Guardians of the Air: In-Device Detection of 5G Control-Plane Threats

Tianwei Wu, Abdullah Al Ishtiaq, Tianchang Yang, Yilu Dong, Kai Tu, Zeyu Song,
Ridwanul Hasan Tanvir, Md Toufikuzzaman, Shagufta Mehnaz, Syed Rafiul Hussain
The Pennsylvania State University
{*tvw5452, abdullah.ishtiaq, tzy5088, yiludong, kjt5562, zzs5287, rpt5409, mpt5763, smehnaz, hussain1*}*@psu.edu*

***Abstract*—We present 5GShield, the first in-device framework for detecting and mitigating control-plane threats in 5G networks. 5GShield works with two complementary modules called ConnSentinel and ExFinder. By utilizing a novel observation of temporal and spatial consistency in broadcast messages among cells under the same tracking area and frequency, ConnSentinel inspects initial cell broadcast messages from nearby base stations to identify and block suspicious base stations that expose anomalous configuration before connection establishment. On the other hand, a machine-learning-based ExFinder module continuously monitors observable control-plane traffic to detect ongoing protocol-level attacks. For ExFinder, we develop a novel graph representation construction mechanism and integrate it with a hybrid pipeline for anomaly detection and attack classification. To support training and evaluation, we curate the first comprehensive dataset combining diverse benign traces from commercial 5G deployments with malicious traces derived from known 4G and 5G control-plane attacks. Experimental results show that 5GShield achieves 99.6% precision and 97.0% recall in detecting both known and zero-day attacks, based on anomaly detection over UE-visible control-plane behavior. Furthermore, 5GShield is lightweight, consuming less than 3.75% of the memory and is deployable on modern commercial devices.**

1. Introduction

The rapid growth of 5G devices reflects widespread deployment of next-generation connectivity, enabling gigabit speeds, network slicing, and ultra-low latency. These devices support critical systems such as autonomous vehicles, IoT, and infrastructure networks [1]–[3]. However, their complex architecture and large attack surface make them attractive targets for adversaries, including state-sponsored actors. Recent studies have reported that the latest 5G devices are infected with several critical vulnerabilities in their control plane protocol design [4]–[6] and implementations [7]–[13] that can lead to severe exploits, including authentication bypass, location tracking, zero-click SMS, and denial-of-services (DoS), launched by fake base stations (FBSs) or similar devices [14]–[17]. The most common form of FBSs primarily exploits the lack of authentication of the cell broadcast messages and spoofs the cell configurations of legitimate base stations to force the victim device to connect to them. Although 3GPP [18], the mobile communications standards body, has considered these concerns and improved the safety measures in the 5G standards [19], the mitigation of those flaws is significantly slow—often taking six months to several years—and some are postponed to later releases or next-generation systems (e.g., 6G) [20]. In addition, adversaries keep finding new flaws, adapting previous exploits, and inventing new attacks. Since FBSs establish direct radio links with victims without traversing operator networks, most operator-level protection mechanisms are ineffective. On the other hand, the user devices lack runtime protections, leaving them exposed to such control-plane threats. Hence, to improve resilience against such adversaries and reduce the impact of critical exploits, it is crucial to integrate threat detection frameworks into 5G devices.

Several prior works have proposed both device-centric and network-centric solutions to detect or mitigate control-plane attacks and the presence of fake 5G base stations (gNBs) in earlier generations of cellular networks (e.g., 3G and 4G) [21]–[24]. However, these approaches mostly employ signature-based detections and exhibit several limitations. Approaches based on pre-defined rules or supervised ML-generated signatures cannot generalize attack behavior [21], [24]–[27] and fail to detect zero-day or variant attacks. Another major limitation of these approaches is they depend on labeled examples of malicious behavior, which are often unavailable or simulated. This data, in particular, often fails to capture the complexity of real-world network environments and the diversity of attackers, such as machine-in-the-middle (MitM) relays [4], [12], [28] and signal injectors [29]–[31], which are stealthier than FBSs [29] and reveal malicious behavior long after the initial interaction with the user devices. Although one can argue for using existing anomaly detection techniques for general-purpose systems [32], [33], they are ineffective in 5G environments due to the highly complex, stateful, and interleaved nature of control-plane procedures.

To address these challenges, we have developed 5GShield, an in-device 5G threat detection framework that monitors control-plane traffic, identifies a wide range of known and unknown threats (zero-day) when they manifest as observable anomalies, including attacks launched by different types of attackers, and triggers countermeasures at runtime with high accuracy. 5GShield is designed to operate entirely on the user device side and addresses

two complementary objectives: (1) inspect broadcasted cell configurations to identify and prevent suspicious base stations before connection establishment, and (2) continuously monitor observable control-plane traffic spanning both pre-connection and post-connection signaling using lightweight machine learning to detect anomalous behaviors and stop attacks in progress. To achieve this, 5GShield implements a two-layer detection mechanism working in parallel: ConnSentinel and ExFinder for each objective, respectively.

Unlike prior approaches that rely on outdated and imprecise public base station databases, ConnSentinel builds on the empirical observation that configurations broadcast by benign base stations (e.g., SIB1 messages) within the same tracking area and frequency are highly stable and similar, with most fields exhibiting minimal variation across cells over time. ConnSentinel flags base stations whose broadcasts deviate from benign configurations of those surrounding cells. By preventing such connections early, 5GShield not only thwarts fake base stations that cannot perfectly impersonate all configuration parameters, but also detects attacks that modify or inject broadcast fields [29]. This design further obviates the need to encode all configuration fields as features for the machine learning (ML) module, ensuring its on-device efficiency.

In parallel, ExFinder actively monitors all traffic to and from the device and detects anomalous behavior indicative of 5G control-plane attacks at runtime. To identify previously unseen (zero-day) attacks, ExFinder adopts a hybrid learning approach. First, it employs an unsupervised model trained solely on benign traces to learn protocol transitions and valid control-plane states. However, as prior work [24], [27] shows, comprehensively capturing benign behavior in cellular networks is non-trivial because real deployments generate a large number of auxiliary or low-significance messages whose relevance is highly context-dependent. Some seemingly unimportant messages can trigger critical state transitions if not tracked carefully or they could also introduce noise (e.g., MeasurementReport) and disrupt the semantic flow of higher-level procedures under different protocol states. To address this challenge, ExFinder constructs a semantically enriched graph that preserves the temporal order of messages while introducing structural edges that encode protocol semantics, enabling it to distinguish incidental messages from those that drive meaningful cross-layer interactions. We trained an unsupervised masked-attribute graph neural network (GNN) to learn and generate contextualized embeddings. Finally, a lightweight supervised classifier, trained on these embeddings, distinguishes known (n-day) attacks from previously unseen threats to enable targeted remediation. ExFinder's capability inherently relies on the presence of observable anomalies in UE-visible control-plane behavior. If an attack remains fully specification-compliant and its resulting behavior is sufficiently common in benign commercial traffic, then it can become indistinguishable from normal operation and fall outside the effective scope of ExFinder.

To develop 5GShield, we have curated the first comprehensive dataset of 5G traces collected from commercial networks operated by four major carriers across two continents, using commercial off-the-shelf (COTS) devices. The dataset further includes 26 unique attacks and variations, bridging the gap created by the lack of publicly available 5G control-plane datasets. To demonstrate practicality, we deploy 5GShield on commercial 5G devices and validate its detection capabilities and performance in real environments.

Evaluations of 5GShield on 26 unique known 5G attacks and their 869 variations show it achieves an F1-score of 98.3%, with a false positive rate (FPR) of 0.029% and a false negative rate (FNR) of 2.95%. Furthermore, 5GShield achieves 95% F1-score on detecting unknown (zero-day) attacks, significantly outperforming the most recent work FBSDetector [27] (75% F1-score), highlighting its robustness and generalization capability in real-world scenarios. We also deployed 5GShield in the wild and identified two attack attempts, demonstrating its usefulness in real commercial networks. We also demonstrate that 5GShield is effective on networks from different operators, based on a two-week test which achieved a false positive rate of 0.037%. We further show that 5GShield is practical, with only 12% higher CPU usage and an additional 427.75 mAh/hour of power consumption.

Open-Sourcing. We'll make the dataset and 5GShield publicly available at https://github.com/SyNSec-den/5GShield.

Contributions. In summary, we contribute the following:

- We curate the first 5G control-plane dataset, including comprehensive benign traffic and 26 real-world attacks.
- We design and implement a two-pronged, in-device 5G attack detection framework that preemptively detects malicious base stations and accurately identifies both known (n-day) and previously unseen (zero-day) exploits.
- We develop a novel strategy for constructing graph presentations and a hybrid graph-based ML component to capture complex, stateful, and cross-protocol interactions.
- We deploy 5GShield on commercial off-the-shelf 5G devices and demonstrate its practicality and low overhead.

2. Preliminaries

5G Network. The 5G network primarily consists of three components: 5G Core Network (5GC), 5G-NR base station (gNB), and User Equipment (UE) [34]. 5GC manages essential network functionalities such as user authentication, data session management, billing, etc. The gNB, connected to the 5GC, provides the radio interface for the UE. UEs are end-user devices such as smartphones or IoT devices. They establish wireless connections with the gNB and communicate with the 5GC to access network services.

Cell Scan and RACH Procedure. To establish communication, UE first performs a cell scanning procedure to identify nearby gNBs [35]. For this, the UE identifies available cells, and subsequently reads the master_info_block (MIB) and system_info_block_1 (SIB1) messages to obtain initial network parameters that are necessary for cell selection and the Random Access Channel (RACH) procedure. Upon identifying the cell that provides the best service, the UE initiates the RACH procedure.

3. Overview of 5GShield

3.1. Threat Model & Scope

We consider adversaries that can arbitrarily intercept, inject, modify, or drop messages between the UE and the gNB. However, as the attackers lack operator keys, they are limited to manipulating and transmitting unprotected cellular protocol messages and cannot generate encrypted or integrity-protected messages. In this work, we focus on protocol-level adversaries capable of exploiting weaknesses in 5G control-plane procedures by manipulating and transmitting unprotected messages. The attack setups are described in the following and are summarized in Table 1.

(i) FBS Attacker. An adversary can deploy a fake gNB by spoofing all or some parameters in `MIB` and `SIB1` [10], [11], [13], [36], [37]. We call full replication *perfect broadcast impersonation/mimicry*. In *partial/imperfect broadcast mimicry*, attackers often use different and irregular values for some parameters than those of the legitimate gNBs due to functional limitations of commercial gNBs [38], [39], limited domain expertise, interference and detection risks [38]–[40], or incomplete support in open-source stacks [41], [42]. The FBS broadcasts such perfectly or imperfectly forged `MIB` and `SIB1` messages with higher signal strength than nearby legitimate gNBs (*pre-connection* phase in Table 1). This lures UEs to disconnect from their serving cell and connect to the FBS, thereby enabling the adversary to subsequently transmit or alter non-integrity-protected messages to the victim UE (i.e., exploitations in *post-connection* phase). For instance, an attacker can issue a plaintext `identity_req` to extract sensitive identifiers [11] (e.g., International Mobile Equipment Identity (IMEI), or send a `reg_reject` message to downgrade the UE to less secure previous-gen (2G-4G) networks [28], [43]. Note that perfect broadcast impersonations do not reveal the attacker's malicious behavior during the *pre-connection* phase and are therefore difficult to detect, as it is challenging to distinguish such malicious messages from legitimate ones.

TABLE 1: Attacker setups, and exploitation strategies in two phases and 5GShield's detection methods. Detection methods: ConnSentinel, ExFinder, not detected.

Exploitation strategy	Attacker Setup			
	FBS	**MitM**	**Signal Injector (SI)**	
Pre-conn.	Perfect broadcast configuration impersonation	Perfect broadcast configuration impersonation	Overshadow broadcast configuration messages	Pre-conn.
	Irregular broadcast configuration impersonation	Irregular broadcast configuration impersonation	Overshadow paging messages	
	RRC Connection Establishment			
Post-conn.	Send plaintext messages	Send plaintext messages Alter intercepted messages	Inject plaintext messages Overshadow broadcast messages	Post-conn.
	Control Plane Security Activation			
	N/A	Send plaintext messages Replay protected messages	Overshadow broadcast messages	

(ii) MitM Attacker. A MitM attacker uses two software-defined radios (SDRs) or similar devices to simultaneously impersonate a legitimate gNB to a victim UE and a legitimate UE to the actual gNB [4], [12], [28]. Beyond FBS's exploitation strategies in phases 1 and 2, this dual-role enables MitMs to intercept, replay, drop, inject, or modify messages even *after control plane security* is activated.

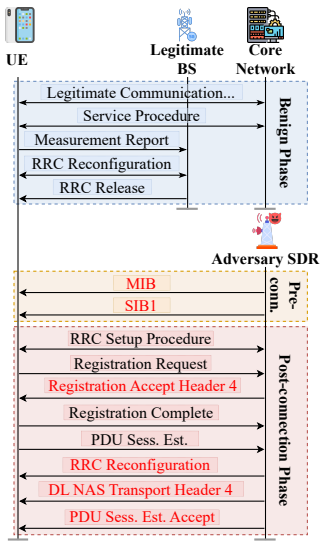


Figure 1: Illustration of 5G AKA bypass attack [10].

(iii) Signal Injector (SI). Unlike MitM and FBS attackers, a signal injector does not establish a connection with either the UE or the gNB [29]–[31]. Instead, it transmits carefully timed malicious messages that temporarily overpower legitimate signals. This enables the attacker to overshadow paging or `SIB` messages before connection establishment, or inject malicious plaintext messages after connection establishment.

Scope. 5GShield aims to detect Layer-3 (L3) control-plane attacks, which leverage FBS, MitM, or signal injector to transmit malicious messages in the Non-Access Stratum (NAS) and Radio Resource Control (RRC) layers. Note that detecting the exact attacker setup and attacks whose distinguishability depends primarily on lower-layer effects, e.g., manipulation through MAC-, RLC-, or PHY-layer messages, are out of scope for this work. Our detection further relies on the presence of observable anomalies in UE-visible L3 control-plane behavior. Within this scope, we aim to prevent the UEs from connecting to malicious gNBs/BSs and identify exploitation attempts when malicious messages produce observable anomalies.

3.2. Motivating Example

Prior work analyzing the security of cellular networks has discovered several new and complex control-plane attacks in 5G [4], [7], [8], [10] and earlier generations [11], [13], [28], [44]. To illustrate the complexity and subtlety of such attacks, we present the 5G Authentication and Key Agreement (AKA) bypass attack in Figure 1. Before the adversary initiates the attack, the UE may already be connected to a legitimate base station, performing regular communication, or it may attempt to connect to the network. We refer to this phase as the *benign phase*. The adversary begins the attack, comprising two phases, by first luring the victim UE to connect to a malicious gNB through broadcasting forged `MIB` and `SIB1` messages with a stronger signal than benign gNBs. We call this phase the *pre-connection* phase. In this

phase, signal injectors and MitM can also launch specific categories of DoS attacks, such as manipulating specific parameters in broadcast messages [29], [45] to prevent a UE from connecting to legitimate cells, called *cell-barring attack*. Finally, in the *post-connection* phase, the adversary can send malicious messages to perform the exploit and achieve the goal. During this phase, the adversary may use any attacker setup (§3.1) to launch an attack.

In the 5G AKA bypass attack, the adversary sends a plaintext `reg_accept` message with *security header type 4* during the *post-connection* phase and registration procedure [10], [31], without conducting authentication or establishing a security context. *Security header type 4* is meant to be used only with a ciphered and integrity-protected `nas_sm_comp` message. Due to this protocol deviation, some UEs incorrectly accept the unprotected plaintext `reg_accept` and respond with a `reg_comp` message. After receiving `reg_comp` and `pdu_sess_est` messages, the attacker sends additional messages, such as `rrc_reconf`, `dl_nas_transport`, and `pdu_sess_est_accept`, to establish a user plane connection, i.e., packet data unit (PDU) session. Such a PDU session is established without any security context, inducing the UE to access the Internet via the malicious BS. This enables the rogue BS to provide unencrypted and unauthenticated user-plane traffic. Consequently, the attacker gains full visibility and control over the UE's traffic, enabling more sophisticated attacks such as phishing or DNS poisoning. Unfortunately, as we show in later sections, existing threat detection mechanisms cannot effectively detect or alert users to such attack attempts.

TABLE 2: Limitations of existing approaches.

Solution	Cellular Gen.	UE/Network Centric	Detection Type	Detection Stage	No Special Hardware	Detect Attack Variations	Detect Unknown	Defense Action	COTS UE Deployment
Crocodile Hunter [46]	4G	UE	Database	Pre-conn.	✗	✗	✗	✗	✗
IMSI Catcher [22]	2G/3G	UE	Database	Pre-conn.	✓	✗	✗	✗	✓
SnoopSnitch [21]	2G—4G	UE	Rule	Pre-/Post-conn.	✓	✗	✗	✗	✓
Wuthier et al. [47]	4G/5G	UE	Rule	Pre-conn.	✗	✗	✗	✓	✗
Li et al. [48]	5G	UE	Rule	Pre-conn.	✗	✗	✗	✗	✗
Phoenix [25]	4G	UE	Rule	Post-conn.	✓	✗	✗	✗	✗
FBSDetector [27]	4G	UE	Supervised ML	Post-conn.	✓	✓	✓	✗	✓
5GSpector [24]	5G	Network	Rule	Post-conn.	✓	✓	✗	✗	✗
CellGuard [49]	2G—5G	UE	Database/Rule	Pre-/Post-conn.	✓	✗	✗	✗	✓
Rayhunter [50]	4G	UE	Rule	Post-conn.	✗	✗	✗	✗	✗
`5GShield`	5G	UE	Rule/Hybrid ML	Pre-/Post-conn.	✓	✓	✓	✓	✓

3.3. Limitations of Existing Approaches

Table 2 summarizes prior work on detecting fake gNBs or malicious actions in cellular networks. Unfortunately, these approaches fall short against sophisticated and evolving attack strategies due to limited scope or methodology. As demonstrated in §3.2, many attacks on UEs start with the *pre-connection* phase and fully manifest in the *post-connection* phase, yet most defenses cover only one phase. As a result, *pre-connection*-only methods miss attacks that remain dormant during this phase via perfect impersonation and then manifest malicious behavior in the *post-connection* phase. Moreover, such systems rely on coarse, often outdated public databases [21], [22], [46], [49] with limited parameters (e.g., location, frequency, cell ID), making them ineffective at distinguishing legitimate from fake cells or detecting manipulations of unrecorded broadcast fields [29].

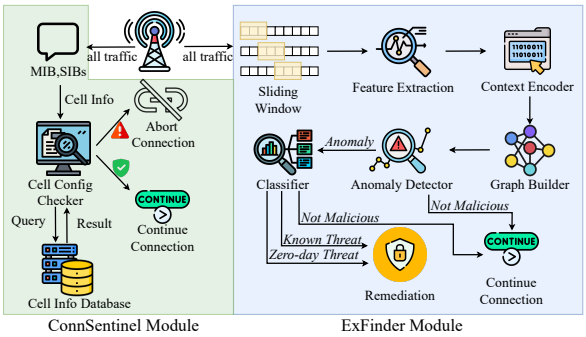


Figure 2: Overview of `5GShield`.

On the other hand, previous approaches focusing on *post-connection* phase threat detection cannot generalize the attack patterns and adapt to evolving attacks, i.e., attack variants and unknown attacks. Broadly, such approaches fall into two categories: rule- or signature-based detection [21], [22], [24], [25], [49], [50], and supervised machine learning-based detection [27]. To illustrate their limitations, consider the attack described in §3.2. Because this attack is recent, existing detection methods based on handcrafted signatures or trained on past patterns cannot identify it [21], [22], [25], [27], [46]. Even if these systems are updated to recognize this specific attack, which relies on the misuse of *security header type 4*, they still fail to detect plausible variants crafted to evade detection. For example, an attacker could replace the security header field with different reserved values in `reg_accept` and `dl_nas_transport` messages to achieve the same exploit while avoiding existing signatures. These limitations are inherent to rule-based and supervised systems, as they rely on prior knowledge of attack behaviors, which attackers can easily circumvent by modifying packet fields or injecting benign-looking messages. As a result, attacks that deviate substantially from known signatures or training distributions (e.g., [21], [24], [25]) may go undetected, while indiscriminate rule expansion can inflate false positives and frequent retraining can substantially increase operational effort (§7.2.5). Some defenses consider both phases [21], [49], but they typically rely on existing single-phase approaches, inheriting prior limitations, e.g., reliance on public databases, signature-based attack detection, and poor generalization to unseen patterns.

Apart from methodological limitations, most prior approaches [21], [22], [25], [27], [46], [50] target earlier cellular generations and are not easily extensible to 5G systems. Moreover, several solutions rely on specialized hardware or require modifications to the protocol, making them impractical for commercial deployment [46]–[48], [50].

3.4. `5GShield` Approach Overview

We design `5GShield` to address these limitations (workflow summarized in Figure 2). It has two primary components, ConnSentinel and ExFinder, working in parallel for efficient and effective control plane threat detection.

ConnSentinel. It aims to preemptively detect malicious gNBs that transmit broadcast messages (e.g., MIB and SIB1) with anomalous cell configuration data and to prevent the UE from connecting to them (§5). Unlike prior database-driven defenses [21], [22], [46], [49], which rely on rigid, centralized, and often outdated records, as well as coarse configuration attributes (e.g., frequency, cell identity), ConnSentinel introduces a novel *similarity-based* detection mechanism, leveraging empirical evidence showing that configuration parameters (e.g., those broadcast in SIB1) remain highly stable (*temporal consistency*) and similar (*spatial consistency*) among benign gNBs operating under the same Tracking Area Code (TAC) and frequency. Instead of relying on static global databases, ConnSentinel running on a UE builds and maintains an offline internal database of gNB configurations with which the UE has registered. ConnSentinel uses these observations to identify malicious base stations that deviate from benign patterns without using gNBs' public database. Moreover, our similarity computation logic in ConnSentinel assigns higher influence to fields that are consistently stable across benign deployments, making the system sensitive to even slight perturbations in SIB1 fields (as in overshadowing attacks [29]), while down-weighting inherently noisy parameters so that this increased sensitivity does not introduce additional false positives. On the other hand, ConnSentinel cannot detect malicious gNBs that use perfect broadcast mimicry. Hence, 5GShield, instead of ConnSentinel, relies on ExFinder to monitor attack exploitation by such gNBs, as discussed below.

ExFinder. It continuously monitors *all* control plane traffic (including the broadcast and control-plane messages during *pre-* and *post-connection* phases) exchanged between the UE and surrounding gNBs, and detects any anomalous communication patterns (§6). To address the limitations of the previous rule, signature, or supervised learning-based approaches [21], [22], [24], [25], [27], [46]–[50], ExFinder adopts a hybrid machine learning-based design. It combines (i) a novel graph neural network–based embedding method, (ii) an embedding-level, distance-based self-supervised anomaly detector, and (iii) a supervised attack classifier. The self-supervised design enables ExFinder to uncover unknown variations and even novel attack types, since its learning is not tied to predefined attack patterns. The supervised classifier, trained on labeled benign and known attack traces, then categorizes detected anomalies into specific attack classes, enabling tailored remediation. Notably, as a trace-driven module, ExFinder models benign behavior from observed traffic rather than exhaustively enumerating all standard-compliant behaviors, so some unseen but legitimate behaviors may still appear anomalous.

By considering 5GShield to be deployed in end-user devices, i.e., UEs, both machine learning models are intentionally lightweight, as we explicitly embed domain knowledge during graph construction to strengthen representation without adding architectural complexity. We have successfully deployed ExFinder on COTS UEs, demonstrating its practical feasibility and impact. Table 1 summarizes the detection capabilities of ExFinder across different attacker setups (highlighted in blue), which is evaluated in §7.2.1.

Rationale of Two-Component Design. Although no existing studies suggest that attackers can perfectly replicate legitimate gNBs, we consider such advanced adversaries possible. In this case, neither 5GShield's ConnSentinel (Table 1) nor prior frameworks [21], [22], [24], [25], [27], [46]–[50] would detect them before an attack is launched. However, ExFinder can still identify malicious activity, even if the adversary with perfect mimicry has successfully replicated gNB configurations to bypass ConnSentinel and lured the UE to connect. The integral role of ConnSentinel is to provide a lightweight and explainable best-effort prefilter that is effective against imperfect impersonation, including many practical commercial fake gNB deployments, rather than protecting against adversaries capable of perfect or near-perfect impersonation. Specifically, it preemptively blocks malicious gNBs based on anomalous broadcast configurations, without relying on complex machine learning models. Our evaluation (§7.2.2) shows it detects real FBSes while operating in a UE within real networks. While such detection could theoretically be performed using machine learning or integrated into ExFinder, we deliberately isolate it to avoid overfitting to the specific configuration values that appeared during training without reasoning about cross-cell relationships, and to ensure scalability, transparency, interpretability, and more importantly, lightweight, fast, and highly accurate detection capability—a prerequisite for in-device threat monitoring systems. This separation allows ExFinder to concentrate on subtle, semantically-rich, and context-dependent behavioral anomalies (i.e., attacking communication flows or single anomalous messages), while ConnSentinel focuses on configuration-level inconsistencies (i.e., broadcast attributes). We observed that incorporating these attributes into ExFinder increases total model parameters by 77.4%, affecting efficiency and energy consumption.

3.5. Challenges of Designing 5GShield

C1: Unavailability of Suitable Dataset. Effective detection of 5G control-plane attacks requires a comprehensive dataset containing both benign and malicious traces across realistic deployment scenarios and attack variations. However, no such dataset exists for 5G networks, or even for earlier generations such as 4G LTE. While existing efforts, such as FBSDetector [27], typically rely on small-scale simulated testbeds, they fail to capture the complexity and diversity of real-world commercial deployments. These testbeds also lack variation in factors like population density, terrain, and cell configurations, leading to datasets that do not generalize well. Moreover, no public *attack* datasets exist to date.

C2: Complicated Protocol Semantics. 5G protocols involve complex control-plane procedures composed of numerous message types. In practice, messages from different procedures often interleave in unpredictable ways, creating highly non-linear and context-dependent message flows. For example, during the registration procedure, the UE may simultaneously receive gNB broadcast messages such as MIB, SIB1, and Paging, which are unrelated to the call

flow of ongoing registration procedure. Additionally, the UE may respond to periodic measurements or mobility-related procedures in parallel. Therefore, semantically related messages (e.g., `reg_req` and `reg_accept`) can be separated by many unrelated broadcasts or heartbeat messages, complicating the task of inferring expected control flows. Additionally, the interpretation of message validity is also highly dependent on the UE's current state. For instance, plaintext NAS messages are considered valid before a security context is established but are anomalous if observed after successful authentication and security mode completion. This context-sensitive behavior means that the same message may be either benign or malicious depending on when it appears in the protocol execution.

Due to these complications, it is extremely difficult for unsupervised models to learn a coherent representation of benign behavior, especially under such noisy and interleaved conditions. To avoid this complexity, many prior approaches rely on supervised learning [27], [51], where it is easier to model specific attack signatures rather than the full spectrum of benign protocol flows. While graph-based techniques such as GNNs [52] offer valuable capabilities in capturing structural and temporal dependencies, they still fall short in modeling the rich state transitions and procedure-level semantics unique to cellular protocols.

C3: In-Device Model Deployment and Runtime Threat Detection. A major challenge in realizing runtime threat detection on UE lies in the inherent trade-off between efficiency and accuracy. Control-plane signaling between the UE, base stations, and the core network is frequent and highly dynamic. Designing a system that can continuously monitor this traffic and accurately identify anomalies in real-time without disrupting normal device operation necessitates a lightweight yet effective architecture. Simple models with small footprints may be computationally efficient, but they often fail to capture the rich contextual dependencies inherent in traffic. Conversely, complex models with deep or dense layers might capture such contextual information, but they introduce a substantial increase in model parameters, resulting in significant demands on processor cycles, memory consumption, and ultimately, user experience. Achieving a balance between these two extremes is a critical research challenge, requiring novel representation methods and model architecture optimization.

4. Dataset Construction

To address **C1**, we construct a dataset that is realistic and representative, and includes both benign and malicious traces while avoiding the ethical and technical issues of launching attacks in commercial networks. For that, we collect large-scale benign traces from commercial networks using COTS UEs with Cellular-Pro [53] and generate diverse attack traces in a controlled lab environment, covering multiple variations of known attacks. We apply trace synthesis techniques to further enhance the dataset with diverse attack traces. We present details on dataset construction in Appendix A.1 while providing a high-level overview below.

Benign Trace Collection. We collected benign traces from four major commercial networks across 2 countries on different continents, resulting in 6.6 GB of signaling traces from real UEs. These traces reflect a wide range of realistic usage conditions (e.g., walking, driving, attending major events, etc.), geographic conditions (e.g., metropolitan cities, rural areas, etc.), and user mobility patterns (e.g., browsing, calling, streaming, etc.). Additional details on the benign dataset are discussed in Appendix A.1.1.

Attack Trace Construction. We first collect malicious traces in a controlled lab testbed using COTS UEs and SDR-based malicious base stations. Each attack trace consists of two stages: the *pre-connection phase* and the *post-connection phase* (§3.4). For the pre-connection phase, we configure fake gNBs with varying levels of impersonation fidelity and include attacks launched during this stage. For the post-connection phase, we reproduce and implement 26 known 5G control-plane attacks from prior works [4], [10], [11], [28]–[30], [36], [43], [54], including *single-message* attacks such as reject-message-based DoS [4] and *multi-message* attacks such as 5G AKA bypass shown in Figure 1 and identity-linking attacks [54] that span one or more protocol sessions. We create $\sim$33 variants per attack on average using domain-specific knowledge, such as modifying message fields or altering the order. To increase diversity and realism, we synthesize traces by combining benign, pre-connection, and post-connection segments in various ways to emulate different adversary setups and attack scenarios. We further insert benign traffic between attack stages to craft delayed attacks, and introduce packet insertion or reordering to reflect evasion techniques that obscure attack intent.

Attack Validation. To ensure that the synthesized traces are valid and effective in triggering the intended attack behaviors, we validate them on vulnerable COTS UEs to confirm the results. This step ensures the generated traces are realistic and meaningful. Notably, during this process, we discovered a previously unknown attack, which `5GShield` can successfully detect without prior training on it, demonstrating its ability to generalize to unknown threats (§7.2.3).

5. Suspicious gNB Detection

Since `MIB` and `SIB1` are essential broadcast messages that base stations transmit during the pre-connection phase, a fake gNB requires broadcasting these messages to initiate an attack. Hence, as discussed in §3.4, to provide a lightweight and explainable first line of defense, `5GShield` first employs ConnSentinel to monitor these messages and preemptively identify fake gNBs that do not achieve perfect or near-perfect impersonation.

When broadcasting `MIB` and `SIB1`, attackers often do not (and cannot) spoof all, but only a subset of all configurations of legitimate base stations for several reasons. First, hardware constraints often limit what attackers can emulate. In practice, rogue base stations may be implemented using a range of hardware platforms, from commodity or mid-range commercial equipment to higher-end (and often much more expensive) systems and software-defined radio (SDR) based

setups. These platforms differ substantially in the bands/frequencies they support and in the level of low-level configuration they expose [38], [39], making perfect, field-by-field replication of commercial base stations often unlikely to be uniformly achievable across attacker deployments. Second, software constraints in available software stacks further limit attackers' capabilities. If a field is supported by commercial implementations but not by open-source ones, the attacker must implement it, requiring in-depth knowledge of 5G specifications, signal processing, and engineering effort. Third, commercial broadcast values are tuned to the operator's RF environment and infrastructure, blindly copying all of them (e.g., `preambleReceivedTargetPower`, `ra-ResponseWindow`) without matching the actual radio channel environment, PHY timing accuracy, and receiver sensitivity can cause RACH and lower-layer failure, connection retries, instability, interference and the risk of detection by network operators [40]. In other words, perfect replication is often strategically suboptimal. In such cases, ConnSentinel aims to preemptively detect anomalously configured gNBs and prevent UEs from connecting to them.

5.1. Stability & Similarity of gNB Parameters

To evaluate the feasibility of detecting malicious gNBs during the *pre-connection* phase, we conducted an extensive empirical analysis of MIBs and SIBs of 1,352 distinct commercial cells operated by 4 major network providers across 2 countries on different continents, covering 49 frequencies and 83 unique tracking areas. Our analysis consistently revealed two key observations: (i_1) several fields in `MIB` and most fields in `SIB1` remain largely stable over time (temporal consistency); and (i_2) benign gNBs operating under the same tracking area and on the same frequency exhibit similar configuration for the fields in `MIB` and `SIB1` (spatial consistency). We have validated these observations in detail in Appendix §A.2. Leveraging these insights, ConnSentinel can validate the legitimacy of gNBs by analyzing `MIB` and `SIB1` configurations before a UE initiates a connection, enabling `5GShield` to preemptively detect and mitigate attacks from malicious gNBs. Although ConnSentinel can be extended to incorporate other optional System Information messages (e.g., `SIB2` and `SIB3`) when present, we focus on `MIB` and `SIB1` to maintain robustness and universal applicability. If an attacker perfectly mimics all configuration fields, ConnSentinel cannot identify it, but `5GShield`'s ExFinder later detects anomalous traffic once the attack proceeds.

5.2. Detection Method

Building on those observations, we designed ConnSentinel to detect anomalous gNBs during the *pre-connection* phase to avoid connecting with suspicious gNBs by analyzing `MIB`/`SIB1` configurations.

Similarity Score-based Detection. As only legitimate gNBs know the cryptographic keys and can successfully complete security-sensitive procedures such as registration or service request procedures, and neither ConnSentinel nor ExFinder raises an alarm, `5GShield` marks such a cell as a *verified cell* and ConnSentinel running locally on the UE stores the corresponding `MIB`/`SIB1` messages (or updates if in database) in its internal verified-cell database for later use.

Based on the insight (i_2) from §5.1 that legitimate gNBs in the same TAC and frequency have highly similar `SIB1` configurations, ConnSentinel attempts to detect anomalous configurations introduced by the malicious gNBs. However, a high overall similarity within a TAC–frequency pair does not imply that all fields in `MIB`/`SIB1` are consistent or identical. For example, `timeAlignmentTimerCommon` and `defaultPagingCycle` tend to be highly consistent across cells in the same pair, whereas others reflect cell-specific or low-level physical layer settings, such as `ss-PBCH-BlockPower` and `totalNumberOfRA-Preambles`, that naturally vary across cells. Hence, treating all fields equally likely leads to false positives. To address this, we analyze per-field consistency. More specifically, ConnSentinel first identifies the field type, such as categorical $\mathcal{C}$ (e.g., `uac-BarringTime`) or numeric/linear $\mathcal{N}$ (e.g., `preambleReceivedTargetPower`), and then computes stability using a type-specific variability metric. It computes the standard deviation of each numeric `MIB`/`SIB1` field and the entropy of each categorical `MIB`/`SIB1` field across cells within every TAC-frequency pair, since standard deviation captures dispersion for continuous values, whereas entropy better measures how concentrated or fragmented categorical values are. It uses the maximum standard deviation ($\sigma_{i,\max}$) and entropy ($e_{i,\max}$) observed over all TAC–frequency pairs as a conservative upper bound on the inconsistency of that field across the same operator's network under benign environments. This prevents underestimating potential inconsistencies in noisier regions and reduces false positives in similarity computation.

At runtime, for any cell not present in the database or whose `MIB`/`SIB1` values differ from its stored record, ConnSentinel calculates per-field differences against (i) the cell's most recent record (insight (i_1)), if available, otherwise (ii) neighboring cells stored in the database within the same TAC–frequency pair (insight (i_2)) according to Equation 3, where diff_i is the absolute difference of the i_{th} field in `MIB`/`SIB1` between the target cell and the chosen reference cell. In `SIB1`, most categorical fields are ordinal rather than purely nominal (e.g., timer indices, periodicity, etc.), where the integer codes represent an ordered scale of configuration values. Depending on field type, we apply different scaling to the raw absolute difference diff_i, yielding the scaled difference d_i. The detailed handling logic is in Appendix A.3.

$$w_i = \min\left(1, \frac{\beta}{\chi_i + \varepsilon}\right), \qquad \chi_i \in \{\sigma_{i,\max},\, e_{i,\max}\}. \quad (1)$$

$$\text{Similarity} = \max\left(0,\ 1 - \frac{\sum_i w_i\, d_i}{\sum_i w_i}\right) \quad (2)$$

To compute whether a target field is similar to that of the reference cell, ConnSentinel assigns larger weights to

the fields that are highly stable in benign deployments, and smaller weights to the variable fields. Equation 1 presents how we compute weight w_i, which also reflects how reliable a field in `MIB`/`SIB1` is as an indicator of irregularity. Here, each per-field difference is weighted by a capped inverse of its corresponding standard deviation or entropy stored in the database, χ_i denotes the maximum standard deviation or maximum entropy of the i_{th} field in `MIB`/`SIB1` stored for the current operator in the database and β is an empirically determined scaling parameter that sets the relative emphasis on low standard deviation versus high standard deviation fields. We also include $\varepsilon = 10^{-3}$ to avoid divide-by-zero. Thus, d_i and w_i play complementary roles, as d_i encodes the magnitude of deviation and w_i encodes the trustworthiness of the field. The weighted differences are aggregated and normalized by the total weight to measure dissimilarity. Finally, the overall similarity score is computed as shown in Equation 2.

This calculation ensures that the resulting similarity score primarily reflects anomalies that appear in stable or consistent fields across cells within the same TAC–frequency pair, while minimizing potential false positives caused by inherently noisy or highly variable fields. Thus, any cell with a similarity score below the threshold is flagged as an anomaly by ConnSentinel.

6. Exploitation Attempt Detection

If the adversary uses perfect or near-perfect impersonation of legitimate base stations, ConnSentinel may fail to preemptively stop the adversarial influence. To address this, `5GShield` incorporates a machine learning-based attack detection module, ExFinder, which identifies anomalous traffic on 5G UEs and triggers appropriate remediation actions through two novel contributions. *First*, to ensure both robustness and efficiency in detection, we introduce a novel graph-based representation of network traces. This representation captures temporal, procedural, and stateful dependencies within the trace, allowing the model to encode contextual information more effectively than prior approaches [27], [49]. *Second*, to address the challenge of detecting previously unknown attacks, ExFinder adopts a hybrid learning strategy that combines unsupervised and supervised learning. We first employ a self-supervised masked auto-encoder GNN trained solely on benign traces. This model learns the normal behavioral patterns of communications between UEs and gNBs and is used to flag deviations that may indicate malicious activity. Next, a supervised model is trained using the embeddings produced by the unsupervised GNN, along with labeled traces of known attacks. This second model serves two key purposes: **(i)** it classifies the detected anomaly into known attack types or flags it as a potentially novel attack, enabling `5GShield` to take tailored remediation actions or report unknown attacks for further investigation; **(ii)** it serves as a fallback mechanism to reduce the risk of false positives from the unsupervised model. Specifically, if the unsupervised model classifies a trace as malicious but with low confidence (i.e., close to the decision threshold), and the supervised model classifies it as benign with high confidence, ExFinder treats the trace as benign. As shown in our evaluation (§7.2.1), this dual-model approach reduces false positives while introducing only a minimal increase in false negatives. The supervised model is carefully calibrated to avoid undermining the ability of the unsupervised model to detect previously unknown attacks. Furthermore, ExFinder allows the second-stage fallback detection to be disabled, enabling stricter detection policies for users who prioritize comprehensive threat detection and can tolerate slightly higher false-positive rates.

6.1. Representing Network Traces as Graphs

Data Preprocessing. While it is possible to add all packets and fields as features in the embedding model, this significantly increases the size of the model, resulting in increased latency and energy consumption, making on-device deployment virtually infeasible (**C3** in §3.5). In particular, using all message-field features as in prior work [27] would increase the model size by 390.1%, resulting in a substantial performance degradation. Thus, to satisfy strict performance constraints while retaining a semantically rich representation of each packet, we extract 11 protocol-level features based on domain knowledge of 5G signaling semantics and attack vectors. For instance, the 5G Mobility Management (5GMM) cause value indicates why a procedure (e.g., registration or deregistration) was rejected, helping capture both the message and its context [55]. Such signals are useful for single-message attack detection. Other features, such as the NAS header and protocol discriminator, convey structural and security-related information about the packets, which are manipulated by real attacks [10], [56]. All features are one-hot encoded to retain their discrete nature.

Graph Representation Construction. As discussed in §3.5, adversaries can craft diverse attack variations and interleave benign messages to evade detection. Graph constructions that rely only on immediate, time-ordered connections miss critical long-range dependencies and control-flow semantics essential to cellular workflows. To address this, we introduce a novel graph construction approach that incorporates domain knowledge to embed cellular network-specific contextual information within each window.

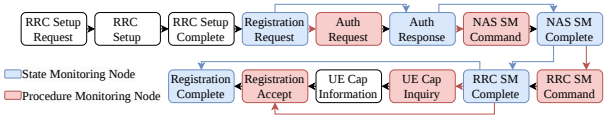


Figure 3: Graph representation construction method.

To account for state transitions, procedural behavior, and potential attack indicators, we define two key message lists based on 5G specifications [35], [57]: (i) *State Monitoring Nodes:* UE response messages that indicate or alter internal UE states (e.g., `auth_resp`, `nas_sm_comp`); and (ii) *Procedure Monitoring Nodes:* messages that initiate new procedures (e.g., `identity_req`, `nas_sm_cmd`) or may be used in attacks (e.g., `reg_reject`, `reg_accept`).

The lists are detailed in Appendix A.4. Using these lists, we construct a domain-informed graph for each trace window. Each packet is added as a node with the encoded message type as its attribute. A directed edge is created to connect each node to its immediate predecessor to capture the simple sequential temporal relations (black edges in Figure 3). To explicitly model protocol state transitions, we leverage the state monitoring node list. Specifically, if a packet matches an entry in this list, indicating that it represents a state change, we add a directed edge from the last encountered state-monitoring node to the current node. This edge captures the semantic transition between protocol stages, enabling the model to learn how high-level protocol states evolve over time rather than just the immediate relationship between consecutive packets. Similarly, if a packet is in the procedure monitoring list, we connect it to the most recent state-monitoring node. This connection reflects procedural dependencies, enabling the model to infer the protocol state context of the currently initiated procedure.

These residual-like connections (shown in red and blue in Figure 3) allow the model to capture long-range dependencies that are critical for identifying control-plane behaviors. In traditional graph convolutional networks, information is propagated through multi-hop message passing, where each layer aggregates features from immediate neighbors. However, as the number of hops increases, the influence of distant nodes tends to diminish due to oversmoothing, making it difficult for the model to retain meaningful dependencies between protocol messages that are temporally or semantically distant. Similar to skip connections in neural networks [58], these residual-like edges directly bridge such gaps, allowing critical state transitions or procedure contexts to remain accessible during aggregation, even when the messages are separated by many intermediate steps. For example, in Figure 3, a model trained on a purely time-ordered graph (i.e., without the red and blue edges) may focus only on immediate message transitions and incorrectly learn that a `reg_accept` typically follows a `ue_cap_Info`. Without contextual awareness, the model fails to recognize that this procedure occurs after an `rrc_sm_comp`. An attacker can exploit this by sending a plaintext `ue_cap_Inq` and, on a vulnerable UE that responds before security is established [10], injecting a plaintext `reg_accept` that matches the learned pattern with a purely time-ordered graph. As a result, the model may misclassify this malicious flow as benign, since it lacks the necessary semantic and procedural context to detect the anomaly. While purely temporal edges only capture sequential transitions, our enriched graph structure links procedural and state information explicitly, allowing the model to reason over message sequences more effectively. This improves both learning efficiency and the semantic richness of the generated embeddings.

6.2. Benign Trace Learning & Threat Detection

In this section, we describe how the unsupervised phase enables the model to learn benign behaviors. The supervised classification is then discussed in §6.3, where the graph

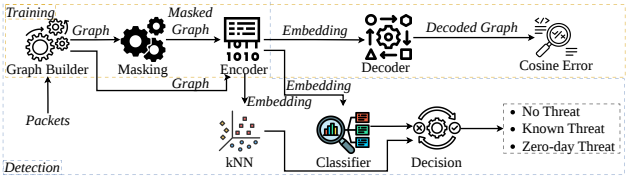


Figure 4: ExFinder architecture.

embeddings are used to classify identified malicious traces from the first phase.

6.2.1. Node Attribute Masking. To address Challenge C2 (§3.5), ExFinder adopts a node attribute masking strategy during its self-supervised training [59]–[62]. This technique encourages the model to learn meaningful contextual patterns, rather than simply memorizing input features. During each forward pass, a random subset of nodes is masked at a predefined rate, and the model predicts their attributes from graph structure and context. For example, in Figure 3, if the node representing a `nas_sm_cmd` message is masked, the model must leverage its surrounding context, such as the preceding `auth_resp` node, the following `nas_sm_comp` node, and their interconnections via semantic edges, to accurately reconstruct it. This mechanism ensures that the learned embeddings capture not just raw features, but also procedural flow, inter-message dependencies, and high-level semantics embedded in the graph structure.

6.2.2. Self-Supervised Autoencoder Training. ExFinder adopts an encoder-decoder architecture to reconstruct the masked node features during training. The encoder transforms the raw graph input into latent node embeddings that capture the relationship derived from the graph's topology and the semantic meaning embedded in nodes and edges while the decoder constructs the masked features using those embeddings. We train the autoencoder in a self-supervised manner with only benign data where the supervision signal is derived from the input data itself without requiring any external labels. This allows the model to learn generalizable representations purely from benign data. Moreover, relying directly on raw node features without contextual encoding often fails to capture the complex relational dependencies and topological patterns present in input data, leading to poor generalization and sensitivity to noise. To address this, we employ a transformer-based graph convolutional layer with a multi-head attention mechanism to dynamically weight neighboring nodes based on learned relevance. This mechanism allows the model to focus on the most informative parts of the graph during message passing. Additionally, we integrate edge attribute information to enrich node representations with protocol-specific interaction within the graph representation. Specifically, the updated representation of node i can be defined as:

$$h_i'^{(k)} = M_1^{(k)}\, h_i + \sum_{j \in \mathcal{N}(i)} \beta_{i,j}^{(k)} \left(M_2^{(k)}\, h_j + M_3^{(k)}\, e_{i,j} \right)$$

where h_i is the initial node feature vector of node i, $e_{i,j}$ is the edge feature between nodes i and j, $\mathcal{N}(i)$ is the set of neighbors of node i, $\beta_{i,j}^{(k)}$ is the attention weight from node i to neighbor node j computed by head k, and $M_1^{(k)}$, $M_2^{(k)}$, $M_3^{(k)}$ are learnable linear transformations (matrices). After computing $h_i'^{(k)}$ from each head, the outputs are concatenated to form the final updated representation of i.

On the other hand, we project the node embeddings into the decoder space using a linear transformation to align the encoder output with the input feature space of the decoder. The decoder is composed of a similar transformer-based graph convolution layer, and the reconstruction performance is evaluated through a scaled cosine loss function [61]:

$$\mathcal{L}_{\text{SCE}} = \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} \left(1 - \frac{h_i^\top \tilde{h}_i}{\|h_i\| \, \|\tilde{h}_i\|}\right)^\gamma$$

where h_i is the original feature of node i, $\tilde{h}_i$ is its reconstructed feature, $\mathcal{N}$ is the set of marked nodes, and γ is a hyperparameter controlling the angular difference. This scaled cosine loss mitigates large norm differences and provides a balanced comparison between the original and reconstructed features [61].

6.2.3. Anomaly Detection. Since the encoder is trained exclusively on benign data, it learns to generate node-level embeddings that capture typical behavior patterns and structural dependencies inherent to benign traffic. In contrast, input graphs that contain malicious or abnormal activities (e.g., exhibit irregular message sequences, anomalous feature distributions, or unexpected transitions) exhibit deviations from the learned patterns. As a result, the encoder produces latent representations for such graphs that diverge significantly from those of benign samples. Thus, to identify outliers, we employ a distance-based anomaly detection strategy grounded in k-nearest neighbor (k-NN) analysis over graph-level embeddings. First, the node-level embeddings generated by the encoder are aggregated by a mean pooling function into graph-level embeddings, which summarize critical features and contextual interactions.

During inference, after the graph-level embeddings are generated by the encoder, we query the stored training embeddings to retrieve the k-nearest neighbors and calculate the Euclidean distance to those neighbors, yielding $d_{\text{test}} = (d_{\text{test},1},\ d_{\text{test},2},\ \ldots,\ d_{\text{test},k})$. Importantly, this comparison is not based on matching the test sample to a specific cell, location, or per-cell database. Instead, the stored embeddings capture benign control-plane behavior patterns in the learned embedding space. Because these patterns arise from 3GPP-defined protocol procedures, they are not inherently confined to a particular geographic area. From these distances, we define two signals: (1) Mean Distance, $\bar{d}_{\text{test}} = \frac{1}{k}\sum_{j=1}^{k} d_{\text{test},j}$, which measures the local density around the test point. (2) First-Neighbor Distance, $d_{\text{test},1}$ representing the distance to the single nearest neighbor, which provides an additional perspective on how closely the sample lies to its most similar benign counterpart. Prior work [62] relies solely on the mean distance to a fixed number of nearest neighbors for k-NN-based outlier detection. While this captures overall isolation, it assumes a relatively uniform distribution of neighbors in the embedding space. Our method instead combines the mean neighbor distance with the first-neighbor distance to produce a more informative anomaly score, with the first-neighbor distance serving as a local reference to the closest benign behavior observed during training. The preliminary composite score is calculated as:

$$s = w \cdot \min\left(\frac{\bar{d}_{\text{test}}}{\mu_{\text{train}}},\ 100\right) + (1-w) \cdot \min\left(\frac{d_{\text{test},1}}{\mu_{\text{train}}},\ 100\right)$$

where μ_{train} is the mean neighbor distance calculated among benign training samples and w is the hyperparameter that controls the relative weight of the mean and first-neighbor distances. Both ratios are clipped to limit extreme outliers. We then normalize s to 0-100, and classify an embedding as anomalous if its score exceeds the threshold θ.

6.3. Attack Classification & Remediation

To categorize detected anomalies and support targeted remediation, ExFinder augments its unsupervised anomaly detection pipeline with a lightweight supervised classifier. Specifically, we trained an XGBoost classifier that operates on graph-level embeddings generated by the encoder of the autoencoder module. The classifier is trained on labeled attack samples from our dataset to identify known attack types (i.e., n-day attacks) and to help distinguish them from unknown threats. During inference, this classifier is applied only to those instances that have been flagged as anomalous by the preceding unsupervised module.

Fallback Benign Classification. If the classifier maps an anomalous instance to a known attack category, ExFinder considers the instance to align with existing attack signatures and labels it as the corresponding n-day attack. However, supervised classifiers may misclassify new (unknown) attacks as benign due to the lack of similar training examples. To mitigate this limitation, ExFinder introduces an additional threshold, θ_{high}, above the default anomaly detection threshold θ. If an instance exceeds the high anomaly threshold ($> \theta_{high}$) but is predicted as benign by the classifier, ExFinder flags it as a potential zero-day attack. The intuition is that the instance is highly deviant in the embedding space despite being classified as benign. When the anomaly score lies between θ and θ_{high}, ExFinder further considers the classifier's confidence score f for the benign prediction. If f exceeds a tunable threshold, the instance is conservatively flagged as an unknown (zero-day) attack. Conversely, if f is below the threshold, the sample is classified as a false positive from the unsupervised detector, under the assumption that the supervised classifier, having seen both benign and malicious examples during training, can compensate for false alarms. To allow flexibility, both θ_{high} and f are tunable hyperparameters that can be calibrated to balance sensitivity and specificity. Users may disable fallback logic entirely by setting $\theta_{high} = \theta$, enabling broader zero-day attack detection at the cost of increased false positives.

Attack-Specific Remediation. Table 3 in the Appendix shows the specific remediation 5GShield takes based on the detected attack. Our application may perform the following 4 operations: **(i)** notify the user, **(ii)** turn off the radio, **(iii)** switch to another cell, and **(iv)** temporarily block the malicious cell. If the user is under a fake base station attack, it cannot trust the current connected cell. Our general approach is to notify the user, switch to another cell, and add the current cell to a temporary block list. However, for certain attack scenarios where the phone may respond with sensitive information (e.g., IMEI-Catcher) or further exploitation can be performed (e.g., 5G AKA bypass), 5GShield instantly turns off the radio to prevent the device from transmitting the response. For a zero-day previously unknown attack, 5GShield first disables and immediately re-enables the radio and forces the baseband to reselect a verified cell (**R2** in §5.2) (if available), or a different cell. Importantly, being marked as verified does not exempt the cell from monitoring; both modules continue to monitor it. If no gNB is available (e.g., all channels are blocked by attackers), it disables the cellular radio and prompts user intervention (e.g., move to a different area or use Wi-Fi). There is a potential case in which a benign cell is first verified by the system, after which an attacker replicates its broadcast parameters, and the remediation mechanism reselects the spoofed cell. In this situation, the mitigation is subject to the same attacker assumptions as ConnSentinel and may not detect adversaries capable of perfect or near-perfect cell configuration mimicry. However, ExFinder continues to monitor the traffic after reselection and will disconnect the suspicious gNB and remove the cell from the verified database if anomalous activity is detected. Beyond on-device remediation, users can opt to share suspicious traces with us to enhance detection accuracy, and share the locations of potential fake base stations with operators or authorities to protect more users.

7. Evaluation

In this section, we evaluate 5GShield's effectiveness and practicality, and compare it with existing solutions.

7.1. Evaluation Setup

To build our dataset, we collected over 1.67 million packets from commercial networks operated by 4 service providers across 2 countries, covering 6 major cities and the highways connecting these cities. We utilize CellularPro [53] on 5 COTS UEs from different vendors to collect real-time traces of 5G NAS and RRC layers. For malicious traces, we implemented 26 existing attacks in 5G with different adversary types and ∼33 variations for each attack on average as mentioned in §4 using srsRAN [41] and OpenAirInterface [42] for the gNodeB and Open5GS [63] for the 5G Core. Overall, our attack model covers 19 types of single-message attacks and 7 types of multi-step sequence attacks, including 16 specification-non-compliant attacks and 10 specification-compliant attacks, as summarized in Table 3. When constructing the malicious trace dataset, we considered temporal order and protocol states, however, we did not consider physical-layer manipulations. Accordingly, attacks whose distinguishability depends primarily on lower-layer radio effects, rather than observable L3 control-plane anomalies, fall outside the effective scope of our system. The attack traces were generated on multiple open-source cellular platforms, specifically srsRAN [41] and OpenAirInterface [42] for the gNodeB and Open5GS [63] for the 5G core. The whole collected dataset is split into

TABLE 3: Attacks in 5G NAS/RRC.
NU: Notify User, TR: Turn off Radio, SC: Switch to another cell, TB: Temporarily Block the malicious cell. Non-compliant: The attack violates the protocol specification by sending invalid messages or field values, transmitting messages in improper states, or breaking required message ordering or procedure flow.

Attack	Attack Type	Adversary	Impact	Remedial Actions
Plaintext AuthReq accept after SMC [10]	single msg, non-compliant	FBS/MitM/Signal Injector	Location Tracking, DoS	NU + SC + TB
CounterCheck Finger Printing [10], [11]	single msg, non-compliant	FBS/MitM/Signal Injector	Fingerprinting	NU + TR + SC + TB
Plaintext IdReq accept after SMC [10]	single msg, non-compliant	FBS/MitM/Signal Injector	Location Tracking	NU + SC + TB
IMEI-Catcher [11]	single msg, compliant	FBS/MitM/Signal Injector	Information leak	NU + TR + SC + TB
IMEISV-Catcher [11]	single msg, compliant	FBS/MitM/Signal Injector	Information leak	NU + TR + SC + TB
NAS SMC w/ NIA0 [10]	single msg, non-compliant	FBS/MitM/Signal Injector	Information leak	NU + SC + TB
NAS SMC replay [10]	single msg, non-compliant	FBS/MitM/Signal Injector	Location Tracking	NU + TR + SC + TB
Registration Reject DoS [43]	single msg, compliant	FBS/MitM/Signal Injector	DoS, Downgrade	NU + SC + TB
Plaintext RRC Reconf DoS [4]	single msg, non-compliant	FBS/MitM/Signal Injector	DoS	NU + SC + TB
RRC SMC bypass [36]	single msg, non-compliant	FBS/MitM	Information leak, Downgrade	NU + SC + TB
RRC SMC w/ NIA0 [10]	single msg, non-compliant	FBS/MitM	Information leak	NU + SC + TB
Plaintext RRC SMC DoS [10]	single msg, non-compliant	FBS/MitM/Signal Injector	DoS	NU + SC + TB
Numb Attack [28]	single msg, compliant	FBS/MitM/Signal Injector	DoS, Downgrade	NU + SC + TB
Service Reject DoS [30]	single msg, compliant	FBS/MitM/Signal Injector	DoS, Downgrade	NU + SC + TB
5G AKA bypass [10]	multiple msg, non-compliant	FBS/MitM	Information leak, Phishing	NU + TR + SC + TB
Auth Sync Failure [4]	single msg, non-compliant	FBS/MitM/Signal Injector	DoS	NU + SC + TB
Deregistration Attack [28]	single msg, non-compliant	FBS/MitM/Signal Injector	DoS	NU + SC + TB
Paging Panic [28]	multiple msgs, compliant	FBS/MitM/Signal Injector	Life threatening	NU + SC + TB
NAS counter reset [4]	multiple msgs, non-compliant	FBS/MitM	DoS	NU + SC + TB
UL NAS counter desync [4]	multiple msgs, non-compliant	FBS/MitM/Signal Injector	DoS	NU + SC + TB
Lullaby [4]	single msg, non-compliant	FBS/MitM/Signal Injector	DoS	NU + SC + TB
Incarceration [4]	multiple msgs, non-compliant	FBS/MitM/Signal Injector	DoS	NU + SC + TB
SUCI-Catcher [54]	multiple msgs, compliant	FBS/MitM	Information leak	NU + TR + SC + TB
Cell Barring [29]	single msg, compliant	Signal Injector	DoS	NU + TR + SC + TB
Adaptover DoS [30]	single msg, compliant	Signal Injector	DoS	NU + TR + SC + TB
Service Request Loop	multiple msgs, compliant	FBS/MitM	DoS	NU + SC + TB
Unknown attack	Unknown	FBS/MitM/Signal Injector	Unknown	NU + TR + SC + TB

training, validation, and test sets in a ratio of 0.7, 0.1, and 0.2, respectively. The validation set is used to monitor training status and select hyperparameters. After training, the model's performance and robustness are evaluated on the test set containing unseen data. We evaluated 5GShield to answer the following research questions:

- **RQ1.** Can 5GShield identify known 5G attacks (§7.2.1)?
- **RQ2.** Can 5GShield detect real-world attacks (§7.2.2)?
- **RQ3.** Can 5GShield detect unknown attacks (§7.2.3)?
- **RQ4.** Does 5GShield generate false positive alerts on commercial networks (§7.2.4)?
- **RQ5.** How does 5GShield compare against existing intrusion detection approaches (§7.2.5)?
- **RQ6.** What is the computational overhead of 5GShield deployment on commercial UEs (§7.2.6)?
- **RQ7.** How impactful is the graph representation construction technique on the effectiveness of ExFinder (§7.2.7)?
- **RQ8.** Can 5GShield generalize to other commercial networks that are not in the training dataset (§7.2.8)?
- **RQ9.** Is 5GShield effective on previous cellular generations (§7.2.9)?

7.2. Evaluation Results

7.2.1. Known Attack Detection of 5GShield. We report ConnSentinel's and ExFinder's efficacy separately (**RQ1**).
Effectiveness of ConnSentinel. On our dataset of diverse attack traces consisting of all three phases with different attacker capabilities (§4), ConnSentinel achieves 1.0 precision, 0.807 recall, 0.893 F1-score, and 0.807 accuracy on

TABLE 4: Comparison of known attack detection.

IDS	Precision	Recall	F1-Score	Accuracy
ExFinder (hybrid)	0.996	0.970	0.983	0.997
ExFinder (unsupervised)	0.961	0.970	0.970	0.997
ExFinder (supervised)	0.996	0.975	0.986	0.997
Deeplog [32]	1.000	0.613	0.760	0.970
Deepcase [33]	0.227	0.749	0.349	0.804
FBSDetector [27]	0.925	0.725	0.813	0.991
Phoenix w/ DFA [25]	0.070	0.182	0.101	0.919
Phoenix w/ MM [25]	0.661	0.653	0.657	0.983
Phoenix w/ PLTL [25]	1.000	0.062	0.116	0.929

malicious gNB detection. This shows that ConnSentinel can reliably prevent most luring attempts by adversaries when software/hardware constraints or lack of operator configuration knowledge hinder them from perfectly replicating legitimate gNBs. We also applied ConnSentinel to our all collected benign traces, including 1352 distinct cell configurations from 4 major operators across 2 continents. For each operator, we divide the data into two parts. The first part is used to compute the standard deviation or entropy of each `MIB`/`SIB1` field. The second part consists of benign traces from the same operator, collected at different locations or times from the first part. This second subset is used to evaluate the false positive rate of the ConnSentinel, during which the module achieved an average false-alarm rate of 0.18% across all four operators.

Effectiveness of ExFinder. We evaluate ExFinder on the test dataset of known attacks and report precision, recall, F1-score, and accuracy. In addition to the full hybrid model, consisting of the unsupervised anomaly detector and the supervised classifier, we evaluate ExFinder's unsupervised and supervised models independently to understand their contributions. Table 4 summarizes the results. Among the ExFinder variants, the supervised model achieves the highest performance when classifying known attacks. However, as demonstrated in §7.2.3, its performance degrades significantly when encountering previously unseen attacks, highlighting the importance of `5GShield`'s hybrid approach. The hybrid model exhibits higher precision than the unsupervised model but slightly lower recall, showing the trade-off of the supervised fallback mechanism, which helps reduce false positives but may slightly increase false negatives. We compare ExFinder with other approaches in §7.2.5.

Effectiveness of ExFinder across Attacker Setups. 5G adversaries can have either FBS, MitM, or signal injector setups to perform attacks on victim UEs (§3.1). Table 5 shows that ExFinder can consistently demonstrate robust performance across attacks launched by different attacker setups. In the case of attacks based on signal injectors [29], [30], the monitored traffic contains anomalous or unusual field values in broadcast messages. As discussed in §6.1, embedding these critical fields in ExFinder allows it to detect such signal injector-based attacks as well with high accuracy (99.6%) and F1-score (97.3%). Conversely, if an attacker does not intend to perform these attacks, rather only makes subtle changes in overshadowed broadcast configurations, such cases may be missed by both ConnSentinel and

TABLE 5: Effectiveness of ExFinder across attacker setups.

Attacker Setup	Precision	Recall	F1-Score	Accuracy
FBS	0.996	0.988	0.996	0.999
MitM	0.970	0.949	0.959	0.998
Signal Injector	0.994	0.959	0.973	0.996

ExFinder. However, if no actual exploit is performed through these overshadowed configurations, it is safe to ignore such cases. Also, similar to `5GShield`, no prior work ([21], [24], [25], [27], [46]–[50], [64]) can detect those.

7.2.2. Attack Detection Effectiveness in Commercial Networks. To answer **RQ2**, we collected real network traces containing both regular operations and attacks from a network security solution provider (anonymized because of non-disclosure agreement) and evaluated them with `5GShield`. The company performed a manual analysis of these traces and detected two traces containing interactions with fake gNBs. When given the network traces to `5GShield`, both fake gNBs were detected by ConnSentinel as they used irregular `SIB1` configurations. Additionally, ExFinder independently detected the post-connection phase exploits in both traces. Our collaborator confirmed these findings. No other traces were flagged by `5GShield` in this provider dataset, which aligns with the ground truth, proving `5GShield`'s reliability. We analyze false positives in the commercial dataset in §7.2.4.

7.2.3. Unknown Attack Detection Efficacy of `5GShield`. To evaluate **RQ3**, we assess the robustness of `5GShield` in detecting previously unseen (zero-day) attacks. Specifically, we craft unknown attack scenarios by randomly selecting a number of attack classes, removing all their traces and variants from the training dataset, and then evaluating the model's performance on *only* the excluded attacks. Figure 5 presents the results for experiments where 1, 3, 5, and 10 attack classes are excluded from training. For each setting, we conduct 5 independent experiments, each time selecting a different random subset of attack classes to omit. The bars in the figure represent the mean precision, recall, F1-score, and accuracy across the experiments, while the error ticks indicate the lower and upper bounds. The prior approach, which employs fully supervised learning [27], exhibits significant performance degradation as more attack classes are excluded. In comparison, `5GShield` exhibits good resilience as both the full hybrid model and the unsupervised-only model of `5GShield` maintain stable performance across all metrics, as they learn solely from benign behavior and are not dependent on specific attack signatures. While the supervised-only variant of `5GShield` shows a slight performance drop as the number of excluded classes increases, it still outperforms prior supervised approaches and exhibits minimal sensitivity to dropped attacks. These results confirm that the hybrid design of `5GShield` offers good generalization and robustness in detecting unknown attack variants.

Zero-Day Attack Discovered by `5GShield`. We uncover a previously unknown vulnerability in two COTS UE models during malicious trace collection, which en-

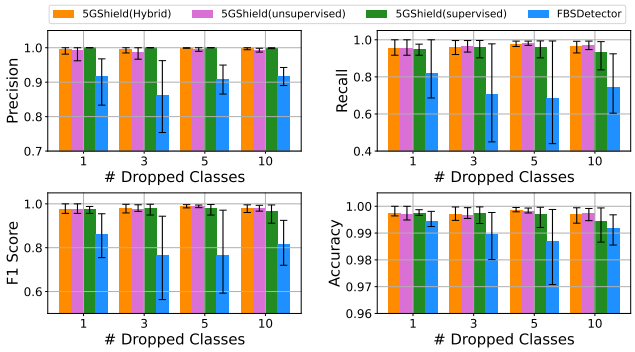


Figure 5: Evaluation results of unknown attack detection.

ables a novel *service request loop* attack that causes DoS and battery depletion. In this attack, the adversary uses a malicious gNB to attract the victim UE and trigger a `service_request` from the UE. Instead of sending `service_accept` or `service_reject` message in response, the attacker waits for a couple of seconds and sends a `rrc_release` message. The UE again attempts to connect to the gNB and sends `service_request`, which the gNB replies with a `rrc_release`, leading to an infinite loop. This behavior persists until the gNB eventually issues a `service_reject`, prompting the UE to search for a new cell. Attackers can exploit this vulnerability to launch a prolonged DoS attack by trapping the UE in an endless cycle of connection attempts and preventing it from connecting to legitimate networks. This also leads to battery depletion due to repeated service attempts. Notably, this attack was inadvertently triggered by our FBS and subsequently recognized as anomalous by the model, despite the absence of any prior exposure during training. Even when we removed this attack from our training dataset, `5GShield` can still detect it with 99% accuracy and 95% F1-score.

7.2.4. False Positives in Benign Network (RQ4). The false positive rate of ExFinder on our benign dataset is only 0.029%, which demonstrates its reliability. Further, We deployed `5GShield` on 5 COTS devices on a commercial network, running continuously for multiple seven-day periods. This experiment yields 6 false warnings for each period on average, which is less than 0.003% of the received messages overall.

For deeper insights, we further analyze several specific cases of false positives. Specifically, as discussed in §4, many known attacks leverage protocol-compliant reject and IMSI-exposing messages [4], [11], [28], [30], [43], [54]. As these messages closely align with attack scenarios, we examine whether they are significant sources of false positives. The results show that among the 47 reject messages (0.003% of the benign dataset), only 2 are flagged as attacks by ExFinder. On the other hand, among the 31 IMSI-exposing messages (0.002% of the benign dataset), only 8 were flagged as attacks. This proves that ExFinder can correctly account for contextual information while dealing with such reject and IMSI-exposing messages. We discuss how `5GShield` distinguishes the benign and attack case utilizing such messages in §7.3. On the other hand, RRC rejections (e.g., `rrc_reestablishment_reject`) in our benign dataset only caused the victim to reconnect to the gNB once, whereas an attack scenario requires repeated transmission of RRC rejections, which is easily distinguished by ExFinder.

7.2.5. Comparison with Existing Approaches (RQ5). We compare `5GShield` against several state-of-the-art attack detection systems: FBSDetector [27], which uses a supervised graph model; Phoenix [25], which relies on rule-based signature matching, and general-purpose intrusion detection systems, including DeepLog [32] and DeepCase [33]. Since FBSDetector was originally designed for 4G networks, we retrained it on our 5G dataset for a fair comparison. Phoenix employs three distinct signature representations: Deterministic Finite Automata (DFA), Mealy Machine (MM), and Propositional-Linear-Temporal-Logic (PLTL). We manually construct 5G versions of the provided state machines and PLTL formulas for Phoenix.

The evaluation results are presented in Table 4. Both DeepLog and DeepCase suffer from high false positive rates, demonstrating that they fail to model the complex and stateful nature of cellular protocols. FBSDetector cannot capture complex interactions within message flows and shows worse performance in detecting unknown attacks due to its supervised model (§7.2.3). The signature-based detection Phoenix employs results in both high false positive and false negative results. To demonstrate, Phoenix's signatures explicitly flag any received reject message as indicative of an attack to identify attacks, including registration reject, numb attack, and service reject attack. However, this signature also leads to benign reject messages sent by the legitimate network being misclassified as malicious. Moreover, Phoenix struggles to detect attack variants. Since they only detect malicious messages received at certain states or particular sequences, an adaptive attacker can easily bypass the detection by transmitting attack messages out of the expected order. Apart from these approaches, we also perform an end-to-end comparison of `5GShield` against all existing solutions with publicly available apps, as detailed in Appendix A.5.

7.2.6. Performance of `5GShield` (RQ6). We deploy `5GShield` on 5 COTS 5G devices (Table 7 in Appendix) connected to commercial 5G networks to evaluate its performance in terms of inference speed, CPU utilization, memory usage, and power consumption. A video demonstration of our Android implementation is available at [65]. On average, `5GShield` processes 41 packets per second, sufficient for real-time detection since control-plane traffic is typically 1–2 packets/s and rarely exceeds 20 even during signaling-intensive operations such as registration and handovers. To evaluate resource consumption, we triggered frequent cell changes to induce intensive control-plane activity. On average, `5GShield` increased CPU usage by 11.2% over idle conditions and consumed less than 300 MB of memory on average (∼3.75% of an 8 GB memory COTS device). We observed an average power increase of 427.75 mAh per

hour, representing about 8–9% of total battery usage. By contrast, a variant trained with all fields as features and deployed in the app incurred at least 1.97× higher CPU inference time than the current version and at least a 28.2% CPU increase over idle conditions.

7.2.7. Impact of Novel Graph Construction (RQ7). To analyze the effectiveness of our masking strategy and novel graph representation construction method, we conducted an ablation study comparing model performance with and without these methods. As shown in Figure 8 in the Appendix, incorporating protocol semantic edges leads to improvements in F1 score, precision, and recall. The masking strategy also leads to improvement in all metrics. These enhancements can be attributed to improved propagation of semantic and procedural information, which enables the model to focus on more relevant contextual cues.

7.2.8. Effectiveness in Other Networks (RQ8). To address **RQ8**, we conducted a cross-network evaluation by training our model exclusively on a dataset containing operators from country A and subsequently testing it on a dataset comprising operators from country B. `5GShield` achieved a false-positive rate of 0.037%, demonstrating good generalization ability across diverse network environments.

7.2.9. Effectiveness in Previous Cellular Generations. To address **RQ9**, we construct a 4G dataset of commercial benign traces and malicious traces spanning 23 distinct attack types (shown in Table 8 in Appendix), and evaluate ExFinder using this dataset. The results show that ExFinder achieves an accuracy of 99.7%, precision of 96.7%, recall of 93.7%, and F1 score of 95.2%, demonstrating its generalizability and effectiveness even in legacy networks.

7.3. Observable Signals and Assumptions for Single-Message Compliant Attack Detection

Since `5GShield`, especially ExFinder, infers on window-level graph embeddings, it is critical to evaluate if `5GShield` can detect attacks triggered by a single protocol-compliant message. Because such attacks may not produce clear flow-level anomalies or explicit protocol violations, they can be difficult to distinguish at the window level. We discuss this case explicitly to clarify that `5GShield` relies not only on anomalous message sequences, but also on the features/-context of individual messages preserved in each window. This enables `5GShield` to detect single-message compliant attacks, achieving a 97.4% detection rate in our evaluation.

For instance, in *direct identity-capture attacks* using `identity_req` [39], such messages may also appear in benign networks. To better distinguish malicious from legitimate cases, ExFinder explicitly considers the requested `identity type`, i.e., SUCI, TMSI, IMEI, or IMEISV. In 5G, a direct `identity_req` cannot expose IMSI, because IMSI is not transmitted directly by protocol design; instead, the UE returns SUCI, which is a concealed form of the subscriber identity. Thus, an attacker cannot derive the user's identity from a single `identity_req` alone. Prior work, however, has shown that `identity_req` messages requesting SUCI can still be abused to facilitate IMSI exposure and UE linking through multi-message attacks [54], [66], which ExFinder can detect. In 4G or 5G NSA, an attacker can send a single `identity_req` asking for IMSI, in this case, `5GShield` cannot reliably separate it from benign network behavior; therefore, 5GShield raises an alert and relies on user judgment for subsequent action. Other attack cases with a single `identity_req` requesting permanent identifiers other than IMSI, such as IMEI or IMEISV. These requests are rare in benign networks and are therefore classified as highly anomalous. Although such requests may still occur in benign settings, based on our collected dataset, their occurrence ratio is below 0.0001%. Given this extremely low prevalence, we consider the resulting false-positive tradeoff acceptable in practice.

For *indirect identity-capture attacks* induced by reject messages [39], we incorporate both the reject cause value and the PLMN identity of the transmitting cell as input features to the model. This choice is based on our observation that reject messages intended to trigger identity exposure are, in most cases, sent by cells advertising a non-home PLMN. As a result, if an attacker attempts to launch such an attack using a fake gNB configured with the victim UE's home PLMN, the attack can still be captured through these features. Note that this signal is less discriminative when the attacker advertises a non-home PLMN. However, such a strategy generally reduces attack efficiency, since the UE typically prefers home-PLMN cells as long as the signal conditions of nearby legitimate cells are sufficiently good.

For *reject-based service-area blocking attacks* [67], `5GShield` follows the same detection principle as for reject-based identity-catching attacks: the cause value and PLMN identity serve as discriminative features that enable the model to flag anomalous reject messages. For cell service-barring attacks that manipulate SIB fields such as `uac-BarringTime`, ConnSentinel can effectively detect them. since achieving persistent denial of service generally requires assigning this field an unusually large or small value, which creates a clear deviation from the configurations broadcast by surrounding benign cells. This substantial inconsistency significantly reduces the similarity score and enables reliable detection. We also acknowledge that our approach has limited sensitivity to attacks that manipulate only a single SIB field, and that the manipulated value differs only slightly from those of neighboring benign cells; the resulting anomaly signal may be insufficient for detection.

For downgrade attacks carried out through reject messages with specific cause values, such as cause #42 [67], ExFinder treats these messages as direct anomaly indicators. This is because such cause values are exceedingly rare in benign networks; indeed, they did not appear in our collected benign dataset. As a result, although this design may in principle introduce false positives if such messages arise in rare legitimate scenarios, we consider the tradeoff acceptable in practice due to their negligible benign occurrence. For downgrade attacks realized through manipulation of SIB

fields, our current implementation cannot detect cases in which the modified fields are outside MIB/SIB1, since ConnSentinel presently focuses only on MIB/SIB1 analysis. We note that ConnSentinel can be extended to support additional SIBs. In such cases, detectability would depend on the deviation introduced by the attacker: attacks that require conspicuously different field values are more likely to be detected, whereas attacks that succeed with only slight modifications relative to normal configurations may remain difficult to identify. For example, an attacker overshadows the broadcast `cellReselectionPriority` from a high value to a very low value, or vice versa; such a substantial change would be identified by ConnSentinel.

8. Discussion and Limitations

Limitations of trace-driven benign modeling. Although 5GShield collects broad and diverse data across operators, improving detection robustness, it does not fully eliminate false positives. Heterogeneous real-world deployments may still exhibit previously unseen yet legitimate behaviors. Therefore, ExFinder is considered to provide practical rather than complete coverage of all benign behaviors, and its deployment may require operator- or environment-specific calibration, along with careful alert handling.

Practical deployment challenges. 5GShield currently requires root access on Android, as user-level apps lack access to control-plane signaling. Without direct baseband access on commercial devices, it operates at the application layer, observing messages only after decoding, which introduces millisecond-level detection delay. Hence, for single-message attacks (e.g., identity exposure or reject-based) that take effect immediately, detection occurs post-execution and mitigation is only retroactive. This limitation is inherent to all application-layer UE-side approaches, not specific to 5GShield, and earlier intervention would require vendor integration into baseband firmware.

9. Conclusion

We present 5GShield, an in-device framework designed to detect and mitigate control-plane threats in 5G networks. 5GShield leverages ConnSentinel to preemptively identify malicious base stations, and ExFinder to employ a novel graph-based representation and hybrid learning to detect and mitigate active exploitation attempts. 5GShield demonstrates high effectiveness in identifying diverse n-day and zero-day 5G attack scenarios, achieving over 99.7% detection accuracy.

Acknowledgment

We thank the anonymous reviewers and the shepherd for their feedback and suggestions. We also thank the corresponding developers for cooperating with us during our responsible disclosure. This work has been supported by the NSF under grants 2145631, and 2215017, and the NSF and Office of the Under Secretary of Defense—Research and Engineering, ITE 2326898 and 2515378, as part of the NSF Convergence Accelerator Track G: Securely Operating Through 5G Infrastructure Program.

Ethics considerations

Our work is intended to enhance the security and privacy of cellular users. All data collection and experimentation were conducted adhering to the following considerations to avoid harm to operational networks or end users.

Dataset Collection. We collected data from commercial networks using our own FCC-certified devices with valid data plans. All collection was strictly passive. We only observed traffic between our device and network and did not transmit abnormal or non-standard messages. This ensured that our equipment behaved identically to a normal subscriber device.

Attack Trace Generation. Attack traces were generated exclusively in a controlled laboratory environment. We reproduced previously reported exploits inside a physically and logically isolated Faraday cage and carefully limited the transmission power of our software-defined radios to prevent interference with commercial networks. Our procedures followed established best practices for secure RF experimentation [37].

In-Device Detection. We developed an Android application to detect threats in-device to protect users, which prioritizes user privacy and control: voluntary use, local computation and not uploading private data to remote servers.

Vulnerability Discovery. We discovered 1 new vulnerability. We followed a coordinated disclosure process by confidentially reporting our finding to the corresponding vendor, who has acknowledged our reports. Additionally, we reported the identified fake base stations found in the data traces to our collaborators, who provided the data.

LLM usage considerations

We, the authors, are fully responsible for the originality and correctness of all content in this paper, including the technical contributions, experimental design, analysis, evaluations, and conclusions. LLMs were used solely for editorial purposes in this paper, and all outputs were inspected and edited by the authors to ensure accuracy and originality.

References

[1] Third Generation Partnership Project, "5g system overview," https://www.3gpp.org/technologies/5g-system-overview, 2022.

[2] Qualcomm Technologies, "Everything you need to know about 5g," https://www.qualcomm.com/5g/what-is-5g, 2024.

[3] Amazon Web Services, "5g system overview," https://aws.amazon.com/what-is/5g, 2024.

[4] 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 *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19, 2019.

[5] D. Basin, J. Dreier, L. Hirschi, S. Radomirovic, R. Sasse, and V. Stettler, "A formal analysis of 5G authentication," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 1383–1396.

[6] C. Cremers and M. Dehnel-Wild, "Component-based formal analysis of 5G-AKA: Channel assumptions and session confusion," in *Network and Distributed System Security Symposium (NDSS)*, 2019.

[7] C. Yu, S. Chen, Z. Wei, and F. Wang, "Secchecker: Inspecting the security implementation of 5g commercial off-the-shelf (cots) mobile devices," *Comput. Secur.*, vol. 132, no. C, Sep. 2023.

[8] 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*, ser. WiSec '23, 2023.

[9] A. A. 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 *33rd USENIX Security Symposium*, 2024.

[10] K. Tu, A. A. 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 *33rd USENIX Security Symposium*, Aug. 2024, pp. 3063–3080.

[11] 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 *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1325–1342.

[12] D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper, "Breaking lte on layer two," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 1121–1136.

[13] S. M. M. Rashid, T. Wu, K. Tu, A. A. Ishtiaq, R. H. Tanvir, Y. Dong, O. Chowdhury, and S. R. Hussain, "State machine mutation-based testing framework for wireless communication protocols," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '24, 2024.

[14] S. Kumar, E. Hamed, D. Katabi, and L. Erran Li, "Lte radio analytics made easy and accessible," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 211–222, 2014.

[15] J. D. Roth, M. Tummala, J. C. McEachen, and J. W. Scrofani, "On location privacy in lte networks," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 6, pp. 1358–1368, 2017.

[16] Sanjole, "WaveJudge 5000 Wireless Test System for LTE and WiMAX."

[17] Software Radio Systems, "AirScope — SRS Products."

[18] *3GPP Standard*, www.3gpp.org.

[19] 3GPP, "5g; security architecture and procedures for 5g system," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 33.501, 2025, version 18.7.0.

[20] *3GPP Release 20*, [Online]. Available: https://www.3gpp.org/specifications-technologies/releases/release-20.

[21] SRLabs, "Snoopsnitch - mobile network security tool," https://github.com/srlabs/snoopsnitch.

[22] "Android imsi-catcher detector," https://github.com/CellularPrivacy/Android-IMSI-Catcher-Detector.

[23] 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 *NDSS*, 2017.

[24] H. Wen, P. Porras, V. Yegneswaran, A. Gehani, and Z. Lin, "5g-spector: an o-ran compliant layer-3 cellular attack detection service," in *Network and Distributed System Security Symposium*, 2024.

[25] M. Echeverria, Z. Ahmed, B. Wang, M. F. Arif, S. R. Hussain, and O. Chowdhury, "Phoenix: Device-centric cellular network protocol monitoring using runtime verification," in *Network and Distributed System Security Symposium, NDSS*, 2021.

[26] P. K. Nakarmi, M. A. Ersoy, E. U. Soykan, and K. Norrman, "Murat: Multi-rat false base station detector," *arXiv preprint arXiv:2102.08780*, 2021.

[27] K. S. Mubasshir, I. Karim, and E. Bertino, "Fbsdetector: Fake base station and multi step attack detection in cellular networks using machine learning," *arXiv preprint arXiv:2401.04958*, 2024.

[28] S. Hussain, O. Chowdhury, S. Mehnaz, and E. Bertino, "Lteinspector: A systematic approach for adversarial testing of 4g lte," in *Network and Distributed Systems Security (NDSS) Symposium 2018*, 2018.

[29] 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 *28th USENIX Security Symposium*.

[30] S. Erni, M. Kotuliak, P. Leu, M. Roeschlin, and S. Capkun, "Adaptover: adaptive overshadowing attacks in cellular networks," in *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, 2022, pp. 743–755.

[31] S. Luo, M. Garbelini, S. Chattopadhyay, and J. Zhou, "Sni5gect: A practical approach to inject anrchy into 5g nr."

[32] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 1285–1298.

[33] T. Van Ede, H. Aghakhani, N. Spahn, R. Bortolameotti, M. Cova, A. Continella, M. van Steen, A. Peter, C. Kruegel, and G. Vigna, "Deepcase: Semi-supervised contextual analysis of security events," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022.

[34] 3GPP, "5g; system architecture for the 5g system (5gs)," 3GPP, TS 23.501, 2025, version 18.7.0.

[35] 3GPP, "Radio resource control (rrc); protocol specification," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.331, 2023, version 17.6.0.

[36] H. Kim, J. Lee, E. Lee, and Y. Kim, "Touching the untouchables: Dynamic security analysis of the LTE control plane," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019.

[37] A. Shaik, J. Seifert, R. Borgaonkar, N. Asokan, and V. Niemi, "Practical attacks against privacy and availability in 4g/lte mobile communication systems," in *23rd Annual Network and Distributed System Security Symposium, NDSS, 2016*.

[38] PKI Electronic Intelligence, "Advanced imsi catcher," https://pki-electronic.com/products/imsi-catcher/advanced-imsi-catcher/.

[39] PTC, "4g imsi catcher for detecting imsi/imei number of mobile phone," https://ptcjammer.en.made-in-china.com/product/qnbpoIGlbhkv/China-4G-Imsi-Catcher-for-Detecting-Imsi-IMEI-Number-of-Mobile-Phone.

[40] T. J. Schlangen, M. H. Hutchison, and P. M. Stephens, "Physical-layer cell identity (PCI) conflict detection," US Patent Application US 20120275315A1, 2012. [Online]. Available: https://patents.google.com/patent/US20120275315A1/en

[41] srsran, "srsran - open source 4g and 5g software radio suite," https://github.com/srsran/srsRAN_Project, 2024, accessed: 2024-11-10.

[42] oai, "Openairinterface - 5g software alliance for democratising wireless innovation," https://github.com/OPENAIRINTERFACE/openairinterface5g, 2024, accessed: 2024-11-10.

[43] B. Karakoc, N. Fürste, D. Rupprecht, and K. Kohls, "Never let me down again: Bidding-down attacks and mitigations in 5g and 4g," in *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2023, pp. 97–108.

[44] 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, pp. 1082–1099.

[45] N. Ludant and G. Noubir, "Sigunder: A stealthy 5g low power attack and defenses," in *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*.

[46] Electronic Frontier Foundation (EFF), "Crocodile hunter." [Online]. Available: https://github.com/EFForg/crocodilehunter

[47] S. Wuthier, J. Kim, J. Kim, and S.-Y. Chang, "Fake base station detection and blacklisting," in *33rd International Conference on Computer Communications and Networks*, 2024.

[48] X. Li, K. Zheng, S. Guo, and X. Ma, "Precheck sequence based false base station detection during handover: a physical layer security scheme," in *2023 IEEE 6th International Conference on Computer and Communication Engineering Technology*. IEEE, 2023.

[49] L. Arnold, M. Hollick, and J. Classen, "Catch you cause i can: busting rogue base stations using cellguard and the apple cell location database," in *Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses*, 2024.

[50] Electronic Frontier Foundation, "Rayhunter." [Online]. Available: https://github.com/EFForg/rayhunter

[51] E. E. Abdallah, A. F. Otoom *et al.*, "Intrusion detection systems using supervised machine learning techniques: a survey," *Procedia Computer Science*, vol. 201, pp. 205–212, 2022.

[52] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[53] "Cellular pro," https://play.google.com/store/apps/details?id=make.more.r2d2.play.cellular_pro&hl=en.

[54] M. Chlosta, D. Rupprecht, C. Pöpper, and T. Holz, "5G suci-catchers: still catching them all?" in *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2021.

[55] 3GPP, "Study on 5g security enhancements against false base stations," 3rd Generation Partnership Project (3GPP), TR 33.809.

[56] 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 *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 3547–3563.

[57] 3GPP, "Non-access-stratum (nas) protocol for 5g system (5gs); stage 3," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 24.501, 2023, version 17.8.0.

[58] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[59] D. Bahdanau, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[60] P. Mishra, A. Piktus, G. Goossen, and F. Silvestri, "Node masking: Making graph neural networks generalize and scale better," *arXiv preprint arXiv:2001.07524*, 2020.

[61] Z. Hou, X. Liu, Y. Cen, Y. Dong, H. Yang, C. Wang, and J. Tang, "Graphmae: Self-supervised masked graph autoencoders," in *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, 2022.

[62] Z. Jia, Y. Xiong, Y. Nan, Y. Zhang, J. Zhao, and M. Wen, "MAGIC: Detecting advanced persistent threats via masked graph representation learning," in *33rd USENIX Security Symposium*, 2024.

[63] open5gs, "Open5gs - a free and open-source 5g core network implementation," https://github.com/open5gs/open5gs, 2024.

[64] A. Dabrowski, N. Pianta, T. Klepp, M. Mulazzani, and E. Weippl, "Imsi-catch me if you can: Imsi-catcher-catchers," in *Proceedings of the 30th annual computer security applications Conference*, 2014.

[65] *5GShield Demo Video*, https://drive.google.com/drive/folders/1ToEmapLTLUqtnrdaXiy6HvBrzycn1WnG?usp=sharing.

[66] S. R. Hussain, M. Echeverria, O. Chowdhury, N. Li, and E. Bertino, "Privacy attacks to the 4g and 5g cellular paging protocols using side channel information," *Network and distributed systems security (NDSS) symposium2019*, 2019.

[67] W. Liu, Y. Li, H. Li, Y. Chen, Y. Wang, J. Lan, J. Wu, Q. Wu, J. Liu, and Z. Lai, "The dark side of scale: Insecurity of direct-to-cell satellite mega-constellations," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 445–464.

Appendix A. Supplemental Materials

A.1. Dataset Construction

`5GShield`'s dataset includes both realistic benign and attack traces. We show the high-level overview of our dataset construction method in Figure 6. In what follows, we illustrate `5GShield`'s data collection process in detail.

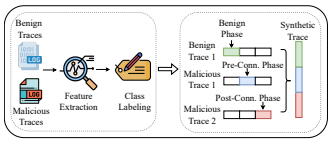


Figure 6: Dataset construction.

A.1.1. Benign Trace Collection. To compile a wide range of traffic patterns reflective of real-world conditions, we collected benign traces from user devices operating within different commercial networks across various contexts, totaling 6.6 GB of data.
(A) Different Use Cases: We collected traces from UEs engaged in various activities, including web browsing, video streaming, texting, calling, idle periods, and location-based services (LBS), e.g., maps and navigation.
(B) Diverse Geographic and Environmental Conditions: We compiled traces from areas with varying population densities (e.g., metropolitan cities, suburban neighborhoods, and rural regions), different terrain types (e.g., mountains, hills, and valleys), and special events (e.g., art festivals, security conferences, etc.), ensuring a diverse range of conditions.
(C) Varied User Mobility: We also collected traces from UEs in different mobility scenarios: stationary, walking with moderate mobility, triggering intra-cell handovers, and driving with high mobility, triggering frequent handovers and connection reestablishment events.

A.1.2. Attack Traces Collection. Attacks on victim COTS 5G UEs involve two main phases: **(1)** *pre-connection* phase, and **(2)** *post-connection* phase.
Pre-connection Phase. For FBS/MitM attacker, to effectively impersonate a legitimate BS, the attacker first configures various parameters of the malicious BS, closely mimicking those of the legitimate one, e.g., the Public Land Mobile Network (PLMN), Physical Cell ID, Tracking Area Code, and frequency. It then exploits the lack of authentication of `MIB` and `SIB1` messages and spoofs them with a higher

signal power to attract potential victims. Based on attacker capabilities and intent, we considered and configured the following types of configurations for malicious BS: **(i)** perfect impersonation by using the configurations of a legitimate nearby BS, **(ii)** partial impersonation by randomly altering a few configuration parameters, and **(iii)** unknown BS impersonation by using configurations of a gNB from a different area or by using random legitimate values.

Post-connection Phase. To collect *post-connection phase traces*, we first implemented 26 distinct control-plane attacks on commercial UEs (including those in *pre-connection* phase). Prior research has identified these attacks in 5G networks or in earlier generations of cellular networks, which can be adapted to 5G, as detailed in Table 3. We also manually verified that our implementation affects the UE as intended. We further create attack variants by modifying malicious message fields and message order.

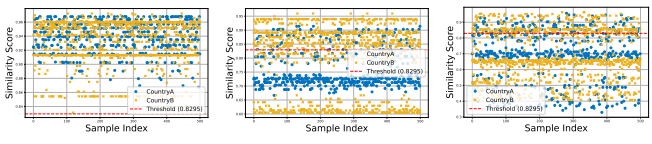
Attack Traces Synthesis. To further increase the diversity, we synthesized diverse attack traces by concatenating the three types of trace segments (as defined in §3.2) and randomly selecting segment variants. These synthesized traces capture multiple adversarial scenarios. Direct concatenation models a victim lured from a benign BS to a malicious SDR before exploitation, while inserting benign traffic between the two attack phases delay attack execution.

A.2. Stability & Similarity of gNB Parameters

TABLE 6: Number of field changes in `SIB1` configurations.

#Days	#BS observed	#BS with changes	#Unique changed fields		
			Max	Min	Average
< 1	1352	3	3	2	2.67
1	333	12	7	1	3
3	278	13	7	1	1
5	214	17	9	1	3
7	136	21	14	1	6.25
30	108	24	14	1	2
90	98	106	14	1	2.95

Validating Stability of Configurations. We monitored all covered commercial gNBs over a period ranging from 1 to 90 days and tracked changes in the unique fields of `SIB1` messages during this time. The results, summarized in Table 6, show that `SIB1` configurations remain largely stable. Specifically, only about 5% of the observed gNBs exhibited any change in their `SIB1` configurations over a 7-day window, and just 6% of all configuration fields changed throughout the observation period. These findings suggest that a *high degree of change or inconsistency in `SIB1` fields over a short duration is indicative of anomalous behavior.*



(a) (b) (c)

Figure 7: Distribution of similarity scores.
(a) same TAC/Freq (b) same TAC, different Freq (c) different TAC/Freq

Validating Similarity of Configurations. To validate the similarity of configurations across different gNBs of the same operator, we conducted a comparative analysis of `SIB1` messages by computing their similarity scores. To properly evaluate the similarity scores, we computed these scores for pairs of gNBs and categorized those pairs into three groups: (i) same TAC and frequency, (ii) same TAC but different frequencies, and (iii) different TACs and frequencies. The results, shown in Figure 7, demonstrate clear trends. We consistently observed that gNBs sharing the same TAC and frequency exhibit significantly higher configuration similarity, achieving similarity scores greater than 0.8081. In contrast, gNBs with differing TACs or frequencies show notably lower similarity. This strong consistency is expected, as many fields in `SIB1` (e.g., frequency, bandwidth, PRACH configuration) are determined by physical-layer parameters that are tightly coupled with network planning and performance requirements. These parameters are non-trivial to modify without degrading the functionality or coverage of the base station. Therefore, a malicious gNB attempting to impersonate a legitimate one, especially without detailed knowledge of local deployment parameters, is likely to introduce anomalies in `SIB1`. These deviations make impersonations detectable, thus enabling ConnSentinel to effectively flag malicious gNBs based on configuration mismatches.

A.3. Handling Raw Difference Calculation

We first distinguish fields based on their variability. For fields with non-zero standard deviation or non-zero entropy in the system database ($\mathsf{sd_i}, \mathsf{e_i} > 0$), we directly use the raw absolute difference $\mathsf{diff_i}$ as the per-field gap. Fields with zero standard deviation ($\mathsf{sd_i} = 0$) or zero entropy

$$d_i = \begin{cases} \mathrm{diff}_i, & \sigma_{i,\max} > 0 \text{ or } e_{i,\max} > 0 \\ \gamma_{\text{bool}}, & e_{i,\max} = 0,\ F_i \in \mathcal{B}, \\ \eta_{\text{cat}}\,\mathrm{diff}_i, & e_{i,\max} = 0,\ F_i \in \mathcal{C}_{\text{seen}}, \\ \gamma_{\text{cat}}\,\mathrm{diff}_i, & e_{i,\max} = 0,\ F_i \in \mathcal{C}_{\text{unseen}},\ \mathrm{diff}_i > \mathrm{R}_i/2, \\ \mathrm{diff}_i, & e_{i,\max} = 0,\ F_i \in \mathcal{C}_{\text{unseen}},\ \mathrm{diff}_i \le \mathrm{R}_i/2, \\ \mathrm{diff}_i(\alpha\,\mathrm{diff}_i)^2, & \sigma_{i,\max} = 0,\ F_i \in \mathcal{N}, \end{cases} \tag{3}$$

($\mathsf{e_i} = 0$) in the database are treated as highly stable configuration anchors and are handled more carefully because although they show no variation in our analytical data, future benign reconfigurations may still introduce small changes, we want to ensure that ConnSentinel exploits deviations on these fields as strong anomaly signals without turning rare but possible legitimate updates into false positives. Among these, for boolean categorical fields ($\mathcal{B}$) (e.g., `cellReservedForOperatorUse`), any mismatch between the target and reference cell incurs a fixed penalty γ_{bool}. For non-boolean categorical fields with zero entropy, if the target cell value has previously been observed by ConnSentinel for the same operator, we down-scale its contribution by multiplying $\mathsf{diff_i}$ with a factor η_{cat}, reflecting that such values are part of the benign configuration space and therefore should contribute less strongly to the similarity

score, which in turn reduces the likelihood of false positives compared to using the raw absolute difference. If the value is unseen, we apply a stronger penalty by adding a factor γ_{cat} when diff_i exceeds a reliable (empirically set to $R_i/2$) where R_i is the range of this field. Numeric fields with zero standard deviation are further amplified using a nonlinear scaling equation, where α is a scaling parameter that controls how aggressively large deviations on stable numeric fields are emphasized. We select the value of α on validation data and observe that the overall detection performance is robust to moderate changes around this value.

A.4. Node List for Graph Construction

State Monitoring Nodes: Registration/Attach Request, Authentication Response, NAS Security Mode Complete, RRC Security Mode Complete, Registration/Attach Complete, Service Accept, RRC Reestablishment Complete, RRC Reconfiguration Complete, TAU Complete.
Procedure Monitoring Nodes: Service Request/Reject, Registration/Attach Reject, Authentication Reject, Deregistration/Detach Request DL, NAS Security Mode Command, RRC Security Mode Command, Registration/Attach Accept, Identity Request, RRC Reject, RRC Release, Counter Check, RRC Reconfiguration, RRC Reestablishment Request, UE Capability Enquiry, TAU Request, TAU Reject.

Algorithm 1 Graph construction

procedure CONSTRUCTGRAPH($L_{imp\_node}, L_{mon\_node}, T_{encoded}$)
  $G \leftarrow None$
  $V_{prev} \leftarrow None$
  $V_{imp\_new} \leftarrow None$
  for each packet $p_i \in T_{encoded}$ **do**
    Create a node V_i
    $V_i.feature \leftarrow p_i.featrure$
    if $V_{prev} \neq None$ **then**
      G.ADDEDGE(V_{prev}, V_i)
    end if
    if $V_i \in L_{imp\_node}$ **then**
      if $V_{imp\_new} \neq None$ **then**
        G.ADDEDGE(V_{imp_new}, V_i)
      end if
      $V_{imp\_new} \leftarrow V_i$
    else if $V_i \in L_{mon\_node}$ and $V_{imp\_new} \neq None$ **then**
      G.ADDEDGE(V_{imp_new}, V_i)
    end if
    $V_{prev} \leftarrow V_i$
  end for
    return G
end procedure

A.5. Additional Evaluations

End-to-End Comparison with Existing Apps We also conducted an end-to-end comparison with all existing solutions that provide publicly available apps, including SnoopSnitch [21], AIMSICD [22], FBSDetector [27], and CellGuard [49]. We downloaded the latest versions of these apps from their official websites. Since SnoopSnitch, AIMSICD, and FBSDetector do not support 5G fake gNB detection, we instead configured a fake eNB in our controlled lab environment with different configurations, as described in Appendix A.1.2. Using a victim device with our commercial SIM card, we ran these apps in real time. SnoopSnitch and AIMSICD could not operate due to compatibility issues, which is consistent with the prior report that they have not been maintained for several years [27]. For FBSDetector, the app runs but fails to detect any of the fake eNBs we configured, including our attack scenarios. We were unable to determine the root cause of this failure, as the App's source code is not publicly available. CellGuard [49] is the only one that supports fake gNB detection. To evaluate its effectiveness, we deployed a fake gNB in a controlled experimental environment with the same settings as above and launched all types of attacks from our dataset against a test victim device running CellGuard. The results indicate that it detected only fake gNBs using Configuration C (impersonation of an unknown base station). By closely replicating the fake gNB's basic parameters (e.g., PLMN, cell id, and bandwidth) of a nearby legitimate base station, an attacker could evade its detection based on configuration, distance, and signal strength. In contrast, `5GShield` is capable of detecting fake gNBs in the *pre-connection* phase if an attacker fails to *precisely* replicate the `SIB1` fields. Furthermore, CellGuard can only detect 6 of the 26 attack types that rely on sending reject messages.

A.6. Additional Tables & Figures

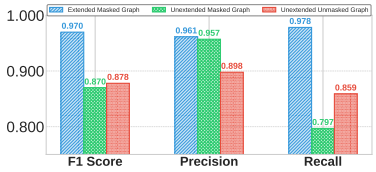


Figure 8: Ablation study for graph construction technique.

TABLE 7: List of devices.

#	Device Name	SoC Model
1	Motorola Edge+ 2022	Snapdragon 8 Gen 1
2	Motorola Edge 2025	MediaTek Dimensity 7400
3	Xiaomi 12	Snapdragon 8 Gen 1
4	Oneplus 13R	Snapdragon 8 Gen 3
5	Oneplus 10 Pro	Snapdragon 8 Gen 1

TABLE 8: Attacks in 4G NAS/RRC.

Attack	Adversary	Impact	Remediation Actions
CounterCheck Finger Printing [10]	FBS/MitM/Signal Injector	Fingerprinting	NU + TR + SC + TB
IMSI-Catcher [64]	FBS/MitM/Signal Injector	Information leak	NU + TR + SC + TB
IMEI-Catcher [11]	FBS/MitM/Signal Injector	Information leak	NU + TR + SC + TB
IMEISV-Catcher [11]	FBS/MitM/Signal Injector	Information leak	NU + TR + SC + TB
Attach Reject DoS [43]	FBS/MitM/Signal Injector	DoS, Downgrade	NU + SC + TB
Plaintext RRC Reconf DoS [4]	FBS/MitM/Signal Injector	DoS	NU + SC + TB
NAS SMC replay [10]	FBS/MitM/Signal Injector	Tracking	NU + TR + SC + TB
RRC SMC bypass [36]	FBS/MitM	Information leak	NU + SC + TB
Numb Attack [28]	FBS/MitM/Signal Injector	DoS, Downgrade	NU + SC + TB
Service Reject DoS [30]	FBS/MitM/Signal Injector	DoS, Downgrade	NU + SC + TB
Auth Sync Failure [4]	FBS/MitM/Signal Injector	DoS	NU + SC + TB
Paging Panic [28]	FBS/Signal Injector	Life threatening	NU + SC + TB
TAU Reject [4]	FBS/MitM/Signal Injector	DoS	NU + SC + TB
Detach Attack [28]	FBS/MitM/Signal Injector	DoS	NU + SC + TB
NAS counter reset [4]	FBS/MitM	DoS	NU + SC + TB
NAS SMC w/ EIA0 [10]	FBS/MitM/Signal Injector	Information leak	NU + SC + TB
RRC SMC w/ EIA0 [10]	FBS/MitM	Information leak	NU + SC + TB
Energy Depletion [28]	FBS/MitM/Signal Injector	Battery depletion	NU + SC + TB
UL NAS counter desync [4]	FBS/MitM/Signal Injector	DoS	NU + SC + TB
Lullaby [4]	FBS/MitM/Signal Injector	DoS	NU + SC + TB
Incarceration [4]	FBS/MitM/Signal Injector	DoS	NU + SC + TB
Cell Barring [29]	Signal Injector	DoS	NU + TR + SC + TB
Adaptover DoS [30]	Signal Injector	DoS	NU + TR + SC + TB
`Unknown attack`	FBS/MitM/Signal Injector	Unknown	NU + TR + SC + TB

Appendix B.
Meta-Review

The following meta-review was prepared by the program committee for the 2026 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

B.1. Summary

This paper presents 5GShield, an in-device framework for detecting and mitigating control-plane threats in 5G networks. The system consists of two complementary components: ConnSentinel, which analyzes broadcast configurations to detect suspicious base stations prior to connection establishment, and ExFinder, which monitors control-plane traffic to identify anomalous behavior during runtime. The approach combines rule-based consistency checks with a hybrid learning pipeline based on graph representations of protocol behavior. The evaluation is conducted on datasets collected from commercial networks and includes both known and unknown attack scenarios, demonstrating the effectiveness of the system in detecting control-plane threats observable at the UE. As the system is trained on and models behavior from commercial network traces, standard-compliant but previously unseen benign behaviors may be flagged as anomalous, and attacks that closely mimic benign configurations or rely on limited message-level deviations may be difficult to distinguish from normal operation.

B.2. Scientific Contributions

- Provides a New Data Set For Public Use.
- Creates a New Tool to Enable Future Science.
- Provides a Valuable Step Forward in an Established Field.

B.3. Reasons for Acceptance

1) The paper presents a practical in-device architecture for detecting control-plane threats directly at the UE, addressing a well-known gap in cellular security where operator-side defenses are insufficient.
2) The design cleanly separates pre-connection and post-connection detection (ConnSentinel and ExFinder), providing a well-motivated and deployable system architecture.
3) The paper introduces a comprehensive dataset collected from commercial 5G deployments, including both benign and attack traces, which is valuable for future research.
4) The system demonstrates strong empirical performance in detecting both known and previously unseen attacks, while maintaining low overhead suitable for deployment on commercial devices.