Thwarting Smartphone SMS Attacks at the Radio Interface Layer

Haohuang Wen
The Ohio State University
wen.423@osu.edu

Phillip Porras
SRI International
porras@csl.sri.com

Vinod Yegneswaran
SRI International
vinod@csl.sri.com

Zhiqiang Lin
The Ohio State University
zlin@cse.ohio-state.edu

***Abstract*—The short message service (SMS) is a cornerstone of modern smartphone communication that enables inter-personal text messaging and other SMS-based services (e.g., two-factor authentication). However, it can also be readily exploited to compromise unsuspecting remote victims. For instance, novel exploits such as Simjacker and WIBAttack enable transmission of binary SMS messages that could surreptitiously execute dangerous commands on a victim device. The SMS channel may also be subverted to drive other nefarious activities (e.g., spamming, DoS, and tracking), thereby undermining end-user security and privacy. Unfortunately, neither contemporary smartphone operating systems nor existing defense techniques provide a comprehensive bulwark against the spectrum of evolving SMS-driven threats. To address this limitation, we develop a novel defense framework called RilDefender, which to the best of our knowledge is the first *inline prevention system* integrated into the radio interface layer (RIL) of Android smartphones. We describe an implementation of RilDefender on three smartphone models with five Android versions of the Android Open Source Project (AOSP), and show that it is able to protect users from six types of SMS attacks spanning four adversary models. We evaluate RilDefender against 19 reproduced SMS attacks and 11 contemporary SMS malware samples and find that RilDefender detects all and automatically prevents all but one of these threats without affecting normal cellular operations.**

I. Introduction

The Short Message Service (SMS) is a low-bandwidth text-based transmission protocol introduced prior to the advent of smart Internet-connected mobile devices. However, it is less known that the SMS service can actually be used in a variety of ways that are opaque to mobile phone users. Particularly, the SMS service can be weaponized in ways that violate users' privacy and security, including the existence of SMS functions that can be inserted into the SIM supply chain and remain undocumented by mobile providers [1].

We posit that it is incorrect for one to view SMS as a text-based service, as an SMS message can carry binary attachments that cause vulnerable SIMs to execute dangerous operations on a smartphone without the recipient's awareness or consent. An attacker can thus exploit this vulnerability by crafting binary-embedded SMS messages and delivering them to users, which can cause covert dialing or the sending of SMS, the opening of a browser with a specified URL, and even the execution of AT commands [2]. Two instances of such binary SMS attacks are Simjacker [3] and WIBAttack [4], affecting mobile users from 29 countries worldwide [3]. Launching such an attack only requires a pre-paid SIM and a GSM-compatible USB modem, which can be acquired for as little as \$20.

Other exploit paths exist within the SMS protocol that can violate a mobile user's privacy and security. For example, silent and flash SMS messages are two separate types of SMS that are used by mobile operators and governments to track a smartphone owner's geo-position or to post emergency alerts, respectively [5]. Unfortunately, both of them can also be exploited by a larger pool of adversaries to perform tracking and denial-of-service (DoS) attacks against mobile devices [5]–[7]. Fraudulent and spammed SMS messages can also be sent from a GSM fake base station (FBS) to impersonate a legitimate source in order to spoof the recipient [8], [9]. The increasing availability of recent software stacks [10] and software-defined radio hardware [11], [12] makes such attack campaigns more accessible to unsophisticated adversaries. Indeed, FBS SMS represents widespread threats within some countries and causes billions in monetary loss per year [9].

SMS has also been usurped as a communication medium that enables malware-infected phones to receive instructions, exfiltrate data from infected mobile devices, and accelerate malware propagation [13]. Mobile malware developers are adept at luring users to install and grant privileges to them, for example, by masquerading their malware as legitimate postal-service apps [14] and COVID-19 notification apps [15]. Once the malware is trusted by the user, SMS offers adversaries a channel to quickly spread malicious links to other users in the victim's contact list. The recipients are more likely to access these links as they originate from a known source.

Reports of aforementioned attacks have not gone unnoticed. In particular, contemporary research efforts on defenses designed to help end users detect malicious SMS attacks may be broadly categorized into two classes. The first class includes device-centric defenses, which install additional software plug-ins to mobile devices. For example, Android-based monitoring apps can detect silent SMS and FBS attacks [16]–[18]. The second class is network-based defense, which involves the deployment of cellular hardware infrastructures within the

Network and Distributed System Security (NDSS) Symposium 2023
27 February - 3 March 2023, San Diego, CA, USA
ISBN 1-891562-83-5
https://dx.doi.org/10.14722/ndss.2023.24432
www.ndss-symposium.org

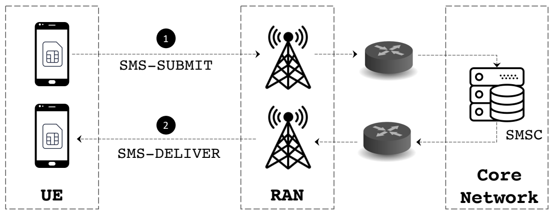


Fig. 1: SMS transmission path in a cellular network.

radio access network to monitor cellular traffic. For instance, city-wide cellular network monitors [8], [19]–[21] were deployed to hunt for malicious FBS messages. While these solutions help end-users understand the prevalence of certain SMS attacks, they suffer from three key limitations. First, most defenses focus on passive detection and simply produce alerts for attacks that are observed [8], [16]–[21]. Second, they often have poor scalability, e.g., network-based solutions require the deployment of hardware infrastructures at a large scale [8], [19]–[21]. Third, existing solutions do not comprehensively address the menagerie of aforementioned SMS attacks, such as Simjacker [3], WIBAttack [4], and SMS messages generated from proactive SIMs [1].

In response, we design and develop a system-level defensive framework called RILDEFENDER, which is the first inline prevention system integrated into the Android smartphone OS's *Radio Interface Layer (RIL)*. RILDEFENDER is capable of addressing a wide range of SMS attacks and offers several critical advantages over existing SMS defense technologies. First, unlike existing SMS defenses [16], [17], RILDEFENDER provides the capability to not only detect SMS attacks but also intercede (block) them automatically in real-time. Second, RILDEFENDER's deployment at the RIL makes it agnostic to a wide range of vendor-specific smartphone OSs. Third, RILDEFENDER is designed to be extensible to new SMS threats by providing user-configurable attack signatures and policies. We demonstrate the utility of this policy language by presenting inline threat mitigation rules for six types of SMS attacks from four different adversary classes.

As a proof-of-concept, we present a prototype of RILDEFENDER integrated as an extension to the Android Open-Source Project (AOSP) [22]. We have implemented and tested RILDEFENDER on three representative Android smartphone models with five legacy and most recent Android OS versions (up to Android 13). To evaluate RILDEFENDER's robustness against the six types of SMS threats, we replicated all these attacks using an SMS testbed atop a BladeRF 2.0 software-defined radio [11]. We tested our system against 19 reproduced SMS attack cases, 11 SMS malware samples, and real-world SMS traces of 7-days. We find that RILDEFENDER accurately prevented all these attacks (except one) while not affecting normal cellular operations. In addition, we show that RILDEFENDER imposes acceptable overhead in terms of power consumption, memory, storage, and computation. In the worst-case scenario, it imposes 1% battery consumption per hour and a 100ms delay on cellular operations for our tested devices. Further, RILDEFENDER can potentially be extended to support

Field	SCA	FO	OA/DA	PID	DCS	...	UDL	UD
Length	1-12	1	2-12	1	1	...	1	0-140

Fig. 2: Typical format of an SMS message.

SMS over IP Multimedia Subsystem (IMS) [23], which has been widely deployed in recent 4G and 5G networks.

In summary, our paper makes the following contributions:

- We present the first system-level defense residing at the RIL that detects and automatically mitigates SMS attacks.
- We assess the ability of RILDEFENDER to comprehensively defend against six types of SMS attacks targeting end-user devices from four adversary classes.
- We present a prototype of RILDEFENDER, which is integrated as an extension to the Android Open Source Project (AOSP), and evaluate its effectiveness and overhead.

II. BACKGROUND

A. Cellular Network and SMS Workflow

SMS transmissions traverse through user equipment (UE) such as smartphones, radio access networks (RAN), and the core network [24]. Figure 1 presents the typical workflow of how cellular networks process SMS messages exchanged by end-user mobile devices. First, the UE initiates an SMS by sending an SMS-SUBMIT message to its subscribed base station (BS). The base station then relays the message to an SMS center (SMSC) within the core network hosted by the receiver's network provider. Upon receiving the message, the SMSC repackages the message and forwards it to a base station close to the target UE. Finally, this base station delivers the message to the receiver's UE using an SMS-DELIVER message [24].

The structures of an SMS-SUBMIT and SMS-DELIVER message share some information in common, as shown in Figure 2 [24]. Specifically, the message starts with its service center address (SCA) field, indicating the SMSC address ranging from 1 to 12 bytes. Next, the first octet (FO) includes several encoded bits to convey some meta information. For instance, the first MTI bit indicates whether the SMS is an SMS-SUBMIT or an SMS-DELIVER. The originator address (OA) or destination address (DA) is also specified, followed by the protocol identifier (PID) and the data coding scheme (DCS). The PID and DCS fields define the type of SMS, such as a binary or silent SMS (discussed later). Finally, an SMS carries variable-length user data (UD), restricted by its length (UDL) up to 140 bytes. The UD normally carries the plain-text content of the SMS. An SMS with content longer than 140 bytes is split into separate SMS messages and reassembled upon arrival by the destination UE.

B. Radio Interface Layer

The *radio interface layer* (RIL) is an abstraction layer between the radio hardware and the OS, which generally exists in smartphone UEs. In iOS, such an abstraction is known as the *CommCenter* [25]. We use the Android RIL as an example

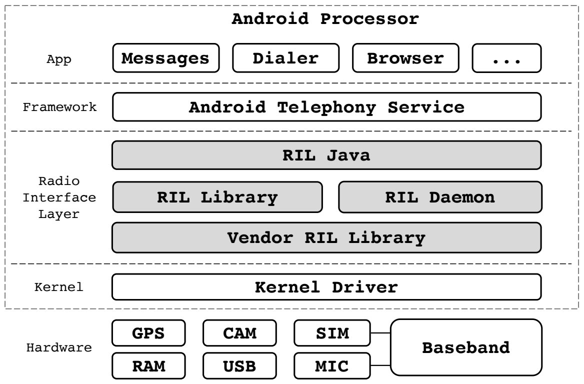


Fig. 3: Typical architecture of an Android UE.

for illustration, and we show its high-level architecture in Figure 3 [26]. As highlighted in grey, the RIL includes a variety of binaries and libraries. The RIL Java component serves as an intermediate layer between the Android telephony service and the native RIL libraries. Specifically, it provides encapsulated APIs for upper telephony functions, such as dialing and SMS. There is an RIL daemon (RILD), which handles the life cycle of the RIL process and communicates with the low-level radio hardware (e.g., baseband) through the vendor RIL library. The RIL defines specific commands for communication. Commands issued to the baseband through the RIL are called *solicited commands* (e.g., `DIAL` and `HANGUP` [26]), and the commands on the opposite direction are *unsolicited commands* (e.g., `CALL_STATE_CHANGED` and `NEW_SMS` [26]). The RILD, RIL libraries, and RIL Java are vendor-neutral, while vendor-specific (e.g., Samsung and Qualcomm) functions and customization are provided within each manufacturer's custom RIL library. For the vendor RIL library to communicate with the Linux kernel, there are multiple ways depending on the vendor's design, such as shared memory, serial communication, or TCP socket. We further illustrate the internal workflow of SMS at the RIL in Appendix §B, for readers of interest.

Cellular Data Processing. All cellular data is first processed by the baseband hardware. This initial processing phase is completely transparent to the kernel as well as its user-space programs. The baseband executes within an independent *baseband processor (BP)* and its implementation is often closed-source and tamper-resistant [27]–[29]. Once processed by the baseband, cellular transmissions are converted into requests, which are transferred to the kernel in the *Android processor (AP)* and delivered to the RIL to invoke cellular functions in user-space applications (e.g., the dialer and SMS app). The baseband also has direct access to some UE hardware components, such as the SIM and the microphone.

III. Adversary Model and Scope

A. Adversary Model

We consider four distinctive adversary models that can manifest violations of the privacy, integrity, or availability expectations of the UE owner. To scope this discussion, we assume that the baseband is trusted to the same extent as the underlying hardware, as it is often protected against unauthorized modifications [27]–[29]. Furthermore, as we focus on the SMS attack surface, we exclude consideration of system- and hardware-level attacks and assume that the kernel, RIL, and most of the hardware (excluding the SIM) are also trustworthy. In the following, we dive into the details of the four adversary models.

(I) **UE to UE (U2U) Attacks**. The U2U model applies to adversaries who employ the normal cellular SMS delivery channel to transmit malicious SMS messages to target UEs. Attack transmission can occur by using a commodity USB modem (e.g., a ZTE MF833V dongle [30]) to craft a malicious SMS in Protocol Data Unit (PDU) mode, which is then sent to the target UE. The attacker performs such transmissions by subscribing to a mobile operator network, such as through a prepaid SIM card. There is no location restriction as the SMS payload is delivered through SMSCs. For example, adversaries can use this engagement path to deliver a binary SMS with instructions that are then interpreted and executed without the consent and awareness of the victim UE owner. More precisely, the adversary is targeting the UE's SIM, which interprets a set of instructions within the SMS binaries, leading to information theft or availability attacks against the UE.

(II) **RAN to UE (R2U) Attacks**. The RAN that services communications between the UE and the core network also presents a potential engagement path for adversaries seeking to present malicious SMS payloads to the target UE. In particular, fake base stations (FBS) have been increasingly prevalent due to the availability of cellular software stacks [10], [31] along with the accessibility of software-defined radios (SDRs) [11], [12]. This adversary model involves the use of an FBS as the channel to deliver SMS payloads to victim UEs. Thus, the adversary must first trick the target UE into connecting with the FBS, by various means such as transmitting high signal power [32]–[34] and downgrading the victim's device to legacy GSM network [35], [36] (e.g., by simply jamming the current network frequency [34]). The R2U model can also serve as another feasible path for U2U attacks, such as delivering a binary SMS through an FBS to nearby victims.

(III) **Internet to UE (I2U) Attacks**. The I2U adversary model involves scenarios in which the adversary implants malicious application logic within the target UE (such as Android malware applications) and then uses the SMS service as a covert communication path. For example, SMS-based mobile botnets have demonstrated centralized and P2P SMS-based command and control (C&C) channels [37]. Malware, such as the Android remote access tool (RAT) trojan [38], also has the ability to send SMS messages to users within the phone's contact list without the owner's awareness. In either case, malicious applications exploit the SMS service for their own purposes, while suppressing indicators to the device owner of this use.

(IV) **SIM to UE (S2U) Attacks**. The S2U adversary model represents concerns regarding the supply-chain insertion of logic within the SIM to proactively initiate outbound SMS messages without the consent or awareness of the UE owner.

Attack	Category	Adv. Location	Threat
Binary SMS	U2U/R2U	Remote/Nearby	Command injection
Silent SMS	U2U/R2U	Remote/Nearby	Tracking, DoS
Flash SMS	U2U/R2U	Remote/Nearby	Spoofing, DoS
FBS SMS	R2U	Nearby	Spoofing, Spamming
Malware SMS	I2U	Remote	Spoofing, Spamming
Proactive SIM SMS	S2U	Remote	Data exfiltration

TABLE I: Classification of SMS attacks targeting the UE

One functional legacy of SIM architectures prior to the development of smartphone was the extension of the SIM from simply an identity management unit to a computing platform that independently executes embedded applications, including an entire SIM-local browser [3], [4]. Mobile operators may install their customers' SIMs with applications that can access and exfiltrate sensitive UE-local data back to the mobile core using the SMS cellular service. We consider such SIM functions as exfiltration because these operations often occur without the user's consent or awareness, and thus could potentially be exploited by supply-chain attackers.

B. Our Focus: SMS Attacks Targeting UEs

SMS attacks targeting UEs can be broadly classified into six categories as summarized in Table I and described below.

1) Binary SMS. There is a special type of SMS in which the user data field carries binary data or executable code. These messages were originally designed for tasks such as over-the-air (OTA) SIM updates, voice mail notifications, and device configuration [17]. Unfortunately, this feature also poses an attack vector, which has been demonstrated in prior exploits such as WIBAttack [4] and Simjacker [3]. These exploits allow remote attackers to use a binary SMS to inject surreptitious code logic in victim devices without the knowledge or consent of the UE owner. Example functionality enabled includes dialing a specified number, sending SMS, launching a browser with a specified URL, or acquiring the victim's location. These functions are processed by the SIM, bypassing any OS-level notifications. Fundamentally, such exploits are feasible due to insufficient access controls within browser applications installed on SIM cards.

There are three prerequisites for binary SMS attacks: (1) a victim UE that has installed a vulnerable SIM, (2) an SMSC that does not filter binary SMS messages, and (3) an attacker who can craft and send a malicious binary SMS payload in PDU mode. It is known that a large portion of SIMs produced in 29 countries are vulnerable to Simjacker [3]. In addition, it requires a coordinated effort across operators to defend against binary SMS attacks because attackers can arbitrarily specify the SMSC address in the SMS payload [24]. As a result, it is difficult to completely prevent binary SMS attacks within the current cellular network ecosystem.

2) Silent SMS. Another special type of SMS silently handled by UEs is called *silent SMS* or *type zero SMS* [24]. Upon receiving a silent SMS, a UE typically responds with a delivery report for acknowledgment. The baseband, SIM, and RIL handle this SMS, without involvement from user-space applications. Thus, the UE owner is completely unaware that a silent SMS message has been delivered [24]. Law enforcement often uses silent SMS (in some countries) to track individuals through their mobile phones [5], since a silent SMS forces the UE to respond with a delivery report back to the sender. The response SMS is then used to perform distance measurements and triangulation that infer the owner's approximate geolocation. In addition, silent SMS can be abused for malicious purposes, such as conducting denial-of-service (DoS) attacks via silent SMS flooding [39], and extracting a user's Temporary Mobile Subscriber Identity (TMSI) over the air [40].

3) Flash SMS. Also known as class 0 SMS, a flash SMS received by a UE will be displayed on the screen through a pop-up window and is deleted once the window is dismissed [24]. As a result, flash SMS is often used by authorities to push alerts to users. However, flash SMS can also be exploited by malicious attackers to crash smartphones [6], [7]. Attackers can also craft a malicious flash SMS with fake alerts to spoof users, as its sender information will not be shown. Similar to binary and silent SMS, the attacker requires only a USB modem to inject flash SMS messages to remote target UEs.

4) Fake Base Station SMS. It has been shown that GSM (i.e., 2G) fake base stations (FBS) hosted on SDRs by an attacker can send fraud and spam SMS to nearby users [8], [9]. For instance, the attacker may impersonate a legitimate contact of the victim to send out a spoofed SMS to ask for money transfer. FBS SMS messages are prevalent in some countries. For example, in China, mobile users received more than 5.7 billion FBS messages in 2015 [8]. Establishing a fake base station is also feasible and profitable for an attacker. A study in 2017 [8] found that an attacker could establish an FBS for as little as $700, while the exploits from the FBS could generate revenues of more than $1,400 a day.

5) Malware SMS. Although modern smartphones have deployed permission mechanisms for access control, malware still remains a prevalent threat. It is shown that more than 10M devices were infected by a single malware in 2021 [13]. Malware links are often spread over SMS [14] or hidden in mobile app stores [13], luring users to install and grant privileges to them (e.g., by impersonating legitimate postal-service apps [14] or COVID-19 notification apps [15]). When permissions are granted, the remote attacker uses the malware to initiate calls or send SMS without the UE owner's awareness. Moreover, it has also been shown that malware does not require SMS permission to abuse IMS-based SMS silently [23]. Apparently, the existing OS-level defense (e.g., permission scheme) is not sufficient against this malware.

6) Proactive SIM SMS. Program logic can be inserted into the supply chain of mobile SIMs, enabling them to transmit SMS messages to the mobile operator without the knowledge or consent of the UE owner. We define this type of SMS as *proactive SIM SMS*, which can be triggered by nearly all the SIMs in the market today [1]. Most recently, AT&T SIMs were found to include embedded proactive logic to send SMS messages when the SIM detects hardware or firmware configuration changes. This discovery occurred as a result of a car accident lawsuit. One of the involved drivers was accused of being distracted due to an SMS sent from the driver's smartphone at a time close to when the accident occurred [41]. However, this SMS was found to have been sent from the AT&T SIM to an AT&T managed phone number dedicated to proactive SIM message collection. These SMS messages and AT&T's proactive logic are entirely undocumented, and even under subpoena, an AT&T engineer would not disclose the full structure of these SMS payloads nor the proactive generation logic. While proactive SIM SMS can serve benign purposes, it can also be weaponized by supply-chain attackers to produce malicious SIMs to exfiltrate sensitive user data.

C. Existing Defenses Against SMS Attacks

We summarize existing defenses that are designed (or are potentially extensible) for SMS attacks in Table II and discuss their pros and cons below.

Name	Layer	S-SMS	B-SMS	F-SMS	FBS-SMS	M-SMS	P-SMS	Mitigation
FBS-RADAR [8]	Network	✗	✗	✗	✓	✗	✗	✗
PHOENIX [46]	Baseband	○	○	○	○	✗	○	✓
AIMSICD [16]	App	✓	✗	✗	✗	✗	✗	✗
SNOOPSNITCH [17]	App	✓	✓	✗	✗	✗	✗	✗
SPAM-DETECTOR [9]	App	✗	✗	✗	✓	✗	✗	✗
SCAT [47]	App	○	○	○	○	✗	○	✗
MOBILEINSIGHT [43]	App	○	○	○	○	✗	○	✗
RILANALYZER [48]	App	○	○	○	○	✗	○	✗
RILDEFENDER	RIL	✓	✓	✓	✓	✓	✓	✓

TABLE II: Existing defenses against cellular attacks (S-SMS, B-SMS, F-SMS, M-SMS, P-SMS stand for silent/binary/flash/malware/proactive-SIM SMS. ○ means not implemented but potentially extensible).

Network-based Defenses. One defensive strategy is to detect attacks at the radio access network by deploying cellular network monitors to collect a large amount of network traffic for analysis [8], [19]–[21], [34], [42]. One notable example of this approach is FBS-RADAR [8], which detects fake base station SMS at scale based on crowdsourced data collected from mobile users. Its detection algorithm depends on a series of network heuristics including unusual signal strength, invalid BS broadcast parameters, incorrect geo-location, and fraudulent SMS content. However, network-based defenses suffer from coverage issues as numerous network monitoring hardware systems must be deployed, making coverage difficult to be achieved in practice.

UE-centric Defenses. UE-centric defenses are deployed directly on UE devices to defend against the threats locally. Compared to network-based defenses, they have extremely low-cost and are easier to deploy as no additional hardware is required. Instead, they simply require the installation of a smartphone app or an updated system firmware, and generally have acceptable overhead on UEs [43]. As summarized in Table II, there are seven UE-centric defenses (except FBS-RADAR), which are deployed on the Android platform. From a deployment perspective, such a defense can at either the baseband or the app layer, as described below accordingly.

- **Baseband-layer Defenses.** Ideally, baseband-layer defenses are at a processing layer that can prevent all cellular attacks targeting the UE, as the baseband provides full visibility and control over low-level cellular network traffic. Unfortunately, it is extremely difficult for an unauthorized party to deploy baseband-layer defenses (e.g., using binary hardening [44]) as they are closed-source and often protected by tamper-resistance signature verification, preventing unauthorized modifications [27]–[29]. To our knowledge, some baseband vendors have deployed defenses such as FBS detection [45]. However, they cannot comprehensively address the SMS threats and the details are not publicly disclosed.

  Particularly, while not focusing on SMS attacks, PHOENIX [46] provides both detection and mitigation for a series of cellular control-plane attacks. It retrofits the cellular stack of an open-source UE, namely srsLTE [31], which is equivalent to baseband modification. However, such a design is hard for an unauthorized party to achieve in practical UEs due to design constraints. Moreover, such a design does not generalize well due to highly-diversified baseband implementations across vendors.
- **App-layer Defenses.** Compared to baseband-layer defenses, app-layer defenses are much easier to deploy for end users. Among the defenses in Table II, three support SMS attack detection, including silent SMS [16], [17], binary SMS [17], and FBS SMS [9]. Two defenses [43], [47] are designed for network traffic monitoring and diagnosis in UEs. In summary, these defenses mainly rely on reading low-level cellular traffic from the baseband chips or using high-level Android APIs to sense the cellular network parameters (e.g., signal strengths of nearby base stations). *Notably,* RILANALYZER *[48] is also an app-layer defense that monitors 3G traffic from baseband logs, and thus is fundamentally different from* RILDEFENDER (discussed in §IV-A).

  Modern smartphone OSs also have deployed defenses such as SMS permissions and spam SMS filters. While they are useful to mitigate some SMS threats, they are not sufficient against more sophisticated SMS attacks such as silent and binary SMS. Moreover, though the permission scheme restricts access to critical resources and functions, it cannot eliminate the malware SMS threat and a second line of defense (discussed later in §V-D) is needed when the permission access control is bypassed.

IV. DETAILED DESIGN

We present RILDEFENDER, the first inline prevention system deployed at the *Radio Interface Layer (RIL)*. The design of RILDEFENDER strives to balance the benefits of monitoring at the upper application layer and bottom baseband layer, which translates to a trade-off between deployability and visibility. We consider RILDEFENDER as a software-only defense integrated into the UE operating system, which can be deployed via OS upgrades. Note that such upgrades can be issued directly from OS manufacturers, such as Google. In addition, we also consider professional users who can integrate RILDEFENDER into the UE's firmware image and flash it into the UE using officially provided tools such as FASTBOOT. In the rest of the section, we highlight the key distinctions of RILDEFENDER in §IV-A, discuss the design challenges in §IV-B, and describe the detailed design in §IV-C and §IV-D.

A. *Key Distinctions of* RILDEFENDER *Over Existing Defenses*

RILDEFENDER emphasizes the following three key distinctions over the existing UE-centric defenses described in §III-C:

- **Inline Control.** RILDEFENDER can not only detect a wide range of SMS attacks but also effectively provide inline control to automatically block them by manipulating corresponding internal-control logic. This cannot be achieved by most existing defenses in Table II, which can only provide after-the-fact detection. One notable exception is PHOENIX, a baseband-level defense. However, it is deployed at srsLTE and does not target practical UEs as mentioned in §III-C. We believe that this distinguishing inline control capability is extremely valuable as it enables real-time mitigation of threats as and when the attacks occur.
- **Generality.** As RIL is vendor-agnostic and generally applicable to the OS of Android UEs [26], RILDEFENDER can be deployed in smartphone UEs without much vendor-specific modification that adds to support costs and complexity. The only vendor-dependent component is an optional baseband monitor (§IV-C) integrated to address non-interactive binary SMS attacks. In contrast, existing UE-centric defenses rely on parsing vendor-specific baseband traffic [16], [17], [46].
- **Extensibility.** RILDEFENDER can be extended with new SMS attack signatures and mitigation policies, which are modeled as propositional logic encoded by `YAML` [49] language. Note that new extensions (e.g., signatures and policies) can be integrated through the RILDEFENDER app by the users without redeploying RILDEFENDER at the OS layer. However, existing defenses are mostly static and cannot be easily extended for new attacks [16], [17], [43].

B. *Challenges and Solutions*

Detecting Baseband-only SMS Attacks. RIL-based defenses generally do not have full visibility to low-level cellular information. In particular, we notice there are two instances of binary SMS (detailed later in §V-A) that are directly handled within the baseband without being passed to the RIL. To detect these baseband-only SMS attacks, we integrate a *Baseband Monitor* component in RILDEFENDER. It establishes a communication channel with the baseband processor (BP) through the diagnostic interface to parse baseband traffic and identify the SMS events of interest. Note that the diagnostic interface (e.g., `/dev/diag` in Qualcomm devices [43]) is prevalent among heterogeneous Android and even iOS devices [17], [43], [47]. There are publicly available libraries [17], [43], [47] that support baseband monitoring of major vendors (e.g., Qualcomm, Samsung, and MediaTek), which serve as the building blocks to develop a baseband monitor component.

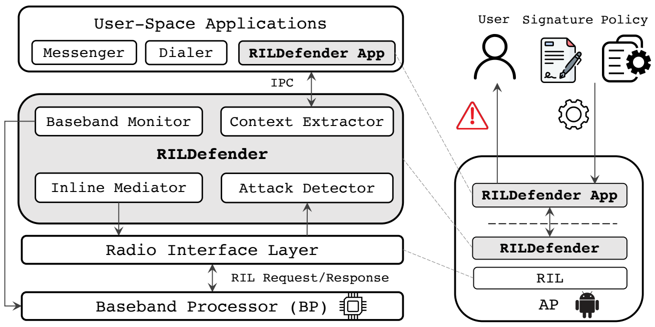


Fig. 4: A high-level illustration of RILDEFENDER.

Tracking SMS Context. We notice that relying on the structural information of an SMS (i.e., the SMS fields in Figure 2) is not sufficient to detect some SMS attacks. For instance, only with an outbound SMS payload, RILDEFENDER cannot determine it is triggered by a legitimate user, a proactive SIM, or a malicious application, to determine the presence of a malicious outbound SMS attempt. Similarly, to detect an FBS SMS, one needs to know the signal strengths and parameters broadcast from the connected BS when the SMS is received [8]. That is, RILDEFENDER must know the context information when an SMS is captured at the RIL. To address this challenge, we develop a *Context Extractor* component that tracks the SMS context (e.g., the originating source and connected BS states). As a system-layer defense, RILDEFENDER can identify the source of an outbound SMS by making inter-process communication calls [50] to the user-space applications. Note that this is a unique advantage of RILDEFENDER as a system-level deployment feature, and cannot be achieved by any of the app-layer defenses [16]–[18].

C. RILDEFENDER *at the Radio Interface Layer*

Figure 4 shows a high-level illustration of RILDEFENDER deployed at both the RIL and the application layer. We first describe RILDEFENDER at the RIL, which mainly consists of the following four logical components:

1) **Baseband Monitor** is based on existing libraries [17], [43], [47] to monitor and interpret baseband traffic from proprietary protocols through the diagnostic interface. We tailor existing library implementation to only extract the SMS traffic of interest (to detect baseband-only SMS attacks).

2) **Context Extractor** interacts with user-space applications to track the necessary SMS context (e.g., the process that initiates an outbound SMS). Meanwhile, it also records other context information from the system layer such as the signal strength and broadcast parameters from the connected BS.

3) **Attack Detector** instruments the internal RIL logic to detect SMS attacks. To comprehensively cover all inbound and outbound SMS, the detection logic sits within the RIL handler (i.e., `RIL.java` [26]) that processes all RIL requests

Category	SMS Feature	Description
SMS Fields	$sms.mti$	Message type indicator
	$sms.smsc$	SMSC address
	$sms.oa$	Originating address
	$sms.da$	Destination address
	$sms.scts$	Service centre time stamp
	$sms.pid$	Protocol identifier
	$sms.dcs$	Data coding scheme
	$sms.udl$	User data length
	$sms.ud$	User data payload
	$sms.proactiveCmd$	Proactive cmd in user data payload
SMS Context	$sms.src$	Source process initiating the SMS
	$sms.ts$	Timestamp when SMS is received
	$bs.\overline{ss}$	Average signal strength in RSSI
	$bs.mcc$	Mobile country code of BS
	$bs.mnc$	Mobile network code of BS
	$bs.cid$	Cell id of BS
	$bs.lac$	Location area code of BS
	$bs.arfcn$	BS frequency band
	$bs.rat$	Radio access technology
SMS Events	$evnetCount$	Number of SMS alert events
	$\{sms_1, ..., sms_n\}$	List of SMS alert events

TABLE III: SMS features that can be utilized to design attack signatures for RILDEFENDER.

```
<Rule>    → <RuleName>: <Expr>
<Expr>    → { lvalue: <Value>
              opCode: <Op>
              condition: <Cond>
              rvalue: <Value> }
<Value>   → <Feature> | <Const> | <Expr> | List(<Expr>)
<OpCode>  → + | - | * | / | & | | | ^ | && | || | << | >>
<Cond>    → == | != | > | < | >= | <=
<Feature> → sms.mti | sms.smsc | sms.oa | sms.da | sms.ud
            | sms.pid | sms.scts |sms.dcs | sms.udl
            | sms.src | sms.ts | sms.proactiveCmd |
            | bs.ss | bs.mcc | bs.mnc | bs.cid | bs.lac
            | bs.arfcn | bs.rat | eventCount | sms_n
<Const>   → <Integer> | <Float> | <String>
```

Listing 1: High-level representation of propositional logic-based SMS attack signatures with YAML language.

and responses from an architectural point of view. To detect SMS attacks, it acquires the SMS payloads from the RIL traffic and also gathers information from the baseband monitor and context extractor to perform analysis against the criteria defined within its attack signature set.

4) **Inline Mediator** instruments the RIL logic and interacts with the RIL to provide inline control to mitigate the detected SMS attacks. Each RILDEFENDER policy is individually configurable by the user to produce alerts only or to additionally perform inline mitigation of the detected malicious SMS event.

D. RILDEFENDER *at the Application Layer*

RILDEFENDER also includes a user interface app at the application layer, which displays alerts and allows users to configure operating parameters and load custom attack signatures (see Appendix §A for details and a UI snapshot).

Real-time SMS Attack Alert. When an SMS attack is detected by RILDEFENDER and the user would like to receive real-time alerts, a broadcast intent is generated along with the attack event details (e.g., the SMS source, destination, and payload). The broadcast intent is asynchronously delivered to the RILDEFENDER app, which will display the alert using Android's notification center to show the event details.

Extensible Attack Signature. RILDEFENDER uses propositional logic to describe SMS attack signatures. To support extensible attack signature design, RILDEFENDER collects a wide range of SMS features listed in Table III. In the following, we describe them based on the below three categories:

- **SMS Fields.** The SMS fields are extracted from the payload as shown in Figure 2. For example, an SMS's PID and DCS are denoted by *sms.pid* and *sms.dcs*, respectively.
- **SMS Context.** The SMS context indicates the context information when an SMS is sent or received. For an outbound SMS message, the originating process that initiated the SMS can be tracked (denoted by $sms.src$) through IPC calls. RILDEFENDER also recorded the attributes of the currently connected BS (denoted by bs), such as its average signal strength ($bs.\overline{ss}$), cell id ($bs.cid$), and frequency band ($bs.arfcn$).
- **SMS Events.** The past SMS events can also be tracked in a list $\{sms_1, ..., sms_n\}$ in temporal order, and each event contains the corresponding SMS fields and context. Although the detection of the six types of SMS attacks (as described later in §V) does not require these features, RILDEFENDER support them for future attacks that require the reasoning of temporal SMS events.

To make the SMS attack signatures easily configurable for end-users, RILDEFENDER encodes them with the `YAML` language [49]. Note that `YAML` is a human-readable language to express structural data, and has been widely adopted in many platforms to describe network policies and configurations [51]. In Listing 1, we present the high-level representation of `YAML`-encoded signatures. To add a custom attack signature, the user needs to configure a few mandatory attributes. As in the example, `lvalue` defines the left operands in the signature, which can use any of the SMS features defined in Table III. `opcode` and `condition` define the computation (e.g., arithmetic and logical) and rational operators, respectively, and `rvalue` indicates the right operand. Further details about the extensible signatures are described in §A in Appendix.

Flexible Mitigation Policy. The mitigation policies are configured by users upon attack signatures with four different types: `Allow`, `Notify`, `Block without Notify`, and `Block and Notify`, which represent different levels of mitigation of a specific attack.

V. IMPLEMENTATION

We have implemented a proof-of-concept prototype of RILDEFENDER[1] atop the Android Open Source Project (AOSP) [22]. The baseband log monitor is implemented using the libraries from SNOOPSNITCH [17] for Qualcomm

[1]The source code of RILDEFENDER is available at https://github.com/OSUSecLab/RILDefender.

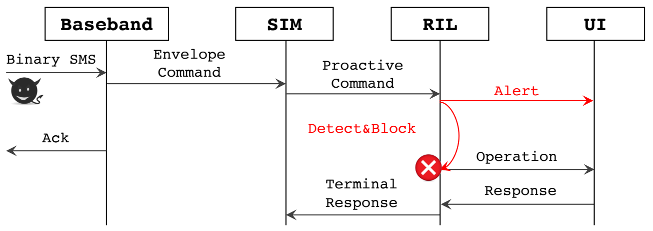


Fig. 5: Interactive binary SMS detection and mitigation.

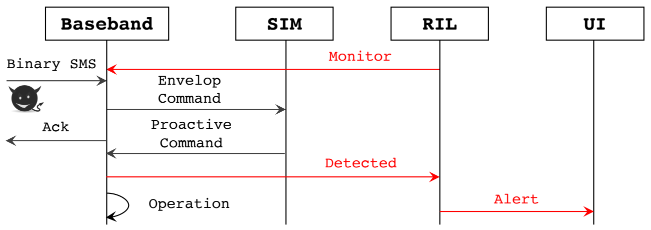


Fig. 6: Non-interactive binary SMS detection and mitigation.

baseband. Note that RILDEFENDER could be extended for non-AOSP devices due to the generality of RIL. This section details how RILDEFENDER is implemented to detect and mitigate the spectrum of SMS attacks described in Table I.

A. Binary SMS

When a binary SMS is sent to a mobile device (or UE), the baseband first parses the SMS payload and determines its type. If it is a binary SMS, the baseband relays it to the SIM for further processing. Binary SMS contains a series of proactive UICC commands [52] in the user data field, which are processed by the embedded applications in the SIM, such as the S@T browser [3] and the Wireless Internet Browser (WIB) [4]. When the SIM's internal browser is configured with the least security protection, it will blindly instruct the AP to execute the proactive UICC commands contained in the binary SMS [3], [4]. In this paper, we focus on Simjacker and WIBAttack under this category. There are other types of binary SMS targeting the UE OS, such as WBXML SMS for MMS/APN configurations, which can be exploited to trigger an integer overflow bug in Samsung Galaxy devices [53]. While not being our main focus, these forms of binary SMS can be handled similarly with other binary SMS attacks.

Within our scope, a binary SMS can be classified as either an *interactive binary SMS*, which when received will initiate user engagement to process the message, or it may be a *non-interactive binary SMS*. Correspondingly, RILDEFENDER employs different attack signatures to handle each case.

Interactive Binary SMS. This type of binary SMS invokes interactive functions by transmitting a UICC proactive command to the RIL, such as `SET_UP_CALL`, `LAUNCH_BROWSER`, `DISPLAY_TEXT`, and `PLAY_TONE` [54]. For instance, a binary SMS with `LAUNCH_BROWSER` can spawn the Internet browser with a specified URL embedded as an SMS parameter. As an interactive binary SMS requires user interaction, it must be delivered to the application layer to trigger corresponding user-interactive operations. The high-level workflow of an interactive binary SMS attack is illustrated in Figure 5. First, when the binary SMS arrives at the UE, the baseband converts it into an envelope command [52] and relays it to the SIM. The SIM then interprets it and instructs the AP to execute the specified function with a proactive UICC command. Next, the RIL is notified of the proactive command (e.g., through an unsolicited RIL command `UNSOL_STK_PROACTIVE_COMMAND` [26]), and handles it with a Card Application Toolkit (CAT) service handler. Based on the command type, the RIL further invokes the corresponding UI operation at the application layer. Finally, the response is sent back to the SIM. Based on the distinct behavioral signature, RILDEFENDER uses the following detection rule:

$$(sms.proactiveCmd \neq \emptyset) \rightarrow SMS(Binary, Interactive)$$

where RILDEFENDER detects an interactive binary SMS when the RIL is notified for a proactive command from the SIM. To mitigate the SMS, the inline mediator intercepts the proactive command at the RIL and blocks the proactive commands from being executed at the application layer.

However, there is one exceptional proactive command `SET_UP_CALL`. Interestingly, when this command is executed, the SIM will not transmit it to the RIL but instead directly instructs the baseband to initiate a voice call. Next, the RIL is notified for an initiated outgoing voice call (e.g., by an unsolicited message `CALL_STATE_CHANGED` [26]), and further spawns a voice call dialog at the dialer app. To detect this attack, RILDEFENDER needs to correctly detect if the voice call is triggered by a binary SMS rather than a normal user operation, which ensures benign operations are not mistakenly blocked. As a result, RILDEFENDER uses an intent-aware detection to monitor the user intent from the application layer context. When the attack is detected, RILDEFENDER suspends the voice call by sending a solicited RIL command `HANGUP` [26] to the baseband.

Non-interactive Binary SMS. For non-interactive binary SMS, we mainly focus on two proactive command types: `SEND_SMS` and `RUN_AT_CMD`, which instruct the UE to send out an SMS and run AT commands [2], respectively. Note that there are also other non-interactive proactive commands, such as `PROVIDE_LOCAL_INFO`, which fetches the UE's information including location, IMEI, IMSI, and so on [52]. However, they need to be chained with the previous commands such as `SEND_SMS` to achieve certain attack goals (e.g., leaking the victim's IMEI through SMS [3]).

Figure 6 illustrates the workflow. Compared with interactive binary SMS, detecting a non-interactive binary SMS attack is more challenging as the proactive command from the SIM will be relayed to the baseband instead of the RIL. Fundamentally, this is because no user interaction is needed for the operation, and thus the event is not transmitted to the AP, making it non-observable at the RIL. Therefore, RILDEFENDER utilizes the SMS traffic from the baseband monitor for detection.

Specifically, we notice that a binary SMS has distinct structural features, as it needs to have its PID field to be 0x7F (indicating USIM data download), and the DCS field to indicate a class-2 message [24]. Based on this insight, RILDEFENDER adopts the following attack signature:

$$(sms.pid = \texttt{0x7F}) \wedge (sms.dcs \ \&\ \texttt{0x3} = 2) \wedge (sms.proactiveCmd \in \{\texttt{SEND\_SMS}, \texttt{RUN\_AT\_CMD}\}) \rightarrow SMS(Binary, NonInteractive)$$

Based on the signature, RILDEFENDER parses the SMS from the baseband monitor traffic and synthesizes the structural information of the SMS payload to detect the attack. After detection, the ideal mitigation is to instruct the baseband to prevent the outcome of the message (e.g., stop the outbound SMS event). Unfortunately, having complete inline control over the baseband is not feasible from the AP's perspective (as discussed in §III-C). Thus, RILDEFENDER utilizes the alert mechanism to inform the user about the concrete threat. To achieve this, it further analyzes the proactive commands and parameters from the SMS payload and generates an alert to the RILDEFENDER app. For example, for a SEND_SMS command, the user is notified about the destination and content of the SMS sent; for a RUN_AT_CMD massage, the user is notified about the specific AT command executed.

B. Silent and Flash SMS

We combine the discussion of silent and flash SMS together as they are detected by similar signatures. According to our observation, silent SMS and flash SMS are handled by the RIL, making both detection and mitigation possible. In particular, for a silent SMS, the RIL instructs the baseband to send a delivery report. For a flash SMS, the RIL informs the upper system UI to display the message on the screen. The detection of these two SMS messages is relatively straightforward, as they all have distinct structural signatures based on the 3GPP specification [24]. Specifically, when an SMS arrives, RILDEFENDER detects the presence of silent SMS based on the following rule:

$$(sms.pid = \texttt{0x40}) \rightarrow SMS(Silent)$$

in which RILDEFENDER dissects the SMS's PDU encoding to determine whether the PID filter equals 0x40. Similarly, RILDEFENDER detects a flash SMS based on

$$(sms.dcs \ \&\ \texttt{0x3} = 0) \rightarrow SMS(Flash)$$

where RILDEFENDER performs a bit-wise and operation on the DCS field to detect class 0 SMS. As a silent or flash SMS is detected at the RIL (e.g., by an SMS handler InBoundSmsHandler), RILDEFENDER can invoke the inline mediator component to invoke the instrumented RIL logic to block further actions. In particular, RILDEFENDER blocks the SMS by stopping the delivery report generation (silent SMS) or displaying the message to the user (flash SMS).

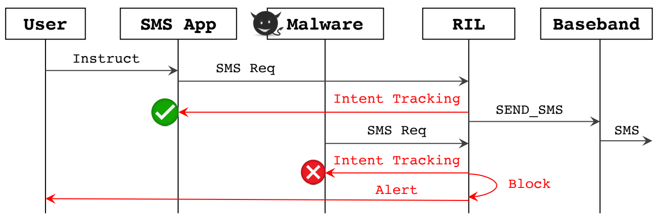


Fig. 7: Malware SMS detection and mitigation.

C. Fake Base Station SMS

A fake base station (FBS) SMS is sent from an adversary-controlled GSM base station to impersonate a legitimate source for spoofing and spamming purposes. While previous works discuss phishing, spoofing, and spamming attacks using FBS SMS [8], [9], it can also be exploited by an attacker to send out a binary, silent, or flash SMS to attack a neighboring UE, thereby providing an alternative delivery vector for attack messages when SMS filters are deployed at some SMSCs.

Since an FBS SMS payload has the same structure as a benign SMS message and its source identity can be spoofed, we cannot simply rely on structural signatures. In particular, previous works have demonstrated using cellular state parameters (e.g., the broadcast parameters and signal strength of the connected base station) to estimate whether a UE is connecting with an FBS [8], [9]. Consequently, we integrate the following heuristics-based detection rule from prior works:

$$(bs.\overline{ss} > -40dBm) \vee (Invalid(bs.params)) \rightarrow SMS(Fbs)$$

This rule examines the SMS context information from the currently connected BS. It first checks if the average signal strength is unusually high. We adopt a conservative threshold -40dBm which has been used by prior works [8], [9] to minimize false positive rate. This threshold is the approximate signal strength when a UE is placed right below a legal BS [8]. Additionally, we adopt another rule to check if the broadcast parameters (summarized as $bs.params$) from the BS are syntactically invalid or do not match with the true location, which can be checked against a well-established syntax table. Although these parameters (e.g., MMC and MNC) could be manipulated by an FBS attacker, this rule still turned out to be effective and has detected many FBSes in practice [8]. Note that in addition to text-based FBS SMS, RILDEFENDER also blocks any binary, silent, and flash SMS messages sent from an FBS, which were not demonstrated in any prior works [8], [9].

D. Malware SMS

As described in §IV-B, simply relying on structural signatures is not sufficient to detect SMS generated from malicious applications. Therefore, the key is to track the source of an outbound SMS from its context and determine whether it is triggered by the user's intention. Recall in §V-A, RILDEFENDER also uses the same strategy to detect the binary SMS attack with SET_UP_CALL command. In the following, we detail how the intent-aware detection approach works.

Intent-aware Detection. Figure 7 shows how an SMS is sent from malware and how intent-aware detection works. First, we notice that application-layer programs interact with the RIL through inter-process communication (IPC) [50], as they reside in different processes. When an outbound SMS is intercepted by RILDEFENDER, it invokes system APIs (e.g., `getCallingPid`) to locate the source of the IPC call, and acquires the process initiating the IPC (denoted by $sms.src$). If the source does not come from a trusted list (configurable through the RILDEFENDER app) such as malware, RILDEFENDER considers it suspicious and intercepts the SMS intent from being passed to the baseband. In summary, RILDEFENDER applies the following signature:

$$Suspicious(ProcessOf(sms.src)) \rightarrow SMS(Malware)$$

Similarly, to detect the `SET_UP_CALL` binary SMS attack in §V-A, RILDEFENDER detects if the voice call comes from a trusted source (e.g., the default dialer app).

One possible way to bypass this defense is to spawn the default SMS app to send the SMS, as the SMS will be sent from a valid source. For instance, Android malware can submit an intent to invoke the SMS app UI with a prepared SMS destination and content. However, this approach is unlikely to succeed, as it needs to be approved by the user on the screen, and thus the user will be fully aware of the attack. We also find that no real-world malware (discussed later in §VI-A) uses this strategy.

E. Proactive SIM SMS

An SMS may also be generated from a SIM card proactively by the embedded logic in the SIM defined by the mobile operators or supply-chain attackers. Such a procedure is relatively straightforward according to our observation. Specifically, the SIM first generates a proactive SMS request to the RIL through a unique SIM-RIL channel (e.g., `UiccSmsController` [55]). Upon receiving the request, the RIL proceeds to send out an SMS message specified in the request payload. Similar to the aforementioned malware SMS and non-interactive binary SMS, the key difference between a proactive SIM SMS and a benign SMS is that the former is directly initiated from the SIM to the RIL without user consent. As a result, we can apply the intent-aware detection approach to intercept such an SMS attempt at the RIL:

$$(sms.src = Sim) \rightarrow SMS(ProactiveSim)$$

If RILDEFENDER detects an outbound SMS generated from the SIM, it invokes the inline mediator to prevent the SMS from being delivered to the baseband.

VI. EVALUATION

We have implemented RILDEFENDER on three Android UEs with five different Android versions of the Android Open Source Project (AOSP) [22], as shown in Table IV. For Nexus 6 and Pixel XL, we target their latest possible Android versions (10.0.0 and 7.1.1). For Pixel 5, we cover three most recent Android versions (11, 12, and 13) to prove RILDEFENDER's generality. To evaluate the implementations, we answer the following three research questions:

Device	Chipset	OS Ver.	AOSP Build	LoC
Nexus 6	QCOM Snapdragon 805	7.1.1	N6F26Q	3,342
Pixel XL	QCOM Snapdragon 821	10.0.0	QP1A.190711.019	3,462
Pixel 5	QCOM Snapdragon 765G	11.0.0	RQ3A.211001.001	3,476
Pixel 5	QCOM Snapdragon 765G	12.0.0	SQ1A.220205.002	3,476
Pixel 5	QCOM Snapdragon 765G	13.0.0	TP1A.221005.002	3,482

TABLE IV: Smartphone UEs and AOSP versions that RILDEFENDER has been implemented on and evaluated.

- **RQ1.** Can RILDEFENDER effectively detect and mitigate all six types of attacks in Table I (no false negatives)?
- **RQ2.** Will RILDEFENDER incorrectly block benign cellular operations in practice (or does it manifest false positives)?
- **RQ3.** What is the overhead (power, memory, storage, and computation) introduced by RILDEFENDER?

To answer RQ1, we evaluate the robustness of RILDEFENDER by testing it with reproduced SMS attacks and real-world SMS malware (§VI-A). To answer RQ2, we test if RILDEFENDER incorrectly classifies benign cellular operations as attack behaviors with real-world trace analysis (§VI-B). To answer RQ3, we measure the overhead imposed by RILDEFENDER to the system, including the power consumption, memory, storage, and computation (§VI-C).

Experiment Setup. To evaluate RILDEFENDER, we need to replicate the SMS attacks in Table I. To this end, we used a BladeRF 2.0 xA9 software-defined radio (SDR) [11] and the open-source GSM cellular software YateBTS [10] to establish a private SMS testbed. To ensure no real-world devices and SMSCs were affected, we minimize the transmission power of our SDR and configured its subscriber list such that it could only connect with our testing devices.

A. Robustness (FN) Evaluation

The goal of our robustness evaluation is to test whether RILDEFENDER can correctly recognize and mitigate all the aforementioned SMS attacks. To this end, we evaluated RILDEFENDER with 19 reproduced SMS test cases, 4 open-source remote access trojans (RAT), and 11 real-world SMS malware samples as in Table V. Note that each type of SMS attack may require multiple test cases (e.g., different binary SMS variants). We further show the SMS payloads of the inbound SMS test cases in Table VI in Appendix, for readers of interest. Among all these cases, RILDEFENDER *was able to correctly recognize and mitigate all the SMS attacks on all the five implementations (i.e., no FN)*. In the following, we detail how our test cases are designed.

- **Binary SMS**. Recall in §V-A, binary SMS attacks are either interactive or non-interactive. Correspondingly, in Table Va we show the eight and two variants for interactive and non-interactive binary SMS, respectively, with different proactive commands. RILDEFENDER successfully detected and mitigated all of them. Note that for non-interactive binary SMS attacks, RILDEFENDER cannot block them as they are handled only by the baseband (mentioned in

Attack	SMS Payload			Cellular Network Params.			D	B
	PID	DCS	Proactive CMD	TxPower	MNC	MCC		
Binary SMS (Interactive)	0x7F	0xF6	DISPLAY_TEXT	-	-	-	✓	✓
	0x7F	0xF6	SET_UP_CALL	-	-	-	✓	✓
	0x7F	0xF6	LAUNCH_BROWSER	-	-	-	✓	✓
	0x7F	0xF6	PLAY_TONE	-	-	-	✓	✓
	0x7F	0xF6	GET_INPUT	-	-	-	✓	✓
	0x7F	0xF6	SELECT_ITEM	-	-	-	✓	✓
	0x7F	0xF6	SET_UP_MENU	-	-	-	✓	✓
	0x7F	0xF6	GET_INKEY	-	-	-	✓	✓
Binary SMS (Non-interactive)	0x7F	0xF6	SEND_SMS	-	-	-	✓	✗
	0x7F	0xF6	RUN_AT_CMD	-	-	-	✓	✗
Silent SMS	0x40	0x00	-	-	-	-	✓	✓
Flash SMS	0x00	0x18	-	-	-	-	✓	✓
FBS SMS	0x00	0x00	-	>-40dBm	MNC	MCC	✓	✓
	0x00	0x00	-	<-40dBm	MNC*	MCC*	✓	✓
	0x00	0x00	-	>-40dBm	MNC*	MCC*	✓	✓
	0x40	0x00	-	>-40dBm	MNC*	MCC*	✓	✓
	0x00	0x18	-	>-40dBm	MNC*	MCC*	✓	✓
	0x7F	0xF6	DISPLAY_TEXT	>-40dBm	MNC*	MCC*	✓	✓
Proactive SIM SMS	0x00	0x00	-	-	-	-	✓	✓

(a) SMS test cases (MNC* and MCC* stand for illegal values).

Attack	SMS Payload		Malware Type	Malware Name	D	B
	PID	DCS				
Malware SMS	0x00	0x00	Open-source RAT	AndroRAT [38]	✓	✓
	0x00	0x00	Open-source RAT	AhMyth [38]	✓	✓
	0x00	0x00	Open-source RAT	BetterAndroidRAT [38]	✓	✓
	0x00	0x00	Open-source RAT	Android Trojan [38]	✓	✓
	0x00	0x00	Real-world malware	FakeSpy [14]	✓	✓
	0x00	0x00	Real-world malware	Corona Updates [56]	✓	✓
	0x00	0x00	Real-world malware	Anubis [56]	✓	✓
	0x00	0x00	Real-world malware	Dendroid [56]	✓	✓
	0x00	0x00	Real-world malware	Ginp [56]	✓	✓
	0x00	0x00	Real-world malware	Golden Eagle [56]	✓	✓
	0x00	0x00	Real-world malware	SilkBean [56]	✓	✓
	0x00	0x00	Real-world malware	WolfRAT [56]	✓	✓
	0x00	0x00	Real-world malware	BlackRock [56]	✓	✓
	0x00	0x00	Real-world malware	Cerberus [56]	✓	✓
	0x00	0x00	Real-world malware	Mandrake [56]	✓	✓

(b) Malware SMS test cases.

TABLE V: All test cases for robustness evaluation (D: Detected, B: Blocked).

§V-A), but it was still able to detect the attacks and generate notifications as expected.

- **Silent and Flash SMS**. Silent and flash SMS both require one test case, based on their definitions [24]. Specifically, we reproduced silent and flash SMS attacks by specifying the PID and DCS fields accordingly as shown in Table Va.
- **FBS SMS**. We used different cellular network parameters to generate FBS SMS test cases. As shown in Table Va, we configured the SDR to generate attack SMS with high signal strength (>-40dBm) and illegal identity parameters (mobile country code (MCC) and mobile network code (MNC)). Based on these FBS configurations, we further generated silent, binary, and flash SMS attacks, demonstrating another feasible path for exploitation.
- **Proactive SIM SMS**. As proactive SIM SMS depends on the specific SIM logic, it is not realistic to test all available SIMs worldwide. Therefore, we tested three activated SIM cards coming from three major U.S. cellular providers, and fortunately one of them contains proactive SIM SMS logic that can be easily triggered. Specifically, a proactive SIM SMS can be generated when the SIM is inserted to the UE. Eventually, we did confirm that RILDEFENDER can effectively detect and mitigate this type of SMS.
- **Malware SMS**. We tested RILDEFENDER with two types of SMS malware: (1) open-source remote access trojans (RAT) from a public list [38] and (2) real-world SMS malware samples after 2020 from the MITRE list [56], as shown in Table Vb. Note that real-world malware samples are not always publicly available, and we successfully acquired 11 out of 15 in total, with best effort. Besides, only a few RAT tools in the list are actually open-source, compilable, and involve SMS exploits. We found 4 open-source RATs that match our requirements. Specifically, a RAT works by instructing the victim UE to open a communication channel (e.g., TCP sockets) to the malware producer, enabling remote control of the UE. Real-world malware samples are more heterogeneous and sophisticated, and they often contain obfuscated code to exfiltrate contact data and send SMS messages without user's consent for malware propagation or spoofing.

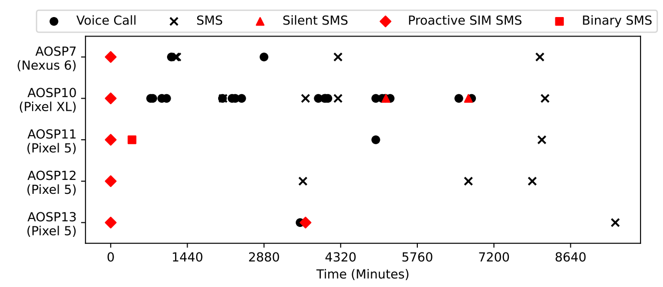


Fig. 8: Real-world SMS events in 7 days collected by RILDEFENDER on the five AOSP implementations.

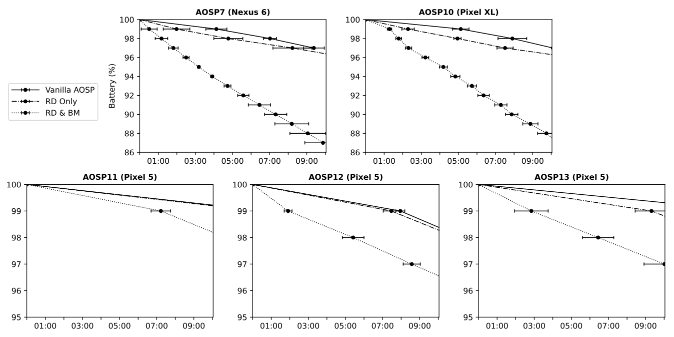
B. Correctness (FP) Evaluation

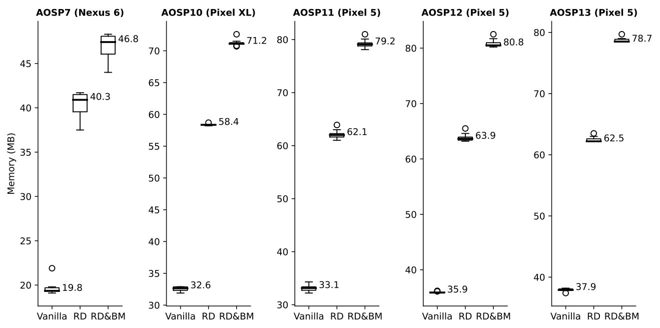
To evaluate whether RILDEFENDER may incorrectly classify benign cellular functions as attack behaviors, we evaluate the correctness of RILDEFENDER with real-world trace analysis by conducting a 7-day test campaign with RILDEFENDER on the five implementations. During the experiment, all devices were under the same cellular settings and received SMS and voice calls in the wild. We plot all the received events during the tests in Figure 8, and verified that RILDEFENDER's decisions were indeed correct, indicating *no benign events under our categorization were mistakenly blocked (i.e., no FP)*.

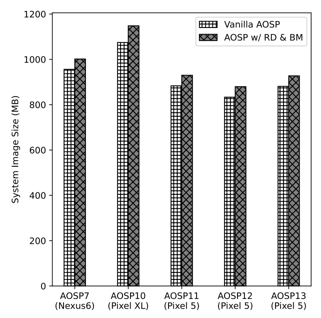
Interestingly, RILDEFENDER captured nine special SMS events in the wild, proving RILDEFENDER's capability to detect and prevent practical SMS attacks. The SMS events include two silent SMS, six proactive SIM SMS, and one binary SMS detected on the three devices. In particular, we observed that proactive SIM SMS messages were triggered right after the SIM was inserted into the UEs to start the tests. To summarize, the destination addresses of these special SMS are unusual three or four-digit private IDs that are likely owned by the carrier. As there is no public documentation about these SMS, we suspect they are related to carrier-specific functions (e.g., to configure UE settings and silently ping the device).

C. Overhead Evaluation

We evaluate four aspects of the RILDEFENDER's overhead introduced to the vanilla AOSP, namely power, memory, storage, and computation. We conduct our measurements under three experiment settings: (1) vanilla AOSP (as the baseline),


(a) Power.


(b) Memory.


(c) Storage.

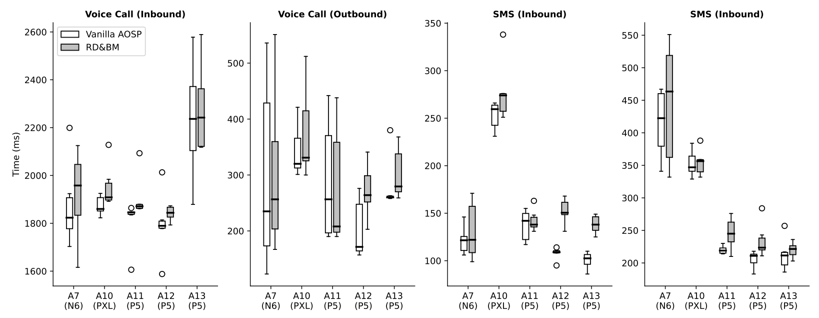

(d) Computation.

Fig. 9: Overhead of RILDEFENDER (A: AOSP, N6: Nexus 6, PXL: Pixel XL, P5: Pixel 5).

(2) AOSP with RILDEFENDER only, and (3) AOSP with RILDEFENDER and the baseband monitor (BM).

Power. We measure the power consumption overhead of each of the five implementations. The tested UEs were under the same device settings and network, with their screens, Wi-Fi and Bluetooth function switched off. For each test, we plot the average battery usage over time for 10 hours and the error bars in Figure 9a. Due to the discrepancies in battery capacity, we discuss RILDEFENDER's overhead for each Android device model. For Nexus 6 (AOSP 7.1.1), it consumed 3%, 3%, and 13% of the total battery capacity in 10 hours for setting (1), (2), and (3), respectively. Pixel XL (AOSP 10.0.0) exhibits a similar power consumption, with 2%, 3%, and 12% under each of the three settings. As the battery capacity of Pixel 5 is significantly larger, the battery percentages consumed by RILDEFENDER are much smaller. Among the three RILDEFENDER implementations (i.e., AOSP 11, 12, and 13), the average power consumption is nearly 1% for setting (1)(2) and 3% for setting (3).

To summarize, RILDEFENDER and BM impose approximately 1% battery consumption per hour on our tested devices in the worst-case scenario. However, the power consumption of RILDEFENDER is negligible, as it is nearly 0.1% per hour, 9X smaller than with RILDEFENDER and BM combined. The results indicate that the BM contributes most to the power consumption, as it needs to periodically poll and interpret baseband traffic. One possible optimization is to enlarge the baseband polling frequency discussed in Appendix §A.

Memory. In Figure 9b, we present the measurement of memory overhead imposed by RILDEFENDER under the three settings. To have a comparison baseline, we measured the memory consumption of the telephony system process for setting (1). For setting (2) and (3), we further measured the memory usage of the telephony process with the RILDEFENDER and BM services deployed. We find that RILDEFENDER and BM collectively introduce an average memory overhead of 40MB among the five implementations. Such overhead only accounts for less than 1% of the device's total RAM (i.e., 4GB, 3GB, and 8GB for Pixel XL, Nexus 6, and Pixel 5, respectively).

Storage. We measured the size of the compiled AOSP system image before and after integrating RILDEFENDER and the BM. As shown in Figure 9c, RILDEFENDER and BM introduce an additional 5.5% storage overhead on average for the five implementations. Such overhead is mainly caused by RILDEFENDER app and the baseband monitor component. The discrepancies in storage overhead are caused by the slightly different implementations, as reflected in the LoC statistics in Table IV. For instance, a handful of telephony and notification APIs [55] are unavailable in Android 7.1.1.

Computation. To evaluate the computational overhead, we measured the execution time of four cellular operations (inbound/outbound voice call and SMS) as RILDEFENDER requires to insert control logic at the RIL for them, which brings additional overhead in computation. Figure 9d plots the distribution of the computation time under different operations under settings (1) and (3) for the five implementations. Overall, RILDEFENDER imposes an acceptable overhead, with an average execution time overhead of 9.6% among all test cases. In the worst-case scenario (i.e., outbound voice call for AOSP12 on Pixel 5), the average overhead is less than 100ms, indicating a very small delay for the operation. In summary, the computation overhead introduced by RILDEFENDER and BM is considered negligible as it is nearly transparent to end-users even in the worst-case scenario (i.e., a 100ms delay).

VII. DISCUSSION: LIMITATIONS AND FUTURE WORK

Limitation. As discussed in §V-A, RILDEFENDER cannot prevent baseband-only SMS attacks (i.e., non-interactive binary SMS) and relies on a baseband monitor to detect them, which requires manual adaptation to different baseband chip vendors. As such, our current implementation focuses on Qualcomm baseband and Android platform; however, RILDEFENDER could potentially be ported to the radio abstraction layer in iOS [25]. Another limitation is the false negatives introduced by the heuristics-based FBS detection, which has been evaluated by prior work [8], [9] and is well-known in the literature [57]. In addition, while we only focus on the generic RIL for scalability concerns, smartphone manufacturers have also integrated many vendor-specific functions through vendor RIL libraries, which may be leveraged to further extend RILDEFENDER's defense capability.

Extensibility. First, although the current implementation of RILDEFENDER cannot support SMS via IMS due to the functional restriction of AOSP, we confirm that IMS-based SMS is handled in a similar way based on the RIL workflow. We provide a detailed discussion in Appendix §B. Second, RILDEFENDER can be easily extended for legacy or future Android OS versions with minimal manual effort, due to the stable architecture of RIL and the standard Android APIs used in the implementations. For instance, as shown in Table IV, RILDEFENDER's implementation on AOSP11 is directly compatible with AOSP12 without any changes. Third, while our implementation focuses on the six types of SMS attacks, RILDEFENDER's generic design allows it to be extensible for new attack signatures and policies at a very low cost.

Law-Enforcement Tracking. Silent SMS has been used for tracking in some countries by law enforcement [5]. RILDEFENDER does not attempt to distinguish between such SMS from attack SMS. However, balancing individual privacy and providing law-enforcement access to data has been a vexing and controversial issue [58], as exemplified by research in cryptography and anonymity networks. Therefore, RILDEFENDER is designed with a particular focus on user privacy, and extending it to support law enforcement (and other potential benign use cases) is future work.

VIII. RELATED WORK

Cellular Security. In early GSM (2G) networks, fake base station (a.k.a., an IMSI-Catcher) was a major threat due to the absence of mutual authentication between the UE and BS [34]. This weakness was addressed in the succeeding 3G and 4G network standards. While the 4G LTE network has become the most dominant, over the past decade, it remains vulnerable to many attacks, such as network message attacks [35], [59], [60], passive sniffing [59], downgrading [36], spoofing [32], [33], [61]–[63], signal injection [64], and eavesdropping [65]. Such widespread vulnerabilities, which compromised the security and privacy of users, stimulated the development of numerous security analysis frameworks and defenses that attempt to mitigate these attacks. In particular, PROCHECKER [66] and DIKEUE [67] automatically analyzed LTE protocol implementations for specification-deviated flaws; PHOENIX [46] used a signature-based approach to detect network intrusion attacks and Android-based apps were developed to detect IMSI-Catcher attacks [16]–[18]. More recently, the latest 5G network protocol has been shown to be vulnerable to many attacks, including SIGUNDER [68], SUCI-Catcher [69], and various availability and privacy attacks [36], [70], [71]. In response, new defenses have been proposed to provide security- or privacy-enhanced cellular telematics [72] and protocols (e.g., privacy-preserving authentication and key management (AKA) [73], [74] as well as broadcast authentication [75]).

In particular, for SMS threats, Mulliner et al. performed SMS fuzzing using a GSM BS and triggered bugs that could prohibit a smartphone's communication [76]. In the 4G era, it was shown that the IMS-based SMS service is vulnerable to SMS abuse, spoofing, DoS, and spamming [23]. In addition, there are systems that detect spamming SMS messages [77], [78] and measurement studies that characterize the SMS spamming ecosystem and analyze the root causes [9], [79]. As mentioned in §III-C, there are four existing defenses deployed within the RAN or at the app-layer of UE to detect SMS attacks [8], [9], [16], [17]. In contrast, RILDEFENDER is the first RIL-based mitigating defense against a wide range of SMS threats. While RILANALYZER [48] seems closely relevant, it is in fact an app-layer 3G traffic monitor relying on baseband log parsing.

Baseband Security. Only a handful of studies have attempted to analyze security flaws in UE baseband implementations. In 2012, Weinmann [80] discovered a memory corruption vulnerability in a baseband that allows remote exploitation. Later in 2015, Golde et. al. [29] made the first attempt to reverse engineer Samsung's Shannon baseband, and also uncovered a few memory corruption bugs. Most recently, BASESPEC [81] advanced this direction by statically analyzing 18 baseband images from 9 vendors and uncovered many implementation flaws, and ARISTOTELES [25] analyzed the baseband interface protocol in iOS to uncover a new attack surface. Additionally, many new exploits were found targeting other baseband chips, such as Huawei [82]. It was also shown that baseband emulation is possible [83], [84] to facilitate dynamic firmware analysis. However, due to the closed-source and heterogeneous nature of commercial baseband software, conducting security analyses remains a significant technical challenge.

IX. Conclusion

We presented RilDefender, the first inline defensive service integrated into the radio interface layer of Android smartphone UEs. RilDefender is distinguished from existing UE-centric defenses in that its approach provides both detection and effective inline defense capability to thwart incoming SMS threats, and is broadly applicable across heterogeneous smartphone OS. As a proof-of-concept, we have implemented a prototype of RilDefender on three UEs with five different Android versions of AOSP. We have evaluated RilDefender by using 19 reproduced SMS attack cases and 11 real-world SMS malware samples. Our evaluation shows that it can detect *all* six types of SMS attacks across four different adversary models, and automatically mitigate all but one in real-time. In the worst-case scenario, RilDefender imposed an hourly battery consumption overhead of 1% and 100 ms delay for cellular operations on our tested devices.

Acknowledgment

We thank the anonymous reviewers for their constructive feedback, as well as Keith Skinner and Martin Fong for their contribution to RilDefender. This research was supported in part by NSF awards 2112471 and 2226443. Any opinions, findings, conclusions, or recommendations expressed are those of the authors and not necessarily of the NSF.

References

[1] "Proactive sims," https://deepsec.net/docs/Slides/2021/Proactive_SIMs_David_Burgess.pdf, November 2021.

[2] D. J. Tian, G. Hernandez, J. I. Choi, V. Frost, C. Raules, P. Traynor, H. Vijayakumar, L. Harrison, A. Rahmati, M. Grace *et al.*, "Attention spanned: Comprehensive vulnerability analysis of {AT} commands within the android ecosystem," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 273–290.

[3] "Simjacker," https://simjacker.com.

[4] "Wibattack – sim card browser bug let hackers take control over mobile phones to make calls & sms," https://gbhackers.com/wibattack-sim/amp.

[5] M. Monroy, "Significantly more "silent sms" with german police authorities," https://digit.site36.net/2019/02/25/significantly-more-silent-sms-with-german-police-authorities.

[6] "Google nexus phones are vulnerable to attack via flash sms messages," https://www.computerworld.com/article/2486382/google-nexus-phones-are-vulnerable-to-attack-via-flash-sms-messages.html.

[7] "ios 8.2 stops attackers being able to restart your iphone with a malicious flash sms," https://grahamcluley.com/ios-8-2-stops-attackers-being-able-to-restart-your-iphone-with-a-malicious-flash-sms/.

[8] 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.

[9] Y. Zhang, B. Liu, C. Lu, Z. Li, H. Duan, S. Hao, M. Liu, Y. Liu, D. Wang, and Q. Li, "Lies in the air: Characterizing fake-base-station spam ecosystem in china," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 521–534.

[10] "Yatebts - lte & gsm mobile network compoents for mno & mvno," https://yatebts.com.

[11] "bladerf," https://www.nuand.com/bladerf-1.

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

[13] "Over 10m android phones infected with grifthorse malware," https://www.pcmag.com/news/over-10m-android-phones-infected-with-grifthorse-malware.

[14] "Fakespy android malware spread via 'postal-service' apps," https://threatpost.com/fakespy-android-malware-spread-via-postal-service-apps/157102/.

[15] "Mobile malware: Tanglebot untangled," https://www.proofpoint.com/us/blog/threat-insight/mobile-malware-tanglebot-untangled.

[16] "Android imsi-catcher detector," https://cellularprivacy.github.io/Android-IMSI-Catcher-Detector.

[17] "Snoopsnitch," https://opensource.srlabs.de/projects/snoopsnitch.

[18] "Stingwatch," https://github.com/marvinmarnold/stingwatch.

[19] 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.

[20] 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.

[21] 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.

[22] "Android open source project," https://source.android.com/.

[23] G.-H. Tu, C.-Y. Li, C. Peng, Y. Li, and S. Lu, "New security threats caused by ims-based sms service in 4g lte networks," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1118–1130.

[24] "Etsi ts 123 040 v12.2.0," https://www.etsi.org/deliver/etsi_ts/123000_123099/123040/12.02.00_60/ts_123040v120200p.pdf.

[25] T. Kröll, S. Kleber, F. Kargl, M. Hollick, and J. Classen, "Aristoteles–dissecting apple's baseband interface," in *European Symposium on Research in Computer Security*. Springer, 2021, pp. 133–151.

[26] "Ril refactoring | android open source project," https://source.android.com/devices/tech/connect/ril.

[27] D. Berard and V. Fargues, "How to design a baseband debugger," in *Information and Communication Technology Security Symposium (SSTIC), Rennes, France*, 2020.

[28] C. Bruns, "Modification of lte firmwares on smartphones," Master's thesis, Technische Universität, 2021.

[29] N. Golde and D. Komaromy, "Breaking band: reverse engineering and exploiting the shannon baseband," *Recon 2016, Recon*, 2016.

[30] "Zte mf833v lte/wcdma/gsm usb modem," https://usermanual.wiki/ZTE/MF833V.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] A. Shaik, R. Borgaonkar, S. Park, and J.-P. Seifert, "New vulnerabilities in 4g and 5g cellular access network protocols: exposing device capabilities," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, 2019, pp. 221–231.

[37] Y. Zeng, K. G. Shin, and X. Hu, "Design of sms commanded-and-controlled and p2p-structured mobile botnets," in *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, 2012, pp. 137–148.

[38] "Remote access tool trojan list - android," https://github.com/wishihab/Android-RATList.

[39] N. J. Croft and M. S. Olivier, "A silent sms denial of service (dos) attack," *Information and Computer Security Architectures (ICSA) Research Group South Africa*, vol. 29, 2007.

[40] 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) Symposium*, 2019.

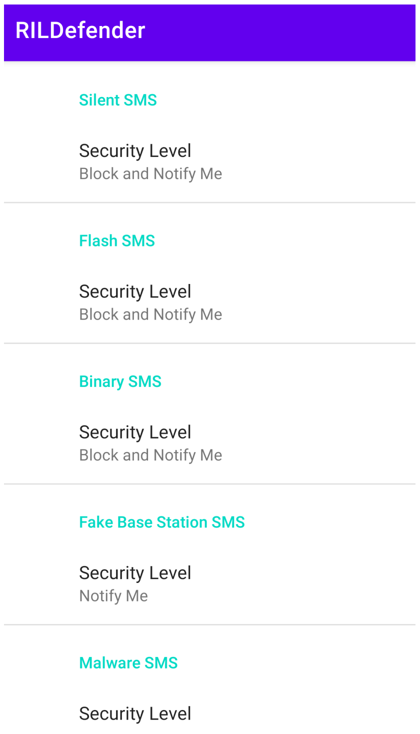
[41] "Proactive sim: When your sim does things on its own without your knowledge," https://washingtonindependent.com/proactive-sim/, December 2021.
[42] P. K. Nakarmi, M. A. Ersoy, E. U. Soykan, and K. Norrman, "Murat: Multi-rat false base station detector," *arXiv preprint arXiv:2102.08780*, 2021.
[43] 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.
[44] M. Payer, A. Barresi, and T. R. Gross, "Fine-grained control-flow integrity through binary hardening," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2015, pp. 144–164.
[45] "Qualcomm reveals demo to prevent smartphones from being hacked by connecting to fake base stations," https://iphonewired.com/news/259588/.
[46] M. Echeverria, Z. Ahmed, B. Wang, M. F. Arif, S. R. Hussain, and O. Chowdhury, "Phoenix: Device-centric cellular network protocol monitoring using runtime verification," in *Network and Distributed Systems Security (NDSS) Symposium*, 2021.
[47] 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.
[48] N. Vallina-Rodriguez, A. Auçinas, M. Almeida, Y. Grunenberger, K. Papagiannaki, and J. Crowcroft, "Rilanalyzer: a comprehensive 3g monitor on your phone," in *Proceedings of the 2013 conference on Internet measurement conference*, 2013, pp. 257–264.
[49] "The official yaml web site," https://yaml.org/.
[50] "Android interface definition language (aidl)," https://developer.android.com/guide/components/aidl.
[51] "What is yaml?" https://www.redhat.com/en/topics/automation/what-is-yaml.
[52] "Ts 131 111 v15.4.0," https://www.etsi.org/deliver/etsi_TS/131100_131199/131111/15.04.00_60/ts_131111v150400p.pdf.
[53] "Cve-2016-7990," https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-7990.
[54] "Etsi ts 102 223 v12.1.0," https://www.etsi.org/deliver/etsi_ts/102200_102299/102223/12.01.00_60/ts_102223v120100p.pdf.
[55] "android.telephony | android developers," https://developer.android.com/reference/android/telephony/package-summary.
[56] "Sms control, technique t1582 - mobile," https://attack.mitre.org/techniques/T1582/.
[57] R. Borgaonkar, A. Martin, S. Park, A. Shaik, and J.-P. Seifert, "White-stingray: evaluating imsi catchers detection applications." USENIX, 2017.
[58] S. Savage, "Lawful device access without mass surveillance risk: A technical design discussion," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1761–1774.
[59] A. Shaik, R. Borgaonkar, N. Asokan, V. Niemi, and J.-P. Seifert, "Practical attacks against privacy and availability in 4g/lte mobile communication systems," in *Network and Distributed Systems Security (NDSS) Symposium*, 2016.
[60] 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.
[61] 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.
[62] D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper, "Imp4gt: Impersonation attacks in 4g networks." in *NDSS*, 2020.
[63] M. Chlosta, D. Rupprecht, T. Holz, and C. Pöpper, "Lte security disabled: misconfiguration in commercial networks," in *Proceedings of the 12th conference on security and privacy in wireless and mobile networks*, 2019, pp. 261–266.
[64] 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.
[65] D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper, "Call me maybe: Eavesdropping encrypted {LTE} calls with revolte," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 73–88.
[66] I. Karim, S. R. Hussain, and E. Bertino, "Prochecker: An automated security and privacy analysis framework for 4g lte protocol implementations," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 773–785.
[67] 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.
[68] 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.
[69] M. Chlosta, D. Rupprecht, C. Pöpper, and T. Holz, "5g suci-catchers: still catching them all?" in *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2021, pp. 359–364.
[70] 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.
[71] 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.
[72] 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.
[73] 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.
[74] R. Borgaonkar, L. Hirschi, S. Park, and A. Shaik, "New privacy threat on 3g, 4g, and upcoming 5g aka protocols," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, pp. 108–127, 2019.
[75] 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.
[76] C. Mulliner, N. Golde, and J.-P. Seifert, "Sms of death: From analyzing to attacking mobile phones on a large scale." in *USENIX Security Symposium*, vol. 168. San Francisco, CA, 2011.
[77] N. Jiang, Y. Jin, A. Skudlark, and Z.-L. Zhang, "Greystar: Fast and accurate detection of {SMS} spam numbers in large cellular networks using gray phone space," in *22nd USENIX Security Symposium (USENIX Security 13)*, 2013, pp. 1–16.
[78] B. Reaves, L. Blue, D. Tian, P. Traynor, and K. R. Butler, "Detecting sms spam in the age of legitimate bulk messaging," in *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, 2016, pp. 165–170.
[79] B. Reaves, N. Scaife, D. Tian, L. Blue, P. Traynor, and K. R. Butler, "Sending out an sms: Characterizing the security of the sms ecosystem with public gateways," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 339–356.
[80] R.-P. Weinmann, "Baseband attacks: Remote exploitation of memory corruptions in cellular protocol stacks." in *WOOT*, 2012, pp. 12–21.
[81] E. Kim, D. Kim, C. Park, I. Yun, and Y. Kim, "Basespec: Comparative analysis of baseband software and cellular specifications for l3 protocols," in *Symposium on Network and Distributed System Security (NDSS)(San Diego, CA, USA). ISOC*, 2021.
[82] M. Grassi, M. Liu, and T. Xie, "Exploitation of a modern smartphone baseband," *BlackHat US*, 2018.
[83] G. Hernandez, M. Muench, T. Tucker, H. Serle, W. Zhu, P. Traynor, and K. Butler, "Emulating samsung's baseband for security testing," *BlackHat USA*, 2020.
[84] D. Maier, L. Seidel, and S. Park, "Basesafe: Baseband sanitized fuzzing through emulation," in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2020, pp. 122–132.

Appendix

A. *Detailed Design and Implementation of* RILDefender

We now describe the design and implementation details of RILDefender that are not covered in §IV.

User Interface. Figure 10 shows the RILDefender app UI as mentioned in §IV. The user interface is developed as a system app integrated in the AOSP system image. The front end of RILDefender app is mainly for policy configurations, browsing past SMS alert events, and configuring SMS whitelist. Users can interact with the app to configure the mitigation policies for each SMS attack. For instance, the user may choose to get notifications for all silent SMS while blocking all incoming binary SMS. The notification service runs in background and keeps listening to the RIL and baseband monitor events for the occurrence of attacks, which consumes very little resource for the overall system.


RILDefender

Silent SMS
Security Level
Block and Notify Me

Flash SMS
Security Level
Block and Notify Me

Binary SMS
Security Level
Block and Notify Me

Fake Base Station SMS
Security Level
Notify Me

Malware SMS
Security Level

Fig. 10: User interface of the RILDefender app.

Extensible YAML Attack Signature. We further describe how new YAML attack signatures can be designed for new SMS attacks and how they are converted into corresponding logic at the RIL. In addition to the six types of SMS attacks in Table I, RILDefender can be extended with new attack signatures by utilizing the SMS features in Table III. For instance, a user can automatically block SMS from a certain user by setting up a signature using the SMS destination ($sms.da$) from the feature list. The user can also prevent silent SMS DoS attacks by using temporal SMS events in the past. When new attack signatures are configured, they are first broadcast to the system layer of RILDefender and stored in persistent memory. Once there is an SMS event, RILDefender will iterate all the stored SMS detection rules to match the event against them. For each of the rules, RILDefender has a YAML signature parser that automatically converts the human-written rules into corresponding program logic, which is computed based on the SMS event features. Finally, the parser outputs Boolean values to indicate the matched rules for the SMS event.

Baseband Monitor. As mentioned in §IV, RILDefender integrates a baseband monitor to capture non-interactive binary SMS attacks as they are not visible to the RIL. While as an implementation choice, we choose the supportive libraries from SnoopSnitch [17] as building blocks to realize the baseband monitor (BM), there are many alternatives such as the well-implemented libraries from SCAT [47] and MobileInsight [43], which provide a wide range of support for different baseband chipsets. However, as the BM needs to constantly poll the baseband for its traffic, it brings additional overhead to the overall power consumption, as discussed in §VI-C. To minimize such overhead, the implementation of RILDefender has tailored the BM component to reserve only the logic for non-interactive binary SMS detection. Additionally, RILDefender provides flexible configurations to adjust the baseband traffic polling frequency, and the user is also allowed to disable the BM component if non-interactive binary SMS detection is not necessary, which significantly reduces the power consumption as well. While reducing the baseband traffic polling frequency reduces power consumption, it also adds to the delay of the attack notification. Our preliminary results show that adding a 1-second delay to the BM component can save 2%-3% of battery consumption for 5 hours.

B. *SMS Workflow within the Radio Interface Layer*

In §II-B, we briefly describe the high-level architecture of the radio interface layer. In this section, we explain in detail the internal workflow of the RIL, with a particular focus on how SMS (inbound and outbound) is handled. In particular, we use RIL in Android for all the illustrations.

Inbound SMS. Figure 11 abstracts the workflow of inbound SMS processing at the RIL [26]. When an SMS arrives at the UE, it is delivered from the baseband to the `RILD` and handled by the `RILJ` as well as `InboundSmsHandler` sequentially. Next, based on UE type, the RIL selectively invokes `GsmInboundSmsHandler` or `CdmaInboundSmsHandler` to process the SMS message, and the handler logic is actually quite similar, which relies on a `MessageDispatcher` component. After parsing the PDU bytes of the SMS, different dispatchers will be called for the SMS. For normal text SMS, the RIL uses `NormalMessageDispatcher`; For special types of SMS including silent, flash, and binary SMS mentioned in the paper, the RIL calls `RadioSpecificMessageDispatcher`. When the dispatcher finishes, the control is back to the `InboundSmsHandler`, which broadcasts the SMS to the

SMS app at the application layer. The workflow clearly shows that the RIL has corresponding processing logic for different types of SMS, which allows us to implement RILDEFENDER to detect and block the SMS attacks.

Outbound SMS. Figure 12 illustrates the workflow of outbound SMS [26]. In summary, the RIL uses different dispatchers to handle outbound SMS, including `ImsSmsDispatcher`, `GsmSmsDispatcher`, and `GsmSmsDispatcher`. There are three different RIL commands corresponding to different SMS types, which allows the RIL to inform the baseband to send GSM, CDMA, or IMS SMS. This workflow facilitates our intent-aware detection approach, as the RIL has a direct communication path with the SMS app at the application layer, allowing us to implement an intent detector to monitor the apps at the upper layer.

IMS SMS. Based on the above illustrations, both inbound and outbound IMS-based SMS are handled within the RIL. We further confirm our hypothesis by dynamically analyzing the internal RIL traffic on our two tested Android devices (both are installed with official factory images and support IMS SMS). Specifically, we observed that inbound IMS SMS is handled by the `GsmSmsInboundSmsHandler`, which is almost the same as traditional SMS handling logic. An outbound SMS is sent by using a `RIL_REQUEST_IMS_SEND_SMS` request to the baseband. In conclusion, both the inbound and outbound IMS SMS logic is visible to the RIL, and RILDEFENDER can be extended to support IMS SMS as well, which is necessary under 4G and 5G networks.

C. *SMS Payload Tested*

In Table VI, we present the raw PDU payload of the SMS test cases shown in Table V. These SMS payloads are of SMS-Deliver types as they are transmitted from our experimental SDR (equivalent to an SMSC) to the tested devices. Note that we have anonymized the destinations of the SMS payloads to zero values. As shown in the table, we present three types of SMS including silent SMS, flash SMS, and binary SMS. For binary SMS, we present all the variant attack payloads based on the proactive commands. For instance, `0x13` and `0x34` are the proactive commands for `SEND_SMS` and `RUN_AT_CMD`, respectively [52]. For the remaining types in Table V, FBS SMS messages have the identical payload as the aforementioned SMS types. Malware and proactive SIM SMS are outbound SMS triggered by an application or the SIM, and thus their contents vary based on the SIM and application logic.

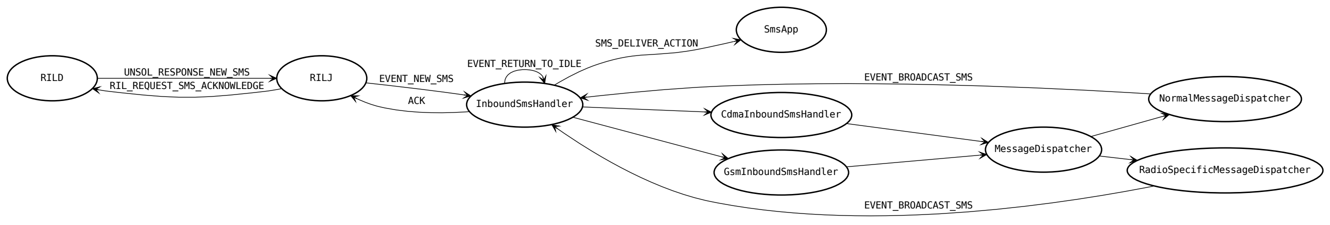


Fig. 11: Internal RIL workflow: inbound SMS.

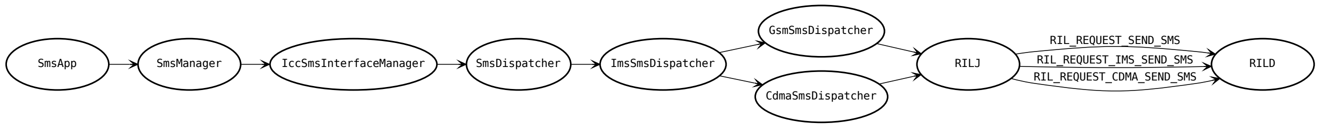


Fig. 12: Internal RIL workflow: outbound SMS.

Case	Type	SMS Payload
1	Silent	010004802143f500150405802143f54000000000000000000005e8329bfd06
2	Flash	010004802143f500160405802143f50010000000000000000006e8329bfd06f6
3	Binary	010004802143f500484405802143f57ff6000000000000003802700000330D000060605053480000000000000422301210 20744382E3130353105160604313035312D0C10038306079121436587009F02B00 (SET_UP_CALL)
4	Binary	010004802143f500484405802143f57ff6000000000000003802700000330D000060605053480000000000000422301210 20744382E3130353105160604313035312D0C2103028D07912143658709F02B00 (DISPLAY_TEXT)
5	Binary	010004802143f500484405802143f57ff6000000000000003802700000330D000060605053480000000000000422301210 20744382E3130353105160604313035312D0C1503020607912143658709F02B00 (LAUNCH_BROWSER)
6	Binary	010004802143f500504405802143f57ff600000000000000040027000003b0D0000606050534800000000000000422b01210 20744382E31303531051e0604313035312D161301828B1131000B911000000000000000AA03C8771A (SEND_SMS)
7	Binary	010004802143f5005f4405802143f57ff6000000000000004f027000004a0D0000606050534800000000000000423a01210 20744382E31303531052d0604313035312D25340082850F56697369626C65206D657373616765280F 4154443132333435363738393b5c72 (RUN_AT_CMD)
8	Binary	010004802143f500524405802143f57ff6000000000000004202700000340D0000606050534800000000000000422d01210 20744382E31303531052006043130353312D18240082028102 8F07AB4974656D20318F07AC4974656D2032 (SELECT_ITEM)
9	Binary	010004802143f500494405802143f57ff6000000000000003902700000340D0000606050534800000000000000422401210 20744382E3130353105170604313035312Df2203028D08912143658709F0102B00 (GET_INKEY)
10	Binary	010004802143f500494405802143f57ff6000000000000003902700000340D0000606050534800000000000000422401210 20744382E3130353105170604313035312Df2003028D08912143658709F0102B00 (PLAY_TONE)
11	Binary	010004802143f500494405802143f57ff6000000000000003902700000340D0000606050534800000000000000422401210 20744382E3130353105170604313035312Df2303028D08912143658709F0102B00 (GET_INPUT)
12	Binary	010004802143f500494405802143f57ff6000000000000003902700000340D0000606050534800000000000000422401210 20744382E3130353105170604313035312Df2503028D08912143658709F0102B00 (SET_UP_MENU)

TABLE VI: Raw PDU payload of the SMS test cases in evaluation (SMS destination anonymized).