Hindawi
Security and Communication Networks
Volume 2021, Article ID 8847803, 13 pages
https://doi.org/10.1155/2021/8847803

Research Article

YAICD: Yet Another IMSI Catcher Detector in GSM

Parimah Ziayi,[1] Seyed Mostafa Farmanbar,[2] and Mohsen Rezvani[2]

[1]*Science and Research Branch, Islamic Azad University, Tehran, Iran*
[2]*Faculty of Computer Engineering, Shahrood University of Technology, Shahrood, Iran*

Correspondence should be addressed to Mohsen Rezvani; mrezvani@shahroodut.ac.ir

Received 27 June 2020; Revised 20 December 2020; Accepted 13 January 2021; Published 29 January 2021

Academic Editor: Stelvio Cimato

In GSM, the network is not authenticated which allows for man-in-the-middle (MITM) attacks. Attackers can track traffic and trace users of cellular networks by creating a rogue base transceiver station (BTS). Such a defect in addition to the need for backward compatibility of mobile networks makes all GSM, UMTS, and LTE networks susceptible to MITMs. These attacks are conducted using IMSI-Catchers (ICs). Most of the solutions proposed for detecting ICs in the literature are based on using specific mobile devices with root access. Also, they cannot identify ICs to which users are not connected. In this paper, we propose an approach called YAICD for detecting ICs in the GSM network. YAICD consists of a sensor that can be installed on Android mobile devices. It detects ICs by extracting 15 parameters from signals received from BTSs. We also established a lab-scale testbed to evaluate YAICD for various detection parameters and for comparing it against existing solutions in the literature. The experimental results show that YAICD not only successfully detects ICs using the parameters but also identifies ICs to which users are not yet connected to the network.

1. Introduction

Cell sites known as base transceiver stations (BTSs) constitute the underlying infrastructure of today's cellular networks. They connect end-users of mobile devices to a wider network (for example, a cellular carrier network and the Internet) by sending audio streams, short messages, and IP data packets [1]. Unfortunately, vulnerabilities of the Global System for Mobile Communications (GSM) make it possible to create fake Base transceiver station (FBTS). In fact, the GSM standard does not require cellular devices to confirm the BTS [2]. Currently, millions of BTS simultaneously support GSM, universal mobile telecommunications systems (UMTS), and Long-Term Evolution (LTE) networks and serve billions of mobile phones. Also, most mobile devices nowadays support GSM, UMTS, and LTE networks, and in the presence of several networks, they tend to choose the one with the highest signal strength [3]. This allows attackers to launch their own GSM cell sites with high signal strengths. By sending malicious signals, attackers can even downgrade 3G/4G GSM-compatible mobile phones to the GSM mode. Such defects of GSM networks, along with the need for backward compatibility of cellular networks, expose all GSM, UMTS, and LTE networks to man-in-the-middle attacks (MITMs) which make use of FBTSs [4].

In cellular network architecture, mobile devices are always assigned an International Mobile Subscriber Identity (IMSI) and an International Mobile Equipment Identity (IMEI) [5]. During a network connection process, these IDs are exchanged between the mobile phone and BTSs over the air. In order to prevent disclosure of a particular user's IMSI, a Temporary Mobile Subscriber Identity (TMSI) is applied to the security architecture of the mobile phone [6]. However, in two cases before a successful validation, IMSI is transmitted to the network without encryption: (1) when a device is turned on and (2) when the network fails to detect the TMSI. Attackers exploit this GSM weakness in order to introduce themselves to mobile users as a real network. Thus, they acquire the ID of cellular network users. At first, attackers were only able to receive IMSI and IMEI of neighboring mobile users. Hence, they were called IMSI catchers (ICs) [7]. The more advanced models of this malware can launch MITMs in the traffic exchanged between an authentic BTS and a mobile phone [8, 9]. In this way, when an adversary with a "fake" mobile tower acts between the target mobile phone and the service provider's real towers in order to listen and manipulate the communication of a targeted mobile phone, it is considered a MITM attack. It is to be noted that disclosing such information may be further used to achieve confidentiality and privacy violations [10–12].

Many mobile phone operators are planning to abolish GSM. However, it will take years to upgrade cell sites and to do away with old mobile phones [13]. Offenders and criminals currently use ICs to attack users directly. Today, ICs have been seen in the United States, China, India, Russia, Israel, and the United Kingdom [1]. Recently, the widespread use of ICs has been reported at airports and embassies [14, 15]. In the past, FBTSs such as StingRays were quite expensive (ranging from \$68,000 to \$134,000), and they were sold only to judicial and state authorities. But in the last decade, new technical devices designed for the same malicious purposes have become popular and cheap. By spending about \$1,500, it is now possible to design a simple IC that includes a Software-Defined Radio (SDR) and two directional antennas, and it only requires a laptop to run the OpenBTS for free [14, 16].

Given the rise in IC attacks, several solutions have been recently advanced for their identification. In 2017, the FBS-Radar project was developed in collaboration with Chinese mobile network operators to uncover ICs [1]. Meanwhile, FBS-Radar detects only ICs that send spam or unwanted SMS. In the same year, security researchers at the University of Washington devised a system called SeaGlass to report ICs [17]. It collects the data related to BTS signals and transmits them to the server by several sensors embedded in the car. The server-side application subsequently performs necessary analyses. In 2014, SRLabs introduced two client-side tools, called SnoopSnitch and CatcherCatcher, for detecting ICs [18, 19]. The main limitation of these solutions is that they can only be used on specific mobile devices and require root privilege. Moreover, they cannot identify ICs that users are not yet connected to the network. The Android IMSI-Catcher Detector (AIMSICD) project reports ICs by checking BTS signals. It is capable of identifying even ICs which users have not yet connected to the network. However, AIMSICD is still in the alpha phase, and its effectiveness has not been evaluated in practice [20].

In this paper, we propose yet another IC detector, called YAICD, a client-side solution to effectively detect ICs while overcoming the above-mentioned constraints. This solution can be launched on Android mobile devices, regardless of whether they have root privilege or not. In rooted mobile devices, as in SnoopSnitch, a Qualcomm chip is used to collect signal data received from BTSs. In the case of nonrooted Android mobile devices, BTS signals are received using a library called Telephony Manager (https://developer.android.com/reference/android/telephony/TelephonyManager). The Android Telephony Manager library provides information about the telephony services such as subscriber id, sim serial number, phone network type. Telephony Manager is compatible with all phone models. The YAICD sensor uses the signals received from BTSs to detect ICs by extracting fifteen parameters in rooted devices and six parameters in nonrooted devices. The detection process in the YAICD is accomplished based on a threshold value and considering the sum of individual parameters. The YAICD sensor alarms the user when an IC is detected. In addition to detecting ICs to which the user is connected, our solution can identify the neighboring ICs and alerts users before they might decide to get connected to them.

We also created a laboratory-scale testbed at the Shahrood University of Technology to evaluate the proposed method along with existing approaches proposed in the literature. In this testbed, an FBTS is launched using an SDR device and a laptop to run the OpenBTS (http://openbts.org/) software. Three rooted mobile phones (Huawei Y7, LG nexus 5x, and Huawei G620) and three nonrooted phones (Samsung J7, Sony z2, and LG G4) were used to collect the signal data received from the BTSs and the designed FBTS. To protect the privacy of users, only the test mobile phones were permitted to connect to the FBTS. By separately implementing each of the fifteen parameters for identifying ICs in the laboratory environment, we conducted several experiments to compare the performance of our approach against the state-of-the-art approach in the literature. Only the YAIC sensor detects ICs successfully. It is also capable of identifying the neighboring ICs to which users are not yet connected.

YAICD is a client-side solution to effectively detect ICs while overcoming the abovementioned constraints. This solution can be launched on Android mobile devices, regardless of whether they have root privilege or not.

The rest of this article is organized as follows. Section 2 discusses some background and related work. Section 3 elaborates our proposed method. Section 4 discusses the results and compares them with those of previous studies. Finally, Section 5 outlines a few conclusions.

2. Background and Literature Review

In this section, we will first introduce the GSM network. Then, the way ICs operate is described. Subsequently, different parameters used to identify ICs are discussed. Finally, previous works on IC detection are assessed. All the acronyms used in this paper are listed in Table 1.

2.1. *The Global System for Mobile Communications (GSM).*

Today's mobile phones are based on cellular network architecture. The structure of a cellular network is shown in Figure 1.

GSM, a 2G standard, is geographically divided into different location areas (LAs). This classification is performed by the operators. Each LA comprises a number of cells, and each cell is controlled by a BTS [2]. Based on its capacity, power, and radio range, each BTS can serve multiple users. A mobile phone with a SIM card, hereafter simply "mobile phone," is known as a mobile station (MS). The BTS periodically broadcasts network information as System Information Block (SIB) messages. In the GSM, the mobile phone identifies its network using SIB messages based on mobile country codes (MCCs) and mobile network

Table 1: List of acronyms used in this paper.

Field	Description
MITM	Man-In-The-Middle
BTS	Base Transceiver Station
IC	IMSI-Catcher
FBTS	Fake Base Transceiver Stations
GSM	Global System for Mobile
UMTS	Location Update Timer
LTE	Long-Term Evolution
ARFCN	ARFCN of Cell
IMSI	IMSI digits
IMEI	IMEI digits
TMSI	Temporary Mobile Subscriber Identity
MS	Mobile Station
SIB	System Information Block
LA	Location Area
MCC	Mobile Country Code
MNC	Mobile Network Code
LAU	Location Area Updating
LAC	Location Area Code
ARFCN	Absolute Radio-Frequency Channel
AIMSICD	Android IMSI-Catcher Detector
ICD	IMSI catcher Detector

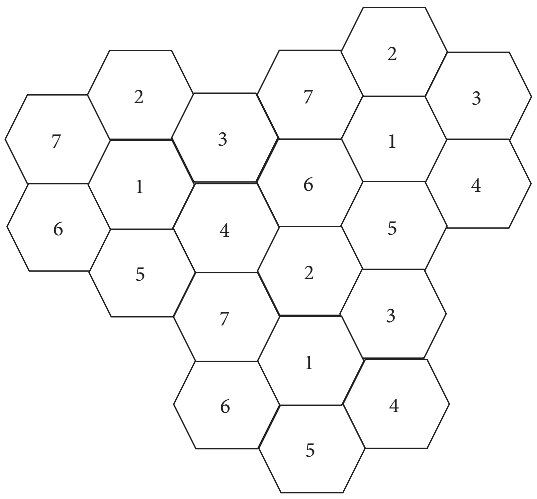

Figure 1: Cellular network structure.

codes (MNCs). When a mobile phone is turned on or enters a new LA, it sends a location area updating (LAU) request to the network. If the mobile phone is already connected to the network, it will send a TMSI for authentication; otherwise, it communicates an IMSI for this purpose. If the network is unable to identify the TMSI, it will send an identification request. The mobile phone sends an ID response that contains the IMSI. Immediately after receiving the IMSI, the network sends the authentication request to the mobile phone, which includes a random number. The mobile phone performs some computations with the random number and then sends it to the network in the form of an authentication response. After successful authentication, the network creates a ciphering mode command and determines the desired encryption algorithm (such as A5/1 or A5/3) to encrypt the user's information. The mobile phone generates encryption keys and transmits them to the network in the form of an encrypted ciphering mode complete message. From this point onward, communications between the BTS and the mobile station (MS) are encrypted. Eventually, the network sends an LAU Accept message with a new TMSI to the mobile phone. It should be noted that both authentication and encryption steps are optional in GSM [21]. Figure 2 illustrates the LAU process in the GSM network [5].

2.2. Fake Base Transceiver Station. In the GSM network, the mobile phone must be authenticated to connect to the network, but network authentication is not required for the mobile phone [1]. GSM is an old technology which makes the mutual authentication between the mobile station and the network very expensive. To remove GSM constraints, UMTS and LTE proposed a two-step verification, which requires BTS authentication via the mobile phone [4]. Due to backward compatibility and the use of GSM as a support network in cases where UMTS and LTE are not available, mobile phones have to downgrade to a GSM connection and be totally exposed to IC attacks [6, 22]. In fact, an IC enables the attacker to get the mobile's IMSI in order to track locations, eavesdrop on phone calls, or impersonate an existing user [7]. Using such tricks, the attacker inserts himself as man-in-the-middle (MITM) attacks [23]. On the GSM network, an IC introduces himself as an authentic BTS by adopting a high signal strength. The IC registers IMSI of all mobile phones in the nearby which want to connect and, thus, obtains the necessary information from the target device [24]. It can also be force connected to mobile phones to use the A5/0 encryption algorithm (without encryption) in order to better eavesdrop on them [25] (Figure 3).

2.3. IC Detection Parameters. References [5, 17, 18, 26] assessed the information of broadcasted IC signals and introduced them as the IC identification parameters. In the following, 15 parameters proposed for IC identification are described.

2.3.1. P1: Decreasing the Security Level of the Encryption Algorithm and Using a Weak Encryption Algorithm. One of the most common tactics used by attackers to eavesdrop on mobile calls is to lead them to choose less secure encryption algorithms. Attackers often force cell phones not to use encryption so that eavesdropping could be facilitated. Hence, being situated in a cell where a mobile phone has already made encrypted communications and the cell is now requesting for downgrading or removing the encryption algorithm indicates the occurrence of this parameter.

2.3.2. P2: Applying Security Settings and Encryptions Incompatible with Network Settings. An attacker is usually unaware of cell phone keys and is, therefore, unable to "break" any kind of encryption. As a result, it suggests users not to choose encryption. Therefore, one has to suspect a cell with disabled encryption in a network where encryption is common.

2.3.3. P3: Rejecting the Location Update Request Immediately after Receiving the ID of the Mobile Phone. If the attacker only targets the victim's IMSI, he or she will disconnect from the cell phone after obtaining that number. This is a well-known pattern which could be used to detect various IC attacks.

2.3.4. P4: Paging the Mobile Phone without the Next Request. The attacker sends paging messages to the victim so as to find if the victim is still within his or her reach, without processing the next steps of SMS or phone call. This behavior could prefigure an attack.

2.3.5. P5: Channel Assignment without the Next Call or SMS Setup. According to the standard, after the station assigns a channel to a mobile phone, the mobile phone sends an idle message to keep the channel open until the SMS or the voice is delivered. In order to locate the victim more accurately, the attacker may misuse this standard by receiving the idle message and discontinuing the process once the channel is assigned to the victim's device.

2.3.6. P6: A Large Number of Paging Groups. The attacker sends invalid data to the paging channel in order to remove the mobile phone from its cell. These data are sent continuously until the mobile phone is removed from the attacker's cell. Therefore, the appearance of a large number of paging-group messages can signal an attack.

2.3.7. P7: Delay in Validating Cipher Mode Complete Messages. An attacker attempting to "break" a cryptographic code online needs some encrypted texts with corresponding plaintexts (Known-Plain attack). To do this, the attacker makes the mobile phone use the International Mobile Station Equipment Identity Software Version (IMEISV) code. On the other hand, by delaying the response to the Cipher Mode Complete message, the attacker makes the mobile phone re-send this message in an encrypted format, thereby obtaining a number of encrypted texts with specific plaintexts and making it easy to find the key.

2.3.8. P8: Being the Only Visible Cell in a Location Area. To attract the mobile phone and make it send a LAU request, the attacker creates a fake cell with a Location Area Code (LAC) different from other neighboring cells and assigns a higher signal strength to it. Consequently, this cell will be the only one in the related area.

2.3.9. P9: Unusual LAC. To acquire more information from the victim, the attacker needs to make the mobile phone send an LAU command to the attacker's station. According to the standard, this happens when the LAC of the new station is different from that of the previous station. On the other hand, the attacker must also send a list of the neighboring cells to the mobile phone. Thus, the cell phone receives a list of cells, one of which is incompatible with others.

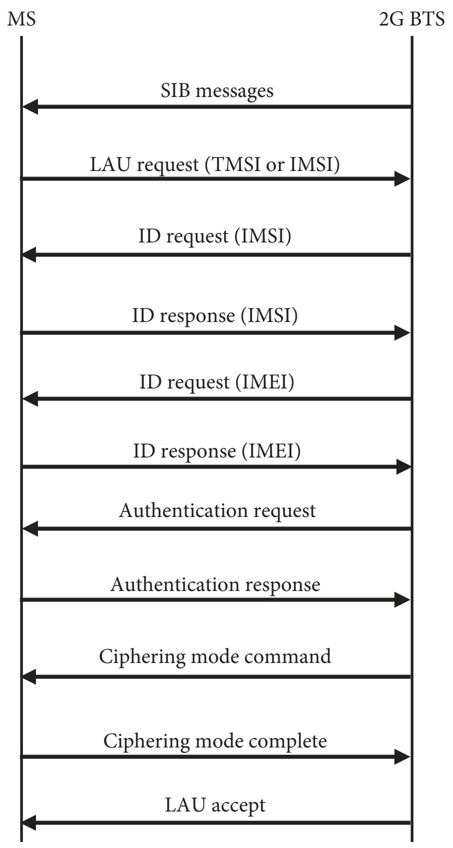

Figure 2: LAU process in GSM.

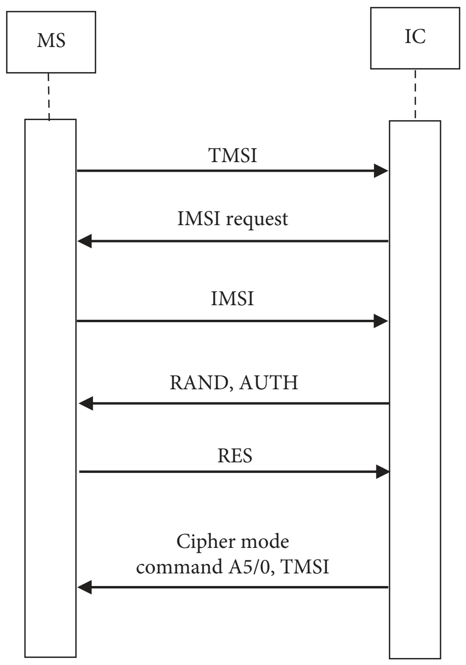

Figure 3: Register process in GSM.

2.3.10. P10: Providing Similar LACs/Cell IDs for Two Different Absolute Radio-Frequency Channel Numbers (ARFCNs). An attacker sometimes uses the cell ID and LAC of a real station to prevent disclosure. In this case, to avoid wave interference, the attacker usually makes use of a frequency different from that of the real station. An attack detection module can identify threats in case two different frequencies have the same cell ID.

2.3.11. P11: No Introduction of Neighboring Cells by the Station. As the attacker tries to keep the victim's mobile phone connected, he or she announces that there are no cells in the neighborhood, thus preventing the device from searching other stations.

2.3.12. P12: List of Incompatible Neighboring Cells. The neighborhood is a two-way relationship. In fact, when a station introduces its neighboring cells to a cell phone, it is expected that the stations in the neighboring cells will also recognize it as their neighbor. Hence, if the victim does not find the attacker's cell ID in the list of neighboring cells, it can be inferred that an attack has taken place.

2.3.13. P13: High Signal Strength. This parameter is activated if the signal strength of one cell is considerably different from that of other cells received up to that point.

2.3.14. P14: Request for Shortening Registry Intervals. The attacker can reduce the time period of sending LAU to at least 6 minutes by communicating the T3212 parameter to the mobile phone. This causes the victim's cell phone to send its information every six minutes, making it easier to track the victim. Therefore, the mobile phone will be at risk if it receives this message.

2.3.15. P15: High Cell Reselect Offset. The attacker amplifies the Cell Reselect Offset parameter to prevent the cell phone from connecting to neighboring BTSs.

2.4. Previous Studies. In recent years, the increasing number of IC attacks has prompted researchers to look for solutions to identify them. Some of these solutions are client-side. A number of these tools are discussed in Reference [5]. Examples include SnoopSnitch [18], GSM Spy Finder [27], Cell Spy Catcher [28], and AIMSICD [20]. These Android applications utilize BTS information as well as BTS-mobile phone communications in order to identify suspicious BTSs.

In 2014, SRLabs developed two client-side IC diagnostic tools, namely, SnoopSnitch and CatcherCatcher [18, 19]. SnoopSnitch is an Android application that alerts users to IC presence by analyzing signals received from available cells. According to Reference [5], SnoopSnitch makes use of the highest number of IC detection parameters. However, this software only works on rooted mobile phones with Qualcomm chipset. Similarly, CatcherCatcher explores suspicious behaviors on mobile networks and detects IC activities, but it just operates on Osmocom phones. Therefore, they have not gained much popularity. The Android IMSI-Catcher Detector (AIMSICD) identifies ICs by testing the BTS based on various criteria. Two of these criteria include monitoring the signal strength and investigating the accuracy of BTS information. Nevertheless, AIMSICD is still in the alpha version, and its effectiveness has not been evaluated in practice.

Scott et al. proposed a method that measures the signals of an area by periodically examining frequency bands and configurations of neighboring cells. They suggested monitoring unexpected location updates, mobile phone verification, and requests for downgrading encryption [7]. Anti Spy is a mobile application in which suggest a machine-learning-baed approah to detecting fake BTSs [29]. Unfortunately, neither of these methods has come up with a solution for scalability to cover vast areas. Broek et al. also suggested replacing the IMSI of devices by changing their nickname to counter IC attacks. The limitation of this method is that changes to the SIM card and authentication server are extremely costly for both users and operators [30].

FBS-Radar is a system solution that includes several cell phones and the Guard Cell Phone software. Guard Cell Phone is a network security assessment software for both Android and iOS systems. In collaboration with mobile network operators, FBS-Radar utilizes data related to network cell locations in order to identify ICs. This method can only detect ICs that send spam or fake SMS and cannot identify other attacks such as catching IMSI or tracking and eavesdropping on mobile phones [1]. SeaGlass was developed by security researchers at the University of Washington to report ICs across a given city. This system includes a number of sensors that are installed on volunteer vehicles and the server. SeaGlass sensors move around the city to collect BTS signals and transmit them to the server. ICs are identified at the server. These sensors have low power consumption and easy maintenance/support and could be installed on vehicles traveling long distances. It should be added that each sensor costs \$502 [17]. FBS-Radar and SeaGlass are system solutions for identifying ICs and should be implemented by operators and judicial authorities because they need database information of cell locations. They are also expensive to implement and cannot be used by individual mobile users.

Generally, IMSI Catcher Detectors (ICDs) can be classified into three categories: app based, sensor based, and network based, whose characteristics are illustrated in Figure 4.

Furthermore, we also evaluate their strengths and weaknesses based on their detection capabilities in Table 2.

In this paper, a client-side solution is proposed for IC detection. It consists of a sensor that can be installed on both rooted and nonrooted mobile devices. The YAICD sensor uses signal information from nearby BTSs to detect ICs. In addition to the ability to detect ICs to which the user is connected, this sensor can also identify nearby ICs and provide necessary alerts before users may connect to them.

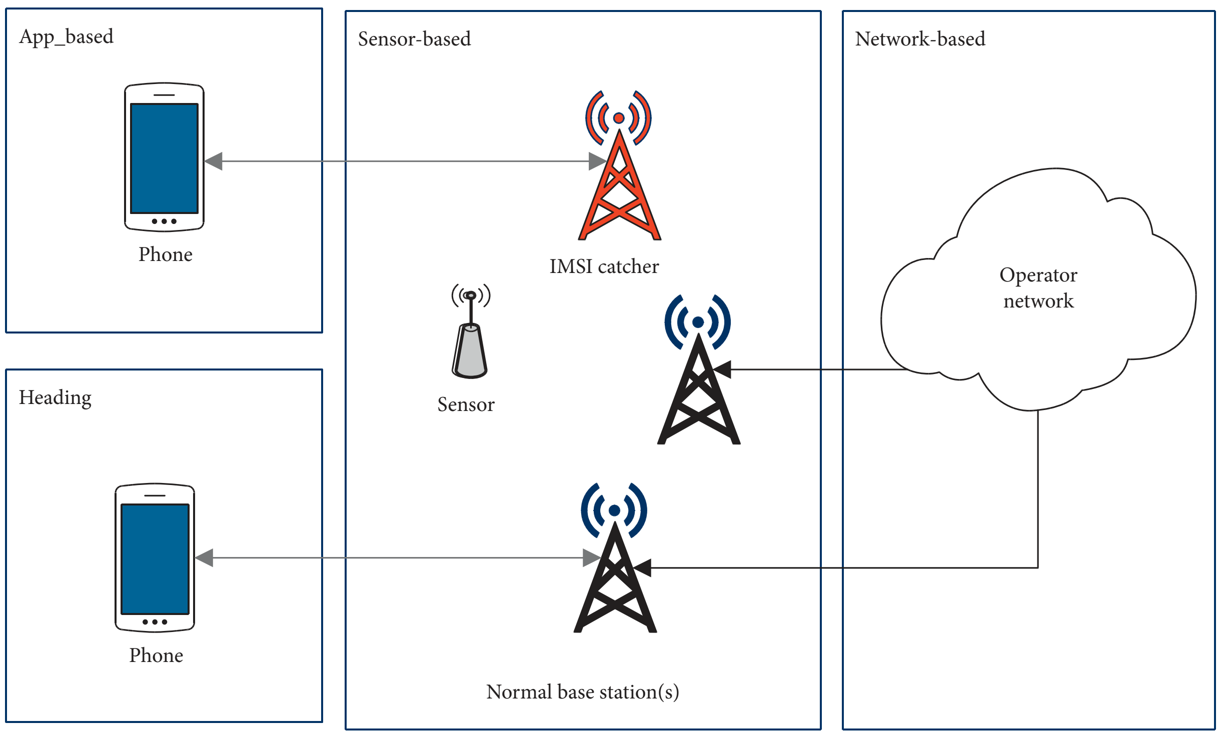

Figure 4: Overview of IMSI catcher detection methods [31].

Table 2: Overview of IMSI catcher detection methods.

Category	Examples	Advantages	Disadvantages
App-based	SnoopSnitch [18], AIMSICD [20], Cell Spy catcher [28], and GSM Spy Finder [25]	App-based ICDs can give the user a direct real-time notification shortly upon IMSI catcher activity because they are directly facing the user.	The smartphone itself does not know the network deployment status, and some apps present false notifications to the user. Furthermore, there is a potential burden on a smartphone due to increased battery consumption and side effects such as some security features being disabled caused by rooting the smartphone.
Sensor-based	SeaGlass [8, 18, 32].	ICD sensors can monitor a larger area compared to smartphones, thanks to the bigger antenna size than smartphones. This allows a detailed and focused analysis of a group of base stations, while app-based ICDs are bound to the mobility of the phone. Moreover, sensors provide more computing power for IMSI catcher detection. Since they are powered from a mains power unlike smartphones, they do not run any additional background process that may hinder the analysis of incoming signaling messages.	Sensors need long observation times and require some time to detect the IMSI catcher event and deliver a warning to the phone. As a result, an IMSI catcher might already succeed in an attack well before it is detected by a sensor. While individual sensors could be easily built with minimal cost, maintaining backend infrastructure for sensor management and analysis requires recurring cost (power, Internet, place rental, and management), which will also increase when service coverage is expanding.
Network-based	FBS-radar [2, 26, 33]	Operators retain up-to-date information of the cellular network deployment; hence, detection of unknown cells and base station identities are a clear sign of an IMSI catcher. Furthermore, this reduces false-positives and increases accuracy. Deployment cost is low as it only requires software upgrades within the network.	Implementation requires cooperation with network operators, which is not always possible. Like sensor-based detectors, the detection is performed after the phone was released from the IMSI catcher.

3. The Proposed Solution

In this section, a new method, called the YAICD, is described for IC identification. It is a client-side tool that can be launched on Android mobile devices, whether they have root privilege or not. In rooted devices, like SnoopSnitch, it uses a Qualcomm chip to collect signal data from BTSs. In nonrooted Android devices, BTS signal data are received from a library called TelephonyManager. The YAICD sensor detects ICs by extracting 15 parameters from BTS signals based on a threshold value and the sum occurrence of several parameters. The architecture of the proposed method and its implementation are described below.

3.1. The Proposed Architecture. Our solution is a sensor that includes two components: data collection and attack detection. Figure 5 illustrates the architecture of this instrument. Different components of the YAICD sensor and their relationships are discussed in the following lines.

3.1.1. Data Collection Components. The mobile phone is directly connected to available BTSs. These stations broadcast their information in the form of messages called System Information Block (SIB). The data collection unit receives information from all signals broadcasted by the nearby BTSs. These received data are stored in their respective databases. The YAICD sensor's database consists of four important tables entitled Cell_Info, Session_Info, Catcher, and NeighborCatcher. Different pieces of information related to the data collection component are assigned to the Cell_Info and Session_Info tables, while the attack detection component analyzes the data of these tables to identify ICs. The Catcher and NeighborCatcher tables contain the information of spotted ICs. In the following, these four tables are explained:

Cell_Info table: it includes information broadcasted by all BTSs that the YAICD sensor has observed.

Session_Info table: this table registers information about transactions (such as calling and sending/receiving SMS) conducted between the mobile phone and the BTS.

Catcher table: a separate record is created for each identified IC to which the mobile phone is connected. In each record, the parameter information of the identified IC is registered.

NeighborCatcher table: it contains information related to neighboring ICs to which the mobile phone is not currently connected but the YAICD sensor has identified. In each record, parameter data of each of these ICs are registered.

3.1.2. Attack Detection Component. This is the most important component of the YAICD sensor. It attempts to report the presence of ICs using data gathered by the data collection component. This is accomplished based on the sum of occurring parameters and a detection threshold. The YAICD sensor extracts IC identification parameters, introduced in Section 2.3, by analyzing the signals received from BTSs. The number of these parameters varies depending on whether the mobile phone is rooted or not (see Section 3.2).

3.1.3. Communications between Components. Mobile phones are in direct contact with BTSs and always receive SIB messages. In the data collection component, information on transactions between the mobile phone and BTS as well as the broadcasted information of all BTSs observed by the YAICD sensor are separated in the Session_Info and Cell_Info tables. The attack detection component deploys the information of these two tables to extract IC identification parameters.

As mentioned, the YAICD sensor can detect both ICs to which mobile phones are connected and those neighboring ICs to which mobile devices are not yet connected. To detect connected ICs, the YAICD sensor may extract all 15 IC identifiers by means of Cell_Info and Session_Info tables. However, only parameters p8 to p15 can be calculated to spot nearby ICs. These parameters only use the data of Cell_Info table (Figure 6). Meanwhile, in both cases, the number of parameters depends on whether the mobile device is rooted or not.

As one can see in Figure 6, the Data Collection module aims to collect all the data related to the BTS devices. Such data are organized and stored in two tables named here *cell info* and *session info*. The Data Analysis module uses the data stored in these two tables to evaluate the 15 detection parameters in order to detect ICs. After detection of any IC based on the data collected from BTS devices, the system generates corresponding alerts.

3.2. Implementation. The proposed sensor was implemented on both rooted and nonrooted mobile phones. The implementation of each mode is described below.

3.2.1. Rooted Sensor. Like SnoopSnitch, our rooted sensor uses the Qualcomm chip to collect signal information from BTSs. To operate in this mode, the YAICD sensor requires a rooted Android mobile phone with a Qualcomm chip. In fact, receiving data transmitted between the BTS and mobile phone (such as the type of encryption and some information relayed by the BTS) is possible when the mobile device has both root access and the Qualcomm chip. In this case, by accessing most information received from BTSs, one may calculate all 15 parameters of IC identification. The information gathered by the data collection module, in this case, is presented in Table 3.

3.2.2. Nonrooted Sensor. The nonrooted sensor can be installed on all mobile phones, but it will not have access to low-level data since it uses a library called TelephonyManager on the Android system to collect signal information from BTSs. Consequently, it does not have access to the information (such as encryption type) exchanged between the BTS and the mobile phone and is only

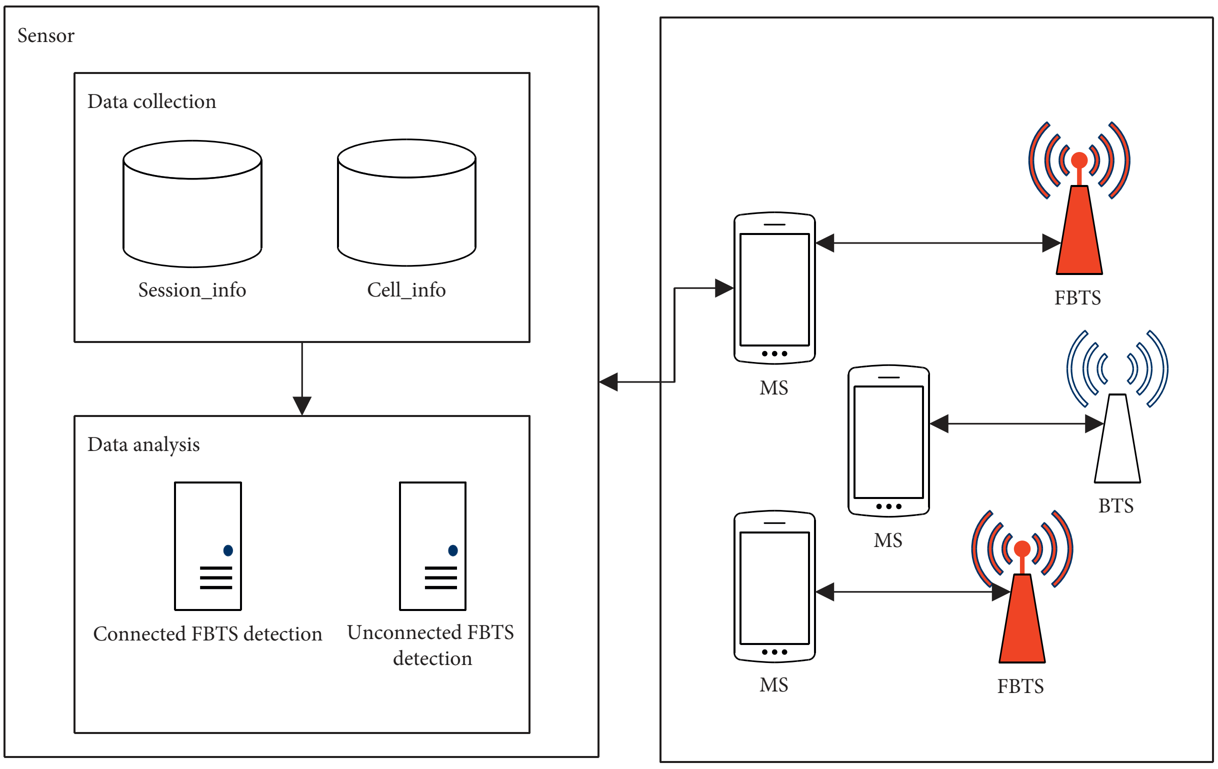

Figure 5: View of external components associated with the YAICD sensor.

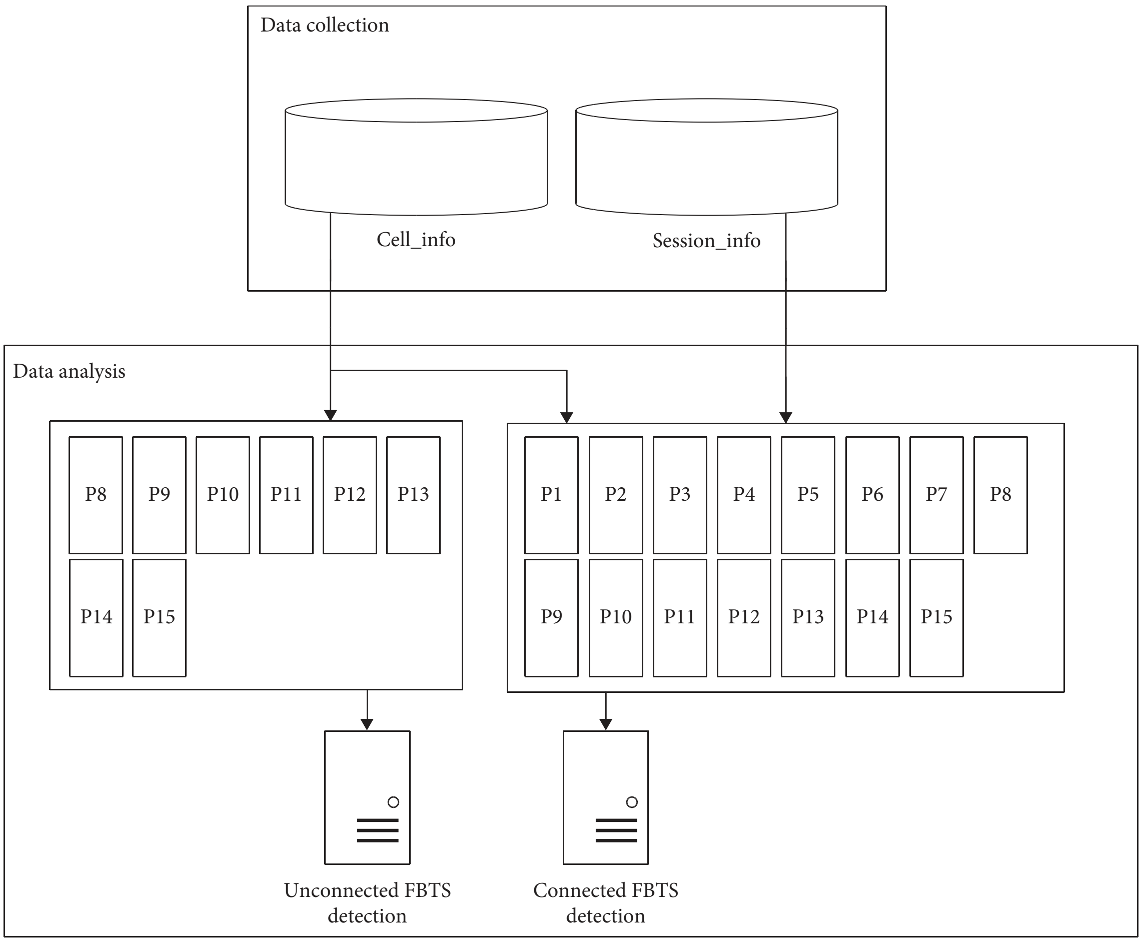

Figure 6: View of component communication.

Table 3: Received information from QUALCOMM and root mobile device.

Field	Description
ARFCN	ARFCN of Cell
IMSI	IMSI digits
IMEI	IMEI digits
CID	Cell ID
CRO	Cell reselect offset
T3212	Location update timer
Neigh	Neighboring cell count
PWR	Signal power
GPS_lat	GPS Latitude
GPS_lang	GPS Longitude
MNC	Mobile network code
MCC	Mobile country code
RAT	Radio access technology
LAC	Location area code
auth	Authentication was performed
cipher_delta	Delta time to complete ciphering
cipher	Cipher type (A5/x, UEA/x, EEA/x)
Duration	Transaction duration in ms
sms_presence	Transaction contains SMS data
call_presence	Transaction contains a call setup
Assign	Transaction contains an assignment to other channels
iden_IMEI_bc	IMEI was requested before ciphering
iden_IMEI_ac	IMEI was requested after ciphering
iden_IMSI_bc	IMSI was requested before ciphering
iden_IMSI_ac	IMSI was requested after ciphering
lu_reject	Location update was rejected
lu_acc	Location update was accepted
Mobile_org	Mobile originated transaction
Paging_mi	Paging mobile identity type
Mobile_termin	Mobile terminated transaction

able to receive limited information from the nearby BTS signals. As such, in this case, only the parameters p8 to p13 can be evaluated. The information received from this library is given in Table 4.

4. Experiments

In this section, we first describe how to set the threshold parameters of IC detection. After introducing the experimental platform and the procedure for implementing every single parameter, we compare the performance of YAICD in terms of IC detection with existing client-side applications. Eventually, the performance of the sensor in consuming resources such as CPU and memory is evaluated.

4.1. Cell Data Collection. In this study, in order to determine the threshold parameters, several mobile phones with IR-MCI, Irancell, and RighTel SIM cards were chosen. The YAICD sensor was then installed on every mobile phone. Subsequently, the mobile devices began collecting information in some areas of the two cities of Tehran and Shahrood, Iran (we published the datasets on github and all the datasets are publicly available at https://github.com/parimahziaei/YAICD-Yet-Another-IMSI-Catcher-Detector-in-GSM). The YAICD sensors gathered data from 1254 BTSs. The threshold of each IC detection parameter was calculated based on the observations. It was found that parameters p2, p3, p4, p8, p9, p10, p11, and p12 had no threshold value. In case any of these parameters occurs, the YAICD sensor detects it and the rest of the parameters are determined according to their observed values. Table 5 presents the required score for each parameter.

The final threshold value for IC detection was estimated at 2.6. It is necessary to mention that the user can change threshold values by referring to the sensor's configurations. Furthermore, the value chosen for each parameter and the final threshold value was obtained by using the SnoopSnitch application.

As mentioned in [19], the threshold values are obtained based on some evaluations of the attack scenarios using ICs. As the results of these evaluations, they obtained some weights and threshold values for the detection parameters as reported in [19]. Since we were not able to deploy any IC in a real environment (because of the privacy issues and regulations), we decided to conduct our experiments in a testbed using the threshold values reported in [19]. This helps us to fairly evaluate the performance of our approach against SnoopSnitch.

We believe that having the detailed configurations of the real BTS devices can improve the process of choosing the threshold values for the detection parameters. An interesting research issue is to investigate the possibility of designing a systematic approach for choosing the threshold values when we have information about the configurations of the BTS devices for each network operator.

It is to be noted that we have no access to any real-world attack scenario launched by an FBTS. Thus, we conducted our experiments by simulating the attacks in our test bed and we could not evaluate the performance of our approach based on the false alarming and the ROC curve. It must be noted that the threshold value controls the sensitivity of our detection rate. In other words, the lower value of this threshold makes our detection mechanism more sensitive and therefore increases the false alarm. It is due to the fact that by smaller values of the threshold, the probability of labeling a real BTS as a fake BTS increases. We also left the threshold value to be configured by the user of the system, which provides the possibility to manipulate the sensitivity of our detection mechanism.

4.2. Comparing the Performance of YAICD with Other Conventional Client-Side Applications Used for IC Parameter Detection. This section compares the YAICD sensor with some other client-side programs devised for identifying IC parameters. These available applications include SnoopSnitch, AIMSICD, Cell Spy Catcher, and GSM Spy Finder. An IC was simulated in a testbed environment in order to investigate the performance of the YAICD sensor. Testbed environment was made possible by using OpenBTS software along with the USRP N210 device. Figure 7 illustrates the testbed environment used to test the YAICD sensor. In addition, the following hardware and software were employed in this study to collect data from BTSs and assess the YAICD sensor:

Table 4: Received information from telephony manager.

Field	Description
ARFCN	ARFCN of Cell
IMSI	IMSI digits
IMEI	IMEI digits
CID	Cell ID
Neigh	Neighboring cell count
PWR	Signal power
GPS_lat	GPS Latitude
GPS_lang	GPS Longitude
MNC	Mobile network code
MCC	Mobile country code
RAT	Radio access technology
LAC	Location area code

Table 5: desired parameters along with the threshold and corresponding value.

Parameter	Threshold 1	Value 1	Threshold 2	Value 2
P1	If 80% of sessions are encrypted	2	—	1
P2	—	2		
P3	—	6		
P4	—	1		
P5	8000 ms	1		
P6	>32	1		
P7	2000 ms (with IMEISV)	2	2000 ms (without MEISV)	1
P8	—	1		
P9	—	0.5		
P10	—	1		
P11	—	1		
P12	—	1		
P13	−57	1		
P14	1–60 minutes	1.5	61–240 minutes	0.7
P15	>8	1		

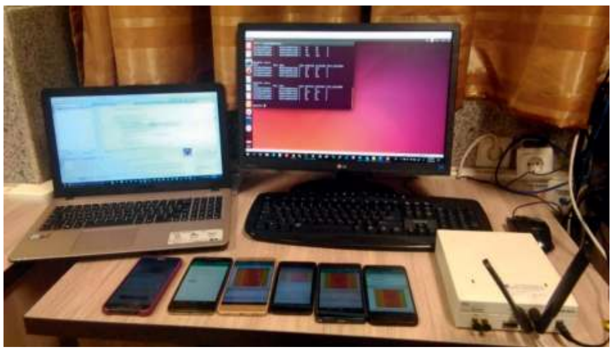
Figure 7: Laboratory of testing sensor.

(i) Three rooted mobile phones including Huawei Y7, LG nexus 5x, and Huawei G620, and three nonrooted devices including Samsung J7, Sony z2, and LG G4

(ii) Six SIM cards from different telecommunications service providers of Iran (IR-MCI, Irancell, and RighTel)

(iii) OpenBTS installed on the virtual machine

(iv) A USRP N210

To protect the privacy of users, only the test mobile phones were allowed to connect to the simulated FBTS. The performance of the proposed sensor was compared with other available applications by separately implementing each of the 15 parameters to detect FBTSs in the laboratory environment. Tables 6 and 7 provide the results of comparing the performance of the YAICD sensor with those of similar user-side tools in identifying parameters of connected nearby ICs.

As can be understood from Table 6, SnoopSnitch succeeds in identifying parameters of those ICs to which the mobile phone is connected (except p13). However, this application can only be installed on rooted devices that are equipped with Qualcomm chip. Cell Spy Catcher, AIMSICD, and GSM Spy Finder could detect fewer parameters than SnoopSnitch. Meanwhile, the YAICD sensor could recognize all parameters implemented in the testbed environment. The advantage of our model is that it can be installed on Android mobile devices, whether they have root privilege or not. Since it has access to most BTS information in the rooted mode, it uses the largest number of parameters possible to detect ICs. In the nonrooted mode, however, it can only access high-level information. As a result, it uses fewer parameters to perform its task.

Table 6: Results of parameters testing for connected in apps.

	Root		Nonroot			
Parameter	YAICD	SnoopSnitch	AIMSICD	Cell spy catcher	GSM spy finder	YAICD
P1	✓	✓	✓			
P2	✓	✓	✓			
P3	✓	✓				
P4	✓	✓				
P5	✓	✓				
P6	✓	✓				
P7	✓	✓				
P8	✓	✓	✓	✓	✓	✓
P9	✓	✓	✓	✓	✓	✓
P10	✓	✓	✓	✓	✓	✓
P11	✓	✓	✓	✓	✓	✓
P12	✓	✓	✓	✓	✓	✓
P13	✓		✓			✓
P14	✓	✓				
P15	✓	✓				

Table 7: Results of parameters testing for disconnected in apps.

	Root		Nonroot			
Parameter	YAICD	SnoopSnitch	AIMSICD	Cell spy catcher	GSM spy finder	YAICD
P8	✓		✓			✓
P9	✓		✓			✓
P10	✓		✓			✓
P11	✓		✓			✓
P12	✓		✓			✓
P13	✓		✓			✓
P14	✓					
P15	✓					

Table 7 shows the results of identifying neighboring ICs. Accordingly, the YAICD sensor detects the highest number of parameters for nearby ICs. Moreover, SnoopSnitch, while good at identifying parameters of ICs to which the user is connected, is unable to detect neighboring ICs to which the user is not connected. Of the available applications, only AIMSICD is capable of detecting nearby ICs, but it is still in the Alpha version and has not been fully investigated. Besides, compared with our sensor, it detects fewer parameters of the nearby ICs.

4.3. Assessing Resource Consumption of the YAICD Sensor. Nowadays, CPU and memory usage is one of the most decisive indicators in software performance testing. In this section, the performance of the YAICD sensor is compared with that of SnoopSnitch in terms of the number of resources, including CPU and memory, each consumes.

The AnotherMonitor (https://f-droid.org/en/packages/org.anothermonitor/) software was used to measure the CPU and memory consumption of the YAICD sensor and SnoopSnitch. To this end, both the YAICD sensor (root) and SnoopSnitch were installed on Huawei Y7 phone and their CPU and memory usage were calculated for 4 minutes. Also, to calculate CPU and memory usage on nonrooted phones, the sensor was installed on the LG G4 phone. The following is a comparison of the amount of CPU and memory resources used by the rooted YAICD sensor, the nonrooted YAICD sensor, and SnoopSnitch. Figure 8 displays the degree of CPU used by the rooted YAICD sensor, the nonrooted YAICD sensor, and SnoopSnitch, while Figure 9 presents the amount of memory used by the rooted YAICD sensor, the nonrooted YAICD sensor, and SnoopSnitch.

As shown, the YAICD sensor and SnoopSnitch require almost the same amount of memory and CPU. Accordingly, it is evident that the sensor, in addition to its excellent identification of ICs in different modes, performs optimally with regard to CPU and memory consumption.

5. Conclusion

Due to the dramatic rise of IC attacks on cellular networks, several system and client-side solutions have been developed to identify ICs. FBS-Radar and SeaGlass are among system solutions for IC detection. These methods need database information of cell locations, and their implementation is costly. Therefore, they cannot be used by laymen and must

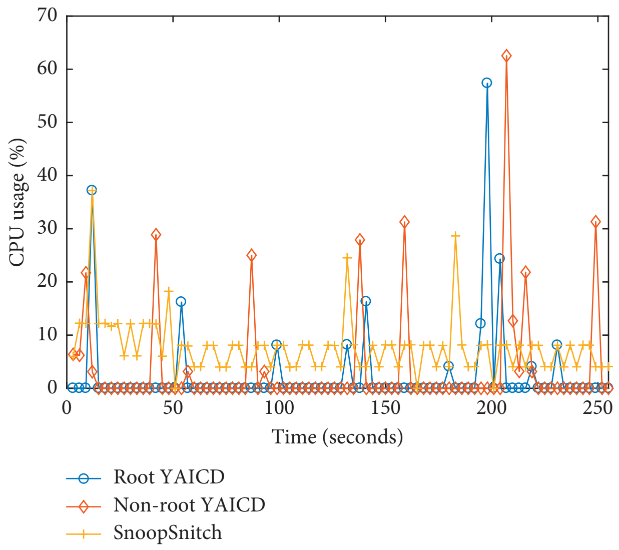

Figure 8: Comparison of CPU usage for the YAICD sensor in root and nonroot mode and SnoopSnitch.

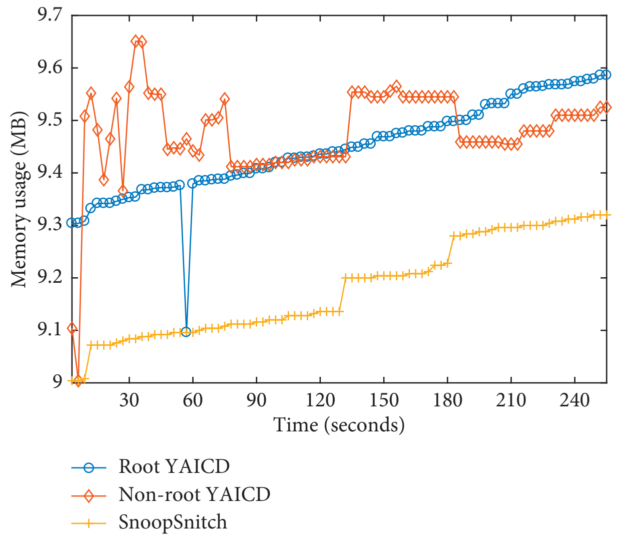

Figure 9: Comparison of memory usage for the YAICD sensor in root and nonroot mode and SnoopSnitch.

be implemented by operators and judicial authorities. SnoopSnitch and AIMSICD are well-known examples of client-side applications in this area. SnoopSnitch utilizes the highest number of IC identification parameters, but it can only be used on specific mobile devices with root access. In addition, it does not identify ICs to which users have not yet connected. AIMSICD, on the other hand, recognizes ICs to which users have not yet connected, but it is in the alpha version and its practical effectiveness has not been substantiated.

This paper introduced a client-side sensor for IC identification, called YAICD. This comprehensive solution takes account of all IC detection parameters that the existing applications cover. The designed sensor can be installed on both rooted and nonrooted mobile devices. In addition to identifying ICs to which the mobile phone is connected, it also notifies users of nearby ICs.

The amount of resources consumed by a mobile application is one of the basic criteria predicting its acceptability in the market. In this regard, the CPU and memory consumption of YAICD and SnoopSnitch were compared. It was found that, besides its high performance in detecting ICs, the YAICD is not significantly different from SnoopSnitch in terms of resource consumption.

It is noteworthy that the YAICD sensor only alerts the user of ICs and does not provide secure communications between the mobile phone and the base transceiver station. The main advantages of this sensor include:

(1) Alerting the user of MITM or IC attacks to prevent fraud
(2) Alerting the user when an IC is in the vicinity
(3) Capacity to be installed at different strategic locations, such as Parliament and the President's Office, to ensure the absence of ICs.

As future work, we plan to design a mechanism to automatically adjust the threshold values we employed in our detection algorithm. We also plan to design a systematic approach to learn the detection parameters' values based on the configurations set on the real BTS devices by the network operators. Moreover, by collecting all the data in a central server, we believe more sophisticated algorithms can be designed to improve the detection accuracy and to track the attackers.

Data Availability

The data used to support the findings of this study are restricted in order to protect the privacy of the networks tested in this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors acknowledge ParsaSharif Research Institute, Iran, for partial support for this work.

References

[1] T. T. Allen, *Introduction to Discrete Event Simulation and Agent-Based Modeling: Voting Systems, Health Care, Military, and Manufacturing*, Springer, New York, NY, USA, 2011.
[2] Z. Li, W. Wang, C. Wilson et al., "FBS-radar: uncovering fake base stations at scale in the wild," in *Proceedings of the Network and Distributed System Security Symposium*, San Diego, CA, USA, 2017.
[3] M. Mouly, M.-B. Pautet, and T. Foreword By-Haug, *The GSM System for Mobile Communications*, Telecom Publishing, Bucharest, Romania, 1992.
[4] C. Zhang, "Malicious base station and detecting malicious base station signal," *China Communications*, vol. 11, no. 8, pp. 59–64, 2014.

[5] A. Shaik, R. Borgaonkar, N. Asokan, V. Niemi, and J.-P. Seifert, "Practical attacks against privacy and availability in 4G/LTE mobile communication systems," 2015, http://arxiv.org/abs/1510.07563.

[6] S. Park, A. Shaik, R. Borgaonkar, A. Martin, and J.-P. Seifert, "White-stingray: evaluating {IMSI} catchers detection applications," in *Proceedings of the 11th {USENIX} Workshop on Offensive Technologies ({WOOT} 17)*, Baltimore, MD, USA, 2017.

[7] P. F. Scott, "Secrecy and surveillance: lessons from the law of IMSI catchers," *International Review of Law, Computers & Technology*, vol. 33, no. 3, pp. 1–23, 2019.

[8] 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*, pp. 246–255, ACM, Orleans, LA, USA, December 2014.

[9] U. Meyer and S. Wetzel, "A man-in-the-middle attack on UMTS," in *Proceedings of the 3rd ACM workshop on Wireless security*, Philadelphia, PA, USA, October 2004.

[10] A. Montieri, D. Ciuonzo, G. Aceto, and A. Pescape, "Anonymity services tor, i2p, jondonym: classifying in the dark (web)," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 3, pp. 662–675, 2018, þ.

[11] G. Bovenzi, G. Aceto, D. Ciuonzo et al., "H2ID: hierarchical hybrid intrusion detection approach in IoT scenarios," in *Proceedings of the IEEE Global Communications Conference (Globecom)*, Taipei, Taiwan, December 2020.

[12] F. Van Den Broek, R. Verdult, and J. de Ruiter, "Defeating IMSI catchers," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, Prague, Czech Republic, September 2015.

[13] P. Ginzboorg and V. Niemi, "Mobile equipment identity privacy, network node and methods thereof," Google Patents, 2019.

[14] Phony Cell Towers Are the Next Big Security Risk, 2019, https://www.theverge.com/2014/9/18/6394391/phony-cell-towers-are-the-next-big-security-risk Accessed.

[15] K. Norrman, M. Näslund, and E. Dubrova, "Protecting IMSI and user privacy in 5G networks," in *Proceedings of the 9th EAI International Conference on Mobile Multimedia Communications*, Xi'an China, June 2016.

[16] 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," in *26th Annual Network and Distributed System Security Symposium, NDSS 2019*, San Diego, CA, USA, February, 2019.

[17] M. Khan, V. Niemi, and P. Ginzboorg, "IMSI-based routing and identity privacy in 5G," in *Proceedings of the 22nd Conference of Open Innovations Association FRUCT*, Jyvaskyla, Finland, 2018.

[18] P. Ney, I. Smith, G. Cadamuro, and T. Kohno, "SeaGlass: enabling city-wide IMSI-catcher detection," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 3, pp. 39–56, 2017.

[19] Snoopsnitch, 2019, https://opensource.srlabs.de/projects/snoopsnitch.

[20] CatcherCatcher, 2019, https://opensource.srlabs.de/projects/mobile-network-assessment-tools/wiki/CatcherCatcher.

[21] Android IMSI-Catcher Detector, 2019, https://cellularprivacy.github.io/Android-IMSI-Catcher-Detector/.

[22] M. Radio Interface Layer, Specification, Core Network Protocols; Stage [], GPP TS.

[23] D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper, "Breaking LTE on layer two," in *IEEE Symposium on Security & Privacy (SP)*, San Francisco, CA, USA, May 2019.

[24] I. Singh, "Signaling security in LTE roaming," MSc thesis, School of Electrical Engineering, Aalto University, Espoo, Finland, 2019.

[25] 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*, pp. 202–215, ACM, New York, NY, USA, 2016.

[26] A. Dabrowski, G. Petzl, and E. R. Weippl, "The messenger shoots back: network operator based IMSI catcher detection," in *Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses*, Paris, France, September 2016.

[27] H. Alrashede and R. A. Shaikh, "IMSI catcher detection method for cellular networks," in *Proceedings of the 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*, Riyadh, Saudi Arabia, May 2019.

[28] GSM Spy Finder, 2019, https://play.google.com/store/apps/details?id=kz.galan.antispy.

[29] Cell Spy Catcher, 2019, https://play.google.com/store/apps/details?id=com.skibapps.cellspycatcher.

[30] T. van Do, H. T. Nguyen, and N. Momchil, "Detecting IMSI-catcher using soft computing," in *Proceedings of the International Conference on Soft Computing in Data Science*, Berkeley, CA, USA, March 2015.

[31] S. Park, A. Shaik, R. Borgaonkar, and J.-P. Seifert, "Anatomy of commercial IMSI catchers and detectors," in *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society*, pp. 74–86, London, UK, November 2019.

[32] A. Wilson, "SITCH: situational information from telemetry and correlated heuristics," in *Proceedings of the DEF CON*, vol. 24, Las Vegas, NV, USA, 2016.

[33] S. Steig, A. Aarnes, T. Van Do, and H. T. Nguyen, "A network based imsi catcher detection," in *2016 6th International Conference on IT Convergence and Security (ICITCS)*, pp. 1–6, IEEE, 2016.