5G-SPECTOR: An O-RAN Compliant Layer-3 Cellular Attack Detection Service

Haohuang Wen*, Phillip Porras†, Vinod Yegneswaran†, Ashish Gehani†, Zhiqiang Lin*
*The Ohio State University, †SRI International
{wen.423, lin.3021}@osu.edu, {porras, vinod, gehani}@csl.sri.com

***Abstract*—Over the past several years, the mobile security community has discovered a wide variety of exploits against link and session-establishment protocols. These exploits can be implemented on software-defined radios (SDRs) that disrupt, spoof, or flood layer-3 (L3) messages to compromise security and privacy, which still apply to the latest 5G mobile network standard. Interestingly, unlike the prior generations of closed (proprietary) mobile network infrastructures, 5G networks are migrating toward a more intelligent and open-standards-based fully interoperable mobile architecture, called Open RAN or *O-RAN*. The implications of transitioning mobile infrastructures to a software-defined architectural abstraction are quite significant to the INFOSEC community, as it allows us to extend the mobile data plane and control plane with security-focused protocol auditing services and exploit detection. Based on this design, we present** 5G-SPECTOR**, the first comprehensive framework for detecting the wide spectrum of L3 protocol exploits on O-RAN. It features a novel security audit stream called** MOBIFLOW **that transfers fine-grained cellular network telemetry, and a programmable control-plane xApp called** MOBIEXPERT**. We present an extensible prototype of** 5G-SPECTOR **which can detect** 7 **types of cellular attacks in real-time. We also demonstrate its scalability to** 11 **unknown attacks as well as** 31 **real-world cellular traces, with effective performance (high accuracy, no false alarms) and low ($<$2% CPU, $<$100 MB memory) overhead.**

I. INTRODUCTION

Fifth-generation (5G) cellular networks have significantly impacted many sectors, including communication, transportation, entertainment, manufacturing, and healthcare, due to their extremely low latency and high data bandwidth. From the perspectives of development and operation, one significant advancement in 5G is the emerging Open Radio Access Network (O-RAN) [1] paradigm that is informing a next-generation radio access network (RAN). The O-RAN design addresses many limitations of conventional RANs, which are monolithic, all-in-one systems deployed on vendor-proprietary hardware infrastructures [2]. O-RAN brings not only openness and interoperability but also the unprecedented *programmability* that enables stakeholders (e.g., network operators) and innovators to build novel software-defined services on the RAN.

Network and Distributed System Security (NDSS) Symposium 2024
26 February - 1 March 2024, San Diego, CA, USA
ISBN 1-891562-93-2
https://dx.doi.org/10.14722/ndss.2024.24527
www.ndss-symposium.org

The O-RAN design is inspired by the software-defined network (SDN) [3] principles, as it separates the RAN control plane (e.g., network topology management) from the data plane (e.g., packet processing and forwarding). The control plane logic is integrated into a RAN Intelligent Controller (RIC), which serves as a standalone programmable component to manage the RAN nodes from a singular point of view [4]. Custom application-layer services, such as key performance measurement and traffic steering applications, are developed as "plug-n-play" *xApps* on the RIC. They are accompanied by specific *E2 service models* (E2SMs) [5] that define how they should communicate with RAN nodes. Based on this design, sophisticated analytics (e.g., machine learning) can be integrated as xApps into the RIC for a variety of network monitoring and control functions.

From a security perspective, O-RAN enables tremendous opportunities to design and develop novel control-plane services to secure the RAN data plane. Due to the availability of inexpensive commercial-off-the-shelf (COTS) software-defined radios (SDRs) [6] and open-source cellular software stacks [7], [8], there are relatively low economic and technical barriers for practical cellular attacks. Attackers can employ malicious user equipment (UE) [9], [10], [11], fake base stations [10], [12], [13], and Man-in-the-Middle (MiTM) attacks [14], [15], [16], [17], [18]. These threats can trigger service outages, privacy leakage, and quality-level downgrades, compromising the security, privacy, and availability of the network for end-users.

However, it is not trivial to detect these attacks, as they operate through surgical manipulation on cellular control messages [9], [10], [17], [19], such as the layer-3 (L3) protocols including the Radio Resource Control (RRC) [20] and Non-Access Stratum (NAS) [21]. These attacks originate from the vulnerabilities in earlier generations of cellular protocols but most of them still apply in 5G standards [10], [18]. However, while there are existing defenses that address a subset of these attacks from either the UE side [22], [23] or the network side [24], [25], [26], they fall short in having a limited view (e.g., UE-centric defenses cannot detect RAN-targeted attacks [22], [23]) or poor extensibility (e.g., network-based solutions with static defense mechanisms [24], [25], [26]).

To fill this gap, we present 5G-SPECTOR, the *first* comprehensive O-RAN-resident framework to detect L3 cellular attacks. Its objective is to enable network experts to easily program exploit detection logic integrated into an xApp that works as an intrusion detection system (IDS) and has three critical design goals. First, 5G-SPECTOR should operate on telemetry with high *granularity* to support fine-grained security analysis. Second, it should have the *extensibility* that allows new detection mechanisms to be easily integrated for both contemporary and emergent exploits. Third, its *efficiency* should allow the processing of a large volume of packets and report of malicious exploit events in near real-time.

To overcome these challenges, we develop novel extensions and xApps in 5G-SPECTOR that augment the O-RAN data plane and control plane. First, we design and develop MOBIFLOW [27], a security audit stream for fine-grained security analysis, which is produced by an O-RAN compliant service module called SECSM. Inspired by NetFlow [28] in the TCP / IP network, MOBIFLOW aggregates cellular network packets into flow records during network transmission and eventually converts them into packet-level telemetry streams. Next, we develop MOBIEXPERT as an O-RAN control-plane xApp based on the Production-Based Expert System Toolset (P-BEST) language [29]. To this end, it provides MOBIEXPERT with programmability for network operators to easily write production rules to detect attacks with precision and coverage.

We have instantiated 5G-SPECTOR on SD-RAN [30] with 758 lines of P-BEST code programmed as IDS rule sets for 7 types of existing L3 exploits [10], [9], [17], [19], and evaluate 5G-SPECTOR on an O-RAN testbed. We first demonstrate that 5G-SPECTOR can effectively detect the 7 attacks simulated in our testbed, and can be easily extended with 90 LoC to detect additional 11 unknown attack variants. Further, we tested 5G-SPECTOR with 31 network traces collected from COTS UEs [31], [22] and show that 5G-SPECTOR scales to real-world cellular networks without reporting any false alarms. Meanwhile, we demonstrate 5G-SPECTOR's practicality in detecting two end-to-end exploits reproduced over-the-air [9]. Finally, we measure that 5G-SPECTOR has decent data processing speed (>40K MOBIFLOW packets/s), detection latency (<500 ms), and overhead (<100 MB in memory and <2% in CPU) to both the data plane and control plane.

Contribution. Our paper makes the following contributions:

- We present the first framework for O-RAN to detect layer-3 cellular attacks, with a novel telemetry stream (MOBIFLOW) and a programmable xApp (MOBIEXPERT).
- We have instantiated a prototype of 5G-SPECTOR atop an O-RAN testbed and demonstrate its capability of detecting 7 types of practical L3 cellular attacks.
- We comprehensively evaluate our implemented 5G-SPECTOR prototype in terms of detection effectiveness, scalability, practicality, performance, and overhead.

II. BACKGROUND

A. Cellular Network Procedures

5G Cellular networks typically consist of three abstracted entities: (1) the user equipment (UE), such as a smartphone that subscribes to the operational network through a USIM

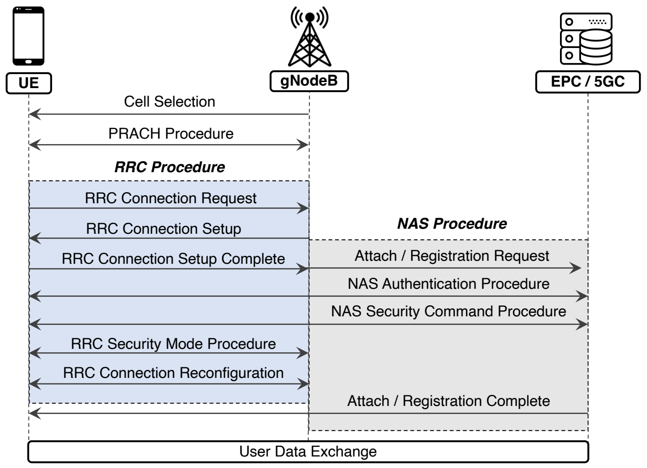

UE
gNodeB
EPC / 5GC
Cell Selection
PRACH Procedure
RRC Procedure
RRC Connection Request
RRC Connection Setup
RRC Connection Setup Complete
RRC Security Mode Procedure
RRC Connection Reconfiguration
NAS Procedure
Attach / Registration Request
NAS Authentication Procedure
NAS Security Command Procedure
Attach / Registration Complete
User Data Exchange

Fig. 1: Typical cellular network procedure when a UE initially attaches to the network.

(Universal Subscriber Identity Module), (2) the gNodeB, or the 5G base station (BS)[1] located within the radio access network (RAN), which connects the UE to the operator's network, and (3) the core network handling multiple network functions such as authentication and key agreement. A gNodeB can connect to either an LTE Evolved Packet Core (EPC) or a 5G Core (5GC), which is known as the non-standalone (NSA) mode or standalone (SA) mode, respectively [32]. Before a UE establishes a data connection, it first needs to attach to a nearby gNodeB and register with the EPC or 5GC through several critical procedures in the cellular control plane. This is illustrated in Figure 1. Below, we explain the details of the three main control-plane procedures.

Cell selection and PRACH procedure. For the UE to select a gNodeB to camp on, it first acquires and decodes the System Information Block (SIB) messages broadcast from each nearby gNodeB. Based on the gNodeB meta information and status in the SIB messages, the UE internally performs measurement to select an optimal gNodeB. Next, the UE and the gNodeB start the physical random access channel (PRACH) in which the UE requests uplink synchronization and is allocated a Radio Network Temporary Identifier (RNTI) for radio access communication [33].

RRC procedure. The Radio Resource Control (RRC) [20] procedure occurs after the initial cell selection and the PRACH procedure. It usually starts from a connection request message initiated by the UE, which contains the UE's establishment cause and identifier (e.g., a random identity or its Temporary Mobile Subscriber Identity (TMSI) if the UE has been allocated one previously). If the gNodeB agrees to set up the connection, it sends a downlink connection setup message with the UE's configuration information. Finally, the UE finishes the handshake with a connection setup complete message.

NAS procedure. Based on an established RRC connection, the Non-Access Stratum (NAS) [21] procedure is performed when

[1]We use the term BS, gNodeB, and RAN interchangeably.

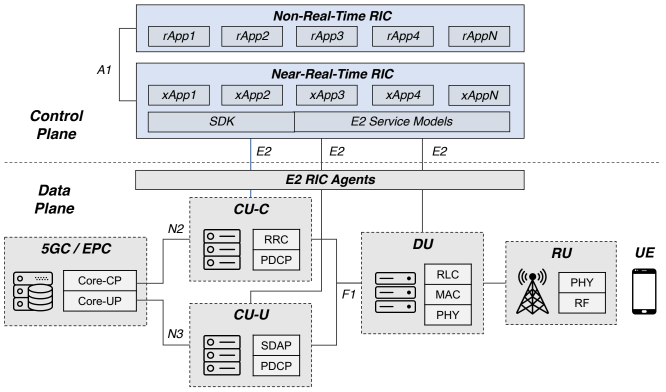


Fig. 2: Open Radio Access Network (O-RAN) architecture.

the UE attempts to attach to the EPC/5GC. More precisely, the NAS messages are transmitted between the UE and the Access and Mobility Management Function (AMF) in a 5GC (or the Mobility Management Entity (MME) in an EPC), and are relayed by the gNodeB. As shown in Figure 1, a typical NAS procedure starts with an Attach Request (LTE) or Registration Request (5G) if this is the initial attachment, which contains the UE's temporary (TMSI) or permanent identifiers such as its Subscription Concealed Identifier (SUCI) and International Mobile Subscriber Identity (IMSI). Next, both parties start the authentication and security mode procedures to activate the encryption and integrity protection for future communication. Finally, the NAS procedure culminates with a downlink Attach or Registration Complete message and the UE starts user data transmission on the cellular data plane. Although not covered in Figure 1, the gNodeB or the EPC/5GC could reject the RRC or NAS connection and generate failure messages according to the specification [21]. For example, a NAS attach reject or registration reject message will be transmitted if the UE's request involves invalid parameters.

B. *O-RAN*

Figure 2 [4] summarizes the O-RAN architecture. We explain its data plane and control plane design in the following.

O-RAN Data Plane. The O-RAN design embraces the functional split in the 3GPP specifications [34]. It breaks down the conventional all-in-one RAN into logical nodes including the *Radio Unit* (RU), *Distributed Unit* (DU), and *Central Unit* (CU). The RUs are typical radio hardware deployed in the front-haul network to handle layer-1 (L1) physical radio signals from surrounding user equipment. The DUs and CUs are logical components that can be hosted at the edge to handle L2 and L3 functions of the cellular protocol. The DU handles L2 functions such as Media Access Control (MAC) and Radio Link Control (RLC). The CU is responsible for L3 control protocols such as RRC, and it is further split into the CU-C and CU-U to forward control-plane and user-plane traffic, respectively. The CU-C and CU-U are further attached to CN functionality, such as the AMF and the User Plane Function (UPF). The O-RAN data plane components are connected via standard and open interfaces. For instance, DUs and CUs are connected by the F1 interfaces [35].

O-RAN Control Plane. The control layer logic of O-RAN is disaggregated from the data plane based on the SDN principles. The O-RAN control functions are realized in the RAN Intelligent Controller (RIC). Its logic is customized with hosted xApps. Fundamentally, the RIC serves as a proxy for control services and connects to the RAN nodes (i.e., CUs and DUs) via the standard E2 interface [36]. The interactions between the RIC and the RAN nodes are defined by four basic E2 operations: *Report*, *Insert*, *Control*, and *Policy*. Based on these operations, xApps can be programmed as "plug-n-play" software on the RIC. An xApp needs to define *E2 Service Models* (E2SMs) as function-specific protocols on top of the generic *E2 Application Protocol* (E2AP) [36] to engage with the O-RAN data plane. For instance, the O-RAN Alliance has demonstrated a few exemplar xApps and E2SMs such as key performance measurements (KPM), RAN slicing management, and traffic steering. According to the latency requirement, the RIC can be classified into the near-real-time RIC (nRT-RIC) and non-real-time RIC. Each control loop of a nRT-RIC completes within the range of 10ms to 1000ms, while non-real-time tasks (e.g., ML model training) are hosted as rApps on the non-real-time RIC with over 1s latency.

III. Overview

A. *Threat Model*

Threat Model and Assumptions. We consider cellular network adversaries who can compromise the availability, confidentiality, and integrity of UEs and RANs, by manipulating and transmitting unprotected cellular protocol messages [37], while respecting the cryptographic properties. To this end, we assume that any UEs can be malicious, and the communication channel between UEs and RANs is subject to man-in-the-middle (MiTM) attacks [14] such as signal injection [19]. More precisely, we focus on *the L3 attack surfaces that exploit unprotected (i.e., not signed or encrypted) cellular messages within the NAS and RRC protocols [38] over the radio frequency (RF) channel*, which have been extensively studied [10], [12], [14], [9], [16], [13], [39], [40], [41], [17]. We provide the list of such unprotected messages [21], [20] in Table V (in the Appendix). To limit the scope, we exclude any passive attacks that are not detectable. We also assume that the O-RAN control plane, the core network, and all reporting RAN nodes that subscribe to the nRT-RIC are trusted.

Adversarial Classes. Based on the threat model, we focus on three distinct adversary classes depicted in Figure 3. In addition to these three classes, fake base station (FBS) attacks [10], [13], [12], [9] are performed by luring victim UEs to connect to an FBS that spoofs a legitimate base station. Although we can certainly deploy existing FBS detection algorithms on O-RAN [42], [24], [11], they have been extensively addressed in prior work and thus are not within the scope of our effort. Next, we describe the three adversarial classes.

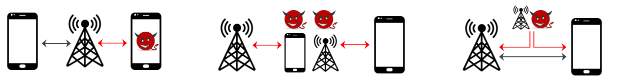

(a) Adversary UE (b) MiTM Attacker (c) Signal Injector

Fig. 3: Adversary models of our focus.

- **Adversarial UE** (Figure 3a). With a COTS SDR running open-source cellular software [7], [8] and a valid subscriber network identity (e.g., SIM), one can set up an adversarial UE. Next, the attacker can modify the cellular protocol stack to perform availability attacks to compromise other victim UEs and the RANs. For instance, a base transceiver station (BTS) resource depletion can be launched by continuously creating massive fabricated RRC connection requests to Denial-of-Service (DoS) a target BS [9], [10]. An adversary may also DoS a legitimate UE by replaying its identity (e.g., TMSI) in a connection request [9]
- **MiTM Attacker** (Figure 3b). A MiTM adversary impersonates a legitimate BS to a victim UE, and a legitimate UE to a victim BS, which requires two SDRs. A MiTM attacker can replay or modify messages in the traffic, by exploiting messages that are not encrypted and digitally signed.
- **MiTM Signal Injector** (Figure 3c). Most recently, it has been shown that MiTM adversaries can use an SDR to inject malicious signals to overshadow the downlink and uplink traffic while maintaining a high level of stealth (e.g., using slightly higher signal strengths such as 3dB more) [16]. As a result, this signal injection (or overshadowing) attack can be viewed as a stealthy version of MiTM attacks. Signal injection can be further exploited to launch privacy and availability attacks, such as a DoS attack [19] and IMSI extraction on UEs [17].

B. Design Goals and Challenges

Based on the threat model, we outline three design goals of 5G-Spector as well as the corresponding challenges.

G1. Granularity. As illustrated in §II-B, an xApp on O-RAN is driven by its underlying E2 service model (E2SM), which is also required by 5G-Spector to perform intrusion detection. Although O-RAN has published several exemplar E2SMs [5], their reported telemetry is too coarse-grained for security analysis (e.g., packet delay and drop rate in E2SM-KPM [2]). While we can build packet-level telemetry that ideally offers the highest granularity, it would be too costly due to the large cellular traffic volume [43].

G2. Extensibility. As cellular attacks continue to evolve, the second goal is to make 5G-Spector extensible for current and future attacks. While we can employ learning-based frameworks (e.g., automata-based [22] and machine-learning-based [2]), they fall short due to the availability of abundant adversary training samples in public [22]. Besides, existing cellular IDS frameworks are mostly static with relatively low flexibility [24], [23], [26], [42]. To achieve this design goal, we build a flexible rule-based expert system that enables users (e.g., network operators) to easily integrate exploit detection signatures in a programmatic manner. However, such a design needs to address several challenges, such as a decoupled architecture (i.e., disaggregation of detection mechanisms from the system) and a low redeployment cost.

G3. Efficiency. The third goal is that the efficiency of 5G-Spector should be high to make it compliant to the nRT-RIC. This will enable 5G-Spector to detect and report attacks in near real-time, and have the potential to be deployed in operational cellular networks in practice (which often need to process a large number of cellular messages simultaneously). As a result, the rule inference engine of 5G-Spector should be written in an efficient programming language that allows it to process a large volume of data packets and generate alerts and events with low latency.

C. 5G-Spector Overview

We present a high-level overview of 5G-Spector's architecture in Figure 4. It is decomposed into the control layer and the app layer, and we have designed key components to achieve the above three design goals. In the following, we first give an overview of the control layer and app layer, followed by the detailed design in §IV and §V.

Control Layer. (§IV) 5G-Spector's control layer serves as an intermediate component to provide basic services for the xApp analytics, adhering to the O-RAN architecture and protocols [36], [5]. Specifically, it generates a fine-grained telemetry stream as MobiFlow by using an O-RAN compliant security module SecSM, and we provide their descriptions below.

- **MobiFlow**. The design of MobiFlow is inspired by *NetFlow*, a network monitoring stream widely deployed in TCP/IP networks [28]. Similarly, MobiFlow aggregates UE- and RAN-specific data into flow and reports them per trigger event, which is later dissected and converted into fine-grained records at the nRT-RIC. Additionally, MobiFlow selectively monitors *L3 control-plane packets* that are necessary for our detection, and a typical UE attachment involves less than 20 control packets [20], [21].
- **SecSM**. SecSM is an O-RAN compliant security module that generates MobiFlow stream records. It consists of a specialized E2SM specification [5] that define the message structures and SecSM agents (located at both the data plane and control plane) that generate MobiFlow records and transmit them to the upper app layer.

App Layer. (§V) The app layer includes the MobieXpert xApp. MobieXpert's design is based on the Production-Based Expert System Toolset (P-BEST) language, which has been widely used for decades in stateful intrusion detection (e.g., in mainframes, operating systems, network traffic, application logs, and for alert correlation) [29], [44], [45], [46], [47]. P-BEST features a decoupled architecture that separates detection mechanisms from the system implementation and also provides an efficient P-BEST language that allows production-based rules to be translated into C programs. For

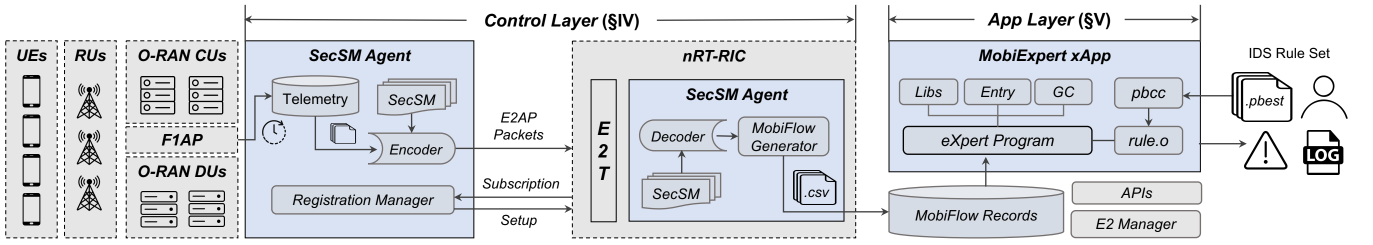


Fig. 4: Architecture of 5G-Spector, which is broken down into the control layer (§IV) and the app layer (§V).

instance, we later show that a 758-lines P-BEST specification can be converted into over 4,000 lines of C code. This language also enables one to comprehensively program sophisticated rules to perform temporal and quantitative analysis for attack inference using MobiFlow.

IV. Control Layer Design

We describe the control layer design of 5G-Spector, including an O-RAN compliant service module SecSM (§IV-A) and the MobiFlow telemetry stream (§IV-B).

A. SecSM

The objective of SecSM is to create O-RAN compliant interfaces that allow *security-focused* xApps to receive the necessary data-plane telemetry that can drive run-time security analyses across the mobile network. To achieve this goal, SecSM incorporates several agents that serve as plugins to extract telemetry for both the RAN data and the control planes. SecSM also defines an E2 service model (E2SM) specification in ASN.1 language for the corresponding communication protocol (e.g., indication packet format) [5]. To summarize, SecSM provides two main services: (1) registration procedures between the E2 nodes and the xApps, (2) telemetry collection and report via the standard E2 interface.

Registration Management. Initially, the E2 nodes (i.e., CUs and DUs) and xApps need to go through registration procedures through the nRT-RIC. The RIC registers an E2 node via an *E2 Setup* procedure. An xApp subscribes to an E2 node via a *RIC Subscription* procedure [36]. Figure 5 presents the workflow of these two procedures.

- **E2 Setup**. Based on an established SCTP connection, an E2 node initiates an `E2 Setup Request` to the RIC. The request conveys the meta information and capability of the E2 node and contains the following four elements [36]. First, a RAN function definition describes the E2 node with its supported E2SM ID. Second, an E2 node list contains meta information such as the public land mobile network (PLMN) ID. Third, the event trigger style defines the supported conditions for when the E2 node should report to the nRT-RIC. For SecSM, it mainly supports periodic reports based on time intervals. Fourth, the report style declares the telemetry that can be collected from the E2 node. Upon receiving a setup message, the RIC relies on either an `E2 Setup Response` or an `E2 Setup Failure` to inform the outcome.

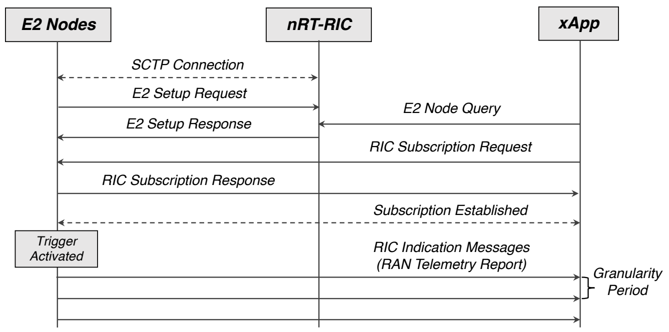


Fig. 5: E2 setup and xApp subscription procedures.

- **RIC Subscription**. After deployment, an xApp hosted by the nRT-RIC can query the connected E2 nodes for their RAN function definitions and identify the nodes to subscribe to. There are four basic subscription types based on the E2 interface operations as mentioned in §II-B: *Report*, *Insert*, *Control*, and *Policy* [36]. In particular, SecSM only makes use of E2's *Report* to collect telemetry. To start a subscription, an xApp initiates a `RIC Subscription Request` to the E2 nodes of interest and selects the appropriate event trigger and report style.

Telemetry Collection and Reporting. The RAN- and UE-related telemetry is collected by a SecSM agent deployed on the CUs and DUs. The agent is considered to be an independent and vendor-agnostic plugin running in parallel with the normal CU and DU processes. More specifically, it instruments the standardized F1 application protocol (F1AP) interfaces defined in 3GPP TS 38.473 [35], which handle various basic procedures (e.g., UE context setup and RRC message relay). Although MobiFlow only requires a subset of the F1AP telemetry, we summarize all supported data that could be collected in Table VI in the Appendix.

The high-level workflow of the telemetry collection procedure is illustrated in algorithm 1. Based on the event trigger defined in the E2 setup message, the SecSM agent collects telemetry per *granularity period* (e.g., 100ms at a near real-time scale) and aggregates them into an indication message. The agent maintains a telemetry buffer for each subscribed UE, and only initiates a report when the specific UE has been updated (e.g., new state or control traffic). For each UE to be reported, its measurement elements (defined in the SecSM specification) are encoded into an indication message with

Algorithm 1: Telemetry collection procedure.

```
1  Function
       // RIC subscription finished
2      for each granularity period do
3          for each $ue$ and $ue$.shouldReport do
4              indicationMsg ← NULL
5              for $i \in$ measurement items of $ue$ do
6                  indicationMsg.add($i$.name, $i$.value)
7              end
8              for $m \in$ ue.msgBuf do
9                  if $m$ == "DLInformationTransfer" or
10                    $m$ == "ULInformationTransfer" and
11                    canDecrypt($m$.dedicatedInfoNAS) then
                       // Encode NAS message
12                     nas ← $m$.dedicatedInfoNAS
13                     encodeMsg ← 0 | (nas.discriminator ≪ 1) |
                         (nas.msgId ≪ 2)
14                 end
15                 else
                       // Encode RRC message
16                     encodeMsg ← 1 | ($m$.channel ≪ 1) |
                         ($m$.direction ≪ 2) | ($m$.msgId ≪ 3)
17                 end
18                 indicationMsg.add(index, encodeMsg)
19             end
20             asn1c_encode_indication_msg(indicationMsg)
21         end
22     end
```

their keys and values. To reduce the payload size, the UE's RRC and NAS message IDs are encoded into an integer value. At line 8, the algorithm iterates over each RRC message that has not been reported since the last granularity period. As NAS messages are also relayed by the RRC traffic [20], it first checks if the message carries a NAS payload (dedicatedInfoNAS) and can be decrypted. If so, the NAS payload's discriminator and message ID [21] are extracted to encode a NAS message. If the RRC message does not relay a NAS payload, its channel information (DCCH or CCCH), direction (uplink or downlink), and message ID are used as different bits to encode the message [20]. To distinguish RRC and NAS messages, a single header bit (either 0 or 1) is used.

After the telemetry is loaded into a `RIC Indication Message`, it is further encoded into an ASN.1 structure based on the SecSM E2SM specification. The indication messages are further embedded in standardized E2AP packets [36] and delivered to the nRT-RIC. The SecSM agent reports one such E2AP packet per granularity period.

B. MobiFlow

The telemetry collected and reported from the F1AP interfaces [35] lack certain fine-grained parameters necessary for security analysis. For instance, it does not express global statistics and stateful information (e.g., how many UEs are currently connected, and what states these UEs are in). Thus, another SecSM agent is deployed at the nRT-RIC to convert this reported telemetry into fine-grained MobiFlow stream records, which requires additional procedures such as state transformation and statistics aggregation. As a result, MobiFlow record generation is not performed on latency-sensitive O-RAN data plane due to performance concerns.

MobiFlow Structure. While MobiFlow can be defined as various structures, we provide an instance of it for detecting L3 cellular attacks in this paper. As shown in Table I, there are two types of MobiFlow, which capture the fine-grained state transitions of UEs and aggregated statistics of RANs, respectively. Both types start with a common header to indicate the message type, ID, timestamp, and reporter ID. Below, we describe the attributes in each MobiFlow type.

Category	Telemetry	Type	UE	RAN
Header	Msg Type	Integer	●	●
	Msg ID	Integer	●	●
	TimeStamp	String	●	●
	Reporter ID	Integer	●	●
Metadata	C-RNTI	Integer	●	○
	S-TMSI, IMEI, IMSI	String	●	○
	MNC, MCC, TAC, CellID	Integer	○	●
	Granularity Period	Integer	○	●
States	Cipher/Integrity Algorithms	Integer	●	○
	RRC/NAS/SEC States	Integer	●	○
	RRC/NAS Msg IDs	Integer	●	○
	Connected/Idle/Max UE Count	Integer	○	●
Timer	RRC/NAS Timer	String	●	○
	Initial/Inactive Timer	String	○	●

TABLE I: Telemetry collected by MobiFlow stream records.

- **UE-centric MobiFlow**. As a UE uses various identities during cellular network connections [33], we track temporary identifiers, including C-RNTI and S-TMSI, as well as permanent identifiers, such as IMEI and IMSI/SUCI. For instance, C-RNTI allows the identification of a unique RRC connection between a UE and the RAN [20], and S-TMSI identifies a unique NAS session between a UE and MME/AMF [21]. The fine-grained UE states are at packet level to represent state transitions at various protocol layers including RRC and NAS [20], [21]. The timing information is tracked via timers that indicate when the UE starts and ends a specific RRC/NAS session.
- **RAN-centric MobiFlow**. This type of MobiFlow aims to provide real-time aggregated statistics of the RAN (i.e., a physical BS in practice). It includes physical RAN identifiers, such as MCC (Mobile Country Codes) and MNC (Mobile Network Codes). The RAN states include statistics, such as the current number of connected and idle UEs, as well as the RAN's maximum capacity. Similar to the UE-centric MobiFlow, timers are recorded to track the life cycle of the RAN.

MobiFlow Generation. Based on the MobiFlow definition, the SecSM agent at the nRT-RIC converts the telemetry into fine-grained MobiFlow stream records. As the reported telemetry is aggregated within a granularity period, additional steps are required to dissect them into packet-level stream records. Since not all measurements defined in Table VI are available from the RAN data plane, further computations and inferences are needed. In the following, we detail some additional steps that are required to generate the MobiFlow records defined in Table I.

- **Fine-grained State Transition**. These states enable analysts to understand the precise status at packet-level granularity (e.g., a UE establishes its security context after transmitting a `SecurityModeComplete`). In 3GPP TS 24.301 [21] and 38.331 [20], fine-grained protocol states are defined as finite-state machines (FSMs) and transit upon receiving certain control messages. For instance, the NAS protocol maintains states such as `EMM_REGISTERED` to indicate an established EMM context with the core. Thus, the SecSM agent maintains a copy of all UE and RAN states and performs state inference using specification-defined FSMs [20], [21]. A security state is also set to track whether a UE has completed the security mode procedures.
- **Timer Recording**. SecSM maintains timers to track the start and end of a session for each UE and RAN based on the state transitions. For example, once a UE enters a `RRC_CONNECTED` or `RRC_INACTIVE` state, the RRC initial or inactive timer will be set accordingly.
- **Statistics Aggregation**. For RAN-centric MobiFlow records, the aggregated UE statistics are computed when any UE states are updated. For instance, the active UE counter is increased when a new UE enters the `RRC_CONNECTED` state. The update of RAN states will also notify the agent to generate and report RAN-centric MobiFlow stream records to the MobieXpert xApp.

V. App Layer Design

Our design of the MobieXpert xApp emphasizes two key considerations: (1) *Programmability* that enables network operators to program production rule-based logic for intrusion detection, and (2) *Flexibility* as it is a light-weight "plug-n-play" xApp and can be easily integrated into the 5G O-RAN control plane without adding much performance overhead. MobieXpert's design is powered by the underlying telemetry streams from the SecSM and MobiFlow at the control layer (§IV), and its programmability is enabled by the P-BEST production rule language [29]. In the following, we first present the architecture of MobieXpert including several key components and then describe the P-BEST language features and syntax.

A. 5G-Spector Architecture

The architecture of MobieXpert is shown in Figure 4. At a high level, MobieXpert runs a standalone *eXpert* program that takes the MobiFlow stream records as input and produces alerts and logs for the networks. More specifically, it incorporates four main components: (1) a *pbcc* translator, (2) a main P-BEST routine, (3) a garbage collection (GC) routine, and (4) a set of static libraries. In the following, we dive into the details of each component.

pbcc Translator. The *pbcc* translator converts a specification file (e.g., `rule.pbest`) written in P-BEST production rule language into a functional C program. Through *pbcc*, elements in the P-BEST domains (facts and rules) are translated into corresponding C variables, structures, and functions. The converted C program is further linked to the main P-BEST routine during compilation.

eXpert Main Routine. The main *eXpert* C routine handles external inputs and the context switch between C and P-BEST domains. More specifically, developers only need to create an entry function that constantly reads MobiFlow records (e.g., from a csv, sparse binary, json, or database), inserts it into the fact pool using the library APIs, and the control is handed over to the internal *inference engine*. This engine is essentially a forward chaining system [29], which iteratively evaluates and activates (if rule conditions are satisfied) rules on each asserted fact until the fact pool is stable (i.e., no facts in the current fact pool bind to the ruleset). Afterward, the control is switched back to the C domain for the next cycle.

Garbage Collection Routine. While programming with P-BEST to create new facts, the corresponding garbage collection rules (GC) must be defined. This is to prevent unused facts from consuming memory resources or to remove facts that no longer require evaluation, as C programs require developers to manage dynamically allocated objects. The objective of GC is similar to invoking the `free()` function in C, but provides programmers with fine-grained semantic control of the fact-purging criteria. For instance, GC may commonly integrate time references to perform interval-based garbage collection or may purge facts based on the state of other facts in the global pool. Therefore, the best practice to program with P-BEST is to always define relevant GC rules when creating one or more facts in the written rules.

P-BEST Libraries. There is a set of static libraries (e.g., `libpb.a`) linked during compilation and provide P-BEST management APIs. For instance, in order to insert a new fact into the system, the *eXpert* main routine shall invoke an `assert()` function with the fact instance and then hand over the control to P-BEST by calling `engine()`.

B. P-BEST Production Rule Language

The P-BEST production rule language emphasizes two main properties: *efficiency* and *usability*. In addition to its high efficiency (as discussed in §III-C), this language has also been proven to have a low learning threshold via a user study, where over 70% of the participating students were able to successfully write a P-BEST system to detect malicious FTP traffic within four hours [29]. The basic elements in P-BEST are *facts* and *rules*, where a fact is considered a piece of knowledge in a *fact pool*, and a rule is the user-defined logic to create, modify, or delete some facts by evaluating the existing facts. At a high level, a P-BEST production rule is formulated as:

$$\frac{\exists S \ \wedge \ \phi_1 \ \wedge \ \phi_2 \ \ldots \ \wedge \ \phi_n}{S' = S \ \cup \ \psi_1 \ \cup \ \psi_2 \ \ldots \ \cup \ \psi_n}$$

where ϕ_1 to ϕ_n express the antecedents, ψ_1 to ψ_n express the produced consequences, and S to S' represent the transition from the previous state to a new state of the fact pool.

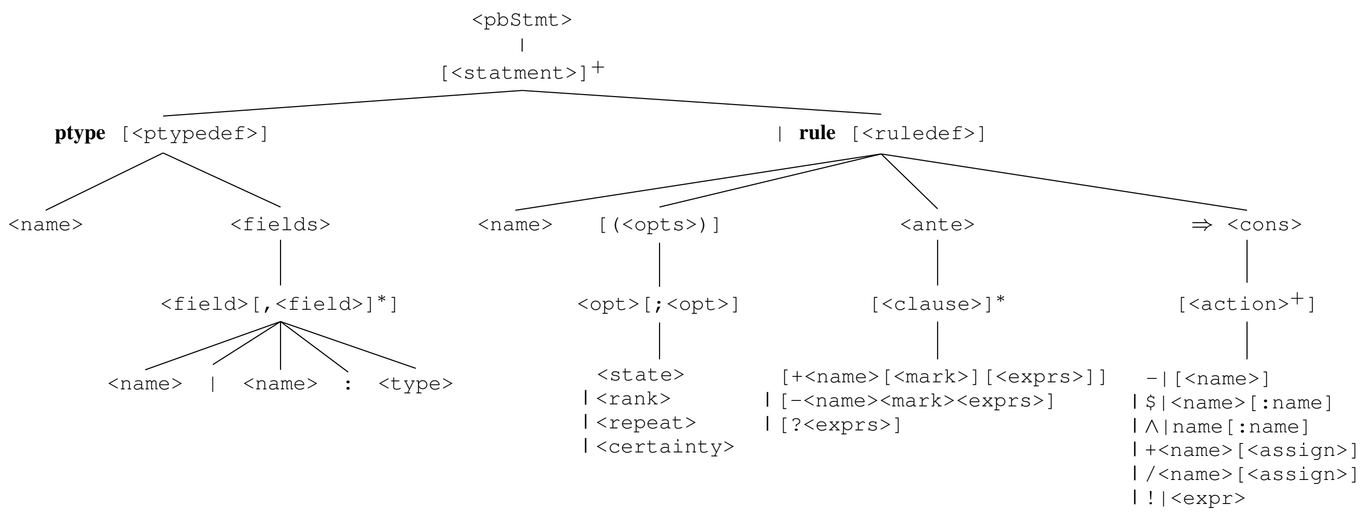


Fig. 6: Simplified abstract syntax tree of P-BEST production rule language for fact and rule definitions.

Figure 6 shows a simplified abstract syntax tree of the P-BEST language describing how facts and rules are defined as `ptypes` and `rules`, which are described in detail below.

- `ptype`. Before creating rules, users must declare certain data structures as pattern types, or `ptypes`. A `ptype` contains several data fields of different types (e.g., integer and string) and is converted into a C structure by the *pbcc*. A `ptype` is mapped to an event or a *fact* that will be evaluated by the user-defined rules. For instance, a MOBIFLOW record or an attack event can be defined as a `ptype` so that a rule can evaluate MOBIFLOW records to produce attack events. The P-BEST language allows one to create, modify, and delete a `ptype` instance with corresponding P-BEST operands (e.g., $+$, $/$, and $-$). A fact can also be masked ($) or unmasked ($\wedge$) to prevent itself from being repeatedly evaluated by a specific rule.
- `rule`. A P-BEST production rule can be written as an "IF...THEN..." structure. A P-BEST rule is defined via the `rule` keyword and uses a $\Rightarrow$ symbol to connect the antecedent and the consequent. When a rule is triggered, the conditions in the antecedent are evaluated sequentially, and corresponding consequent statements are executed if all conditions are satisfied. P-BEST supports a wide range of operations, including logical and arithmetic computation of various data types, as well as invoking (by using a ! operand) standard library functions in C (e.g., `strcmp()`) or user-defined C functions via a C-language bridge. A rule can also be assigned with properties, such as `<state>` (enabled or disabled) and `<rank>` (priority in execution order among other rules).

VI. IMPLEMENTATION

We have instantiated 5G-SPECTOR using SDRAN-in-a-Box (RiaB) [30][2] which is an O-RAN compliant platform with basic RIC services and xApp SDKs. Based on this platform, our implementation includes the MOBIEXPERT xApp and our control plane design (SECSM and MOBIFLOW). For the data plane, we adopt the OpenAirInterface (OAI) radio software suite for the CU, DU, and UE [8]. The core network is the ONF's Open Mobile Evolved Core (OMEC), an LTE EPC.

To realize a prototype of MOBIEXPERT, we first selected 7 L3 cellular attacks from previous literature [10], [9], [17], [19] that match our threat model (§III-A), as shown in Table II. Based on these attacks, we program a P-BEST rule specification with 758 LoC composed of 33 rules with 13 self-defined `ptypes`. These P-BEST specifications are further translated into over $4,000$ lines of C code by the *pbcc*. The implemented rules are based on certain *anomalies* and are categorized into three rule sets for the 7 attacks, as described below in detail.

(1) **Abnormal Quantity Rule Set**. Detection of some attacks requires quantitative reasoning on accumulated events, such as BTS resource depletion [9]. To this end, we define rules to generate anomalous events based on certain criteria that are tracked based on counter values. When the counters reach an adjustable threshold, the rule set concludes and reports an event.

(2) **Abnormal Message Sequence Rule Set**. Many attacks can be detected based on abnormal message sequences of UEs (also viewed as protocol state-machine bugs [22]). As such, this rule set can detect MiTM attacks that manipulate unprotected messages [17], [19]. For instance, an unsolicited `IdentityResponse` message is likely caused by the injection of a `IdentityRequest` message to extract IMSI in plain text [17].

(3) **Abnormal State Rule Set**. This rule set detects whether certain state parameters are abnormal. For instance, null ciphering or integrity attacks [10] trigger a UE into limited service mode with no cipher and integrity protection after security context establishment. These states can be checked at run-time to infer such a malicious event.

[2]Source code is available at https://github.com/5GSEC/5G-Spector.

Attack	Ref	Adv.	Type	Layer	Message Exploited	Alert Level	Rule	Example Inference Rule
BTS Resource Depletion	[9]	UE	A	RRC	ConnectionRequest	●	(1)	$\frac{(\exists cntr(c) \in S) \wedge (c_{value} > T_{ue})}{S' = S \cup \{event \mid c_{bs}, 0, t, \text{“BTS Resource Depletion”}\}}$
Blind DoS	[9]	UE	A	RRC	ConnectionRequest	●	(1)	$\frac{(\exists ue(u_1) \in S) \wedge (\exists ue(u_2) \in S) \wedge (u1_{bs} = u2_{bs}) \wedge (u1_{rrc\_state} = u2_{rrc\_state} = 2) \wedge (u1_{tmsi} = u2_{tmsi}) \wedge (u1_t < u2_t)}{S' = S \cup \{event \mid u2_{bs}, u2_{rnti}, t, \text{“Blind DoS”}\}}$
Downlink DoS	[19]	MiTM	A	NAS	AttachReject / RegistrationReject	●	(2)	$\frac{(\exists ue(u) \in S) \wedge (u_{msg_i} = Auth\_Req) \wedge (u_{msg_{i+1}} \neq Auth\_Resp/Fail)}{S' = S \cup \{event \mid u_{bs}, u_{rnti}, t, \text{“Downlink DoS”}\}}$
Downlink IMSI Extractor	[17]	MiTM	C	NAS	IdentityRequest	●	(2)	$\frac{(\exists ue(u) \in S) \wedge (u_{msg_i} = Identity\_Resp) \wedge (u_{msg_{i-1}} \neq Identity\_Req)}{S' = S \cup \{event \mid u_{bs}, u_{rnti}, t, \text{“Downlink IMSI Extractor”}\}}$
Uplink DoS	[19]	MiTM	A	NAS	AttachRequest / RegistrationRequest	◐	(3)	$\frac{(\exists ue(u) \in S) \wedge (u_{msg_i} = Attach\_Req) \wedge (Blocked(u_{imsi}))}{S' = S \cup \{event \mid u_{bs}, u_{rnti}, t, \text{“Uplink DoS”}\}}$
Uplink IMSI Extractor	[19]	MiTM	C	NAS	AttachRequest / RegistrationRequest	◐	(3)	$\frac{(\exists ue(u) \in S) \wedge (u_{msg_i} = Attach\_Req) \wedge (Unknown(u_{tmsi}))}{S' = S \cup \{event \mid u_{bs}, u_{rnti}, t, \text{“Uplink IMSI Extractor”}\}}$
Null Cipher / Integrity	[10]	MiTM	C	RRC	SecurityModeFailure	◐	(3)	$\frac{(\exists ue(u) \in S) \wedge (u_{sec\_state} = 1) \wedge ((u_{cipher\_alg} = 0) \vee (u_{integrity\_alg} = 0))}{S' = S \cup \{event \mid u_{bs}, u_{rnti}, t, \text{“Null Cipher / Integrity”}\}}$

TABLE II: Targeted attacks and detection rules (A: Availability attack, C: Confidentiality attack, ●: Attack Report, ◐: Warning).

In the following, we walk through two concrete examples to demonstrate how P-BEST and MOBIFLOW can be used to create IDS rules. We further provide the concertized rules in P-BEST language in Appendix (Listing 1 and Listing 2). For the remaining attacks, we show the high-level representation of inference rules in Table II.

Example 1: Detecting BTS Resource Depletion Attacks. This attack was discovered in LTE networks [9] but still applies to 5G networks. In this attack, a malicious UE exploits the unprotected RRC `ConnectionRequest` message [20], which allows it to create massive fabricated RRC connections with random C-RNTIs to perform DoS attacks on the target BS. Specifically, the attacker UE restarts a new RRC session upon receiving a NAS `AuthenticationRequest`. With commercial SDR hardware and compatible cellular software stack [6], [8], this attack can be easily launched in practice, as demonstrated later in §VII-C.

Based on the description, we can intuitively come up with a threshold-based detection approach. This is based on the observation that each fabricated RRC connection is released after the T3460 timer in the MME expires while waiting for the NAS `AuthenticationResponse` [21] from the UE [9]. Therefore, such a malicious connection typically lasts for only a few seconds and thereby creates a distinguishing feature to identify it among other normal UE connections. We define the UE created by such a fake connection as a *transient UE* (denoted by t_ue), and maintain counters for each BS (denoted by cnt) to track the accumulated transient UEs in history. These two data types are defined as `ptypes` in P-BEST. Based on the definitions, we create a small set of rules for attack inference. Initially, when a BS first detects a transient UE, the following rule is triggered:

$$\frac{(\exists ue(u) \in S) \wedge (u_{t_2} - u_{t_1} < T_t) \wedge (\nexists cnt(c) \in S) \wedge (c_{bs} = u_{bs})}{S' = S \cup \{t\_ue \mid u_{rnti}, u_{bs}, t\} \cup \{cnt \mid u_{bs}, 1\}}$$

This rule detects a transient UE based on the timers in MOBIFLOW records (Table I), by subtracting the inactive RRC timer (u_{t_2}) with the initial RRC timer (u_{t_1}). If the connection expires within a user-defined threshold T_t, a transient UE counter is created as 1. Similarly, if a BS has transient UE(s) in the past, another rule is triggered to increase the transient UE counter by 1. Next, when the transient UE counter of the BS exceeds a threshold T_{ue}, an event will be reported:

$$\frac{(\exists cntr(c) \in S) \wedge (c_{value} > T_{ue})}{S' = S \cup \{event \mid c_{bs}, 0, t, \text{“BTS Resource Depletion”}\}}$$

Finally, we still need to program a GC rule to recycle transient UE instances. Otherwise, the accumulated transient UE instances will remain in memory, which causes to program to repeatedly generate undesired alarms. To this end, the GC rule releases a transient UE when its timer expires. If so, the following GC rule is triggered to remove the transient UE and decrease the transient UE counter.

$$\frac{(\exists t\_ue(u) \in S) \wedge (t - u_t > T_{release}) \wedge (\exists cnt(c) \in S) \wedge (c_{bs} = u_{bs})}{S' = S - \{u\} - \{c\} \cup \{cnt \mid c_{bs}, c_{value} - 1\}}$$

Example 2: Detecting Blind DoS Attacks. In contrast to BTS resource depletion, the blind DoS attack pinpoints a UE by establishing an RRC connection using its S-TMSI in LTE or 5G networks [9], [10]. The S-TMSI can be harvested via silent paging attacks [48] or packet sniffing. Based on 3GPP TS 38.331 [20], the BS will delete the victim's RRC security context and release the connection, thus triggering a DoS scenario. Therefore, the intuitive way is to detect blind DoS attacks based on S-TMSI replay with the below rule:

$$\frac{(\exists ue(u_1) \in S) \wedge (\exists ue(u_2) \in S) \wedge (u1_{bs} = u2_{bs}) \wedge (u1_{rrc\_state} = u2_{rrc\_state} = 2) \wedge (u1_{tmsi} = u2_{tmsi}) \wedge (u1_t < u2_t)}{S' = S \cup \{event \mid u2_{bs}, u2_{rnti}, t, \text{“Blind DoS”}\}}$$

This rule checks if a malicious UE (u_2) initiates an RRC connection by replaying the S-TMSI of another actively connected UE (u_1) in the same network. The rule ensures that the S-TMSI of the two UEs are equal and distinguishes the attacker and the victim based on the RRC timers (i.e., the attack establishes the RRC connection after the victim).

Level of Alerts. The above rule sets need to effectively detect malicious attacks while not producing too many false alarms. However, distinguishing attacks from benign traffic is a challenging task in cellular networks (e.g., FBS detection [24], [42], [25], [23] and intrusion detection [22]) and not unique to our system. For this reason, we set up different alert levels for each attack as shown in Table II. For instance, employing null cipher or integrity protection could be caused by a MiTM attacker who injects a failure control message to the traffic, but is still considered valid in the specifications [20], [21]. As a result, for events that MOBIEXPERT cannot deterministically draw conclusions, they are reported as warnings instead of being flagged as malicious attacks.

VII. Evaluation

To evaluate our prototype, we have created an O-RAN testbed for 5G-SPECTOR. Specifically, we use a host machine (Ubuntu 18.04 OS) equipped with 12 Intel i7-8700 cores and 32GB RAM, which runs three virtual machines (VMs) for the core network, RIC, and RAN (i.e., CU and DU), respectively. The RU front-end is either the OAI nFAPI emulator [8] or a physical USRP B210 SDR [6] attached to the RAN machine via USB 3.0. In this evaluation section, we aim to answer the following five research questions:

- **RQ1:** Can 5G-SPECTOR detect known and unknown cellular L3 attacks?
- **RQ2:** How well does 5G-SPECTOR scale to real-world cellular networks?
- **RQ3:** Can 5G-SPECTOR be deployed effectively to detect L3 cellular attacks in practice?
- **RQ4:** What is 5G-SPECTOR's system performance?
- **RQ5:** What is the overhead that 5G-SPECTOR introduces to the original network?

To answer RQ1 (§VII-A), we evaluate 5G-SPECTOR against two sets of simulated attacks, including 7 known attacks in Table II and 11 unknown attacks derived as variants from them. To answer RQ2 (§VII-B), we evaluate 5G-SPECTOR against both benign and abnormal cellular network traffic from public datasets [31], [22]. To answer RQ3 (§VII-C), we evaluate 5G-SPECTOR with two L3 attacks fully replicated over-the-air using COTS SDRs and smartphones. To answer RQ4 (§VII-D), we analyze its system performance, such as throughput and detection latency. To answer RQ5 (§VII-E), we measure the overhead imposed on the O-RAN data plane and control plane, including CPU and memory consumption.

A. Evaluation with Simulated Attacks and Variants

This section answers the following question: does 5G-SPECTOR have the right feature set and programming capability to detect known and unknown L3 attacks? Regarding this issue, we emulated two sets of cellular L3 attacks. The first set involves the 7 exploits (described in Table II) from existing literature [10], [9], [17], [19], constituting 7 *known* attacks. The second set involves 11 *unknown* exploits that are derived as variants from the known attack set. The attack simulation is implemented based on the OAI nFAPI emulator [8] that allows us to emulate the physical layer communication between the UE and the RU. In Table III, we summarize these attacks and variants, and we further describe how we simulate these attacks and derive variants below.

Attack	Layer	Exploited L3 Message	New	Detected
BTS RC Depletion	RRC	ConnectionRequest *(Fabricated)*	○	✓
Blind DoS	RRC	ConnectionRequest *(Replayed TMSI)*	○	✓
Downlink DoS	NAS	AuthRequest ← AttachReject	○	✓
	NAS	SecModeCmd ← AttachReject	●	✓
	NAS	AttachAccept ← AttachReject	●	✓
	NAS	AuthRequest ← ServiceReject	●	✓
	NAS	SecModeCmd ← ServiceReject	●	✓
	NAS	AttachAccept ← ServiceReject	●	✓
Uplink DoS	NAS	AttachReq ← AttachReq *(Invalid IMSI)*	○	✓
	NAS	ServiceReq ← ServiceReq *(Invalid MAC)*	●	✓
Uplink IMSI Extractor	NAS	AttachReq ← AttachReq *(Unknown TMSI)*	○	✓
Downlink IMSI Extractor	NAS	AuthRequest ← IdentityRequest *(IMSI)*	○	✓
	NAS	AuthRequest ← IdentityRequest *(IMEI)*	●	✓
	NAS	AuthRequest ← IdentityRequest *(TMSI)*	●	✓
	NAS	SecModeCmd ← IdentityRequest *(IMSI)*	●	✓
	NAS	AttachAccept ← IdentityRequest *(IMSI)*	●	✓
Null Cipher & Integrity	RRC	SecModeComplete ← SecModeFailure	○	✓
	NAS	SecModeComplete ← SecModeReject	●	✓

TABLE III: All L3 cellular attacks and variants replicated and evaluated ($A \leftarrow B$ indicates message B overwrites A).

To simulate the 7 known exploits, we follow their descriptions from the original paper [10], [9], [17], [19]. More specifically, we emulate them by inserting malicious code logic into the RRC and NAS tasks in the OAI implementation [8]. For example, the BTS resource depletion attack [9] is implemented with only 20 LoC in C based on OAI's RRC stack. This implementation will launch a rogue UE that repeatedly creates RRC connections in our simulated network to ultimately DoS other legitimate UEs. As mentioned in our implementation section (Table II), we have implemented corresponding detection signature sets of these 7 attacks. Further, we created 11 variants from these 7 known attacks, and these variants are considered unknown to 5G-SPECTOR. To achieve this goal, we use two mutation strategies. Given the original attack description and the attack session, we either (1) replace the exploited message with an equivalent one that triggers the same consequence (e.g., `AttachReject` is replaced by `ServiceReject` to achieve the same DoS effect) or (2) move the exploited message to other RRC or NAS phases of the session (e.g., `IdentityRequest` injection [19] can occur during authentication or security mode procedures to ask the victim to report its IMSI in plain text).

Using these strategies, we successfully created 18 attacks in total (including the 7 original attacks). To extend 5G-SPECTOR for the 11 unknown attacks, we further developed additional 90 LoC in P-BEST, based on the original 758 LoC that have been programmed for the known attacks. This indicates that 5G-SPECTOR is extensible to unknown attacks with its existing feature sets and programming capability. Next, we evaluate our new prototype against all 18 attacks, and each attack instance was tested 20 times in total. As summarized in Table III, 5G-SPECTOR is capable of effectively detecting

Name	Ref	UE	Time(s)	#Pkt.	#MF	#Sess.	B	Event
BT-1	[31]	LG LS660	10,597	4,164	1,810	113	✓	0
BT-2	[31]	LG G3 VS985	514	3,803	173	15	✓	0
BT-3	[31]	LG G3 VS985	489	3,766	158	15	✓	0
BT-4	[31]	Galaxy S5	764	2,996	154	13	✓	0
BT-5	[31]	LG G3 VS985	16,324	26,548	1,217	114	✓	0
BT-6	[31]	Galaxy S5	1,459	2,803	97	13	✓	0
BT-7	[31]	Galaxy S5	2,053	4,794	448	27	✓	0
BT-8	[31]	Galaxy S5	6,387	2,839	1,435	113	✓	0
BT-9	[31]	Galaxy S5	1,473	3,755	190	13	✓	0
BT-10	[31]	Galaxy S5	340	1,943	72	9	✓	0
BT-11	[31]	Nexus 6P	1	1,174	2	1	✓	0
BT-12	[31]	LG LS660	2,561	839	310	25	✓	1
BT-13	[31]	Nexus 6	7	322	20	1	✓	0
BT-14	[31]	LG LS660	44,672	13,849	6,649	437	✓	1
BT-15	[31]	Nexus 6P	71	2,553	43	1	✓	0
BT-16	[31]	Nexus 6	9,578	10,315	6,971	163	✓	1
BT-17	[31]	Nexus 6P	1	5	2	1	✓	0
BT-18	[31]	LG LS660	37,153	16,334	5,770	336	✓	1
BT-19	[31]	N/A	127	145	56	2	✓	0
BT-20	[31]	Nexus 6P	569	2,311	94	3	✓	0
BT-21	[31]	Nexus 6	340	209	83	6	✓	0
BT-22	[31]	Galaxy S5	3,216	5,554	506	29	✓	0
AT-1	[22]	N/A	1	632	61	11	✗	0
AT-2	[22]	N/A	1	482	53	8	✗	0
AT-3	[22]	N/A	1	626	59	6	✗	0
AT-4	[22]	N/A	1	715	82	13	✗	1
AT-5	[22]	N/A	1	181	18	3	✗	0
AT-6	[22]	N/A	1	811	108	10	✗	1
AT-7	[22]	N/A	5	1,468	217	28	✗	1
AT-8	[22]	N/A	1	630	85	11	✗	1
AT-9	[22]	N/A	26	359	27	4	✗	0

TABLE IV: Evaluation results using real-world cellular traffic (BT: Benign Trace, AT: Abnormal Trace, B: Benign, #MF: MOBIFLOW counts). Red numbers indicate warning events.

all 18 attacks by generating accurate and reproducible alerts in real-time (indicating a zero false negative rate among these samples). Our results show that 5G-SPECTOR can be programmed and extended with effective IDS rule sets that can detect existing and potential (unknown) L3 cellular attacks.

B. Evaluation with Real-World Datasets

To demonstrate 5G-SPECTOR's scalability to practical cellular networks (i.e., not producing too many false alarms), we evaluated it with datasets with traces collected from real-world UEs [31], [22]. As presented in Table IV, there are 22 benign traces (BT-1 to BT-22) collected from different COTS UEs and network operators, and 9 abnormal UE traces (AT-1 to AT-9 with network failure scenarios) created by academic researchers [22]. Each trace spans varied time (from several seconds to a few hours) and involves many UE sessions with the network. Therefore, we first filter out non-L3 packets (e.g., cell measurement packets) and convert the total $118,145$ raw packets (e.g., `pcap` format) into $26,790$ MOBIFLOW records, based on MOBIFLOW's definition in Table I. Note that the traffic was collected from LTE networks, but it is still applicable to 5G networks due to the consistency of the RRC and NAS protocols [21], [20].

We replayed the generated MOBIFLOW records to 5G-SPECTOR, and the results are reported in Table IV, including the number of events reported for each trace. As shown, MOBIEXPERT produced 8 *warning* events based on the alert level described in §VI, and no attack events were reported. We further investigated these cases and found that our detection rules functioned correctly, and the root cause is that the UEs in these traces employ insecure null cipher or integrity algorithms in practice. For instance, in BT-14, the UE uses EEA0 and EIA1 algorithms [21] for encryption and integrity protection, respectively, which were possibly limited by the UE or network's capability. Based on the rules and alert level described in §VI, MOBIEXPERT does not conclude them as attacks and report them as *insecure practice warnings*, as such activities do not always constitute attacks but indeed bring user's security and privacy at risk. A detailed discussion on distinguishing attacks and insecure practices is presented in §VIII. Overall, our implementation of 5G-SPECTOR produces alerts at a relatively low frequency (i.e., one warning per 374 UE session on average), and does not generate false alarms among the traces we validated.

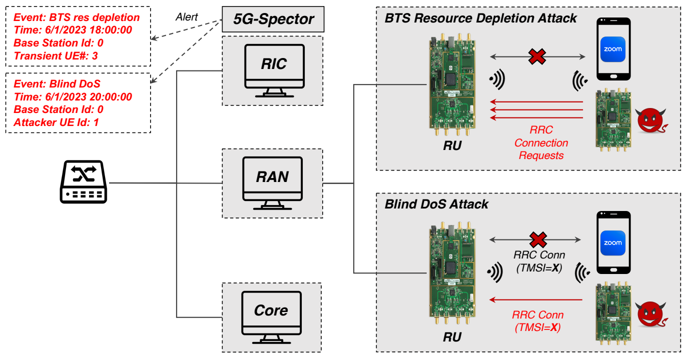


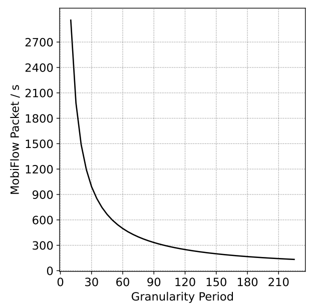
Fig. 7: Two Over-the-air L3 attacks reproduction [9] on our testbed with 5G-SPECTOR's alerts generated.

C. Evaluation with Over-the-Air Attacks

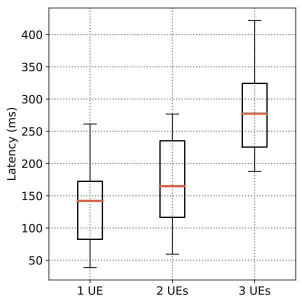
To demonstrate 5G-SPECTOR can be deployed to address L3 attacks in practice, we evaluate it against two over-the-air (OTA) attacks reproduced using our testbed, namely the BTS resource depletion attack and the blind DoS attack [9][3]. Note that OTA reproduction requires a much more sophisticated setup than simulation as we have to also properly set up the physical layer signals in addition to the attack simulation. Figure 7 illustrates our OTA attack setup. As shown, a USRP B210 [6] is the RF front-end and attaches to the RAN VM via USB 3.0. A COTS Pixel 5 and another USRP B210 serve as the victim UE and the attacker respectively. To clearly demonstrate the attack effect (e.g., UE's connection is dropped), we let the Pixel 5 open an active Zoom session as it connects to the data network through the EPC.

The OTA reproduction is performed similarly to our emulation, by inserting malicious logic to the OAI UE code running on the attacker USRP. Additionally, we manipulate the UE's physical layer code when necessary. For instance, the BTS resource depletion attack [9] requires resetting the physical layer parameters when an RRC connection is restarted. At layer 3, we control the malicious UE to restart after `AuthRequest` and continuously create fabricated RRC connections, in order to reproduce the BTS resource depletion attack. For the blind DoS attack, we first connect the victim UE and harvest its

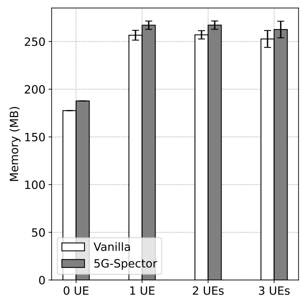
[3]Video demo is available at https://www.5gsec.com/post/5g-spector-demo



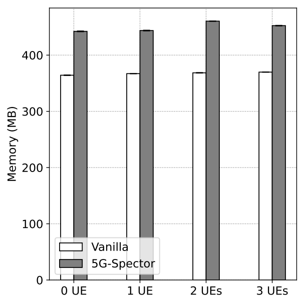
(a) Throughput.



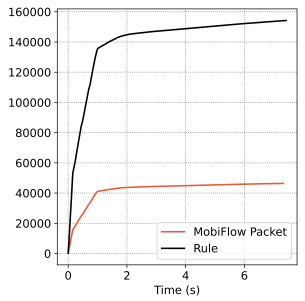
(b) Detection Latency.



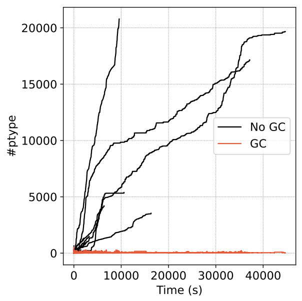
(a) RAN Memory overhead.



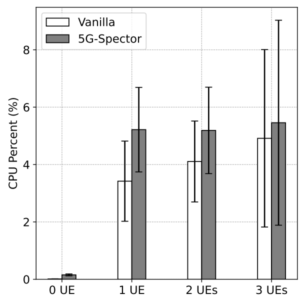
(b) RIC Memory overhead.



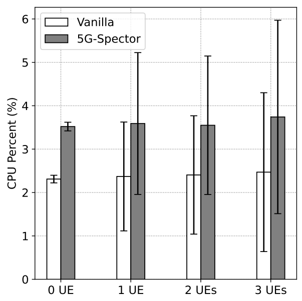
(c) Processing speed.



(d) GC performance.



(c) RAN CPU overhead.



(d) RIC CPU overhead.

Fig. 8: Evaluation of system performance.

Fig. 9: Evaluation of system overhead.

TMSI from the EPC's log. Afterward, we pass the TMSI to the attacker (to simulate an assumption that the attacker knows the TMSI), and command the attacker to initiate an RRC connection with the TMSI. By launching these two attacks, we found the victim's Zoom session froze as its connection was released by the BS because of the exhaustion of BTS resources or the duplicated UE sessions. In the meantime, 5G-SPECTOR correctly generated alerts with the attack details as shown in Figure 7. Through repeated experiments, we confirm that the detection results of 5G-SPECTOR are fully reproducible and do not contain false negatives. To conclude, our evaluation demonstrates 5G-SPECTOR's capability to detect two OTA L3 attacks in a real cellular network.

D. Evaluation of Performance

Throughput. We measure 5G-SPECTOR's throughput which indicates how many MOBIFLOW packets can be generated per time unit. Ideally, assuming that the RIC operates at max network bandwidth, the throughput depends on various parameters and is subject to:

$$T \propto \frac{T_{ric} \cdot v}{gp \cdot size_{mf}}$$

where the throughput is proportional to the RIC machine's throughput (T_{ric}) and its speed for generating MOBIFLOW packets (v). It is inversely proportional to the granularity period (gp) and the size of each MOBIFLOW packet ($size_{mf}$). For our implementation, each MOBIFLOW packet is 406 bytes and the bandwidth between the RAN and RIC VM is 1.76Gbps. In Figure 8a, we show the maximum throughput with respect to the granularity period on our RIC machine.

Detection Latency. The detection latency measures the delay for a UE control packet at the RAN to be converted into MOBIFLOW at the MOBIEXPERT xApp. This is proportional to the granularity period and the number of UEs attached, and inversely proportional to several parameters, including the speed of the network, RF, and data processing (e.g., MOBIFLOW generation in the SECSM agents). Therefore, it is subject to:

$$L \propto \frac{gp \cdot n_{ue}}{v_{network} \cdot v_{process} \cdot v_{rf}}$$

We measure the latency with respect to the number of UEs attached to the network simultaneously, with a granularity period set to be 200ms. Note that we use the emulation setup to generate multiple UE connections, and thus RF latency between UE and RAN is not considered. Due to the limitation of the nFAPI emulator [8], we were only able to emulate at most three UEs. As plotted in Figure 8b, the average latency is 140ms, 160ms, and 280ms under the three settings, respectively. It can also be inferred that the number of UEs indeed has a negative impact on the latency.

Efficiency. We evaluate how many MOBIFLOW packets can be simultaneously processed by the MOBIEXPERT xApp. As shown in Figure 8c, we perform a stressful test by inserting nearly 50K MOBIFLOW records, and find that MOBIEXPERT efficiently processed over 40K (80%) records and executed nearly 140K rules within just one second. However, its speed decreases significantly afterward. This is due to the massive `ptype` creations that constantly call `malloc()`, and GC was not able to trigger within this short period of time.

GC Performance. We measure the performance of the GC mechanism by testing each cellular trace with and without GC in §VII-B and plotting the number of `ptypes` in real-time. As shown in Figure 8d, it is obvious that GC significantly reduces memory consumption by releasing unused `ptypes` at run-time. By deploying GC rules, the number of `ptypes` maintains below a few hundred among all traces tested at all time. In contrast, the number of `ptypes` without GC is significantly higher (200X in the worst case).

E. Evaluation of Overhead

We evaluate 5G-SPECTOR's impact on both the control plane and the data plane. To this end, we measure its CPU and memory overhead on the RIC and RAN machines with respect to the attached UE numbers by using the Python `psutil` APIs, and compare the results with the vanilla RAN and RIC implementation without 5G-SPECTOR. We test it with up to three UEs due to the constraints of the nFAPI emulator [8]. By interpreting the results in Figure 9, our implementation introduces an average 10MB and 80MB memory overhead to the vanilla RAN and RIC, respectively, which is 4% and 20% of the total memory consumption. The memory overhead on the RIC is higher due to the MOBIEXPERT xApp. The CPU overhead varies on the two planes. For the data plane, it decreases as the RAN runs more UEs (from 1.8% to 0.5%), since the cost of the vanilla CPU increases faster. For the control plane, the CPU overhead is around 1.4% on average across the UE numbers. The error bars represent the fluctuation due to the UE attachment procedure with an apparent impact on the CPUs.

VIII. LIMITATION AND FUTURE WORK

False Positives. 5G-SPECTOR's detection rules are based on abnormal states and transmissions of unprotected L3 messages. In practice, such anomalies could occur due to external noise (e.g., bit error) and thus introduce false positives. Additionally, some signatures indicate abnormal activities that do not always correspond to attacks. For example, it is compliant for two UEs to use the same TMSI to attach to the network, as the `RRCConnectionRequest` can be sent from an attacker or a legitimate user. This could represent either the blind DoS attack [9] or an abnormal network condition. However, in our evaluation with repeated attack simulation and a large number of real-world cellular traces, 5G-SPECTOR does not produce false positives, indicating that these scenarios are indeed rare in practice [22]. As a detection service focusing on L3, it is out of 5G-SPECTOR's scope to distinguish attacks from such scenarios. We acknowledge this as a limitation for RAN-based IDS (not unique to 5G-SPECTOR) that should be addressed by future cellular protocol standards.

Scalability to Operational Networks. Due to the limitation of our experimental testbed, we were unable to evaluate the scalability of our system to operational networks that may connect to hundreds of thousands of devices. This is caused by the limitation of both hardware (e.g., experimental SDRs and inadequate COTS UEs) and software (e.g., the OAI [8] and the SD-RAN RiaB platform [30]) that are accessible to researchers. However, as the components we design and develop (i.e., MOBIFLOW, SECSM, and MOBIEXPERT) comply with the O-RAN specifications [49], they can potentially scale to real O-RAN compliant networks.

Future Work. First, MOBIFLOW stream is potentially applicable to machine learning models (e.g., deep neural networks). Therefore, ML-based solutions could be developed at the O-RAN control plane for many other security applications, such as anomaly classification based on traffic patterns [50]. However, it still needs to address challenges such as acquiring adequate adversary samples, which could be collected via large-scale cellular testbed and thus is considered the second future direction. Third, the O-RAN control plane introduces new security risks [2], [51] (e.g., malicious xApps from the supply chain), and thus robust access control policies must be properly enforced. Lastly, 5G-SPECTOR's run-time alerts can operate synergistically with triangulation service and other closed-loop security countermeasures against emerging threats in cellular networks.

IX. RELATED WORK

Run-time Cellular Network Monitoring. To detect cellular network threats in real-time, one cost-effective solution is to deploy run-time monitors [52], [31], [53]. To this end, many Android-based apps were developed for monitoring IMSI catchers [23]. In particular, PHOENIX detects cellular control-plane attacks at UEs using an automata-based approach [22]. These UE-based monitors rely on reading low-level cellular information via baseband APIs provided by the vendors. CELLDAM provides a root-less defense by using a companion data-plane signal analyzer [54]. On the other hand, such monitors can also be deployed at the networks (RAN), such as heuristics-based fake base station detectors [24], [55], [56], [25], [26], [57], [58], and anomalous traffic classifiers [59], [60], [61]. While our system can be viewed as a run-time monitor, it is distinguished from the above works for being the first defense at the 5G O-RAN control plane that offers programmability and extensibility.

Cellular Network Protocol Hardening. A fundamental solution to eliminate the vulnerabilities is to directly harden the cellular network protocols [62]. To this end, one particular focus is to introduce authentication mechanisms to the messages during the connection bootstrapping phase [38], [63]. In addition, there are other proposals to harden certain cellular network procedures such as authentication [64], [65]. These protocol-level defenses still need to address many practical constraints (e.g., performance overhead) before deployment.

O-RAN Security. There have been surveys and security analyses dedicated to O-RAN and its attack surfaces [2], [66], [67]. On the specification level, the O-RAN Alliance has published specifications describing the potential threats and mitigation of the O-RAN control plane, such as malicious xApps, rApps, and ML models [68], [69]. On the implementation level, Atalay et. al. propose a scalable authentication framework to secure the O-RAN control plane against untrusted xApps [51].

Orthogonal to these works, we leverage O-RAN to secure external cellular threats.

X. Conclusion

We present 5G-Spector, an O-RAN compliant L3 attack detection service, which is the first defense deployed at the O-RAN control plane. It features a fine-grained telemetry stream MobiFlow and an O-RAN compliant security service module SecSM as the foundations to facilitate various security applications. We then demonstrate MobieXpert, a P-BEST-language-based xApp on the control plane to enable cellular network operators to program IDS signatures for a wide range of external cellular exploits. We further present a prototype of 5G-Spector and test it with 7 types of L3 attacks in the literature and 31 cellular network traces in practice, and show that it is not only able to effectively detect both existing and unknown attacks but can also scale to real-world scenarios. We also present a comprehensive evaluation of our system and show that it can operate at a high speed with low latency while introducing low overhead to the original RAN.

Acknowledgment

We would like to thank the anonymous reviewers for their constructive feedback. We also thank Keith Skinner, Martin Fong, and Kirk Bradley from SRI International for helping integrate and test 5G-Spector. This research was supported in part by NSF awards CNS-2112471, ITE-2226443 and ITE-2326882. Any opinions, findings, conclusions, or recommendations expressed are those of the authors and not necessarily of the NSF.

References

[1] "O-ran alliance," https://www.o-ran.org/.

[2] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding o-ran: Architecture, interfaces, algorithms, security, and research challenges," *arXiv preprint arXiv:2202.01032*, 2022.

[3] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, "Software-defined networking (sdn): a survey," *Security and communication networks*, vol. 9, no. 18, pp. 5803–5833, 2016.

[4] "O-ran architecture description: O-ran.wg1.o-ran-architecture-description-v07.00," October 2022.

[5] "O-ran.wg3.e2sm-v02.01: Near-real-time ran intelligent controller e2 service model (e2sm)," Martch 2022.

[6] "Usrp software defined radio (sdr)," https://www.ettus.com/products/.

[7] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "srslte: An open-source platform for lte evolution and experimentation," in *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*, 2016, pp. 25–32.

[8] "oai / openairinterface5g," https://gitlab.eurecom.fr/oai/openairinterface5g.

[9] 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, pp. 1153–1168.

[10] 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*, 2019, pp. 669–684.

[11] E. Bitsikas and C. Pöpper, "Don't hand it over: Vulnerabilities in the handover procedure of cellular telecommunications," in *Annual Computer Security Applications Conference*, 2021, pp. 900–915.

[12] 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.

[13] G. Lee, J. Lee, J. Lee, Y. Im, M. Hollingsworth, E. Wustrow, D. Grunwald, and S. Ha, "This is your president speaking: Spoofing alerts in 4g lte networks," in *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, 2019, pp. 404–416.

[14] 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.

[15] ——, "Imp4gt: Impersonation attacks in 4g networks," in *27th Annual Network and Distributed System Security Symposium, NDSS*, 2020.

[16] 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 (USENIX Security 19)*, 2019, pp. 55–72.

[17] M. Kotuliak, S. Erni, P. Leu, M. Roeschlin, and S. Čapkun, "{LTrack}: Stealthy tracking of mobile phones in {LTE}," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1291–1306.

[18] 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*, 2021, pp. 250–260.

[19] S. Erni, M. Kotuliak, P. Leu, M. Röschlin, 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.

[20] 3GPP, "Radio resource control (rrc)," http://www.3gpp.org/DynaReport/38331.htm.

[21] ——, "Non-access-stratum (nas) protocol for evolved packet system (eps)," http://www.3gpp.org/DynaReport/24301.htm.

[22] 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."

[23] R. Borgaonkar, A. Martin, S. Park, A. Shaik, and J.-P. Seifert, "White-stingray: evaluating imsi catchers detection applications." USENIX, 2017.

[24] 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.

[25] P. Ney, I. Smith, G. Cadamuro, and T. Kohno, "Seaglass: Enabling city-wide imsi-catcher detection." *Proc. Priv. Enhancing Technol.*, vol. 2017, no. 3, p. 39, 2017.

[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] H. Wen, P. Porras, V. Yegneswaran, and Z. Lin, "A fine-grained telemetry stream for security services in 5g open radio access networks," in *Proceedings of the 1st International Workshop on Emerging Topics in Wireless*, 2022, pp. 18–23.

[28] "Netflow - cisco," https://www.cisco.com/c/en/us/tech/quality-of-service-qos/netflow/index.html.

[29] U. Lindqvist and P. A. Porras, "Detecting computer and network misuse through the production-based expert system toolset (p-best)," in *Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No. 99CB36344)*. IEEE, 1999, pp. 146–161.

[30] "Sdran," https://docs.sd-ran.org/master/introduction.html.

[31] Y. Li, C. Peng, Z. Yuan, J. Li, H. Deng, and T. Wang, "Mobileinsight: Extracting and analyzing cellular network information on smartphones," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, 2016, pp. 202–215.

[32] 3GPP, "5g system overview," https://www.3gpp.org/technologies/5g-system-overview.

[33] ——, "Numbering, addressing and identification," https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=729.

[34] ——, "Ng-ran architecture description," http://www.3gpp.org/DynaReport/38401.htm.

[35] ——, "Ng-ran f1 application protocol (f1ap)," http://www.3gpp.org/DynaReport/38473.htm.

[36] "O-ran.wg3.e2ap-v02.03: O-ran e2 application protocol (e2ap)," October 2022.

[37] R. P. Jover, "Lte security, protocol exploits and location tracking experimentation with low-cost software radio," *arXiv preprint arXiv:1607.05171*, 2016.

[38] S. R. Hussain, M. Echeverria, A. Singla, O. Chowdhury, and E. Bertino, "Insecure connection bootstrapping in cellular networks: the root of all evil," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, 2019, pp. 1–11.

[39] Y. Chen, Y. Yao, X. Wang, D. Xu, C. Yue, X. Liu, K. Chen, H. Tang, and B. Liu, "Bookworm game: Automatic discovery of lte vulnerabilities through documentation analysis," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1197–1214.

[40] 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.

[41] C. Park, S. Bae, B. Oh, J. Lee, E. Lee, I. Yun, and Y. Kim, "Doltest: In-depth downlink negative testing framework for lte devices," in *USENIX Security Symposium*, 2022.

[42] 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, pp. 246–255.

[43] Y. Zhu, N. Kang, J. Cao, A. Greenberg, G. Lu, R. Mahajan, D. Maltz, L. Yuan, M. Zhang, B. Y. Zhao *et al.*, "Packet-level telemetry in large datacenter networks," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 479–491.

[44] U. Lindqvist and P. Porras, "Expert-bsm: A host-based intrusion detection solution for sun solaris," in *Proceedings of the 17th Annual Computer Security Applications Conference*. IEEE Computer Society, 2001.

[45] R. A. Whitehurst, M. M. Sebring, E. Shellhouse, and M. E. Hanna, "Expert systems in intrusion detection: a case study," in *Proceeding of the 11th National Computer Security Conference*, 1988.

[46] T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, C. Jalali, P. Neumann, H. Javitz, and T. Garvey, "A real-time intrusion-detection expert system," 01 1992.

[47] S. Cheung, U. Lindqvist, and M. Fong, "Modeling multistep cyber attacks for scenario recognition," in *Proceedings of the Third DARPA Information Survivability Conference and Exposition (DISCEX III)*. DARPA, 2003, pp. 284–292.

[48] A. Shaik, R. Borgaonkar, N. Asokan, V. Niemi, and J.-P. Seifert, "Practical attacks against privacy and availability in 4g/lte mobile communication systems," 2016.

[49] "O-ran specification," https://www.o-ran.org/specifications.

[50] A. Afaq, N. Haider, M. Z. Baig, K. S. Khan, M. Imran, and I. Razzak, "Machine learning for 5g security: Architecture, recent advances, and challenges," *Ad Hoc Networks*, vol. 123, p. 102667, 2021.

[51] T. O. Atalay, S. Maitra, D. Stojadinovic, A. Stavrou, and H. Wang, "Securing 5g openran with a scalable authorization framework for xapps," *arXiv preprint arXiv:2212.11465*, 2022.

[52] B. Hong, S. Park, H. Kim, D. Kim, H. Hong, H. Choi, J.-P. Seifert, S.-J. Lee, and Y. Kim, "Peeking over the cellular walled gardens-a method for closed network diagnosis," *IEEE Transactions on Mobile Computing*, vol. 17, no. 10, pp. 2366–2380, 2018.

[53] H. Wen, P. Porras, V. Yegneswaran, and Z. Lin, "Thwarting smartphone sms attacks at the radio interface layer," in *30th Annual Network and Distributed System Security Symposium, NDSS*, 2023.

[54] Z. Tan, J. Zhao, B. Ding, and S. Lu, "{CellDAM}:{User-Space}, rootless detection and mitigation for 5g data plane," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 2023, pp. 1601–1619.

[55] C. Quintin, "Detecting fake 4g {LTE} base stations in real time," 2021.

[56] A. Dabrowski, G. Petzl, and E. R. Weippl, "The messenger shoots back: Network operator based imsi catcher detection," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2016, pp. 279–302.

[57] Z. Zhuang, X. Ji, T. Zhang, J. Zhang, W. Xu, Z. Li, and Y. Liu, "Fbsleuth: Fake base station forensics via radio frequency fingerprinting," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018, pp. 261–272.

[58] P. Ziayi, S. M. Farmanbar, and M. Rezvani, "Yaicd: Yet another imsi catcher detector in gsm," *Security and Communication Networks*, vol. 2021, 2021.

[59] I. A. Karatepe and E. Zeydan, "Anomaly detection in cellular network data using big data analytics," in *European Wireless 2014; 20th European Wireless Conference*. VDE, 2014, pp. 1–5.

[60] P. Casas, P. Fiadino, and A. D'Alconzo, "Machine-learning based approaches for anomaly detection and classification in cellular networks." in *TMA*, 2016.

[61] S. A. Al Mamun and J. Valimaki, "Anomaly detection and classification in cellular networks using automatic labeling technique for applying supervised learning," *Procedia Computer Science*, vol. 140, pp. 186–195, 2018.

[62] 3GPP, "Study on 5g security enhancements against false base stations (fbs)," http://www.3gpp.org/DynaReport/33809.htm.

[63] A. Singla, R. Behnia, S. R. Hussain, A. Yavuz, and E. Bertino, "Look before you leap: Secure connection bootstrapping for 5g networks to defend against fake base-stations," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, 2021, pp. 501–515.

[64] Y. Wang, Z. Zhang, and Y. Xie, "Privacy-preserving and standard-compatible {AKA} protocol for 5g," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021.

[65] 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.

[66] D. Mimran, R. Bitton, Y. Kfir, E. Klevansky, O. Brodt, H. Lehmann, Y. Elovici, and A. Shabtai, "Evaluating the security of open radio access networks," *arXiv preprint arXiv:2201.06080*, 2022.

[67] F. Klement, S. Katzenbeisser, V. Ulitzsch, J. Krämer, S. Stanczak, Z. Utkovski, I. Bjelakovic, and G. Wunder, "Open or not open: Are conventional radio access networks more secure and trustworthy than open-ran?" *arXiv preprint arXiv:2204.12227*, 2022.

[68] "O-ran.sfg.security-near-rt-ric-xapps-v01.00: O-ran study on security for near real time ric and xapps," July 2022.

[69] "O-ran.sfg.non-rt-ric-security-tr-v01.00: O-ran study on security for non-rt-ric," July 2022.

[70] 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.

Appendix A

A. *L3 Protocol Messages*

This section provides details about the L3 protocol (RRC and NAS) messages that are not discussed in §II. Based on the 3GPP TS 38.331 [20] and 24.301 [21], we present a comprehensive list of unprotected RRC and NAS message in Table V that could (potentially) be exploited for attacks. The root cause is that the encryption and integrity protection of these messages are not available or not mandatory as described in the table. For instance, an `ConnectionRequet` in the RRC protocol can be easily exploited by attackers since it is neither encrypted nor integrity-protected, and thus the network cannot verify its validity. We have replicated these L3 exploits by using the OAI stack to create a malicious UE running on a COTS SDR.

Layer	Message	C	I	Attack
RRC	DLInformationTransfer	◯	◯	
RRC	ConnectionReconfiguration	◯	◯	
RRC	ConnectionReestablishment	◯	✓	
RRC	ConnectionReestablishmentRequest	◯	✓	[10]
RRC	ConnectionReject	✗	✗	
RRC	ConnectionRelease	◯	◯	
RRC	ConnectionRequest	✗	✗	[10], [9]
RRC	ConnectionResumeRequest	✓	✓	[10]
RRC	ConnectionSetup	✗	✗	
RRC	ConnectionSetupComplete	✗	✗	
RRC	SecurityModeCommand	✗	✗	
RRC	SecurityModeComplete	✗	✗	
RRC	SecurityModeFailure	✗	✗	
RRC	UECapabilityEnquiry	◯	◯	
RRC	UECapabilityInformation	◯	◯	
NAS	AttachReject	◯	✗	[19]
NAS	AttachRequest	✗	✗	[19]
NAS	AuthenticationFailure	◯	✗	
NAS	AuthenticationReject	◯	✗	
NAS	AuthenticationRequest	◯	✗	
NAS	AuthenticationResponse	◯	✗	
NAS	DetachAccept	◯	✗	
NAS	DetachRequest	◯	✗	
NAS	IdentityRequest	◯	✗	[10], [17]
NAS	IdentityResponse	◯	✗	
NAS	SecurityModeCommand	✗	◯	
NAS	SecurityModeReject	✗	✗	[10]
NAS	ServiceRequest	◯	✗	[19]
NAS	ServiceReject	◯	✗	[19]
NAS	TrackingAreaUpdateReject	◯	✗	
NAS	TrackingAreaUpdateRequest	✗	✗	

TABLE V: L3 Control-plane message of the RRC/NAS(EMM) protocol that are (potentially) exploitable [21], [20] (C: Ciphered, I: Integrity protected, ◯: Protection not mandated).

In Table VI, we show the telemetry available from the F1 Application Protocol (F1AP), which could be collected to extend MobiFlow's feature set. Based on 3GPP TS 38.473 [35], F1AP provides a standard interface for interconnecting CU and DU of an eNB/gNB. All F1AP functions are categorized into different elementary procedures, such as UE context setup and RRC message transfer. For this reason, F1AP is implemented in various CU and DU implementations (e.g., OpenAirInterface [8]), and thus we can extract telemetry by instrumenting this interface. Furthermore, MobiFlow's feature set can be expanded, such as supporting other protocols (e.g., paging) for diagnosis or exploit detection.

F1AP Elementary Procedure	Telemetry
Interface Management	E2 node management traffic
UE Context Management	UE identifiers, DRB/SRB list
RRC Message Transfer	UL/DL RRC/NAS traffic
Warning Message Transmission	PWS traffic
System Information	SI delivery command
Paging	Paging messages
Trace	Trace session traffic
Radio Information Transfer	Radio-related information
IAB	IAB traffic
Self Optimisation Support	Access & mobility information
Reference Time Info Reporting	Reference Time Info
Positioning	Measurement traffic & report
NR MBS	Broadcast context
PDC Measurement Reporting	PDC measurement
QMC	QoE information

TABLE VI: Telemetry that can be collected via the F1AP interfaces [35] categorized by the elementary procedures.

B. *Detection Rule Sets in P-BEST*

We describe the details of the P-BEST detection rules that are not covered in §VI. For the BTS resource depletion attack, the corresponding detection rules are shown in Figure 10, which include four rules written in P-BEST language. For blind DoS attacks, the detection rule is presented in Figure 11. Below, we provide the details of detection rules for the BTS resource depletion attack.

- `bts_depletion_first_transient_ue`. Listing 1 applies when the first transient UE is detected for a BS. Specifically, for a MobiFlow record *ue*, it checks if *ue* is not in `RRC_CONNECTED` state, and its RRC connection period (computed based on the initial and inactive timers as in Table I) is less than a threshold. Meanwhile, it ensures no duplicate transient UE by using the "-" operator. When the rule is activated, a new transient UE instance is created along with a counter to track the total transient UEs for each BS. Finally, *ue* is marked as transient using the "$" operator to prevent this rule from being evaluated on the same *ue* in the future.
- `bts_depletion_add_transient_ue`. Listing 2 functions the same as the first rule except that it applies when there is already a transient counter fact exists for the current BS. To this end, it uses "/" to increase the counter value instead of creating a new one.
- `bts_depletion_release_transient_ue`. Listing 3 defines a GC rule to recycle the created transient UEs, which prevents 5G-Spector from producing false alarms for the attack. We define a `RELEASE_THRESHOLD` as an `xtype` integer. When a transient UE instance is deleted, it simultaneously decreases the transient UE counter of that particular BS.
- `bts_depletion_generate_event`. Listing 4 is activated when the transient UE counter exceeds a certain `UE_THRESHOLD`. If so, a potential BTS resource depletion

```
rule[bts_depletion_first_transient_ue:
    [+ue:ue_mobiflow^TRANSIENT]
    [?|ue.rrc_state < RRC_CONNECTED]
    [?|strcmp(ue.rrc_inactive_timer, "0") != 0 &&
        strcmp(ue.rrc_initial_timer, "0") != 0]
    [?|ue.rrc_inactive_timer-ue.rrc_initial_timer < '
     T_THRESHOLD]
    [-transient_ue_cntr | bs_id==ue.bs_id]
    [-transient_ue | bs_id==ue.bs_id, rnti==ue.rnti]
==>
    [+transient_ue_cntr | bs_id=ue.bs_id, value=1, ts=ue.ts]
    [+transient_ue | bs_id=ue.bs_id, rnti=ue.rnti, ts=ue.ts]
    [$|ue:TRANSIENT]
]
```

Listing 1: `bts_depletion_first_transient_ue`

```
rule[bts_depletion_add_transient_ue:
    [+ue:ue_mobiflow^TRANSIENT]
    [?|ue.rrc_state < RRC_CONNECTED]
    [?|strcmp(ue.rrc_inactive_timer, "0") != 0 &&
        strcmp(ue.rrc_initial_timer, "0") != 0]
    [?|ue.rrc_inactive_timer-ue.rrc_initial_timer < '
     T_THRESHOLD]
    [+tran_ue_cntr:transient_ue_cntr | bs_id==ue.bs_id]
    [-transient_ue | bs_id==ue.bs_id, rnti==ue.rnti]
==>
    [/tran_ue_cntr | value+=1, ts=ue.ts]
    [+transient_ue | bs_id=ue.bs_id, rnti=ue.rnti, ts=ue.ts]
    [$|ue:TRANSIENT]
]
```

Listing 2: `bts_depletion_add_transient_ue`

```
rule[bts_depletion_release_transient_ue:
    [+tran_ue:transient_ue]
    [+tran_ue_cntr:transient_ue_cntr | bs_id==tran_ue.bs_id]
    [?|ts - tran_ue.ts > 'T_THRESHOLD]
==>
    [/tran_ue_cntr | value -= 1]
    [-|tran_ue]
]
```

Listing 3: `bts_depletion_release_transient_ue`

```
rule[bts_depletion_generate_event:
    [+tran_ue_cntr: transient_ue_cntr^BTS_RESOURCE_DEPLETION
     ]
    [?|tran_ue_cntr.value > 'UE_THRESHOLD]
==>
    [$|tran_ue_cntr: BTS_RESOURCE_DEPLETION]
    [!|printlog("[BTS Resource Depletion] Detected for bs %d
     \n", tran_ue_cntr.bs_id)]
]
```

Listing 4: `bts_depletion_generate_event`

Fig. 10: IDS rule sets in P-BEST language for detecting BTS resource depletion attacks [9] using transient UE counters.

```
rule[blind_dos_detect:
    [+ue1:ue_session]
    [+ue2:ue_session^BLIND_DOS]
    [?|ue2.bs_id==ue1.bs_id && ue2.rnti!=ue1.rnti]
    [?|ue2.ts-ue1.ts > 0]
    [?|ue1.rrc_state==RRC_CONNECTED && ue2.rrc_state==
     RRC_CONNECTED]
    [?|strcmp(ue2.tmsi, ue1.tmsi) == 0]
==>
    [$|ue2:BLIND_DOS]
    [!|printlog("[Blind DoS] Detected for ue %d\n", ue2.
     rnti)]
]
```

Fig. 11: IDS rule set in P-BEST language for Blind DoS attack [9] using TMSI replay detection.

attack is detected for the particular BS, and an event will be generated.

C. Other Cellular L3 Attacks

We further discuss 5G-SPECTOR's applicability to other L3 attacks that have been discovered in the literature but are not our targets. To this end, we summarize existing attacks targeting the L3 protocols (e.g., RRC and NAS) in Table VII, by showing the details of the attacks (e.g., the adversary type, attack implication, and message exploited). Most of these attacks were discovered in the 4G LTE standard, but are still applicable to the most recent 5G network. As indicated by the adversary types in the table, most L3 attacks are launched by fake base stations and are well-studied in the literature [10], [12], [48], [70]. More specifically, FBS attacks are performed by using an SDR-implemented rogue base station to impersonate legitimate base stations (e.g., by copying their broadcast parameters and messages that are unauthenticated [38]). After luring nearby victim UE to connect, the FBS then transmits malicious messages to perform DoS or privacy leakage attacks, which are also known as IMSI catchers in legacy 2G and 3G networks [42], [23]. We consider these attacks can be detected by using existing FBS detection (or IMSI catcher detection) techniques [42], [23], [24], [57], [26]. Such techniques, which rely on signal strengths, measurement reports, and other heuristics, can also be deployed at the O-RAN control plane.

In addition, some L3 attacks could be detected with expanded MOBIFLOW feature sets. For example, to detect the NAS Counter Reset attack [10], 5G-SPECTOR needs to have additional features such as the message authentication codes (MAC) and the sequence numbers of L3 messages, which are currently out of the scope of MOBIFLOW but could be extracted from the RRC or NAS message payload. Only a handful of attacks are not detectable by 5G-SPECTOR (and other RAN-based IDS). One category of undetectable attacks is passive attack [10], [70] that exploits side channel vulnerabilities of the protocol. The other category of attack involves the exploitation of unobservable messages. For example, the Exposing Device's TMSI and Paging Occasion attack [10] assumes an adversary can selectively drop the downlink `RRCConnectionRelease` message, by using a MiTM relay. In this attack, there is no available mechanism to confirm the message has been dropped from the network's perspective. We consider this a general limitation for all defenses deployed at the RAN and not unique to 5G-SPECTOR, since the current vulnerable cellular standard lacks verification mechanisms on unprotected L3 messages.

Attack	**Ref**	**Adversary**	**Layer**	**Implication**	**Exploited L3 Message**	**Detectable**
Authentication Sync Failure	[12]	UE	NAS	Availability	AttachRequest	◐
Traceability Attack	[12]	FBS	NAS	Privacy	SecModeCommand	▲
Numb Attack	[12]	FBS	NAS	Availability	AuthenticationReject	▲
Paging Channel Hijacking	[12]	FBS	RRC	Availability	Paging	▲
Stealthy kicking-off Attack	[12]	FBS	RRC	Availability	Paging	▲
Panic Attack	[12]	FBS	RRC	Availability	Paging	▲
Energy Depletion Attack	[12]	FBS	RRC	Availability	Paging	▲
Linkability Attack	[12]	FBS	RRC	Privacy	Paging	▲
Detach/Downgrade Attack	[12]	FBS	NAS	Availability	DetachRequest	▲
NAS Counter Reset	[10]	FBS/MiTM	NAS	Availability	SecModeCommand/Complete	◐
Uplink NAS Counter Desync	[10]	FBS/MiTM	NAS	Availability	SecModeCommand	◐
Exposing NAS Sequence Number	[10]	Passive	NAS	Privacy	SecModeCommand/Complete	○
Neutralizing TMSI Refreshment	[10]	MiTM	NAS	Privacy	ConfigUpdateCommand	○
Cutting off the Device	[10]	MiTM	NAS	Availability	RegRequest/DeRegRequest	○
DoS with RRC Setup Request	[10]	UE	RRC	Availability	ConnectionRequest	●
Installing Null Cipher / Integrity	[10]	MiTM	RRC	Confidentiality	SecurityModeComplete	●
Lullaby Attack	[10]	FBS/MiTM	RRC	Availability	RRCReconfiguration	●
Incarceration Attack	[10]	FBS/MiTM	RRC	Availability	RRCReject	●
Exposing TMSI / Paging / RNTI	[10]	MiTM	RRC	Privacy	RRCRelease	○
BTS Resource Depletion	[9]	UE	RRC	Availability	ConnectionRequest	●
Blind DoS	[9]	UE	RRC	Availability	ConnectionRequest	●
Remote De-registration	[9]	UE	NAS	Availability	AttachRequest	●
AKA Bypass Attack	[9]	Core	RRC	Confidentiality	SecModeCommand/Complete	●
TORPEDO	[70]	Passive	RRC	Privacy	Paging	○
PIERCER	[70]	FBS	RRC	Privacy	Paging	▲
IMSI Cracking	[70]	FBS	RRC	Privacy	Paging	▲
Location Leak Attacks in LTE	[48]	FBS	RRC/NAS	Privacy	Miscellaneous	▲
DoS Attacks in LTE	[48]	FBS	RRC/NAS	Availability	Miscellaneous	▲
Downlink IMSI Extractor	[17]	MiTM	NAS	Privacy	IdentityRequest	●
Uplink IMSI Extractor	[19]	MiTM	NAS	Privacy	AttachRequest	●
Downlink DoS	[19]	MiTM	NAS	Availability	AttachReject	●
Uplink DoS	[19]	MiTM	NAS	Availability	AttachRequest	●

TABLE VII: L3 attacks in the literature (●: Detectable by 5G-SPECTOR, ◐: Detectable with expanded MOBIFLOW feature set, ○: Not detectable, ▲: Has been addressed by fake base station detection techniques [24], [26], [42]).

Appendix B
Artifact Appendix

5G-Spector is the first Open Radio Access Network (O-RAN) compliant layer-3 cellular attack detection service. It is based on the revolutionary O-RAN architecture that brings unprecedented programmability that enables stakeholders (e.g., network operators) and innovators to build novel software-defined services on cellular networks. This artifact includes instructions to access 5G-Spector's source code and replicate 5G-Spector in a controlled environment within a simulated cellular network.

A. Description & Requirements

1) How to access: 5G-Spector's main GitHub repository can be found at https://github.com/5GSEC/5G-Spector. The current deployment of 5G-Spector demands very complex software dependencies, and thus we have prepared a pre-built VM image which is publicly available at https://zenodo.org/records/10154551. We have provided step-by-step instructions to run the artifact at https://github.com/5GSEC/5G-Spector/wiki/5G%E2%80%90Spector-Artifact-in-a-Simulated-LTE-Network.

2) Hardware dependencies: 5G-Spector was evaluated on a host machine (Ubuntu 18.04 OS) equipped with 12 Intel i7-8700 cores and 32GB RAM. To make the experiments readily reproducible, this artifact uses the OAI RF emulator without actual radio hardware. As a result, it only requires a commodity Linux machine with similar (or higher) hardware and a recommended free storage of 100GB.

3) Software dependencies: The 5G-Spector artifact runs on a Ubuntu 18.04 VM. It includes RAN and UE implementations based on OpenAirInterface (`2023.w23` release), LTE core implementations from the OMEC EPC, and control plane (RIC) implementations from ONOS RIC. The whole software environment is deployed within a single VM via the SD-RAN's RAN-in-a-Box (RiaB) model and uses Kubernetes and Docker to manage the network settings and configurations.

4) Benchmarks: None.

B. Artifact Installation & Configuration

Download the 5G-Spector artifact from https://zenodo.org/records/10154551, and then follow the instructions from https://github.com/5GSEC/5G-Spector/wiki/5G%E2%80%90Spector-Artifact-in-a-Simulated-LTE-Network to install dependencies and set up the VM.

C. Major Claims

(C1): 5G-Spector can detect 7 Layer-3 cellular attacks that are published in previous literature.

(C2): 5G-Spector can detect 11 Layer-3 attack variants derived from the 7 known attacks.

D. Evaluation

Before evaluation, please make sure 5G-Spector is successfully deployed. To begin with, stop any eNB and UE instances you are running. You should reach an evaluation state with the OMEC core, ONOS RIC, MobieXpert xApp up and running. Please follow instructions from https://github.com/5GSEC/5G-Spector/wiki/5G%E2%80%90Spector-Artifact-in-a-Simulated-LTE-Network. You can use `kubectl` to verify if all containers are running. Below is an example of a correct evaluation state:

```
$ kubectl get pods -n riab
NAME                   READY   STATUS    RESTARTS   AGE
cassandra-0            1/1     Running   0          18m
hss-0                  1/1     Running   0          18m
mme-0                  4/4     Running   0          18m
onos-a1t-xxx           2/2     Running   0          14m
onos-cli-xxx           1/1     Running   0          14m
onos-config-xxx        3/3     Running   0          14m
onos-e2t-xxx           2/2     Running   0          14m
onos-kpimon-xxx        2/2     Running   0          14m
onos-rsm-xxx           2/2     Running   0          14m
onos-topo-xxx          2/2     Running   0          14m
onos-uenib-xxx         2/2     Running   0          14m
pcrf-0                 1/1     Running   0          18m
sd-ran-consensus-0     1/1     Running   0          14m
sd-ran-consensus-1     1/1     Running   0          14m
sd-ran-consensus-2     1/1     Running   0          14m
sec-sm-xapp-xxx        2/2     Running   0          53s
spgwc-0                2/2     Running   0          18m
upf-0                  4/4     Running   0          16m
```

1) Experiment (E1): [BTS Resource Depletion Attack Detection] [5 human-minutes]

[How to] Run the BTS resource depletion attack (1 instance) and monitor the 5G-Spector xApp's log.

[Preparation] The artifact should be in an evaluation state.

[Execution] Run the RAN (i.e., eNB), and then at a new terminal, start the attack:

```
$ ./run_attack.sh --bts-attack 1 --bts-delay 100
```

[Results] The 5G-Spector xApp, i.e., `sec-sm-xapp` should show a similar detection log as follows:

```
[CRITICAL 2023-09-29 00:39:31,499 PBest.py:71] [PBest]
     Attack event detected
[CRITICAL 2023-09-29 00:39:31,499 PBest.py:72] {
  "Event ID": 1,
  "Event Name": "BTS Resource Depletion",
  "Affected base station ID": 0,
  "Time": "2023-09-29 00:39:31.419822",
  "Number of DoS UE": 4
}
```

2) Experiment (E2): [Blind DoS Attack Detection] [10 human-minutes]

[How to] In this attack, we need to run the blind DoS attack (1 instance) using two UEs, with one being the victim and the other being the attacker who replays the same TMSI to attack the victim.

[Preparation] The artifact should be in an evaluation state.

[Execution] We cannot execute the `lte-uesoftmodem` (OAI UE) twice directly. The solution is to use docker to host two OAI UE instances. To simulate this, first start the eNB:

```
$ ~/run_enb.sh
```

Then start the first (victim) UE, i.e., UE0:

```
$ cd ~/blind_dos
$ docker compose up -d oai_ue0
```

After UE0 has been successfully connected, launch the second (attacker) UE, i.e., UE1:

```
$ cd ~/blind_dos
$ docker compose up -d oai_ue1
```

[Results] The 5G-SPECTOR xApp, i.e., `sec-sm-xapp` should show a similar detection log as follows:

```
[CRITICAL 2023-09-29 16:58:14,533 PBest.py:71] [PBest]
    Attack event detected
[CRITICAL 2023-09-29 16:58:14,533 PBest.py:72] {
  "Event ID": 2,
  "Event Name": "Blind DoS",
  "Affected base station ID": 0,
  "Time": "2023-09-29 16:58:14.486219",
  "Affected UE ID": 14097
}
```

3) Experiment (E3): [Uplink DoS Attack Detection] [5 human-minutes]

[How to] Run the Uplink DoS attack (2 instances) and monitor the 5G-SPECTOR xApp's log.

[Preparation] The artifact should be in an evaluation state.

[Execution] Run attack (`VAR` can be selected from 1-2):

```
$ ./run_attack.sh --uplink-dos-attack <VAR>
```

[Results] The 5G-SPECTOR xApp, i.e., `sec-sm-xapp` should show a similar detection log as follows:

```
[WARNING 2023-09-28 15:47:53,231 PBest.py:67] [PBest]
    Warning event detected
[WARNING 2023-09-28 15:47:53,231 PBest.py:68] {
  "Event ID": 1,
  "Event Name": "Uplink DoS Service Request",
  "Affected base station ID": 0,
  "Time": "2023-09-28 15:47:53.210439",
  "Affected UE ID": 32539
}
```

4) Experiment (E4): [Downlink DoS Attack Detection] [5 human-minutes]

[How to] Run the Downlink DoS attack (6 instances) and monitor the 5G-SPECTOR xApp's log.

[Preparation] The artifact should be in an evaluation state.

[Execution] Run attack (`VAR` can be selected from 1-6):

```
$ ./run_attack.sh --dnlink-dos-attack <VAR>
```

[Results] The 5G-SPECTOR xApp, i.e., `sec-sm-xapp` should show a similar detection log as follows:

```
[CRITICAL 2023-09-28 21:03:22,659 PBest.py:70] [PBest]
    Attack event detected
[CRITICAL 2023-09-28 21:03:22,659 PBest.py:71] {
  "Event ID": 0,
  "Event Name": "Downlink Overshadowing",
  "Affected base station ID": 0,
  "Time": "2023-09-28 21:03:22.658086",
  "Affected UE ID": 32911
}
```

5) Experiment (E5): [Uplink IMSI Extractor Attack Detection] [5 human-minutes]

[How to] Run the Uplink IMSI Extractor attack (1 instance) and monitor the 5G-SPECTOR xApp's log.

[Preparation] The artifact should be in an evaluation state.

[Execution] Run attack:

```
$ ./run_attack.sh --uplink-imsi-extr 1
```

[Results] The 5G-SPECTOR xApp, i.e., `sec-sm-xapp` should show a similar detection log as follows:

```
[WARNING 2023-09-28 15:51:46,687 PBest.py:67] [PBest]
    Warning event detected
[WARNING 2023-09-28 15:51:46,687 PBest.py:68] {
  "Event ID": 1,
  "Event Name": "Uplink IMSI Extractor",
  "Affected base station ID": 0,
  "Time": "2023-09-28 15:51:46.667721",
  "Affected UE ID": 43557
}
```

6) Experiment (E6): [Downlink IMSI Extractor Attack Detection] [5 human-minutes]

[How to] Run the Downlink IMSI Extractor attack (5 instances) and monitor the 5G-SPECTOR xApp's log.

[Preparation] The artifact should be in an evaluation state.

[Execution] Run attack (`VAR` can be selected from 1-5):

```
$ ./run_attack.sh --dnlink-imsi-extr <VAR>
```

[Results] The 5G-SPECTOR xApp, i.e., `sec-sm-xapp` should show a similar detection log as follows:

```
[CRITICAL 2023-09-28 15:11:04,076 PBest.py:61] [PBest]
    Attack event detected
[CRITICAL 2023-09-28 15:11:04,076 PBest.py:62] {
  "Event ID": 6,
  "Event Name": "Downlink IMSI Extractor",
  "Affected base station ID": 4,
  "Time": "2023-09-28 15:11:04.039173",
  "Affected UE ID": 43881
}
```

7) Experiment (E7): [Null Cipher and Integrity Attack Detection] [5 human-minutes]

[How to] Run the Null Cipher and Integrity attack (2 instances) and monitor the 5G-SPECTOR xApp's log.

[Preparation] The artifact should be in an evaluation state.

[Execution] Run attack (`VAR` can be selected from 1-2):

```
$ ./run_attack.sh --null-cipher-integ <VAR>
```

[Results] The 5G-SPECTOR xApp, i.e., `sec-sm-xapp` should show a similar detection log as follows:

```
[WARNING 2023-09-28 15:52:29,684 PBest.py:67] [PBest]
    Warning event detected
[WARNING 2023-09-28 15:52:29,687 PBest.py:68] {
  "Event ID": 4,
  "Event Name": "Null Cipher & Integrity (RRC)",
  "Affected base station ID": 0,
  "Time": "2023-09-28 15:52:29.676571",
  "Affected UE ID": 31084
}
```

E. Notes

While this artifact demonstrates 5G-SPECTOR within a simulated cellular network, 5G-SPECTOR can also run on physical software-defined radios (SDRs) (e.g., a USRP B210) and work with real cellular devices. In our setting, the SDR is attached to a VM via USB 3.0, and the VM has enabled PCI passthrough with QEMU to ensure maximum CPU performance. In addition, 5G-SPECTOR has been successfully integrated and tested within a 5G SA network, with enhanced RIC agent support for OAI 5G RAN implementation and the ONOS RIC. We are working towards a functional 5G SA prototype and will release it in the future.