Catch You Cause I Can: Busting Rogue Base Stations using CellGuard and the Apple Cell Location Database

Lukas Arnold
larnold@seemoo.de
Secure Mobile Networking Lab
TU Darmstadt
Germany

Matthias Hollick
mhollick@seemoo.de
Secure Mobile Networking Lab
TU Darmstadt
Germany

Jiska Classen
jiska.classen@hpi.de
Hasso Plattner Institute
University of Potsdam
Germany

ABSTRACT

Mobile phones connect to the Internet and receive phone calls using a cellular baseband chip. Basebands pose a substantial attack surface, as they do not only process but also decrypt personal data. Cellular attackers usually force a phone to connect with a Rogue Base Station (RBS), e.g., to record identity information and locations, intercept or manipulate traffic, or execute arbitrary code by exploiting vulnerabilities in the baseband stack. RBSes are stealthy, as smartphones attempt to connect to nearby base stations and do not display any indicators of compromise to the user. While their detection with Software-defined Radios (SDRs) is possible, usability and scalability are limited.

We research and expose the baseband interface on recent iPhones for Intel and Qualcomm chips to detect RBS attacks. We integrate these findings into a user-friendly app called CellGuard. Detection even works on non-jailbroken iPhones with the latest security updates and Lockdown mode. We enhance detection by utilizing Apple's internal database with highly accurate cell tower information and in-depth reverse engineering of Apple's baseband interface protocols to find further indicators of compromise. During multiple weeks of evaluation, we collect data on various devices using CellGuard and evaluate the results, along with measurements from our own RBS setup. Our baseband analysis framework BaseTrace will be helpful beyond RBS detection, as it can interact with the baseband and decode any management information exchanged, including satellite communication in the iPhone 15.

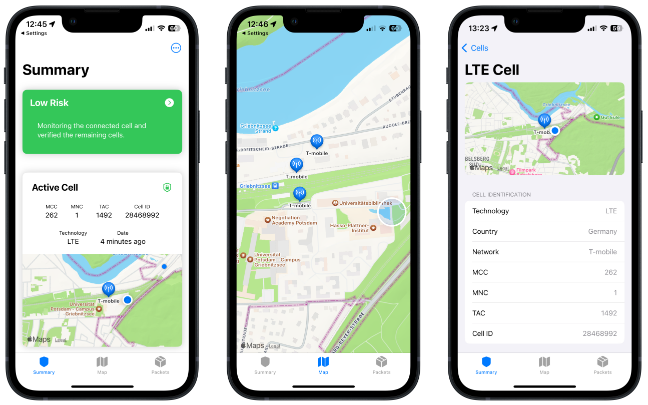
CCS CONCEPTS

• **Security and privacy → Mobile and wireless security**; *Intrusion/anomaly detection and malware mitigation.*

ACM Reference Format:
Lukas Arnold, Matthias Hollick, and Jiska Classen. 2024. Catch You Cause I Can: Busting Rogue Base Stations using CellGuard and the Apple Cell Location Database. In *The 27th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2024), September 30–October 02, 2024, Padua, Italy.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3678890.3678898

This work is licensed under a Creative Commons Attribution International 4.0 License.

RAID 2024, September 30–October 02, 2024, Padua, Italy
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0959-3/24/09
https://doi.org/10.1145/3678890.3678898



Low Risk

Monitoring the connected cell and verified the remaining cells.

Active Cell

MCC	MNC	TAC	Cell ID
262	1	1492	28468992

Technology	Date
LTE	4 minutes ago

CELL IDENTIFICATION

Technology	LTE
Country	Germany
Network	T-mobile
MCC	262
MNC	1
TAC	1492
Cell ID	28468992

(a) Summary. (b) Cell map. (c) Cell data.

Figure 1: CellGuard's user interface.

1 INTRODUCTION

Smartphones and their permanent cellular connection are vital to our modern lifestyle. The underlying technology evolved over five generations, with significant security improvements added over time. Due to lacking security measures, especially in older standards still in use today, threats emerged that could affect the users' privacy and security [34]. Adversaries can set up an RBS, also called an International Mobile Subscriber Identity (IMSI) or Subscription Concealed Identifier (SUCI) catcher, which forces nearby smartphones to connect with them. They collect the smartphone's identifiers, link them with a person's identity, and track their location [19, 52, 63]. An RBS can launch more complex attacks to execute arbitrary code on the connected smartphone's baseband [15, 31, 32, 40, 48, 49, 51, 81, 82, 85]. The security research community is aware of such dangers. Researchers analyzed IMSI catchers available for sale and proposed detection methods [63].

Despite this significant attack surface, there is no openly available approach for usable and scalable tooling to detect RBS-based attacks. Methods requiring special hardware like SDRs are absent in most situations when RBS detection is needed [26, 57]. Detection algorithms within Android apps exist; however, they lack essential metrics [63]. Since the study on these Android apps was published, the core of the RBS detection of these apps stayed unchanged, even though some user interfaces were slightly updated [14, 17, 29, 72, 75]. Packet-based metrics, which are the most effective for app-based approaches, exclusively work on rooted Android devices—thus opening further security issues on the phones that users want to protect against attacks. No attempts have been made to bring packet-based RBS detection to non-rooted devices. None of these approaches, neither SDR-based nor app-based, supports detecting anomalous behavior in 5G.

We create CellGuard for iOS, adding significantly improved detection capabilities through novel metrics and utilizing an Apple-internal database of valid base stations. Analysis of low-level packet information from the baseband works on non-jailbroken phones. Screenshots in Figure 1 show the information CellGuard collects about connected cells of nearby base stations. It supports the latest iPhone 15, including 5G communication. We verified compatibility down to the iPhone 6s for Qualcomm and Intel basebands, resembling full support for nine years of iPhone generations.

Contributions. This paper makes the following contributions:

- We design and implement the iOS app CellGuard. CellGuard verifies collected data with two novel approaches. We evaluate CellGuard's effectiveness in real-world tests over multiple months.
- We retrieve cell information from the Apple Location Services (ALS) database and compare its quality to openly available databases [54, 80]. Our analysis shows that both open databases list many inactive cells while also missing a large portion of legitimate cells. Thus, CellGuard's RBS detection accuracy improves tremendously by using ALS.
- We find multiple metrics indicating RBS presence contained in Intel and Qualcomm baseband interface packets.
- We reverse engineer proprietary baseband interface protocol extensions for iPhones with Qualcomm basebands and develop the framework BaseTrace. It can interact with the iOS baseband chips and create Wireshark dissectors for these automatically.
- The CellGuard app integrates BaseTrace to decode Qualcomm and Intel baseband interface packets on the go.
- We demonstrate BaseTrace is useful beyond RBS detection, e.g., to decode satellite communication on the iPhone 15 [3].
- We provide a library to communicate with iPhone basebands for further experimentation.

CellGuard provides a valuable step forward in mitigating RBS attacks. While it cannot prevent these attacks from happening per se, it uncovers when and where these attacks are happening. Detecting unauthorized attacks against politicians, journalists, or human rights activists enables these individuals to take further action. Supporting the latest operating system updates is mandatory for these at-risk users, as research prototypes running on rooted devices would expose their phones to further attacks.

2 BACKGROUND ON CELLULAR SECURITY

We next detail typical attacker goals, the role of RBSes during attacks, and possible mitigations in cellular networks.

2.1 Attacker Goals

Adversaries install RBSes to reach the following goals.

(1) Personal Information and Location Tracking. An attacker could collect personal information, such as unique identifiers and location information. E.g., an attacker could put up multiple RBSes to get the location profiles of all nearby users.

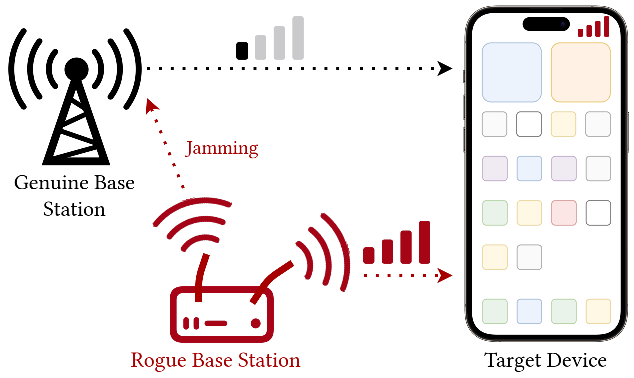


Figure 2: Attack vector of an active rogue base station. Attackers can influence the received signal strength at the target device through jamming.

(2) Traffic Interception and Manipulation. Base stations offer access to Internet and telephony services. An RBS could be used for Machine-in-the-Middle (MitM) attacks to intercept and manipulate traffic. Phone calls and Short Message Service (SMS) are especially at risk, as they lack further protection. Even Rich Communication Service (RCS), the successor of SMS supported by Android and iOS, lacks encryption in some implementations [25, 87]. Transport Layer Security (TLS) secures most Internet connections, limiting the potential impact of MitM attacks.

(3) Baseband Vulnerability Exploitation. An attacker wants to expose as much as possible about their target. With modern encryption widely used, another possibility to obtain this information is the exploitation of vulnerabilities on a target device to exfiltrate this data in decrypted form. Exploitation could result in information leakage or even Remote Code Execution (RCE).

2.2 Rogue Base Station Functionality

The term *IMSI catcher* is often used synonymously for an RBS [63]. Some literature exclusively uses this term when only extracting the IMSI [19, 58, 73]. We use the term RBS, which includes IMSI and SUCI catching functionality.

RBSes listen passively to surrounding cellular network activity and can actively impersonate genuine base stations (see Figure 2). They usually trick target devices into initiating a connection with them by advertising a stronger signal strength than all surrounding cells of a given network [63]. Recent work shows that the latest generation of smartphones still connects to 4G and 5G RBSes [60]. Modern smartphones will not necessarily reconnect to the strongest signal. In this case, jamming a legitimate base station's signal is required to trigger a full network scan and reconnection attempt. Real-world experiments show that all modern smartphones will connect to an RBS with jamming [60]. Using CellGuard, we observe similar behavior, where iPhones even attempt to connect to different mobile networks in case they lose connectivity to their network provider.

When connecting to an RBS, the phone will attempt to authenticate, which might leak identity information *(*1). Once authentication succeeds, the RBS acts as MitM, able to intercept and manipulate network traffic *(*2), and obtain a larger attack surface for RCE *(*3).

2.3 Known Attack Vectors

Cellular standards have a troubled history regarding security. We summarize relevant attacks leading to RBSes being a severe threat to mobile networks and end-users.

Downgrade Attacks. Recent mobile standards have effective attack mitigations in place. A circumvention method for these protections is to selectively force their target devices to use the oldest generation of insecure mobile 2G standards (downgrading) by jamming, e.g., other frequencies of standards from newer generations, allowing them to gather more information [63]. Baseband stacks still support 2G by default and connect to those cells, even though it was disabled in some countries [11]. Once 2G is deprecated, attackers might attempt to downgrade to 3G, which still has fewer mitigations than newer mobile network generations.

Missing Authentication and Integrity Checks. In 2G, only the User Equipment (UE) is authenticated, not the network, allowing attackers to impersonate existing network operators. Moreover, integrity protection is missing. These shortcomings allow for MitM attacks [81]. Since 3G, authentication is mutual, meaning that the network additionally is authenticated to the UE [19]. However, even 4G encrypts user data without integrity protection [68] and has flaws in the authentication protocol [67].

Authentication with Roaming Partners. Roaming enables the usage of the same cellular contract and phone number while traveling abroad. A serving network different from their home network offers them cellular services. The user is in a Visited Public Land Mobile Network (VPLMN), but only their Home Public Land Mobile Network (HPLMN) knows the pre-shared secrets required for successful mutual authentication. In older generations, authentication of the UE is delegated to the VPLMN. In 5G, the UE authentication is always executed in the HPLMN [33, 39]. Either way, authentication works transparently towards the UE, such that messages exchanged are the same regardless of roaming. The core network takes care of forwarding the respective requests.

An RBS collaborating with a malicious network operator with a roaming agreement with the target's home network can successfully authenticate [44]. An RBS can announce any country and operator, also called Mobile Country Code (MCC) and Mobile Network Code (MNC), tricking the UE into connecting to it but not recognizing the roaming procedure in the background. However, for successful authentication, the attacker might need to use the correct MCC and MNC when requesting authentication vectors from the HPLMN. They can hide this circumstance by overriding the display name shown on the UE using Network Information and Time Zone (NITZ), which is not part of the authentication procedure and, thus, not authenticated by the HPLMN.

This type of attack is similar to attacks on TLS certification authorities, allowing adversaries to issue valid certificates for any server. In TLS, the public certificate portion is shown to the user. A falsely issued certificate can be rejected through certificate pinning, and it is generally possible to detect such attacks. However, in cellular networks, roaming is mainly handled within the network core, and end-users are not involved in making trust decisions.

Identity Information Leakage. 3G and 4G networks expose IMSI and International Mobile Equipment Identity (IMEI) during authentication [19]. The latest generation of cellular standards, namely 5G NR, tries to address such issues. Its Subscription Permanent Identifier (SUPI) is comparable to the IMSI of former network generations. In contrast, the mobile device encrypts it with the network operator's public key before sending it to a base station. The encrypted SUPI is known as SUCI. The encryption scheme ensures that the SUCI is unique for each transaction. These measures undermine the operation of traditional IMSI catchers in 5G Standalone (5G SA) networks. *SUCI catchers* work in 5G SA networks and exploit weaknesses in the SUCI scheme to test if a given SUPI is nearby [19].

Another possibility of leaking personal linkable information appears through the sheer nature of wireless signals. An attacker can send silent SMS or hidden traffic over social networks while observing encrypted traffic sent over the air. If a victim is within the cell's location, correlated timings leak identities in 4G networks [73].

Firmware and Mobile Operating System Remote Code Execution. Attackers aim at stealthy exploits, requiring no user interaction while providing high reliability. Cellular baseband stacks pose a huge zero-click attack surface, as they parse much information directly on the baseband chip. Often, these are reliably exploitable because baseband chips lack modern mitigations for performance and cost reasons [15, 31, 32, 40, 48, 49, 81, 82, 85]. This enables an attacker to access decrypted SMS and phone calls, and they can use this as an entry point into the mobile operating system [42, 51].

2.4 Attacker Capabilities

Within the constraints of our attacker model, adversaries can establish an RBS. RBSes are sold as commercial products to various law enforcement agencies and are easy to assemble with commercial off-the-shelf equipment [22, 47]. We limit the scope to an adversary with well-defined and limited capabilities. The attacker can block, intercept, and modify signals sent over the air. It is not within the attacker's capabilities to physically access the target device and modify it any other way except for over-the-air signals. We assume that malware has not infected the smartphone beforehand.

Regular Attackers. We assume an attacker with a reasonable budget to be an individual or a minor criminal organization. They can use open-source software and publicly available attacks. The attacker does not have any organizational influence on network operators.

(1) Personal Information and Location Tracking. Identity information extraction can be achieved for 20 $ in 2G networks. IMSI numbers, country, brand, and operator can be recorded with an open-source project and a DVB-T stick [22, 58]. Attacks to observe identifiers in 4G and 5G networks require more expensive SDR platforms, e.g., the USRP B210, which costs 1800 $ [19, 73].

(2) Traffic Interception and Manipulation. MitM attacks are most cost-effective when first performing a downgrade attack and then using an open-source 2G base station [64, 76, 83]. Bypassing authentication in newer network generations requires flaws in the

authentication protocol. E.g., one setup exploiting such a flaw requires two USRP B210 [67].

(3) Baseband Vulnerability Exploitation. Executing arbitrary code on the target device is only possible with recent public exploits [40, 49, 85]. Due to coordinated disclosure, many devices are up-to-date and not vulnerable to such exploits. Yet, there is a significant patch gap; not all mobile devices receive firmware updates regularly.

State-sponsored Attackers. State-sponsored adversaries are effectively not limited in budget and can coerce network operator collaboration within their jurisdiction. Additionally, they have all regular attacker capabilities.

(1) Personal Information and Location Tracking. Tracking becomes easily accessible to state-sponsored attackers when assuming their collaboration with network operators.

(2) Traffic Interception and Manipulation. State-sponsored attackers can join as roaming partners and redirect traffic, even in 5G deployments. TLS limits lawful interception of network traffic.

(3) Baseband Vulnerability Exploitation. As state-sponsored attackers can afford prices for baseband exploits on the market [88], RCE becomes a realistic threat. We assume zero-days are only used if the attacker cannot achieve the desired goals otherwise and only after identifying a specific target.

2.5 Mitigation

The most effective mitigation is disabling 2G. This raises costs and reduces the scalability of state-sponsored attacks, as they require a more expensive radio setup and need access to the user-specific key material. Android 13 and iOS 17 in Lockdown mode support disabling 2G [59, 65]. 2G is still enabled by default, as most users desire network coverage and emergency services over security.

Yet, more recent cellular network generations are not secure against RBS attacks either. Especially state-sponsored attackers are challenging to detect. Often, various heuristics are combined to gain evidence. Previous studies outline approaches to defend against RBSes, and we compare existing tools with CellGuard in Section 7. Park et al. categorize them as follows [63]. *App-based* methods use baseband debugging data to detect RBSes. They monitor the parameters of the connected base station to find common patterns employed by adversaries [23]. CellGuard belongs to this category. Dedicated sensory devices solely focused on detecting RBSes enable *sensor-based* detection methods. Multiple sensors installed at fixed positions collaborate to identify threats [57]. Network operators can introduce *network-based* detection methods as they already possess data about their network's status and base stations. They can use this information to detect possible violations of their frequency spectrum by RBSes, initiate legal action against responsible entities, and try to block uncovered RBSes.

3 iOS BASEBAND EXPERIMENTATION FRAMEWORK BASETRACE

We build the experimentation framework BaseTrace for Apple devices with a baseband stack, such as the iPhone, cellular iPad, and Apple Watch models. We focus on the following features:

- Support for all baseband chips used in Apple devices.
- Basic functionality is available on non-jailbroken devices.
- Interaction with the baseband chip, allowing to send custom requests and receive responses.
- Integration into existing open-source tools, such as Wireshark and `libqmi` [53, 86].
- Extensibility for future use cases, including automated integration when Apple adds new cellular features, such as satellite communication.

While we use BaseTrace for RBS detection, various other cellular security and performance research applications are achievable with such a framework.

3.1 iOS Baseband Stack Architecture

Modern phones consist of various chips with different responsibilities, which increases performance and reduces battery consumption. Figure 3 shows the separation of the Application Processor (AP) and the baseband chip. The AP runs the operating system and apps. The baseband chip manages over-the-air communication and abstracts details from the AP. E.g., the baseband chip forwards SMS to the AP. There, the baseband daemon named *CommCenter* assembles SMS fragments and then forwards the SMS to the *Messages* app. With baseband chips getting more complex and customized for vendors, so do the underlying protocols. They are usually highly sophisticated and proprietary. The abstracted communication between baseband chip and AP is proprietary. Older iPhones with Intel basebands use the Apple Remote Invocation (ARI) protocol [41]; iPhones supporting 5G use the Qualcomm MSM Interface (QMI) protocol. *CommCenter*, the central baseband daemon on iOS handling phone calls and managing connectivity, parses both protocols.

3.2 Baseband Communication

ARI and QMI carry information that is useful for detecting RBSes. Both protocols hold details about low-level network properties of cells the phone connects to, including failed authentication attempts. Understanding these protocols is essential to build CellGuard.

Baseband Debug Profiles. Apple provides baseband debug profiles, increasing the log verbosity for baseband communication [5]. Installing them on a device requires no jailbreak. They support various hardware, including the Apple Watch and iPad. With a debug profile, logs contain all ARI and QMI protocol message contents in hexadecimal form and properties describing the currently connected cell. Logs can be captured live or retrospectively by creating a sysdiagnose [10]. Debug profiles and logs enable baseband research and CellGuard functionality on non-jailbroken devices.

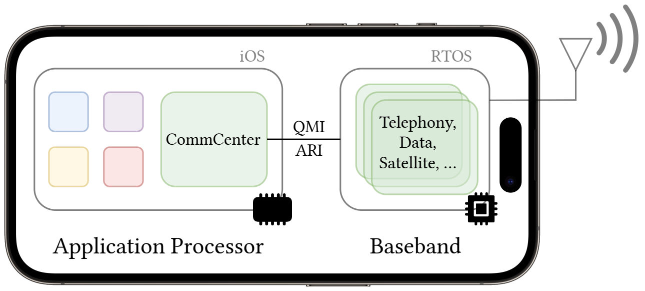


Figure 3: Cellular baseband stack on Apple devices.

Reverse Engineering Protocol Details. Protocol parsing is implemented in *CommCenter* and shared libraries. Apple keeps the ARI and QMI implementations closed-source. We reverse-engineer internals using the following approach. iOS processes use dispatch queues to parallelize data-intense processing tasks. Using dynamic analysis of dispatch queues with Frida on a jailbroken iPhone [6, 21, 66], we get backtraces of all queue handlers. They indicate which functions might process packets. Then, we statically analyze these functions with Ghidra and IDA Pro to determine their implementation details [1, 37]. Insights gained during reverse engineering on a jailbroken phone are also valid on non-jailbroken phones.

Qualcomm MSM Interface. Qualcomm is the largest vendor of cellular basebands by market share [43]. QMI uses a binary protocol and transmits information in packets. The QMI protocol format these chips use is publicly known and implemented, e.g., in the open-source library `libqmi` [53]. Each packet consists of a header followed by multiple elements of varying sizes. The headers define the operation to execute upon packet reception.

QMI services bundle operations of a similar category with a unique numeric identifier. One header field contains a numeric value for the QMI service the operation belongs to [53]. Assigning service names to their numeric identifiers allows us to understand the capabilities of a baseband. `libqmi` includes short and long names for some identifiers used by iOS, but not all. The iPhone's log shows short names for each QMI service. The library `libATCommandStudioDynamic.dylib` contains functions that convert service numbers to human-readable service names, specifically the functions `qmi::asShortString(qmi:: ServiceType)` and `qmi::asLongString(qmi:: ServiceType)`. Table 1 shows all services implemented on iOS 18 beta 3.

Observing and injecting packets requires further understanding of the underlying libraries. Previous research references the function `QMux::State::handleReadData` of the library `libATCommandStudioDynamic.dylib` that handles incoming QMI packets [20]. We locate the function `pci::transport::th::writeAsync` of the library `libPCITransport.dylib`, which sends outgoing packets. When iOS sends outgoing QMI packets to the baseband, it invokes this function with the packet's binary data. By calling this function with Frida, we can send arbitrary QMI packets to the iPhone's baseband.

Apple Remote Invocation. Intel modems use the previously reverse-engineered ARI protocol [42]. We intercept outgoing QMI and ARI packets using the function `writeAsync`. Incoming ARI packets pass the `AriHostRt::InboundMsgCB` function.

Cell Information. The *CommCenter* process parses and logs information about the current cell, independently from the underlying protocol. In addition to baseband interface packets, CellGuard processes this information.

3.3 Interacting with the Baseband Chip

BaseTrace does not only allow passive packet inspection but also active packet injection. These capabilities open BaseTrace for further baseband security analysis.

Table 1: QMI services present as of iOS 18 beta 3.

#	ID	Full Name
0x00	ctl	QMI Control Service
0x01	wds	QMI Wireless Data Service
0x02	dms	QMI Device Management Service
0x03	nas	QMI Network Access Service
0x04	qos	QMI Qos Service
0x05	wms	QMI Wireless Messaging Service
0x06	pds	QMI Position Determination Service
0x08	at	QMI Access Terminal Service
0x09	vs	QMI Voice Service
0x0A	cat	QMI Card App Toolkit
0x0B	uim	QMI User Identity Module
0x0C	pbm	QMI Phonebook Manager Service
0x1A	wda	QMI Wireless Data Administrative Service
0x22	coex	QMI Coexistence Service
0x24	pdc	QMI Persistent Device Service
0x28	787	QMI 5WI 787 Service
0x2A	dsd	QMI Data System Determination
0x2B	ssctl	QMI Subsystem Control
0x2C	mfse	QMI Modem File System Extended Service
0x30	dfs	QMI Data Filter Service
0x52	ms	QMI Media Service Extension
0xE1	audio	QMI Audio Service
0xE2	bsp	QMI Board Support Package Service
0xE3	ciq	QMI Carrier IQ Service
0xE4	awd	QMI Apple Wireless Diagnostics
0xE5	vinyl	QMI Vinyl Service
0xE6	mavims	QMI Mav 5WI Service
0xE7	elqm	QMI Enhanced Link Quality Metric Service
0xE8	p2p	QMI Mav P2P Service
0xE9	apps	QMI BSP APPS Service
0xEA	sft	QMI Stewie Service

Extending `libqmi` *for iPhones.* We modify `libqmi` to interface with outside programs over Unix Domain Sockets. Based on the findings of Section 3.2, we use Frida to set up a bidirectional communication channel with the iPhone's baseband. We connect both with a Python script and thus can utilize `libqmi`'s command line client. The client implements 94 state-retrieving operations, 22 of which the iPhone's baseband can successfully answer. This indicates that `libqmi` features slightly differ from Apple's implementation.

Apple-internal QMI Services. One operation implemented by the `libqmi` client allows requesting all QMI services provided by the baseband. The Qualcomm Snapdragon X55 baseband firmware in the iPhone 12 mini implements 39 services, eleven of which are not implemented by iOS frameworks parsing QMI. We successfully verify that we can address one of the services from our client and that its answer is valid. In contrast, the X55 baseband does not provide three services implemented by iOS. This service mismatch between iOS and baseband could stem from the baseband chipset being an off-the-shelf component developed by an external manufacturer rather than Apple itself. We include a detailed service comparison in Appendix D. As *CommCenter* is the same across all iPhones with a baseband[1], including older versions of Qualcomm, the additional iOS services could be legacy.

3.4 QMI Wireshark Dissector

The significant mismatch in services supported by iPhones versus those implemented by public tools makes it difficult to understand the meaning of many packets. We create a new Wireshark dissector that supports the Apple-internal services. Another Wireshark dissector already supports ARI [42].

Automated QMI Wireshark Dissector Creation. Wireshark is a packet inspection tool supporting a wide range of packet formats [86]. We adapt an existing QMI Wireshark dissector for USB modems to iPhones [61]. The dissector can parse all packets transmitted during various activities but cannot translate most numeric message identifiers or service identifiers to textual representations.

We add all known iOS QMI services to the dissector but rely on `libqmi` data for message identifier translations. Therefore, we expand `libqmi`'s database of QMI messages with iOS-custom packet data. All messages use the superclass `MessageBase` of the library `libQMIParserDynamic.dylib`. By creating a Frida script that intercepts the class constructors and prints their backtraces, we dynamically decipher the meaning of most QMI messages. Some QMI packets are not parsed within the *CommCenter* process and instead are sent to other processes. Using the `ipsw` tool [12], we analyze cross-references between system libraries and uncover the processes *locationd* and *WirelessRadioManagerd*. They also process binary QMI packets.

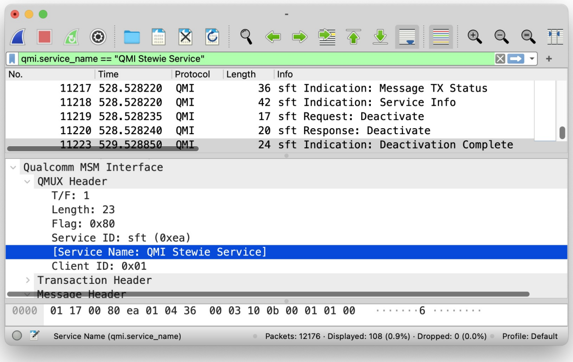
To automate the packet extraction, we devise a message identifier extraction workflow that extracts textual identifiers from iOS binaries using a custom Ghidra plugin and converts them into `libqmi` data.

Live Packet Capture. We provide three different approaches to feed the iPhone's QMI packets into Wireshark alongside the dissector. Two approaches allow us to monitor live QMI packets by extracting them with Frida or from the iPhone's system log. We adapted them from Kröll [41]. The third approach reads QMI packets from past system diagnosis log archives. Both log-based approaches also work on non-jailbroken devices.

Verifying New Features – Satellite Communication. The iPhone 14 lineup introduced the novel *Emergency SOS via Satellite* feature, allowing users to dispatch emergency text messages and share their locations over satellites [3]. The smartphone's Snapdragon X65 baseband implements the lower layers of satellite communication. A new *QMI Stewie Service* appears in the iOS 16 firmware. Stewie is Apple's codename for the satellite project [35]. We update our Wireshark dissector using the message identifier extraction workflow to decode satellite-related QMI packets. Figure 4 shows an excerpt of such a trace.

With this, we demonstrate the flexibility of BaseTrace and its applicability for research on the latest baseband features. The automated Wireshark dissector extraction makes BaseTrace future-proof, even if Apple introduced further novel protocols.

[1]A non-cellular version of *CommCenter* exists for certain iPads.


```
qmi.service_name == "QMI Stewie Service"
No.       Time        Protocol  Length  Info
    11217 528.528220  QMI           36  sft Indication: Message TX Status
    11218 528.528220  QMI           42  sft Indication: Service Info
    11219 528.528235  QMI           17  sft Request: Deactivate
    11220 528.528240  QMI           20  sft Response: Deactivate
    11223 529.528850  QMI           24  sft Indication: Deactivation Complete

Qualcomm MSM Interface
  QMUX Header
    T/F: 1
    Length: 23
    Flag: 0x80
    Service ID: sft (0xea)
    [Service Name: QMI Stewie Service]
    Client ID: 0x01
  Transaction Header
  Message Header

0000  01 17 00 80 ea 01 04 36  00 03 10 0b 00 01 01 00   ·······6 ········

Service Name (qmi.service_name)   Packets: 12176 · Displayed: 108 (0.9%) · Dropped: 0 (0.0%)   Profile: Default
```

Figure 4: Wireshark showing a satellite transmission with a filter for the satellite-related QMI service.

4 ROGUE BASE STATION DETECTION

We combine novel heuristics with heuristics proven effective in prior work to detect abnormal characteristics of RBS. With this, we overcome the limitations of previous app-based approaches [62]. In contrast to the latest app-based detection for Android [72], CellGuard supports both 5G networks and non-jailbroken devices.

The heuristics consume QMI & ARI packets and iOS-internal property sets describing cells, also called cell measurements. Our approach links the packets to cell measurements based on timestamps and allows for a live analysis of the combined datasets and a retroactive scan. Multiple detection criteria rely on data queries from Apple's internal cell database ALS.

4.1 Detection Criteria for CellGuard

We detect RBSes by checking for their typical characteristics. Each characteristic verification step outputs a numeric score. The combined score of a cellular measurement ranges from 0 to 100. We group cellular measurements based on their scores into three categories: suspicious (0-49), anomalous (50-94), and trusted (95-100).

Measurements are still trusted even when missing up to 5 points, as some criteria can subtract a few points, such as the bandwidth or distance calculation. E.g., even when users connect to a genuine base station, bandwidth might be lower in crowded areas, or location measurements might be inaccurate. We thus reduce false positives. If at least one primary detection criterion fails verification, the cell is usually anomalous, and users should keep an eye on it. We cannot determine with high certainty whether such base stations are a threat or genuine. Suspicious cellular measurements identify RBSes with a higher certainty as multiple major verification criteria fail. We employ the following characteristics:

(1) Existence of Cell in ALS Database (20 Points). ALS is Apple's cell location database. All Apple devices query the database to quickly determine an approximate location based on their currently connected cell. We reverse-engineer its endpoint to check if a cell exists in Apple's database. This verification criterion is based on the assumption that RBSes are usually active for a shorter period than genuine ones to evade detection [63]. Consequently, they are

more likely not to be included in Apple's database or, even if so, are quickly removed when they are no longer in operation. We award the cell's measurement 20 points if we receive a successful answer. Otherwise, no points are awarded.

(2) Distance between ALS Cell and User Location (20 Points). If the cell in question is present in ALS, the service returns the cell's approximate location. This verification criterion calculates the distance between the phone's and ALS locations. We detect adversaries cloning the identification of genuine cells present in Apple's database but locate their cloned cells far away from their genuine counterparts. Phones perform a full re-authentication and expose their identifiers when transitioning to a different cellular region, which adversaries can exploit to capture personal information [63]. If a cell is out of place, the likelihood of such an attack or a configuration error by adversaries increases. We calculate a corrected distance that accounts for inaccuracies of the measured and received locations and subtract the maximum range of a cellular tower, which is around 75 km [74].

In our practical experiments outlined in Section 6, we find that an iPhone's location accuracy decreases, especially when moving fast. This is a hardware-based limitation that we cannot overcome. However, we can correct location inaccuracies when awarding our score. Thus, depending on the user's movements, we calculate a percentage of how likely the cell's location is fraudulent by dividing the corrected distance by 150 km, a constant devised from practical experiments. The awarded points result from a multiplication, with the factors being the location's genuine percentage and the criterion's maximum number of points (20). The following formulas outline the score's calculation based on the distance d between the phone's locations and ALS' location in meters, the user's speed v_{user} in meters per second, and inaccuracies Δ_{loc} from the ALS and iOS locations in meters:

$$\begin{aligned}
r_{\text{cell}} &= 75\,000\,\text{m} \\
\Delta_{\text{speed}} &= (v_{\text{user}} \times 3.6/2)^{1.1} \times 1000\,\text{m} \\
d_{\text{corr}} &= d - r_{\text{cell}} - \Delta_{\text{speed}} - \Delta_{\text{loc}} \\
p_{\text{fraud}} &= \max\left(0, \min\left(d_{\text{corr}}/150\,000\,\text{m}, 1\right)\right) \\
s &= (1 - p_{\text{fraud}}) \times 20
\end{aligned}$$

(3) Comparison of Cell's Frequency Channel and PID with ALS (8 Points). ALS responses include additional data, such as the cell's Physical Cell ID (PID) number and Absolute Radio Frequency Channel Number (ARFCN) for some LTE cells. If available, we compare it to the observed cell's attributes. The verification criterion awards 6 points if the correct ARFCN is set and 2 points for a similar PID. We aim to detect unintended configuration errors that could expose RBSes.

Additionally, state-sponsored attackers operating without permission may exploit roaming (see Section 2.3). They can utilize unusual frequencies to set up the cloned RBS so as not to interfere with the operations of network operators. Our detection criteria would notice such a discrepancy.

(4) Bandwidth (2 Points). RBSes with older SDR hardware may not provide the full 20 MHz channel bandwidth. Yet, genuine cells may also decide to lower the channel bandwidth. Therefore, we limit the impact of this verification criterion by awarding 2 points for the full bandwidth of 20 MHz, 1 point for all bandwidths equal to or larger than 10 MHz, and 0 points for all bandwidths below.

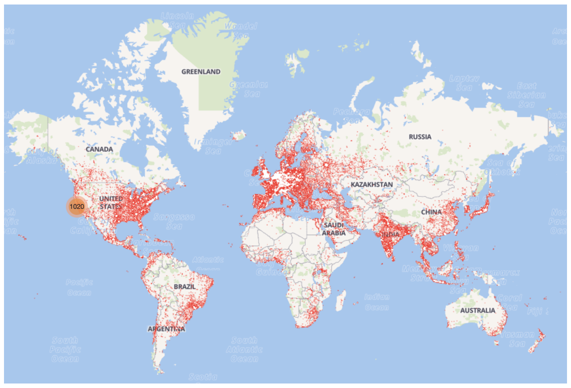
(5) Network Reject Packets (30 Points). When a UE tries to connect to an RBS of a regular attacker that does not bypass the authentication measures of newer network generations, the UE notices this issue after a while and stops further connection attempts. A smartphone's baseband chipset notifies the primary application processor about the rejection from the cellular network of the RBS. QMI notifies the application processor about this with a dedicated *Network Reject* packet. We search all recorded packets in the time frame of a cell measurement for a packet of that type. ARI has no dedicated reject packet. Instead, the reject reason is transmitted within an *IBINetRegistrationInfoIndCb* packet. RBSes emit a special pattern of two packets, one with the reject reason `IBI_NET_REGISTRATION_REJECT_CAUSE_FORB_PLMN` (forbidden public land mobile network) and the other one with the reason `IBI_NET_REGISTRATION_REJECT_CAUSE_INTERNAL_FAILURE`, sent shortly after another. If either the QMI packet or the two ARI packets are present, this criterion awards 0 points, as this is a strong indication for an RBS without authentication bypass measures as shown with an evil twin RBS in Section 6. Otherwise, it awards 30 points.

(6) Signal Strength (20 Points). UEs connect to RBSes with a high signal strength [63]. We extract the signal strengths measured by the UE from QMI and ARI packets and calculate an average value over the time frame the UE is connected to the cell under question. QMI transmits different signal strength measurements for each cellular generation, whereas ARI transmits abstract signal strength and signal quality values that resemble percentages. If they are above a considerable threshold verified in our lab environment and usually only achievable by standing right next to a cell tower, the verification criterion awards 0 points. Otherwise, it awards 20 points. Section 6 describes how we determine and evaluate the threshold values.

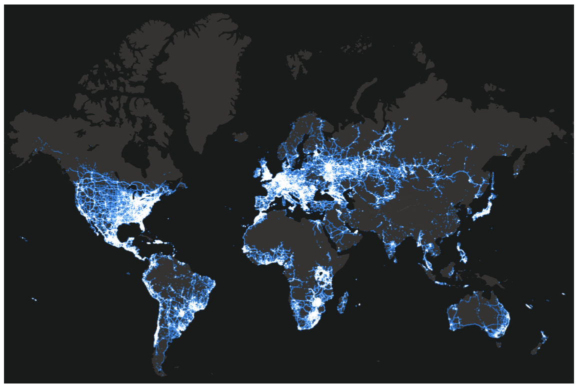
4.2 Cell Location Database Comparison

A sender's signal strength, the receiver's sensitivity, and the propagation properties of radio waves limit their reception radius. A single cell in a cellular network transmits radio waves and, thus, covers a specific geographical area with its signals. Phones use this limitation as an advantage. They can quickly determine an approximate location within the cell's radius based on its unique identification and independent of other technologies such as Global Navigation Satellite Systems (GNSSes). Beforehand, other entities must establish the relationship between the cell and its location. *Cell location databases* store this relational information. To maximize their use, they should accurately represent the current state of all cellular networks globally in service.

There are closed cell databases whose entire dataset has not been published. E.g., each major operating system vendor (Apple, Google, Microsoft) maintains its own cell database. None of them provide an official endpoint to query cells from their database, as the database is only intended for their devices. In contrast, open cell databases provide their full dataset for download. The open databases Mozilla Location Service (MLS) and OpenCelliD rely



(a) OpenCelliD [80]



(b) Mozilla Location Service [54]

Figure 5: Coverage provided by open cell location databases.

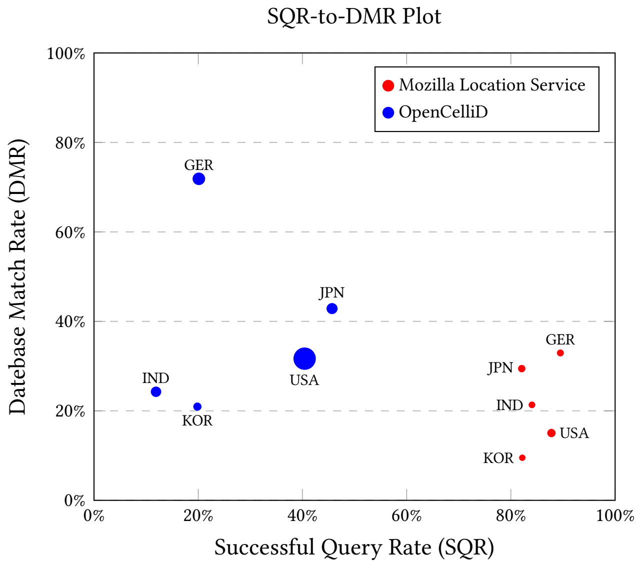


Figure 6: Correlation between SQR, DMR, and the country's number of cells (points' diameter).

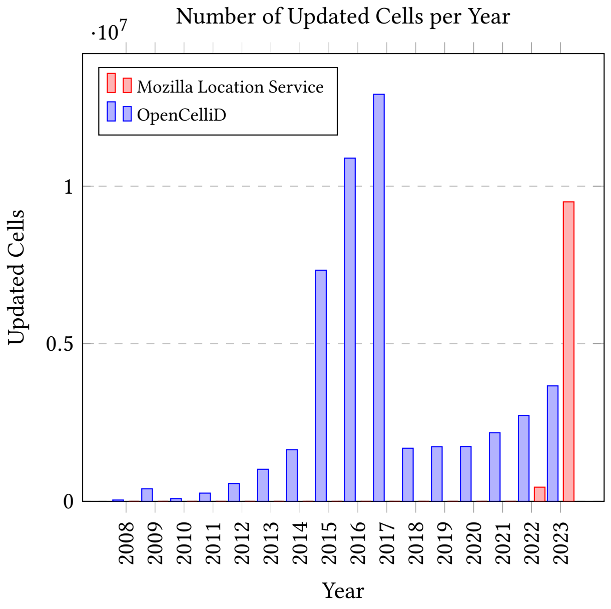


Figure 7: The number of cells per database and year.

on crowdsourced data [54, 80]. Since they rely on voluntary data donations, they lack significant coverage in some areas of the world and contain outdated measurements.

Apple Location Services. Apple's privacy policy for their location services explicitly states that Apple devices query the company's cellular database to determine approximate locations and contribute recent data [4]. iPhones communicate with the database's endpoint at `https://gs-loc.apple.com/clls/wloc`. It is similar to Apple's Wi-Fi location database [79]. All requests and responses start with Apple-custom binary headers, whereas the content is encoded using the binary data interchange format Protocol Buffers. Requests for cells use headers similar to Wi-Fi requests but with different protocol buffer fields. The *locationd* process is the origin of such requests and contains class structures that en- and decode these fields. We extract them using the static code analysis tool IDA Pro and build a Python client to request arbitrary cells from Apple's database. Our client can request cell and location data for the cellular technologies CDMA, TD-SCDMA, GSM, UMTS, LTE, and 5G. Each request contains a single cell, and if this cell is present in Apple's database, it returns location information for the cell and up to 99 neighboring cells. If the cell is absent from the database, the response only holds one invalid cell. In both cases, responses contain coarse locations for up to twenty nearby cellular regions. Invalid cell requests still contain a cell's area code, allowing ALS to answer such queries.

Evaluating Cell Database Coverage. The cell databases WiGLE, CellMapper, OpenCelliD, and Mozilla Location Service (MLS) contain 21.6 M, 5.4 M, 48.8 M, and 9.9 M cells, respectively [13, 16, 54, 80].

Only the last two databases are open, i.e., distribute their full dataset. However, they lack coverage in certain regions, such as Central Asia and Africa.

We assume that the ALS data is up-to-date and accurate as it includes contributions from up to two billion Apple devices [30]. Data from our multi-month real-world test of Section 6 supports this assumption as ALS has a miss rate of cells $\leq 1.6\%$. In further experiments, we find that new legitimate cells take multiple days to end up in Apple's database, making it effective for RBS detection while considering recent changes.

We evaluate the quality of the two open databases OpenCelliD and MLS in comparison with ALS by calculating their similarity using our novel ALS client. Our experiment considers Germany, India, Japan, South Korea, and the USA. It treats ALS as the ground truth, i.e., considers all cells in Apple's database active and those not out of service. For each open database and country combination, we sample 5000 cells based on the country's MCC from the respective database. Our client sequentially requests all 5000 cells from ALS and counts the number of cells in Apple's database. The number of successful requests divided by the number of overall requests is the Successful Query Rate (SQR). If the source-of-truth assumption is valid, the rate shows to what extent the open database contains out-of-service cells. A high value indicates a few inactive cells in the database. ALS returns neighboring cells for each requested cell, which we merge into a list and remove duplicates. The number of intersecting cells between the resulting list and the open database divided by the overall number of cells in the resulting list yields the Datebase Match Rate (DMR). This rate shows how much the open database lacks new or unreported cells. The higher its value, the more cells the open database covers.

We conducted the experiment on November 24, 2023, with the most recent database versions available. Figure 6 aggregates the experiment's results. The plot contains two clusters highlighting the differences in the data characteristics of both databases. OpenCelliD contains more cells than MLS in all selected countries, but fewer are still active. In contrast, MLS contains fewer cells than OpenCelliD in the selected countries, but more are still active. Approximately, MLS and OpenCelliD contain around 15 % and 72 % inactive cells not present in ALS, respectively, and miss around 22 % and 38 % of cells contained in ALS. Figure 7 illustrates that the open databases lack new contributions. Furthermore, old contributions are not purged, especially from the OpenCelliD database. We manually verify this for a site we physically visit and find that OpenCelliD contains reported cells based on a single 10-year-old measurement, with cells no longer existing. Both open databases lack a significant portion of cells making up the networks in each selected country and incorporate a non-negligible amount of cells that are out of service. Therefore, we assume that ALS is the best free source available, even though it is not a database we can download completely.

5 CELLGUARD IMPLEMENTATION

Users in the cross-hair of RBSes are often unaware they are under attack [81]. We create an iOS app named CellGuard to raise awareness about potentially dangerous base stations. Our app continuously records information about the phone's location and its connected cells. It combines the datasets and verifies them with ALS. Then, it combines ALS results and low-level baseband information to calculate a score defined in Section 4.1. If the verification score is *anomalous* or *suspicious*, CellGuard alerts its user.

5.1 iOS App

We create the CellGuard iOS app that implements cell verification according to the criteria in Section 4.1. Figure 8 visualizes the app's architecture. On non-jailbroken devices, it gathers cell information from sysdiagnoses, while on jailbroken devices, all data is collected live with two tweaks. In parallel, the app tracks the phone's location using the iOS Core Location framework. It stores both datasets using iOS' Core Data framework and links locations and cells based on their timestamps. CellGuard verifies the combined dataset using ALS and notifies users upon verification failures.

The app supports iOS 14 to 18 and is distributed with Apple's packaging format IPA. Users can install IPA files with TrollStore [28] or AltStore [78] on all supported devices. We plan to release a version for non-jailbroken iPhones on Apple's App Store—currently, a beta release is available over Apple's TestFlight app. Since users might not be familiar with RBSes, CellGuard has a detailed explanation shown on installation (see Appendix A).

We implement the app using Apple's Swift programming language and large parts of its user interface with the company's SwiftUI framework. The iOS apps AirGuard [36] and SignalReborn [84] inspired elements of CellGuard's design. The app's user interface consists of three tabs, Figure 1 shows two of them. In the Summary tab, a green, yellow, or red risk indicator shows the current risk level for the user at a glance. Below that, CellGuard displays information about the connected cell. Users can open the app's settings, change its permissions, or export their collected data for later analysis. The Map tab displays all cells received from ALS and shows more information upon interaction with markers. Another tap on the information popup opens a new view with a map showing the phone's locations when connected to the cell, data from ALS, and the ability to inspect individual cell measurements and their scoring. Nearby cell markers form groups based on the map's zoom level. Figure 9 shows the app's third Packets tab with a local dissector. The packets can be filtered by different criteria and are constantly updated.

Non-jailbroken iPhones (Sysdiagnose). Apple grants no permission for ordinary applications to read advanced cell information or access QMI and ARI packets. The user creates a sysdiagnose with a baseband debug profile installed as a workaround [5]. Sysdiagnoses are archive files meant to be shared with Apple to analyze bugs. The archives include the system's log data encoded with a proprietary format. The baseband debug profile leads to QMI and ARI packets being included in the sysdiagnose's logs. CellGuard integrates a sysdiagnose log archive parser to extract relevant information from logs, as described in Section 3.2 [50].

CellGuard guides users through this process with verbose explanations as shown in Appendix B and C. Upon sysdiagnose import, CellGuard retrospectively parses and analyzes all information. Sysdiagnoses contain information collected over up to three days. How long a sysdiagnose reaches back depends on how many log messages were created, their size, and further device-specific factors like disk space [9]. After 21 days, the debug profile expires.

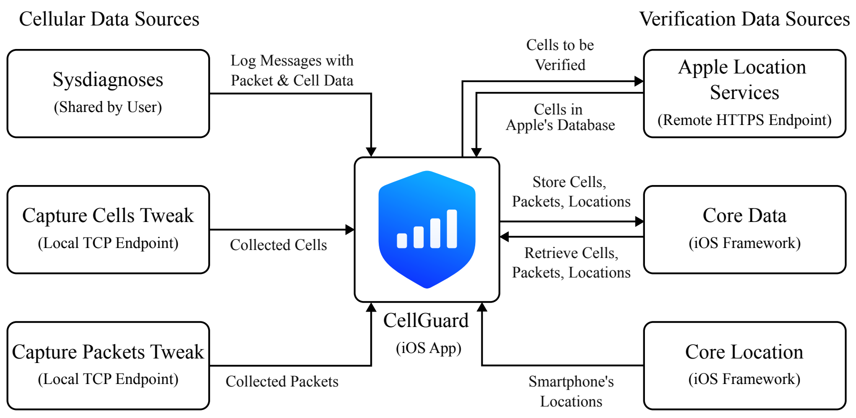
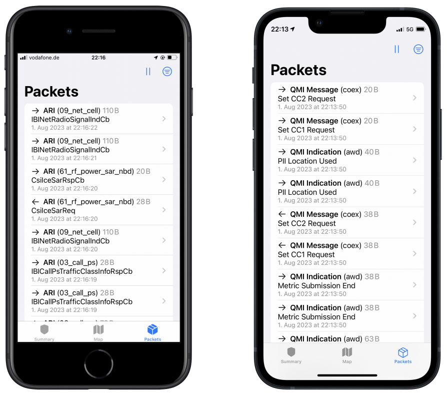


Figure 8: A data-flow diagram of CellGuard's architecture.


vodafone.de 22:16

Packets

→ **ARI** (09_net_cell) 110 B
IBINetRadioSignalIndCb
1. Aug 2023 at 22:16:22

→ **ARI** (09_net_cell) 110 B
IBINetRadioSignalIndCb
1. Aug 2023 at 22:16:21

→ **ARI** (61_rf_power_sar_nbd) 20 B
CsiIceSarRspCb
1. Aug 2023 at 22:16:20

← **ARI** (61_rf_power_sar_nbd) 28 B
CsiIceSarReq
1. Aug 2023 at 22:16:20

→ **ARI** (09_net_cell) 110 B
IBINetRadioSignalIndCb
1. Aug 2023 at 22:16:20

→ **ARI** (03_call_ps) 28 B
IBICallPsTrafficClassInfoRspCb
1. Aug 2023 at 22:16:19

→ **ARI** (03_call_ps) 28 B
IBICallPsTrafficClassInfoRspCb
1. Aug 2023 at 22:16:19

22:13 5G

Packets

→ **QMI Message** (coex) 20 B
Set CC2 Request
1. Aug 2023 at 22:13:50

→ **QMI Message** (coex) 20 B
Set CC1 Request
1. Aug 2023 at 22:13:50

→ **QMI Indication** (awd) 40 B
PII Location Used
1. Aug 2023 at 22:13:50

→ **QMI Indication** (awd) 40 B
PII Location Used
1. Aug 2023 at 22:13:50

← **QMI Message** (coex) 38 B
Set CC2 Request
1. Aug 2023 at 22:13:50

← **QMI Message** (coex) 38 B
Set CC1 Request
1. Aug 2023 at 22:13:50

→ **QMI Indication** (awd) 38 B
Metric Submission End
1. Aug 2023 at 22:13:50

→ **QMI Indication** (awd) 38 B
Metric Submission End
1. Aug 2023 at 22:13:50

→ **QMI Indication** (awd) 63 B

Figure 9: CellGuard's internal dissectors for ARI and QMI.

CellGuard recognizes this and reminds the user to refresh the debug profile.

Using sysdiagnoses, users of non-jailbroken iPhones with the latest security patches and even with Lockdown mode enabled can use CellGuard and regularly check if their phone recently connected to an RBS. CellGuard regularly reminds them to import new sysdiagnoses for scanning. We confirm that this approach works up to the latest iOS version, which is iOS 18.0 beta 3 at the time of writing this paper.

Jailbroken iPhones (Tweaks). Tweaks are small programs that modify the default behavior of iOS and require a jailbroken iPhone to work. We create two tweaks using the cross-platform build system Theos to collect data consumed by CellGuard [24]. Both tweaks operate on iOS 14 to 18 and attach to the *CommCenter* process.

The *Capture Cells* tweak collects iOS-internal data about all cells an iPhone connects to. The function `legacyInfo` of the class `CTCellInfo` returns properties describing the currently selected cell. Our tweak intercepts the function's invocations and caches the captured cell data in a file. It exposes a TCP server on a local port, which our iOS app uses to query the recent cell data. As the `CTCellInfo` class is part of the high-level *Core Telephony* iOS framework, the tweak supports iPhones with Qualcomm and Intel basebands.

The *Capture Packets* tweak records all QMI and ARI packets exchanged between the iPhone's baseband and its primary application processor. It intercepts the invocation of functions transmitting and receiving binary packets of both protocols. Section 3.2 describes how we reverse-engineered iOS' receive and transmit architecture for QMI packets, while Section 3.2 outlines the relevant iOS functions for ARI packets. The tweak encodes the binary packets and additional metadata and stores both in a cache file. It exposes a TCP server on the local port 33067, which our iOS app uses to query the recent packets from the tweak. All iPhones supported by CellGuard, starting with the iPhone 6s, use a baseband chipset manufactured by Qualcomm or Intel. As the tweak supports both of their baseband protocols, it can function on all iPhones in a similar manner.

Mobile Packet Analysis. Anomalous actions and difficult-to-explain baseband behavior might happen when researchers do not have a laptop with Wireshark at hand. Thus, we integrate an ARI and QMI dissector into CellGuard, as shown in Figure 9. The dissector shows basic packet information for both protocols in a human-readable format. The filter function allows users to search for specific packet types, enabling basic packet analysis directly on the phone.

Data Export. Data collected can be exported into a human-readable CSV format. Exported data contains connected cells, the cell cache, locations, and all packets. With this information, retrospective data analysis is enabled, independent of sysdiagnoses. E.g., researchers can import baseband packets collected in the field into Wireshark at home and perform a deeper analysis of the observed phenomena in combination with the exact cell data recorded by their device.

Apple Location Services Integration. CellGuard queries ALS on the go whenever a user encounters a new cell. Cells update regularly, and so does the ALS cell database (see Section 4.2). Users can manually re-run the verification of a potentially malicious cell in CellGuard to see if any of the ALS metrics changed.

We decide against downloading the ALS cell database in its entirety and caching it locally. However, that means that an iPhone must have an Internet connection to complete a cell's verification. For devices running in jailbroken mode, this can delay notifications about malicious cells if an RBS does not offer Internet services. Requests to ALS use TLS, preventing an RBS from tampering with the service's response if offering Internet services.

5.2 Recommended Mode of Operation

CellGuard works on jailbroken and non-jailbroken devices. Depending on the user's needs, both modes of operation have different advantages and disadvantages. In either case, CellGuard only warns after a user connected to an RBS. Further actions are left to the user—such as enabling flight mode or reporting illegal surveillance.

Due to the limitations of openly available interfaces on iOS, live analysis is only possible on jailbroken devices. When using

CellGuard as a sensor for potential RBSes abuse, we thus recommend using it on a jailbroken iPhone. CellGuard works on the iPhone 7 and 8, which support the *palera1n* jailbeak [18]. Refurbished iPhones of these generations cost around USD 100. Thus, CellGuard is an effective, portable, and affordable RBS sensor.

Most baseband attacks target specific users, so an RBS might not establish connections to potentially jailbroken device models. Thus, we recommend installing CellGuard to a user's primary, non-jailbroken device in Lockdown mode, with 2G disabled. The delay in getting warnings about potentially malicious cells is higher. However, it still helps users to take further actions. On a user's primary device, when suspecting a compromised iPhone, they can proceed with forensic analysis to extract further indicators of compromise. Moreover, as many exploits are non-persistent, users can reboot their iPhone [56]. However, rebooting an iPhone might remove some indicators of compromise due to an exploit's non-persistent behavior. Which action is ideal thus depends on the user's threat model.

5.3 Privacy, Transparency, and Modification

CellGuard processes all data locally. Thus, the privacy-sensitive location history and low-level baseband packet logs stay on the iPhone. This differs from other iOS forensics solutions, which extract data from the iPhone to then analyze it on an external PC or in the cloud [38, 55]. Furthermore, CellGuard automatically deletes privacy-sensitive location and packet data after three and seven days, respectively. Users can adjust both retention periods in the app.

CellGuard is open source, which allows users to verify CellGuard's cell measurement algorithm and make adaptions for other metrics. In particular, technically experienced users can tune the classification parameters. Free Apple developer accounts support all permissions required by the iOS app [7]. Anyone with an Apple ID and Xcode can modify CellGuard to their needs.

6 CELLGUARD EVALUATION

We evaluate CellGuard on ten datasets collected on five iPhones, three with Qualcomm and two with Intel basebands. They collected data while accompanying us on our daily trips in the previous months through six European countries. Within Germany, the iPhone connects to the two major German cellular networks of Deutsche Telekom and Vodafone, and outside to their local roaming partners.

iPhone Location Accuracy. Early CellGuard prototypes incorporated a more straightforward threshold-based criterion for verifying the distance between a phone and ALS location, which considered inaccuracies introduced by both. We noticed that the app generated numerous false positives during initial field tests. These stemmed from the iPhone's location being inaccurate in the range of up to tens of kilometers, especially when traveling at speeds of 130 km/h or above. This inaccuracy propagated into the cell scores.

As a countermeasure, we increased the positional accuracy requested from iOS from a hundred to ten meters, even when CellGuard operates in the background. Furthermore, we introduced the novel distance criterion of Section 4.1 used by the latest version of CellGuard. The criterion calculates with high error margins and grants an additional margin based on the user's speed to account for iOS' lack thereof. The cell's corrected distance is the raw distance minus all margins and the maximum cell range of 75 km. If this corrected distance exceeds zero, the verification awards fewer points for the criteria, with 150 km being the absolute distance at which a cell's location is too far away. At this distance, an RBS becomes plausible [74].

We found that this revised approach works well even with high-speed trains. However, suppose a 2nd Gen iPhone SE is used that does not expose speed measurements to CellGuard and records inaccurate location data while flying. In that case, the algorithm cannot always compensate for the difference between the location of a connected cell on the ground and the iPhone's recorded position. This circumstance causes a high number of anomalous cells in dataset #8.

Detection Results. Table 2 lists the resulting numbers from our field test. No cell was marked as *suspicious*. CellGuard found multiple cells where some detection criteria were met, leading to a *anomalous* rating. The cells marked as anomalous might be RBSes or be affected by network reconfigurations. Confirmation would require physical analysis of the base station site to determine if it is accessed by someone other than the legitimate network operator.

ALS Accuracy. Suppose a cell identification is not present in ALS; our verification algorithm subtracts 20 points from the cell's score, as described in Section 4.1, and labels it as anomalous or even suspicious if verifying more criteria fails. Due to our evaluation's low number of anomalous cells, we conclude that ALS covers a significant percentage of cells in our datasets from European cellular networks.

Upon repeated evaluation of our datasets, we find that the cells in ALS change over time. Cells deemed missing in a past evaluation are sometimes successfully verified upon a later reevaluation. Since new cells took multiple days to appear in ALS, we assume that RBSes installed over a short period will not end up in the ALS database, even if operated by state-sponsored attackers. We presume most unknown cells are harmless since network operators regularly update their infrastructure.

Lab Setup with Evil Twin Rogue Base Station. We set up a custom RBS in a controlled lab environment with a USRP X310 inside a Faraday cage [27]. We use *open5gs* and *srsRAN* for the 5G setup, *srsLTE* for the 4G setup, and *YateBTS* for the 3G setup [45, 76, 77]. The RBS is an evil twin of a legitimate nearby base station, cloning the cell information (MCC, MNC, Cell ID). This evades the ALS-based detector of CellGuard, as other iPhones have reported the cell information to the ALS database at nearby locations. We then observe connection attempts to this evil twin RBS using an iPhone 12 mini (Qualcomm modem) and an iPhone SE 2nd Generation (Intel modem). Both have a valid SIM card of the network operator with the same MCC and MNC. Our setup mimics a regular attacker, meaning we cannot access the key material on the SIM card.

Previous work [60] employed a similar setup for 4G and 5G IMSI catching. Palamà et al. found that all 4G and 5G devices connect to an evil twin of a legitimate base station. If the legitimate base station is still in range, some devices only connect if signals of the legitimate base station are jammed. Our setup confirms these

Table 2: Evaluation results of nine datasets, device numbers indicate different devices from the same model.

			Days		Unique Cells				Verification Data Points		
#	iPhone	Baseband	Active	Total	Suspicious	Anomalous	Trusted	Total	Cell Meas.	Packets	Locations
1	12 mini	Qualcomm Snapdragon X55	6	6	0.0 %	2.3 %	97.7 %	440	3661	1 622 064	43 576
2	12 mini	Qualcomm Snapdragon X55	3	3	0.0 %	0.4 %	99.6 %	459	2220	415 199	8275
3	12 mini #2	Qualcomm Snapdragon X55	4	4	0.0 %	1.0 %	99.0 %	674	2327	460 695	20 613
4	13 mini	Qualcomm Snapdragon X60	3	3	0.0 %	6.7 %	93.3 %	45	2003	414 012	21 658
5	2nd Gen SE	Intel XMM 7660	3	3	0.0 %	0.0 %	100.0 %	324	1486	136 005	10 095
6	2nd Gen SE #2	Intel XMM 7660	4	4	0.0 %	0.3 %	99.7 %	289	7256	617 824	27 541
7	2nd Gen SE #2	Intel XMM 7660	21	26	0.0 %	0.6 %	99.4 %	1652	20 260	1 800 338	248 120
8	2nd Gen SE #2	Intel XMM 7660	26	30	0.0 %	9.0 %	91.0 %	796	15 482	884 220	114 737
9	2nd Gen SE #2	Intel XMM 7660	17	25	0.0 %	0.0 %	100.0 %	430	11 635	527 437	90 555
10	2nd Gen SE #2	Intel XMM 7660	67	83	0.0 %	0.3 %	99.7 %	8271	83 244	4 693 314	747 199

findings. The iPhone SE 2nd Generation connects to the 3G and 4G RBS, but its Intel modem does not support 5G [8]. The same holds for the iPhone 12 mini with a Qualcomm modem that supports 5G—the carrier profile belonging to the network operator we spoofed does not support 5G SA mode.

After establishing an initial connection, the iPhones attempt mutual authentication. Without knowing the key material from the SIM card, our evil twin RBS cannot complete authentication. Independent of the wireless standard used, we observe the same error messages indicated by the baseband chips. As outlined in Section 4.1, Qualcomm chips send a *Network Reject* packet, while Intel chips send *IBINetRegistrationInfoIndCb* packets with the reasons `IBI_NET_REGISTRATION_REJECT_CAUSE_FORB_PLMN` and `IBI_NET_REGISTRATION_REJECT_CAUSE_INTERNAL_FAILURE`. CellGuard can reliably detect failed UE authentication attempts in our lab tests by inspecting the baseband and iOS communication.

We observed a few failed authentication attempts appearing during our field tests. Their appearance could also be grounded in valid reasons, such as incomplete handshakes due to bad signal reception.

Received Signal Strength. Advertising a higher signal strength is the primary attack vector of RBSes for forcing UEs to connect with them, as described in Section 2.2. One of our detection criteria of Section 4.1 extracts the UE's perceived signal strength values from QMI and ARI packets. We devised and evaluated thresholds using our custom RBS. Once they are exceeded, a connection to an RBS is likely.

ARI packets transmit the UE's signal strength and quality measurements as percentages. We devise a threshold of 65 % and 85 % for signal strength and quality.

QMI packets contain different raw measurements depending on the current Radio Access Technology (RAT). If 2G (GSM) is active, we can only extract the Received Signal Strength Indicator (RSSI), for which we determine a threshold of −60 dBm. For 4G (LTE), we can extract the RSSI, Reference Signal Received Quality (RSRQ), Reference Signal Received Power (RSRP), and Signal-to-Noise Ratio (SNR) measurements. Based on our experiments, we set a combined threshold of −70 dBm RSSI, −4 dB RSRQ, −100 dBm RSRP, and 200 SNR for the respective measurements. We consider it exceeded only if all measurements are larger than their threshold values. For 5G (NR), the QMI packets contain all measurements present in 4G, except the RSRP. We use a similar combined threshold without the RSRP.

Local regulations limit wireless signal strength allowed for base stations. At the intended receiver, the signal strength decreases at an approximately inverse cubic relation with the distance to the base station. Due to these factors, the expected signal strength at the receiver is relatively low. Our field tests find that our signal strength thresholds produce few false positives while reliably detecting our RBS in the lab.

7 RELATED WORK

RBS detection has a history of Android apps and SDR-based solutions. A detailed study on app-based RBS detection compared smartphone apps and analyzed their shortcomings in practice [63]. Since then, to the best of our knowledge, there have been neither significant updates to these apps nor newly released apps overcoming the flaws pointed out in the study. In the following, we compare existing apps and SDR-based approaches with CellGuard.

CellGuard is the first app-based approach for iOS. It supports iOS 14 to 18 and is built to be compatible with new features. The sysdiagnose feature we use for baseband packet analysis was introduced in iOS 10 [2]. Thus, it is a stable feature and will likely be available for future iOS releases. A side-effect of using sysdiagnoses is that CellGuard supports non-jailbroken phones.

Most previous app-based solutions for Android are no longer maintained (*Cell Spy Catcher*, *GSM Spy Finder*, *AIMSICD*, *Darshak*) and discontinued development several years ago [14, 17, 29, 75]. *SnoopSnitch* still gets UI updates, but the more accurate packet-based detection that requires a rooted phone only supports Android 4.1.2 to 7.2.1 [71, 72]. Thus, users of these apps risk being exploited by unpatched vulnerabilities that have been disclosed since 2016. *SnoopSnitch*'s detection criteria were last updated in 2015 [69, 70]. Even when these apps were recent, packet-based analysis only worked on rooted Android phones [14, 17, 72], further enhancing the risk for users. In contrast, CellGuard works on the latest iOS with all security patches applied and even with Lockdown mode enabled.

FBS-Radar aims to detect unauthenticated 2G RBSes that send SMS spam [46]. Here, RBSes are detected through crowd-sourced collection of the spam messages inside an app rather than low-level cell properties. Thus, *FBS-Radar* cannot detect RBSes that have other goals than sending spam, e.g., location tracking, network traffic interception, or RCE. Furthermore, CellGuard does not require users to share SMS with a cloud service for RBS detection.

CellGuard is the first to fully support 2G, 3G, 4G, and 5G. Rudimentary 4G support was integrated into *SnoopSnitch* [69], while all other app-based approaches only support 2G and 3G [14, 17, 29, 75]. Integrating 5G support into SDR-based approaches is challenging. *SeaGlass* uses a GSM modem for RBS detection [57], as the authors assume only 2G RBSes exist. *Crocodile Hunter* is fully SDR-based and aims to study 4G-only RBSes, but the project development was discontinued in 2022 [26].

8 CONCLUSION

RBSes threaten mobile security—they are the entry point to track users, intercept network traffic, and gain RCE. CellGuard defends against these attacks by warning users immediately when an RBS is suspected to be present. It works on non-jailbroken iPhone models, starting with the 6s up to the iPhone 15, running on iOS 14 to 18. We evaluated CellGuard on 11 813 base stations collected over multiple months on different iPhones and found 187 cells labeled as anomalous by our novel verification algorithm based on six detection criteria. Most commonly, the device's location was inaccurate, or cells were missing from the ALS database. CellGuard collects additional indicators, such as failed authentication attempts, and notifies users about the score. Using our own RBS, we verified its effectiveness and detection capabilities in a controlled environment.

Detection is enabled by our research framework BaseTrace to analyze the communication between iOS and the baseband chip, which is valuable for research beyond RBS detection. BaseTrace supports the analysis of QMI for Qualcomm chips and ARI for Intel chips. We enhanced QMI support for iOS with automated Wireshark dissector generation and added a bridge for `libqmi` that allows sending previously undocumented commands to the baseband.

We enrich our detection with cell location information. Assuming that RBSes are only installed temporarily to avoid detection, they can be found by comparing them to a cell location database. Instead of using the open MLS and OpenCelliD databases, we reverse-engineered Apple's HTTPS endpoint for their ALS database. Our evaluation comparing cell location databases shows that ALS is highly accurate while containing significantly more cells than other open databases, as all Apple users contribute to it automatically.

CellGuard will not only bring RBS detection to end-users, but the frameworks we built around it enable the creation of further tools for science.

AVAILABILITY

A detailed guide on how to set up and utilize CellGuard is hosted on https://cellguard.seemoo.de. As of publishing this paper, Apple approved CellGuard for beta testing. Users can easily install CellGuard via Apple's official TestFlight app on non-jailbroken iPhones without compiling the source code.

CellGuard and the underlying framework BaseTrace are open-source and available on https://github.com/seemoo-lab/CellGuard.

ACKNOWLEDGMENTS

This work has been co-funded by the German Federal Ministry of Education and Research and the Hessian State Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE. This work has been co-funded by the Federal Ministry of Education and Research of Germany in the project Open6GHub (grant number: 16KIS014).

REFERENCES

[1] National Security Agency. 2023. Ghidra. https://ghidra-sre.org
[2] Apple. 2016. WWDC 2016 – Unified Logging and Tracing. https://devstreaming-cdn.apple.com/videos/wwdc/2016/721wh2etddp4ghxhpcg/721/721_unified_logging_and_activity_tracing.pdf
[3] Apple. 2022. Emergency SOS via satellite available today on the iPhone 14 lineup in the US and Canada. https://www.apple.com/newsroom/2022/11/emergency-sos-via-satellite-available-today-on-iphone-14-lineup/
[4] Apple. 2022. Location Services & Privacy. https://www.apple.com/legal/privacy/data/en/location-services/
[5] Apple. 2023. Bug Reporting Profiles and Logs. https://developer.apple.com/bug-reporting/profiles-and-logs/
[6] Apple. 2023. Dispatch. https://developer.apple.com/documentation/DISPATCH
[7] Apple. 2024. Choosing a Membership. https://developer.apple.com/support/compare-memberships/
[8] Apple. 2024. iPhone SE (2nd generation) - Technical Specifications. https://support.apple.com/kb/SP820?locale=en_US
[9] Apple Developer Forum. 2020. Log Retention on iOS Sysdiagnose. https://forums.developer.apple.com/forums/thread/123393
[10] Nikias Bassen and Martin Szulecki. 2023. libimobiledevice. https://libimobiledevice.org
[11] Pete Bell. 2023. 2G and 3G Shutdowns Continue. https://blog.telegeography.com/2g-and-3g-shutdowns-continue
[12] blacktop. 2023. ipsw. https://github.com/blacktop/ipsw
[13] bobzilla, arkasha, and uhtu. 2023. WiGLE: Wireless Network Mapping. https://www.wigle.net
[14] Ravishankar Borgaonkar and Swapnil Udar. 2023. Darshak. https://github.com/darshakframework/darshak
[15] Amat Cama. 2018. A walk with Shannon: A walkthrough of a pwn2own baseband exploit. https://www.youtube.com/watch?v=6bpxrfB9ioo
[16] Cellmapper. 2023. Cellular Coverage and Tower Map. https://www.cellmapper.net
[17] CellularPrivacy. 2023. Android IMSI-Catcher Detector. https://github.com/CellularPrivacy/Android-IMSI-Catcher-Detector
[18] Nick Chan, Lakhan Lothiyi, Nebula, Mineek, and Tom. 2023. palera1n. https://github.com/palera1n/palera1n
[19] Merlin Chlosta, David Rupprecht, Christina Pöpper, and Thorsten Holz. 2021. 5G SUCI-Catchers: Still Catching Them All?. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (Abu Dhabi, United Arab Emirates) *(WiSec '21)*. Association for Computing Machinery, New York, NY, USA, 359–364. https://doi.org/10.1145/3448300.3467826
[20] Jiska Classen. 2020. Fuzzing the phone in the iPhone. https://media.ccc.de/v/rc3-11358-fuzzing_the_phone_in_the_iphone
[21] Jiska Classen. 2023. Frida Scripts. https://github.com/seemoo-lab/frida-scripts
[22] Joseph Cox. 2018. With $20 of Gear from Amazon, Nearly Anyone Can Make This IMSI-Catcher in 30 Minutes. https://www.vice.com/en/article/gy7qm9/how-i-made-imsi-catcher-cheap-amazon-github
[23] Adrian Dabrowski, Nicola Pianta, Thomas Klepp, Martin Mulazzani, and Edgar Weippl. 2014. IMSI-Catch Me If You Can: IMSI-Catcher-Catchers. In *Proceedings of the 30th Annual Computer Security Applications Conference* (New Orleans, Louisiana, USA) *(ACSAC '14)*. Association for Computing Machinery, New York, NY, USA, 246–255. https://doi.org/10.1145/2664243.2664272
[24] Adam Demasi. 2021. Documentation Home. https://theos.dev/docs/
[25] Zak Doffman. 2024. Apple's iOS 18 RCS Release Gets Closer—Encryption Still A Problem. https://www.forbes.com/sites/zakdoffman/2024/03/30/new-apple-iphone-16-pro-max-and-ios-18-leak-googles-imessage-warning/
[26] Electronic Frontier Foundation. 2022. Crocodile Hunter. https://github.com/efforg/crocodilehunter
[27] Ettus Research. 2024. USRP X310. https://www.ettus.com/all-products/x310-kit/
[28] Lars Fröder. 2023. TrollStore. https://github.com/opa334/TrollStore
[29] Galan. 2023. GSM Spy Finder. https://apk.support/app/kz.galan.antispy
[30] William Gallagher. 2023. Apple says 2 billion of its devices are in active use. https://appleinsider.com/articles/23/02/02/two-billion-apple-devices-are-

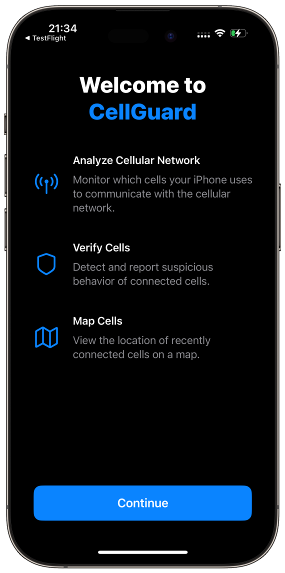
in-active-use
[31] Nico Golde. 2018. There's Life in the Old Dog Yet: Tearing New Holes into Intel/iPhone Cellular Modems. https://comsecuris.com/blog/posts/theres_life_in_the_old_dog_yet_tearing_new_holes_into_inteliphone_cellular_modems/
[32] Nico Golde and Daniel Komaromy. 2016. Reverse Engineering and Exploiting Samsung's Shannon Baseband. https://comsecuris.com/blog/posts/shannon/
[33] Graeme Green. 2021. 5G Security when Roaming – Part 1. *Mpirical* (2021). https://www.mpirical.com/blog/5g-security-when-roaming-part-1
[34] GSMA Intelligence. 2023. The Mobile Economy 2022. https://www.gsma.com/mobileeconomy/wp-content/uploads/2022/02/280222-The-Mobile-Economy-2022.pdf
[35] Mark Gurman. 2021. Apple Plans to Add Satellite Features to iPhones for Emergencies. https://www.bloomberg.com/news/articles/2021-08-30/apple-plans-to-add-satellite-features-to-iphones-for-emergencies
[36] Alexander Heinrich, Niklas Bittner, and Matthias Hollick. 2022. AirGuard - Protecting Android Users from Stalking Attacks by Apple Find My Devices. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (San Antonio, TX, USA) *(WiSec '22)*. Association for Computing Machinery, New York, NY, USA, 26–38. https://doi.org/10.1145/3507657.3528546
[37] Hex-Rays. 2023. IDA Pro. https://hex-rays.com/ida-pro/
[38] iVerify. 2024. iVerify. https://iverify.io/
[39] Ralf Keller, David Castellanos, Anki Sander, Amarisa Robinson, and Afshin Abtin. 2021. Roaming in the 5G System: The 5GS Roaming Architecture. *Ericsson Reports* (2021). https://www.ericsson.com/4981f6/assets/local/reports-papers/ericsson-technology-review/docs/2021/roaming-in-the-5g-system.pdf
[40] Eunsoo Kim, Dongkwan Kim, CheolJun Park, Insu Yun, and Yongdae Kim. 2021. BaseSpec: Comparative Analysis of Baseband Software and Cellular Specifications for L3 Protocols. In *Proceedings of the 2021 Annual Network and Distributed System Security Symposium (NDSS)*. https://syssec.kaist.ac.kr/pub/2021/kim-ndss2021.pdf
[41] Tobias Kröll. 2021. ARIstoteles: iOS Baseband Interface Protocol Analysis. https://doi.org/10.26083/tuprints-00019397
[42] Tobias Kröll, Stephan Kleber, Frank Kargl, Matthias Hollick, and Jiska Classen. 2021. ARIstoteles – Dissecting Apple's Baseband Interface. In *Computer Security – ESORICS 2021*, Elisa Bertino, Haya Shulman, and Michael Waidner (Eds.). Springer International Publishing, Cham, 133–151.
[43] Sravan Kundojjala. 2023. Qualcomm Gains Baseband Share in Q3 2022. https://www.strategyanalytics.com/strategy-analytics/blogs/components/handset-components/handset-components/2023/02/14/qualcomm-gains-baseband-share-in-q3-2022
[44] Swantje Lange, Francesco Gringoli, Matthias Hollick, and Jiska Classen. 2024. Wherever I May Roam: Stealthy Interception and Injection Attacks through Roaming Agreements. In *Computer Security – ESORICS 2024*. Springer International Publishing.
[45] Sukchan Lee. 2024. Open5GS. https://open5gs.org/
[46] Zhenhua Li, Weiwei Wang, Christo Wilson, Jian Chen, Chen Qian, Taeho Jung, Lan Zhang, Kebin Liu, Xiangyang Li, and Yunhao Liu. 2017. FBS-Radar: Uncovering Fake Base Stations at Scale in the Wild.. In *NDSS*.
[47] Berly Lipton. 2022. Police Are Still Abusing Investigative Exemptions to Shield Surveillance Tech, While Others Move Towards Transparency. https://www.eff.org/deeplinks/2022/07/police-are-still-abusing-investigative-exemptions-shield-surveillance-tech-while-0
[48] Dominik Maier, Lukas Seidel, and Shinjo Park. 2020. BaseSAFE: Baseband Sanitized Fuzzing through Emulation. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (Linz, Austria) *(WiSec '20)*. Association for Computing Machinery, New York, NY, USA, 122–132. https://doi.org/10.1145/3395351.3399360
[49] Slava Makkaveev. 2022. Vulnerability within the Unisoc Baseband opens Mobile Phones Communications to Remote Hacker Attacks. https://research.checkpoint.com/2022/vulnerability-within-the-unisoc-baseband/
[50] Mandiant. 2023. macos-UnifiedLogs. https://github.com/mandiant/macos-UnifiedLogs
[51] György Miru. 2017. Path of Least Resistance: Cellular Baseband to Application Processor Escalation on Mediatek Devices. https://comsecuris.com/blog/posts/path_of_least_resistance/
[52] Stig F. Mjølsnes and Ruxandra F. Olimid. 2017. Easy 4G/LTE IMSI Catchers for Non-Programmers. In *Computer Network Security*. Springer International Publishing, Cham, 235–246. https://doi.org/10.1007/978-3-319-65127-9_19
[53] Aleksander Morgado and Dan Williams. 2021. libqmi. https://www.freedesktop.org/wiki/Software/libqmi/
[54] Mozilla. 2023. Mozilla Location Service. https://location.services.mozilla.com
[55] MVT. 2024. Mobile Verification Toolkit. https://docs.mvt.re/en/latest/
[56] National Security Agency. 2024. Mobile Device Best Practices. https://www.documentcloud.org/documents/21018353-nsa-mobile-device-best-practices&xcust=2-1-2330195-1-0-0
[57] Peter Ney, Ian Smith, Gabriel Cadamuro, and Tadayoshi Kohno. 2017. SeaGlass: Enabling City-Wide IMSI-Catcher Detection., Vol. 2017. 36–53. https://doi.org/10.1515/popets-2017-0027
[58] Oros42. 2022. IMSI-catcher. https://github.com/Oros42/IMSI-catcher
[59] Andrew Orr. 2022. Craig Federighi outlines iOS 17 privacy and Apple's stance on AI. https://appleinsider.com/articles/23/06/05/craig-federighi-outlines-ios-17-privacy-apples-stance-on-ai
[60] Ivan Palamà, Francesco Gringoli, Giuseppe Bianchi, and Nicola Blefari-Melazzi. 2021. IMSI Catchers in the wild: A real world 4G/5G assessment. *Computer Networks* 194 (2021), 108137. https://doi.org/10.1016/j.comnet.2021.108137
[61] Daniele Palmas. 2020. Wireshark QMI dissector for Qualcomm based modems. https://github.com/dnlplm/WiresharkQMIDissector
[62] Shinjo Park, Altaf Shaik, Ravishankar Borgaonkar, Andrew Martin, and Jean-Pierre Seifert. 2017. White-Stingray: Evaluating IMSI Catchers Detection Applications. In *11th USENIX Workshop on Offensive Technologies (WOOT 17)*. USENIX Association, Vancouver, BC. https://www.usenix.org/conference/woot17/workshop-program/presentation/park
[63] Shinjo Park, Altaf Shaik, Ravishankar Borgaonkar, and Jean-Pierre Seifert. 2019. Anatomy of Commercial IMSI Catchers and Detectors. In *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society* (London, United Kingdom) *(WPES'19)*. Association for Computing Machinery, New York, NY, USA, 74–86. https://doi.org/10.1145/3338498.3358649
[64] PentHertz. 2023. OpenBTS. https://github.com/PentHertz/OpenBTS
[65] Cooper Quintin. 2022. VICTORY: Google Releases "disable 2g" Feature for New Android Smartphones. https://www.eff.org/deeplinks/2022/01/victory-google-releases-disable-2g-feature-new-android-smartphones
[66] Ole André Vadla Ravnås. 2022. Frida. https://frida.re
[67] David Rupprecht, Katharina Kohls, Thorsten Holz, and Christina Pöpper. 2020. IMP4GT: IMPersonation Attacks in 4G NeTworks.. In *Proceedings of the 2020 Annual Network and Distributed System Security Symposium (NDSS)*. https://www.ndss-symposium.org/wp-content/uploads/2020/02/24283-paper.pdf
[68] David Rupprecht, Katharina Kohls, Thorsten Holz, and Christina Pöpper. 2019. Breaking LTE on Layer Two. In *2019 IEEE Symposium on Security and Privacy (SP)*. 1121–1136. https://doi.org/10.1109/SP.2019.00006
[69] Security Research Labs. 2015. IMSI Catcher Score. https://opensource.srlabs.de/projects/snoopsnitch/wiki/IMSI_Catcher_Score
[70] Security Research Labs. 2015. Snoop Snitch SQL Queries for Detection. https://github.com/srlabs/snoopsnitch/tree/master/analysis/catcher/sql
[71] Security Research Labs. 2018. SnoopSnitch Compatibility. https://opensource.srlabs.de/projects/snoopsnitch/wiki/DeviceList
[72] Security Research Labs. 2022. SnoopSnitch. https://opensource.srlabs.de/projects/snoopsnitch
[73] Altaf Shaik, Ravishankar Borgaonkar, N Asokan, Valtteri Niemi, and Jean-Pierre Seifert. 2016. Practical Attacks Against Privacy and Availability in 4G/LTE Mobile Communication Systems. In *Proceedings of the 2016 Annual Network and Distributed System Security Symposium (NDSS)*. https://arxiv.org/pdf/1510.07563.pdf
[74] Adam Simmons. 2022. Cell Tower Range: How Far Do They Reach? https://dgtlinfra.com/cell-tower-range-how-far-reach/
[75] skibapps. 2023. Cell Spy Catcher (Anti Spy). https://play.google.com/store/apps/details?id=com.skibapps.cellspycatcher
[76] SS7ware. 2024. YateBTS. https://yatebts.com
[77] Software Radio Systems. 2024. srsRAN. https://github.com/srsran/srsran_4g
[78] Riley Testut. 2023. AltStore. https://github.com/altstoreio/AltStore
[79] Mika Tuupola. 2017. Reverse Engineering Apple Location Services Protocol. https://www.appelsiini.net/2017/reverse-engineering-location-services/
[80] Unwired Labs. 2023. OpenCelliD. https://opencellid.org/
[81] Ralf-Philipp Weinmann. 2012. Baseband Attacks: Remote Exploitation of Memory Corruptions in Cellular Protocol Stacks. In *Proceedings of the 6th USENIX Conference on Offensive Technologies* (Bellevue, WA) *(WOOT'12)*. USENIX Association, 2. https://www.usenix.org/system/files/conference/woot12/woot12-final24.pdf
[82] Ralf-Philipp Weinmann. 2017. Did I hear a shell popping in your baseband? https://vimeo.com/214013463
[83] Harald Welte. 2022. OsmoBTS. https://osmocom.org/projects/osmobts/wiki
[84] Amy While. 2022. SignalReborn. https://github.com/elihwyma/SignalReborn
[85] Tim Willis. 2023. Multiple Internet to Baseband Remote Code Execution Vulnerabilities in Exynos Modems. https://googleprojectzero.blogspot.com/2023/03/multiple-internet-to-baseband-remote-rce.html
[86] Wireshark Foundation. 2023. Wireshark. https://www.wireshark.org
[87] Yaru Yang, Yiming Zhang, Tao Wan, Chuhan Wang, Haixin Duan, Jianjun Chen, and Yishen Li. 2024. Uncovering Security Vulnerabilities in Real-world Implementation and Deployment of 5G Messaging Services. In *Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (Seoul, Republic of Korea) *(WiSec '24)*. Association for Computing Machinery, New York, NY, USA, 265–276. https://doi.org/10.1145/3643833.3656131
[88] Zerodium. 2019. Zerodium Payouts. https://zerodium.com/program.html

A ONBOARDING

Upon first installation, CellGuard starts with an onboarding dialogue (see Figure 10). This dialogue explains to the user what CellGuard does, what RBSes are, the concept behind sysdiagnoses, and guides the user through granting location permission. A demo of the onboarding process and sysdiagnose import is also available here: https://youtu.be/ohGHe-RAOT0.

B DEBUG PROFILE INSTALLATION

On non-jailbroken devices, CellGuard requires the user to install a baseband debug profile. CellGuard guides the user through the relevant steps (see Figure 11). While no baseband debug profile is installed, the summary tab displays an installation guide. We ask the user to download the original baseband debug profile directly from Apple's website. After downloading, the debug profile appears in the iPhone's Settings, where users can tap it for installation.



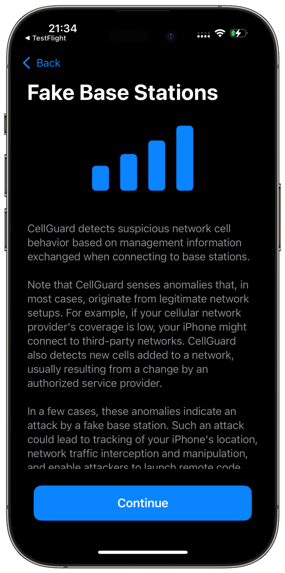
Welcome to CellGuard

Analyze Cellular Network
Monitor which cells your iPhone uses to communicate with the cellular network.

Verify Cells
Detect and report suspicious behavior of connected cells.

Map Cells
View the location of recently connected cells on a map.

(a) Welcome screen.



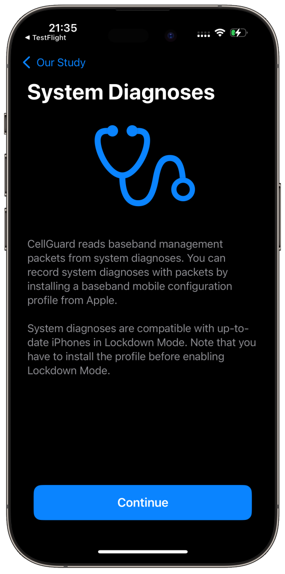
Fake Base Stations

CellGuard detects suspicious network cell behavior based on management information exchanged when connecting to base stations.

Note that CellGuard senses anomalies that, in most cases, originate from legitimate network setups. For example, if your cellular network provider's coverage is low, your iPhone might connect to third-party networks. CellGuard also detects new cells added to a network, usually resulting from a change by an authorized service provider.

In a few cases, these anomalies indicate an attack by a fake base station. Such an attack could lead to tracking of your iPhone's location, network traffic interception and manipulation, and enable attackers to launch remote code

(b) Functionality explained.

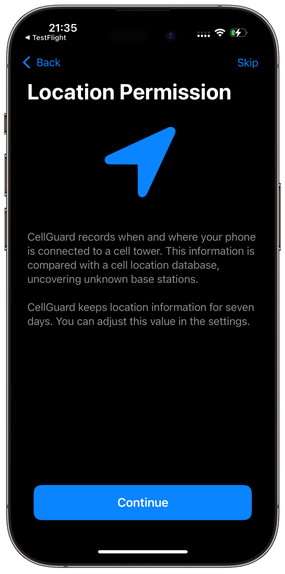


System Diagnoses

CellGuard reads baseband management packets from system diagnoses. You can record system diagnoses with packets by installing a baseband mobile configuration profile from Apple.

System diagnoses are compatible with up-to-date iPhones in Lockdown Mode. Note that you have to install the profile before enabling Lockdown Mode.

(c) Sysdiagnose primer.



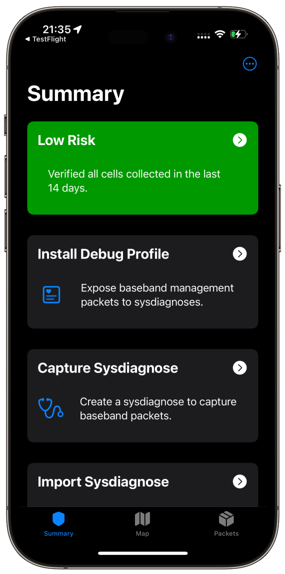
Location Permission

CellGuard records when and where your phone is connected to a cell tower. This information is compared with a cell location database, uncovering unknown base stations.

CellGuard keeps location information for seven days. You can adjust this value in the settings.

(d) Permissions.

Figure 10: CellGuard's onboarding dialogue.



Summary

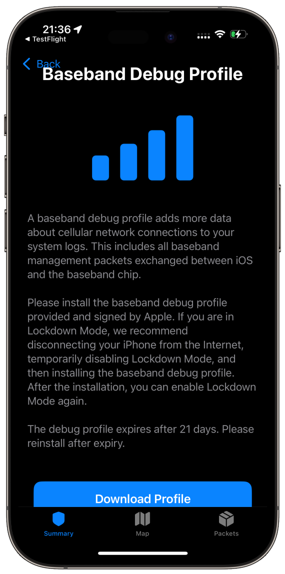
Low Risk
Verified all cells collected in the last 14 days.

Install Debug Profile
Expose baseband management packets to sysdiagnoses.

Capture Sysdiagnose
Create a sysdiagnose to capture baseband packets.

Import Sysdiagnose

(a) Summary w/o imports.



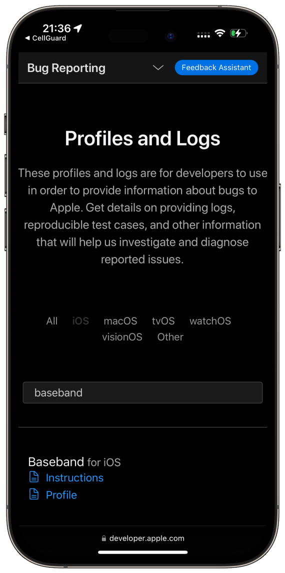
Baseband Debug Profile

A baseband debug profile adds more data about cellular network connections to your system logs. This includes all baseband management packets exchanged between iOS and the baseband chip.

Please install the baseband debug profile provided and signed by Apple. If you are in Lockdown Mode, we recommend disconnecting your iPhone from the Internet, temporarily disabling Lockdown Mode, and then installing the baseband debug profile. After the installation, you can enable Lockdown Mode again.

The debug profile expires after 21 days. Please reinstall after expiry.

(b) Debug profile link.



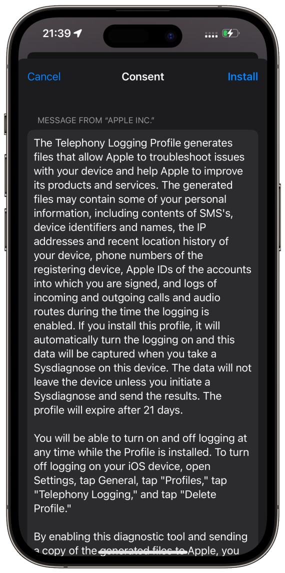
Profiles and Logs

These profiles and logs are for developers to use in order to provide information about bugs to Apple. Get details on providing logs, reproducible test cases, and other information that will help us investigate and diagnose reported issues.

baseband

Baseband for iOS

(c) Apple's website.



MESSAGE FROM "APPLE INC."

The Telephony Logging Profile generates files that allow Apple to troubleshoot issues with your device and help Apple to improve its products and services. The generated files may contain some of your personal information, including contents of SMS's, device identifiers and names, the IP addresses and recent location history of your device, phone numbers of the registering device, Apple IDs of the accounts into which you are signed, and logs of incoming and outgoing calls and audio routes during the time the logging is enabled. If you install this profile, it will automatically turn the logging on and this data will be captured when you take a Sysdiagnose on this device. The data will not leave the device unless you initiate a Sysdiagnose and send the results. The profile will expire after 21 days.

You will be able to turn on and off logging at any time while the Profile is installed. To turn off logging on your iOS device, open Settings, tap General, tap "Profiles," tap "Telephony Logging," and tap "Delete Profile."

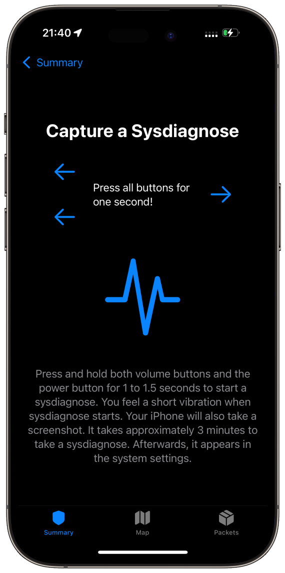
By enabling this diagnostic tool and sending a copy of the generated files to Apple, you

(d) Profile installation.

Figure 11: Debug profile installation steps.

C SYSDIAGNOSE IMPORT

Users must regularly make sysdiagnoses on non-jailbroken devices and import them into CellGuard. CellGuard guides the user through capturing and importing sysdiagnoses (see Figure 12). After taking a sysdiagnose, users can import it through the Settings and share it to CellGuard. CellGuard summarizes imported data upon success.

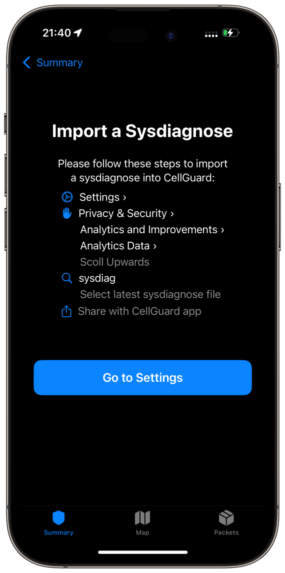


Capture a Sysdiagnose

Press all buttons for one second!

Press and hold both volume buttons and the power button for 1 to 1.5 seconds to start a sysdiagnose. You feel a short vibration when sysdiagnose starts. Your iPhone will also take a screenshot. It takes approximately 3 minutes to take a sysdiagnose. Afterwards, it appears in the system settings.

(a) Take sysdiagnose.

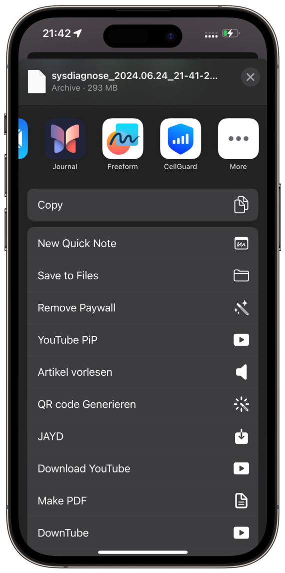


Import a Sysdiagnose

Please follow these steps to import a sysdiagnose into CellGuard:

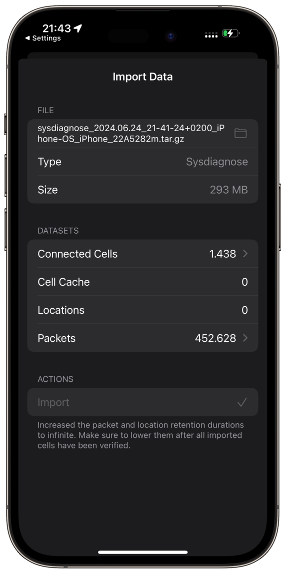
Settings ›
Privacy & Security ›
Analytics and Improvements ›
Analytics Data ›
Scoll Upwards
sysdiag
Select latest sysdiagnose file
Share with CellGuard app

(b) Import helper.



sysdiagnose_2024.06.24_21-41-2...
Archive · 293 MB

(c) Share sysdiagnose.



Import Data

FILE

sysdiagnose_2024.06.24_21-41-24+0200_iPhone-OS_iPhone_22A5282m.tar.gz

Type	Sysdiagnose
Size	293 MB

DATASETS

Connected Cells	1.438 ›
Cell Cache	0
Locations	0
Packets	452.628 ›

ACTIONS

Increased the packet and location retention durations to infinite. Make sure to lower them after all imported cells have been verified.

(d) Import confirmation.

Figure 12: Sharing sysdiagnoses with CellGuard.

D QMI SERVICE COMPARISON

Table 4 shows all services supported by `libqmi` in version 1.33.3 [53]. Table 3 shows all QMI services implemented by the Snapdragon X55 baseband chipset in an iPhone 12. Not all services implemented in the chipset's service are also implemented by iOS. Table 3 indicates if a service is also present in iOS, as previously listed in Table 1.

Table 3: QMI services present in the iPhone's X55 baseband firmware. ● means that there is an iOS implementation for that service, ○ means that there is none.

#	ID	Version	iOS Impl.
`0x00`	ctl	1.5	●
`0x01`	wds	1.177	●
`0x02`	dms	1.79	●
`0x03`	nas	1.25	●
`0x04`	qos	1.17	●
`0x05`	wms	1.10	●
`0x06`	pds	1.18	●
`0x08`	at	1.6	●
`0x09`	vs	2.1	●
`0x0A`	cat	2.24	●
`0x0B`	uim	1.77	●
`0x0C`	pbm	1.4	●
`0x0F`	test	1.0	○
`0x11`	sar	1.0	○
`0x17`	ts	1.0	○
`0x18`	tmd	1.0	○
`0x1A`	wda	1.24	●
`0x1D`	csvt	1.6	○
`0x22`	coex	1.0	●
`0x24`	pdc	1.0	●
`0x29`	rfrpe	1.0	○
`0x2A`	dsd	1.63	●
`0x2B`	ssctl	2.0	●
`0x2C`	mfse	1.0	●
`0x30`	dfs	1.11	●
`0x31`	unknown	1.0	○
`0x44`	unknown	1.4	○
`0x49`	unknown	1.7	○
`0x4A`	unknown	1.1	○
`0x4E`	unknown	1.2	○
`0x52`	ms	1.0	●
`0xE1`	audio	1.0	●
`0xE2`	bsp	1.0	●
`0xE4`	awd	1.0	●
`0xE5`	vinyl	1.0	●
`0xE6`	mavims	1.0	●
`0xE7`	elqm	1.0	●
`0xE8`	p2p	1.0	●
`0xE9`	apps	1.0	●

Table 4: QMI services present in **`libqmi`**.

#	ID	Full Name
`0x00`	ctl	Control Service
`0x01`	wds	Wireless Data Service
`0x02`	dms	Device Management Service
`0x03`	nas	Network Access Service
`0x04`	qos	Quality Of Service Service
`0x05`	wms	Wireless Messaging Service
`0x06`	pds	Position Determination Service
`0x07`	auth	Authentication Service
`0x08`	at	AT Service
`0x09`	voice	Voice Service
`0x0A`	cat2	Card Application Toolkit Service (v2)
`0x0B`	uim	User Identity Module Service
`0x0C`	pbm	Phonebook Manager Service
`0x0D`	qchat	QCHAT Service
`0x0E`	rmtfs	Remote File System Service
`0x0F`	test	Test Service
`0x10`	loc	Location Service
`0x11`	sar	Specific Absorption Rate Service
`0x12`	ims	IMS Settings Service
`0x13`	adc	Analog to Digital Converter Driver Service
`0x14`	csd	Core Sound Driver Service
`0x15`	mfs	Modem Embedded File System Service
`0x16`	time	Time Service
`0x17`	ts	Thermal Sensors Service
`0x18`	tmd	Thermal Mitigation Device Service
`0x19`	sap	Service Access Proxy Service
`0x1A`	wda	Wireless Data Administrative Service
`0x1B`	tsync	TSYNC Control Service
`0x1C`	rfsa	Remote File System Access Service
`0x1D`	csvt	Circuit Switched Videotelephony Service
`0x1E`	qcmap	Qualcomm Mobile Access Point Service
`0x1F`	imsp	IMS Presence Service
`0x20`	imsvt	IMS Videotelephony Service
`0x21`	imsa	IMS Application Service
`0x22`	coex	Coexistence Service
`0x24`	pdc	Persistent Device Configuration Service
`0x26`	stx	Simultaneous Transmit Service
`0x27`	bit	Bearer Independent Transport Service
`0x28`	imsrtp	IMS RTP Service
`0x29`	rfrpe	RF Radiated Performance Enhancement Service
`0x2A`	dsd	Data System Determination Service
`0x2B`	ssctl	Subsystem control Service
`0x2F`	dpm	Data Port Mapper Service
`0xE0`	cat	Card Application Toolkit Service (v1)
`0xE1`	rms	Remote Management Service
`0xE2`	oma	OMA Device Management Service
`0xE3`	fox	Foxconn General Modem Service
`0xE6`	fota	Firmware Over The Air Service
`0xE7`	gms	Telit General Modem Service
`0xE8`	gas	Telit General Application Service