIVs lead to indirect identifier leakage. In particular, when the victim UE receives a reject message with specific causes (*e.g.*, cause #9), it follows the 3GPP-specified recovery procedure [5], deleting its GUTI and security context and transitioning to the **DEREGI.** state. The UE then reattaches using an IMSI-based `Attach Request`, again exposing the permanent identifier over the air.

CIVs leading to attack. We confirm both exposure pathways in practice. One representative scenario is shown in Figure 3(b), in Open5GS, when the attacker sends a spoofed `Attach Request` using the victim's GUTI, the network proceeds with authentication. If the attacker replies with an `Authentication Response`, the network issues an `Authentication Reject`, dissociating the GUTI from the subscriber context. When the victim UE subsequently attempts to reattach using the same GUTI, the network initiates the identification procedure, causing IMSI leakage over the air. In Nokia, we observe the second case. When the victim UE in the **IDLE** state receives a `Service Reject` with cause #9, it deletes its GUTI and reattaches using its IMSI as per the standard. We confirm this behavior using commercial UEs, demonstrating that DoS-induced CIVs can indirectly lead to identifier exposure.

Security impact. Our CIV-based attack exposes IMSI without interacting with the victim's wireless channel. By sending unauthenticated NAS messages to the core network, the attacker causes a legitimate base station to initiate an identification procedure, leading the UE to transmit its IMSI as part of a standard response. Compared to AdaptOver [22], our method does not rely on precisely timing the attack to coincide with the victim's uplink transmission. Since it operates independently of the victim's activity, the attack is simpler to execute and more robust across scenarios. In contrast to FBS-based IMSI catchers [19], which impersonate base stations and are often detectable through abnormal radio characteristics [44, 58], our approach leverages genuine infrastructure and standard signaling behavior. As a result, it evades existing detection mechanisms designed to identify rogue base stations.

7.3 User Presence Detection

Description. We identify CIVs that cause the core network to deviate from normal behavior during reconnection, enabling an adversary to infer whether a specific UE is currently active in the network. If the adversary knows the target's IMSI, they can send a spoofed `Attach Request` using that IMSI to induce a context inconsistency. When the legitimate UE later attempts to reconnect, the network fails to resolve the stored context and initiates an identification procedure, prompting the UE to transmit its IMSI in an `Identity Response`. A passive sniffer can observe this message to confirm the UE's presence within the cell. In addition, as discussed in

§7.2, a `Service Reject` with cause #9 can lead the victim UE to delete its stored context and reattach using its IMSI, which also enables user presence detection. While both this and the IMSI exposure scenario result in IMSI transmission, they differ in how the CIV is triggered: IMSI exposure is caused by spoofing a known GUTI, whereas presence detection assumes knowledge of the IMSI and uses it to provoke the network response.

CIVs leading to attack. We confirm this behavior in practice across multiple core network implementations (see Appendix B). A representative scenario in Amarisoft, shown in Figure 3(c), begins with an adversary sending a spoofed `Attach Request` containing the victim's IMSI and an emergency attach type. The network proceeds with authentication and responds with a `Security Mode Command`. The adversary then replies with a `Security Mode Complete` message containing an SHT value of #4. As a result of this procedure chain, the core network discards the IMSI–GUTI mapping previously associated with the victim UE. Later, when the victim UE attempts to reconnect, the network initiates the identification procedure, causing the UE to transmit an `Identity Response` containing its IMSI. A passive sniffer positioned near the victim can observe this message and determine whether the target UE is currently nearby.

Security impact. This attack enables an attacker to detect the presence of a specific UE near a passive sniffer. Because the victim communicates only with a legitimate network, it has no indication that its presence has been revealed. The sniffer operates passively, making detection by the victim infeasible. The active attacker does not need to be co-located with the victim or sniffer; it only needs to reside within the same MME region to trigger the CIV. While the detection range is constrained by the sniffer's physical coverage, the attack itself can be launched from any location within the MME, significantly increasing the adversary's flexibility and enabling remote, stealthy presence detection.

8 Discussion and Limitations

Standard compliance vs. patch necessity. While LTE and 5G Release 18 specifications [5, 6] explicitly mandate that the core network must not delete an existing context when authentication fails or the UE cannot be verified, this requirement was absent in Release 17 [4] and earlier. The new mandate was introduced via a change request *"to mitigate the DoS attack on a genuine UE, due to unsuccessful authentication."* [2] Prior to Release 18, the principle was only implicitly assumed, according to 3GPP SA.3 representatives.

When we disclosed our CIV findings to Nokia, their response cited compliance with 3GPP TS 24.301 §5.5.1 [3], which states that *"upon receiving an ATTACH REQUEST in EMM-REGISTERED state, the network may delete existing contexts."* This clause does not differentiate whether the request originates from a genuine or malicious UE. Nokia further stated: *"We have not identified any non-compliance with 3GPP specifications; therefore, we will not consider the reported issues as Nokia vulnerabilities."* This response illustrates that adherence to the letter of the specification does not guarantee security. CIVs can still arise in standards-compliant implementations, revealing a critical gap between specification conformance and real-world security guarantees.

Nokia also noted that the tested version (CMM 22.0) had reached End of Life, implying that no patch would be issued. While no new deployments are expected for this version, some operators may still be running it in production. Since Release 18 introduces an explicit check for genuine UEs–a mechanism that is backward compatible–we believe operators using potentially exploitable versions should be strongly encouraged to upgrade.

Extending CIV testing to 5G networks. While this study focuses on LTE networks, our testing methodology can be extended to 5G networks. The underlying NAS protocols and core network context management mechanisms in LTE and 5G share significant similarities, making our approach transferable with minor adjustments. However, a major limitation we encountered was the difficulty of obtaining access to operational 5G core systems with sufficient testing permissions. Unlike LTE, where open-source and commercial-grade testing environments were more readily available, commercial 5G deployments are significantly more restricted, and vendors were reluctant to provide access for security testing. Despite this constraint, we believe that publicly disclosing our findings and releasing the CITesting will encourage broader adoption of CIV testing methodologies.

Simulated radio interface. CITesting targets core network behavior and does not require physical radio transmission. It can operate using a simulated radio interface (*e.g.*, ZeroMQ for srsRAN) to relay NAS messages directly to the core network. Simulated testing is generally faster, as it avoids the overhead of initializing software-defined radios (SDRs). However, we mitigate this overhead in OTA setups by reusing a single SDR initialization across multiple test cases, as described in §6.1. While radio simulation offers performance benefits, OTA testing provides broader compatibility. Many commercial core network implementations—including the Nokia we evaluated—do not support simulated interfaces. In such cases, OTA remains the only viable option. Although our evaluation relied on OTA testing due to these constraints, CITesting is implemented on top of srsUE [25] and can be readily configured to use ZeroMQ when supported by the target network.

Limitations. First, CITesting requires considerable manual effort to analyze standard documents for interpreting protocol messages and semantically pruning IEs. However, this manual effort is a one-time task per cellular network generation and remains broadly applicable across different network equipment without requiring repeated document analysis. Second, as a black-box testing framework, CITesting relies on victim UE behavior as an indirect oracle. This may lead to false positives, as behavioral changes could result from either CIVs or unexpected network errors. Despite this limitation, CITesting's structured testing approach—exercising diverse combinations and generating multiple test cases that target similar CIV conditions—helps identify consistent patterns during post-analysis. In our evaluation, we observed no false positives affecting the findings. Third, CITesting does not precisely identify the underlying root causes of detected CIVs. As discussed in §6.4, the number of distinct root causes was significantly smaller than the number of detected CIVs. While our framework reveals vulnerable procedure chains, subsequent source-level code analysis remains necessary to accurately pinpoint specific implementation flaws. Nevertheless, CITesting provides a practical foundation for discovering attack vectors and guiding further root cause investigations. Finally, CITesting does not model internal sub-states within the core network beyond the explicitly defined EMM and ECM states

in 3GPP specifications. These sub-states are vendor-specific, undocumented, and unobservable in black-box testing environments. Identifying or controlling them would require access to proprietary state machines and precise message sequencing, which is infeasible without source code access. While this limits the granularity of our analysis, CITesting can still indirectly trigger and observe behaviors that may be influenced by internal sub-state transitions through systematic procedure exploration—even without explicitly modeling these sub-states.

9 Related Work

Identifying logical flaws. Several studies have focused on identifying logical flaws in both UEs and core networks through various approaches. The majority of studies employed dynamic testing methods to verify the adherence of UEs and core networks to various security properties [11, 17, 39, 47, 49]. Some researchers extended these dynamic testing approaches by incorporating natural language processing techniques to automatically generate test cases from cellular specifications [14, 15, 36]. Automata learning is another notable direction, often combined with model checking [16] or differential testing [34, 55]. Beyond dynamic approaches, recent research has applied static code analysis to uncover flaws in integrity protection mechanisms [37]. While many studies aim to uncover logical flaws, only a few studies have targeted core networks [15, 17, 20, 39]. Among them, LTEFuzz [39] is the only study that found CIVs, but it lacks the capability for attackers to send messages in various UE states. In contrast, CITesting systematically explores core networks to uncover a broader range of CIVs.

Design flaws. To identify design flaws, many studies have applied formal verification methods on models manually derived from cellular specifications [9, 32, 33, 35]. Some other studies have combined NLP techniques to automatically generate models [7] or test cases [15]. Unlike these works uncovering design flaws, CITesting aims to find implementation flaws, specifically CIVs.

Memory bugs. Another significant research direction is memory bug detection in cellular infrastructures. Several studies have identified such bugs on basebands through reverse engineering [12, 13, 26, 27, 41]. In addition, comparative analysis has been used to detect mis-implementations and undefined behaviors. BASESPEC [38] compares baseband software and specifications to identify deviations, while Klischies *et al.* [40] analyze variances in UE behaviors to uncover undefined behaviors. Beyond static approaches, several studies have adopted fuzzing for dynamic testing on cellular basebands. These fuzzing techniques can be categorized into two types: emulation-based fuzzing [29, 45, 54] and OTA fuzzing [23, 24, 48]. Notably, a recent study [10] presented structure-aware fuzzing on LTE and 5G network components, finding numerous memory bugs. CORECRISIS [20] employs stateful black-box fuzzing with dynamic finite-state machine learning to uncover not only logical flaws but also memory bugs in 5G core networks. Unlike those studies, the goal of CITesting is to uncover logical bugs in the core network.

10 Conclusion

We presented CITesting, the first systematic testing framework for uncovering CIVs in LTE core networks. By systematically exploring procedure chains, varying UE states, and validating context modifications, we identified critical vulnerabilities in both open-source and commercial core implementations. Our findings show that implicit assumptions in pre-Release 18 standards were insufficient, and even current specifications narrowly address only explicit authentication failures. To ensure robust protection, we advocate strengthening the specification to mandate: *"Upon receipt of an unauthenticated or improperly authenticated message, or at any step of an incomplete or unsuccessful procedure that lacks proper authentication, the network shall maintain, if any, the EMM-context and EPS security context unchanged."* We further recommend integrating this into future conformance testing to promote secure implementations prior to full Release 18 adoption. Although we could not evaluate all commercial LTE and 5G cores, we believe this work provides clear definitions and practical methodologies for securing cellular core networks against CIVs.

Acknowledgments

We sincerely appreciate reviewers and shepherd for their valuable comments. This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2024-00397469, Development of Private 5G Security Technology for Integrated Private 5G and Enterprise Network Security).

References

[1] CITesting. https://sites.google.com/view/citesting.
[2] 3GPP. C1-224987. https://www.3gpp.org/ftp/tsg_ct/WG1_mm-cc-sm_ex-CN1/TSGC1_137e/Docs.
[3] 3GPP. TS 24.301, v15.4.0. Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3, 2018.
[4] 3GPP. TS 24.301, v17.6.0. Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3, 2022.
[5] 3GPP. TS 24.301, v18.8.0. Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3, 2024.
[6] 3GPP. TS 24.501, v18.5.0. Non-Access-Stratum (NAS) protocol for 5G System (5GS); Stage 3, 2023.
[7] A. Al Ishtiaq, S. S. S. Das, S. M. M. Rashid, A. Ranjbar, K. Tu, T. Wu, Z. Song, W. Wang, M. Akon, R. Zhang, and S. R. Hussain. Hermes: Unlocking Security Analysis of Cellular Network Protocols by Synthesizing Finite State Machines from Natural Language Specifications. In *USENIX Security Symposium*, 2024.
[8] Amarisoft. Amarisoft Callbox Classic. "https://www.amarisoft.com".
[9] D. Basin, J. Dreier, L. Hirschi, S. Radomirovic, R. Sasse, and V. Stettler. A Formal Analysis of 5G Authentication. In *ACM Conference on Computer and Communications Security*, 2018.
[10] N. Bennett, W. Zhu, B. Simon, R. Kennedy, W. Enck, P. Traynor, and K. R. Butler. RANsacked: A Domain-Informed Approach for Fuzzing LTE and 5G RAN-Core Interfaces. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024.
[11] E. Bitsikas, S. Khandker, A. Salous, A. Ranganathan, R. Piqueras Jover, and C. 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)*, 2023.
[12] A. Cama. A Walk with Shannon. *OPCDE*, 2018.
[13] A. Cama. ASN.1 and Done: A Journey of Exploiting ASN.1 Parsers in the Baseband. *OffensiveCon*, 2023.
[14] Y. Chen, D. Tang, Y. Yao, M. Zha, X. Wang, X. Liu, H. Tang, and B. Liu. Sherlock on Specs: Building LTE Conformance Tests through Automated Reasoning. In *USENIX Security Symposium*, 2023.
[15] Y. Chen, Y. Yao, X. Wang, D. Xu, X. Liu, C. Yue, K. Chen, H. Tang, and B. Liu. Bookworm Game: Automatic Discovery of LTE Vulnerabilities. In *IEEE Symposium on Security and Privacy*, 2021.
[16] M. Chlosta, D. Rupprecht, and T. Holz. On the Challenges of Automata Reconstruction in LTE Networks. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2021.
[17] M. Chlosta, D. Rupprecht, T. Holz, and C. Pöpper. LTE Security Disabled: Misconfiguration in Commercial Networks. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2019.

[18] CICSO ASR 5000 Series (MME Administration Guide). "https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-28/mme-admin/21-28-mme-admin/m_128k-enodeb-connections.html".
[19] A. Dabrowski, N. Pianta, T. Klepp, M. Mulazzani, and E. Weippl. IMSI-Catch Me If You Can: IMSI-Catcher-Catchers. In *ACM Conference on Computer and Communications Security*, 2014.
[20] Y. Dong, T. Yang, A. Al Ishtiaq, S. M. M. Rashid, A. Ranjbar, K. Tu, T. Wu, M. S. Mahmud, and S. R. Hussain. CORECRISIS: Threat-Guided and Context-Aware Iterative Learning and Fuzzing of 5G Core Networks. In *USENIX Security Symposium*, 2025.
[21] Ericsson Mobility Report Q4 2024 update. "https://www.ericsson.com/en/reports-and-papers/mobility-report".
[22] S. Erni, M. Kotuliak, P. Leu, M. Roeschlin, and S. Capkun. AdaptOver: Adaptive Overshadowing Attacks in Cellular Networks. In *International Conference on Mobile Computing and Networking*, 2022.
[23] M. E. Garbelini, Z. Shang, S. Chattopadhyay, S. Sun, and E. Kurniawan. Towards Automated Fuzzing of 4G/5G Protocol Implementations Over the Air. In *IEEE Global Communications Conference (GLOBECOM)*, 2022.
[24] M. E. Garbelini, Z. Shang, S. Luo, and S. Chattopadhyay. 5GHOUL: Unleashing Chaos on 5G Edge Devices. Technical report, SUTD, 2023.
[25] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith. srsRAN: An Open-Source Platform for LTE Evolution and Experimentation. https://github.com/srsran/srsRAN.
[26] M. Grassi and X. Chen. Over the Air Baseband Exploit: Gaining Remote Code Execution on 5G Smartphones. *BlackHat USA*, 2021.
[27] M. Grassi, M. Liu, and T. Xie. Exploitation of a Modern Smartphone Baseband. *BlackHat USA*, 2018.
[28] Public Networks and Operators March 2025. "https://gsacom.com/paper/public-networks-and-operators-march-2025/".
[29] G. Hernandez, M. Muench, D. Maier, A. Milburn, S. Park, T. Scharnowski, T. Tucker, P. Traynor, and K. Butler. FIRMWIRE: Transparent Dynamic Analysis for Cellular Baseband Firmware. In *Network and Distributed Systems Security Symposium (NDSS)*, 2022.
[30] D. T. Hoang, C. Park, M. Son, T. Oh, S. Bae, J. Ahn, B. Oh, and Y. Kim. LTESniffer: An Open-source LTE Downlink/Uplink Eavesdropper. In *16th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2023.
[31] B. Hong, S. Bae, and Y. Kim. GUTI Reallocation Demystified: Cellular Location Tracking with Changing Temporary Identifier. In *Network and Distributed System Security Symposium (NDSS)*, 2018.
[32] S. Hussain, O. Chowdhury, S. Mehnaz, and E. Bertino. LTEInspector: A Systematic Approach for Adversarial Testing of 4G LTE. In *Network and Distributed System Security Symposium (NDSS)*, 2018.
[33] S. R. Hussain, M. Echeverria, I. Karim, O. Chowdhury, and E. Bertino. 5GReasoner: A Property-Directed Security and Privacy Analysis Framework for 5G Cellular Network Protocol. In *ACM Conference on Computer and Communications Security*, 2019.
[34] S. R. Hussain, I. Karim, A. A. Ishtiaq, O. Chowdhury, and E. 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*, 2021.
[35] I. Karim, S. Hussain, and E. Bertino. ProChecker: An Automated Security and Privacy Analysis Framework for Communication Protocol. In *IEEE International Conference on Distributed Computing Systems*, 2021.
[36] S. Khandker, M. Guerra, E. Bitsikas, R. P. Jover, A. Ranganathan, and C. Pöpper. ASTRA-5G: Automated Over-the-Air Security Testing and Research Architecture for 5G SA Devices. In *Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2024.
[37] E. Kim, M. W. Baek, C. Park, D. Kim, Y. Kim, and I. Yun. BaseComp: A Comparative Analysis for Integrity Protection in Cellular Baseband Software. In *USENIX Security Symposium*, 2023.
[38] E. Kim, D. Kim, C. Park, I. Yun, and Y. Kim. BaseSpec: Comparative Analysis of Baseband Software and Cellular Specifications for L3 Protocols. In *Network and Distributed System Security Symposium (NDSS)*, 2021.
[39] H. Kim, J. Lee, E. Lee, and Y. Kim. Touching the Untouchables: Dynamic Security Analysis of the LTE Control Plane. In *IEEE Symposium on Security and Privacy (S&P)*, 2019.
[40] D. Klischies, M. Schloegel, T. Scharnowski, M. Bogodukhov, D. Rupprecht, and V. Moonsamy. Instructions Unclear: Undefined Behaviour in Cellular Network Specifications. In *USENIX Security Symposium*, 2023.
[41] D. Komaromy. Basebanheimer: Now I Am Become Death, The Destroyer Of Chains. *OffensiveCon*, 2023.
[42] M. Kotuliak, S. Erni, P. Leu, M. Roeschlin, and S. Capkun. LTrack: Stealthy Tracking of Mobile Phones in LTE. In *USENIX Security Symposium*, 2022.
[43] S. Lee. Open5GS: Open Source Implementation for 5G Core and EPC. https://open5gs.org/.
[44] Z. Li, W. Wang, C. Wilson, J. Chen, C. Qian, T. Jung, L. Zhang, K. Liu, X. Li, and Y. Liu. FBS-Radar: Uncovering Fake Base Stations at Scale in the Wild. In *Network and Distributed System Security Symposium (NDSS)*, 2017.
[45] D. Maier, L. Seidel, and S. Park. BaseSAFE: Baseband Sanitized Fuzzing through Emulation. In *Conference on Security and Privacy in Wireless and Mobile Networks*, 2020.
[46] Nokia Corporation, CMM 22.0. "https://www.nokia.com".
[47] C. Park, S. Bae, B. Oh, J. Lee, E. Lee, I. Yun, and Y. Kim. DoLTEst: In-depth Downlink Negative Testing Framework for LTE Devices. In *USENIX Security Symposium*, 2022.
[48] S. Potnuru and P. K. Nakarmi. Berserker: ASN. 1-Based Fuzzing of Radio Resource Control Protocol for 4G and 5G. In *17th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2021.
[49] D. Rupprecht, K. Jansen, and C. Pöpper. Putting LTE Security Functions to the Test: A Framework to Evaluate Implementation Correctness. In *USENIX Workshop on Offensive Technologies*, 2016.
[50] D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper. Breaking LTE on Layer Two. In *IEEE Symposium on Security and Privacy (S&P)*, 2019.
[51] D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper. IMP4GT: Impersonation Attacks in 4G Networks. In *Network and Distributed System Security Symposium (NDSS)*, 2020.
[52] A. Shaik, R. Borgaonkar, N. Asokan, V. Niemi, and J.-P. Seifert. Practical Attacks Against Privacy and Availability in 4G/LTE Mobile Communication Systems. In *Network and Distributed System Security Symposium (NDSS)*, 2016.
[53] A. Shaik, R. Borgaonkar, S. Park, and J.-P. Seifert. New Vulnerabilities in 4G and 5G Cellular Access Network Protocols: Exposing Device Capabilities. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2019.
[54] N. Silvanovich. How to Hack Shannon Baseband (from a Phone). *OffensiveCon*, 2023.
[55] K. Tu, A. Al Ishtiaq, S. M. M. Rashid, Y. Dong, W. Wang, T. Wu, and S. R. Hussain. Logic Gone Astray: A Security Analysis Framework for the Control Plane Protocols of 5G Basebands. In *USENIX Security Symposium*, 2024.
[56] USRP B210. https://www.ettus.com/UB210-KIT.
[57] H. Yang, S. Bae, M. Son, H. Kim, S. M. Kim, and Y. Kim. Hiding in Plain Signal: Physical Signal Overshadowing Attack on LTE. In *USENIX Security Symposium*, 2019.
[58] Z. Zhuang, X. Ji, T. Zhang, J. Zhang, W. Xu, Z. Li, and Y. Liu. FBSleuth: Fake Base Station Forensics via Radio Frequency Fingerprinting. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 261–272, 2018.

A Acronyms

3GPP	Third Generation Partnership Project
AS	Access Stratum
CIV	Context Integrity Violation
DoS	Denial-of-Service
ECM	EPS Connection Management
EMM	EPS Mobility Management
eNB	Evolved Node B
EPS	Evolved Packet System
ESM	EPS Session Management
FBS	Fake Base Station
GUTI	Globally Unique Temporary Identifier
IE	Information Element
IMSI	International Mobile Subscriber Identity
KSI	Key Set Identifier
LTE	Long Term Evolution
MAC	Message Authentication Code
MME	Mobility Management Entity
NAS	Non-Access Stratum
SHT	Security Header Type
SIM	Subscriber Identity Module
TAU	Tracking Area Update
UE	User Equipment

B Result Table

We provide the DEREGI. state results in the appendix; full results are available on our website [1].

Table 7: CIVs in DEREGI. State

	Tester UE Initial Message		Procedure Chain (Tester UE ⇋ Network)	Oracle♮					Implication
	Type	Key IE§		ID	Auth	SMC	EMM	Ignore†	
Amarisoft	AR	Emergency AT/IMSI	SM Command » **SM Complete (SHT #4)** » Attach Accept	✓	✓	✓	✓		P
	AR	Emergency AT/IMSI	SM Command » **SM Complete (SHT #4)** » Attach Accept » **Attach Complete (SHT #0)**	✓	✓	✓	✓		P
	AR	Emergency AT/IMSI	SM Command » **SM Complete (SHT #4)** » Attach Accept » **Attach Complete (SHT #1)** » EMM Info.					✓	D
	AR	Emergency AT/GUTI	ID Request » **ID Response (*)** » SM Command » **SM Complete (SHT #4)** » Attach Accept	✓	✓	✓	✓		P
	AR	Emergency AT/GUTI	ID Request » **ID Response (*)** » SM Command » **SM Complete (SHT #4)** » Attach Accept » **Attach Complete (SHT #0)**	✓	✓	✓	✓		P
	AR	Emergency AT/GUTI	ID Request » **ID Response (*)** » SM Command » **SM Complete (SHT #4)** » Attach Accept » **Attach Complete (SHT #1)** » EMM Info.					✓	D
Open5GS	AR	NC 0	Attach Reject	✓	✓	✓	✓		P, E
	AR	NC 0,1,2,3	Auth. Request		✓	✓	✓		-
	AR	NC 0,1,2,3	Auth. Request » **Auth. Fail (Cause #21)** » Attach Reject	✓	✓	✓	✓		P, E
	AR	NC 0,1,2,3	Auth. Request » **Auth. Fail (Cause #20 or #26)** » Auth. Reject	✓	✓	✓	✓		P, E
	AR	NC 0,1,2,3	Auth. Request » **Auth. Response (Invalid Auth. Param.)** » Auth. Reject	✓	✓	✓	✓		P, E
	DR	SHT #1/GUTI	Service Reject	✓	✓	✓	✓		E
	DR	SHT #0/GUTI/KSI 7	Service Reject	✓	✓	✓	✓		E
	SR	*	Service Reject	✓	✓	✓	✓		E
	TAU	GUTI	TAU Reject	✓	✓	✓	✓		E
srsRAN	AR	GUTI	Auth. Request					✓	D
	AR	IMSI/NC 0,1,2,3	Auth. Request		✓	✓	✓		-
	AR	IMSI/NC 0	Auth. Request		✓	‡			D
	AR	GUTI	Auth. Request » **Auth. Response (Invalid Auth. Param.)** » Auth. Reject					✓	D
	AR	GUTI	Auth. Request » **Auth. Fail (Cause #20 or #26)**					✓	D
	AR	GUTI	Auth. Request » **Auth. Fail (Cause #21)** » Auth. Request					✓	D
	AR	IMSI/NC 0,1,2,3	Auth. Request » **Auth. Response (Invalid Auth. Param.)** » Auth. Reject		✓	✓	✓		-
	AR	IMSI/NC 0,1,2,3	Auth. Request » **Auth. Fail (Cause #20 or #26)**		✓	✓	✓		-
	AR	IMSI/NC 0,1,2,3	Auth. Request » **Auth. Fail (Cause #21)** » Auth. Request		✓	✓	✓		-
	AR	IMSI/NC 0	Auth. Request » **Auth. Response (Invalid Auth. Param.)** » Auth. Reject		✓	‡			D
	AR	IMSI/NC 0	Auth. Request » **Auth. Fail (Cause #20 or #26)**		✓	‡			D
	AR	IMSI/NC 0	Auth. Request » **Auth. Fail (Cause #21)** » Auth. Request		✓	‡			D
Nokia	AR	Combined AT/NC 0	Attach Reject		✓	✓	✓		-
	AR	Emergency AT	Attach Reject		✓	✓	✓		-
	AR	Combined AT/GUTI/NC 0,1,2,3	Auth. Request		✓	✓	✓		-
	AR	Combined AT/IMSI/NC 0,1,2,3	Auth. Request » **Auth. Fail (Cause #20 or #26)** » Auth. Reject	✓	✓	✓	✓		P
	AR	Combined AT/IMSI/NC 0,1,2,3	Auth. Request » **Auth. Fail (Cause #21)** » Attach Reject	✓	✓	✓	✓		P
	AR	Combined AT/IMSI/NC 0,1,2,3	Auth. Request » **Auth. Response (Invalid Auth. Param.)** » Auth. Reject	✓	✓	✓	✓		P
	AR	Combined AT/GUTI/NC 0,1,2,3	Auth. Request » **Auth. Fail (Cause #20 or #26)** » ID Request		✓	✓	✓		-
	AR	Combined AT/GUTI/NC 0,1,2,3	Auth. Request » **Auth. Fail (Cause #21)** » Attach Reject		✓	✓	✓		-
	AR	Combined AT/GUTI/NC 0,1,2,3	Auth. Request » **Auth. Response (Invalid Auth. Param.)** » ID Request		✓	✓	✓		-
	AR	Combined AT/GUTI/NC 0,1,2,3	Auth. Request » **Auth. Fail (Cause #20 or #26)** » ID Request » **ID Response (IMSI)** » Auth. Reject		✓	✓	✓		-
	AR	Combined AT/GUTI/NC 0,1,2,3	Auth. Request » **Auth. Fail (Cause #20 or #26)** » ID Request » **ID Response (GUTI or IMEI)** » Attach Reject		✓	✓	✓		-
	AR	Combined AT/GUTI/NC 0,1,2,3	Auth. Request » **Auth. Response (Invalid Auth. Param.)** » ID Request » **ID Response (IMSI)** » Auth. Reject		✓	✓	✓		-
	AR	Combined AT/GUTI/NC 0,1,2,3	Auth. Request » **Auth. Response (Invalid Auth. Param.)** » ID Request » **ID Response (GUTI or IMEI)** » Attach Reject		✓	✓	✓		-
	DR	SHT #1	-		✓	✓	✓		-
	TAU	SHT #0/GUTI	-		✓	✓	✓		-
	TAU	SHT #1/GUTI	TAU Reject		✓	✓	✓		-

§: Key IEs denote the fields that influence the CIV; all other IEs have no effect on the outcome.
†: Network does not respond to the victim's service attempt behavior.
‡: UE sends `Security Mode Reject` and network does not respond to the message.
♮: The oracle for the DEREGI state is a GUTI Attach procedure, and the identified CIVs trigger multiple subsequent procedures.
Procedure abbreviations: (ID: Identification procedure), Auth: (Authentication procedure), SMC: (Security Mode Control procedure), EMM: (EMM Information procedure)
AR: Attach Request, DR: Detach Request, TAU: Tracking Area Update Request, SR: Service Request
AT: Attach Type, SHT: Security Header Type, NC: UE Network Capability, Auth: Authentication, SM: Security Mode, Param: Parameter, Info: Information, IMEI: International Mobile Equipment Identity
P: User Presence Detection, E: IMSI Exposure, D: Denial-of-Service
"NC 0" indicates that the UE supports EPS Ingerity Algorithm 0 (EIA0) and EPS Encryption Algorithm (EEA0) algorithms only, whereas "NC 0,1,2,3" denotes support for EIA0 through EIA3 and EEA0 through EEA3.