IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 16, 2021, accepted July 16, 2021, date of publication July 21, 2021, date of current version July 28, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3099017

An Advertising Overflow Attack Against Android Exposure Notification System Impacting COVID-19 Contact Tracing Applications

HENRIQUE FARIA[1], SARA PAIVA[2], AND PEDRO PINTO[2,3]

[1]Instituto Politécnico de Viana do Castelo, Rua Escola Industrial e Comercial Nun'Álvares, 4900-347 Viana do Castelo, Portugal
[2]ADiT-LAB, Instituto Politécnico de Viana do Castelo, Rua Escola Industrial e Comercial Nun'Álvares, 4900-347 Viana do Castelo, Portugal
[3]INESC TEC, 4200-465 Porto, Portugal

Corresponding author: Henrique Faria (henriquefaria@ipvc.pt)

This work was supported by the Norte Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF), within Project Cybers SeC IP under Grant NORTE-01-0145-FEDER-000044.

ABSTRACT The digital contact tracing applications are one of the many initiatives to fight the COVID-19 virus. Some of these Apps use the Exposure Notification (EN) system available on Google and Apple's operating systems. However, EN-based contact tracing Apps depend on the availability of Bluetooth interfaces to exchange proximity identifiers, which, if compromised, directly impact their effectiveness. This paper discloses and details the Advertising Overflow attack, a novel internal Denial of Service (DoS) attack targeting the EN system on Android devices. The attack is performed by a malicious App that occupies all the Bluetooth advertising slots in an Android device, effectively blocking any advertising attempt of EN or other Apps. The impact of the disclosed attack and other previously disclosed DoS-based attacks, namely Battery Exhaustion and Storage Drain, were tested using two target smartphones and other six smartphones as attackers. The results show that the Battery Exhaustion attack imposes a battery discharge rate 1.95 times higher than in the normal operation scenario. Regarding the Storage Drain, the storage usage increased more than 30 times when compared to the normal operation scenario results. The results of the novel attack reveal that a malicious App can prevent any other App to place their Bluetooth advertisements, for any chosen time period, thus canceling the operation of the EN system and compromising the efficiency of any COVID contact tracing App using this system.

INDEX TERMS Attack, applications, android, denial of service, COVID-19, contact tracing, exposure notification.

I. INTRODUCTION

The worldwide spread of the COVID-19 virus captured society's attention and pushed for efforts to contain and mitigate the impact and the effects of the current pandemic. In the face of this threat against its citizens, governments' reactions have included closing borders, banning travel, and applying strict quarantine procedures. One of the strategies adopted by governments and countries' health authorities consisted of relying on digital contact tracing applications (or Apps), to enable citizens to monitor their exposure to COVID-19.

Contact tracing Apps automatically register and trace the user's contact history and notify users when they have been in contact with an infected user [1]. In comparison to other contact tracing alternatives, the contact tracing Apps allow instantaneous notifications of contacts, avoid loss of information due to recall bias and can be scaled to large amounts of users. These Apps are usually deployed in smartphones and use their already existing interfaces, such as Bluetooth and GPS, to track encounters between users by exchanging anonymous identifiers. The effectiveness of a contact tracing App is dependent on its adoption rate in the general population [2].

In a technical perspective, centralized or decentralized frameworks were developed to support contact tracing Apps. Exposure Notification (EN) system, also known as Google/Apple Exposure Notification (GAEN), was released by Google and Apple, with similarities to an already existing

The associate editor coordinating the review of this manuscript and approving it for publication was Diana Gratiela Berbecaru.

protocol named Decentralized Privacy-Preserving Proximity Tracing (DP-3T), and consists of a decentralized architecture that, according to [3], has the focus on privacy. This architecture ensures interoperability between Apple and Android devices, allows the execution of the contact tracing tasks at the Operating System (OS) level, and prevents the OS from terminating or interfering with a task running on background [2]. The study in [4] highlights that the EN allows to obtain an estimated risk of infection for each day, while recording no more than 30 minutes of exposure contacts to preserve the anonymity of an infection source.

Regarding EN attacks, vulnerabilities exist either in the communication between participants, in the communication with the health authority, or in the contact tracing features [5]. As seen in [6], even though privacy concerns are a primary focus of EN, an attacker can trace a single user by matching his anonymous identifier, replay the messages received as if they were legitimate, amongst other attacks. These vulnerabilities and attacks can compromise the contact tracing strategy, and can be seen as an opportunity by malicious agents to perform high impact attacks, such as terrorist attacks [7], [8].

Since the EN depends on the Bluetooth interface of the mobile devices, it requires Bluetooth access and the availability of its advertising transmission system to exchange the anonymous identifiers. If a contact tracing App using EN is inhibited to use these advertisements transmission slots, the efficiency of the contact tracing can be severely impacted. The inhibition can be inadvertently or maliciously caused by another App, e.g by taking all advertisements transmission slots available in the system.

As main contribution, this paper presents a novel advertisement overflow attack on EN system. This attack allows a malicious mobile App to exhaust all the advertisements transmission slots, thus compromising all other Apps depending on Bluetooth advertisements, including the contact tracing Apps using the EN. In a real scenario, this attack can impair the efficiency of EN system and any contact tracing App that uses it, since the mobile user (target) will not be able to exchange the anonymous identifiers. Thus, if the user is infected with COVID-19 and inserts an infection code on the contact tracing App, the other mobile users will not detect any contact and will not trigger any exposure warning.

Additionally, this paper analyses the impact of internal DoS-based attacks, namely, Battery Exhaustion, Storage Drainage, and the novel Advertisement Overflow attack. Based on our experiments, the impact of the Battery Exhaustion and the Storage Drainage attacks is presented, and tests carried out regarding the Advertising Overflow attack confirmed that a malicious App can overflow the Bluetooth advertisements, impacting any App that requires this interface, including the COVID tracing Apps using EN.

This paper is organized as follows. In section II, background concepts regarding EN, Android, and Bluetooth are explained. Section III presents a systematic review for the existing EN attacks. Section IV details the novel Advertising Overflow attack. Section V presents the impact assessment regarding three DoS-based attacks including the Advertising Overflow attack. Section VI analyzes and discusses the obtained results and their impact. Finally, section VII draws conclusions and points to future work.

II. BACKGROUND

COVID contact tracing Apps have been developed worldwide to assist in the detection of infected people by COVID-19 and in preventing the increase of transmission chains. For the Apps based on EN, Google and Apple defined that each country could choose one digital contact tracing App. Table 1 compiles information from [9] and lists the currently available contact tracing Apps in Europe, the system they use (EN or other), the number of installs registered in Google Play Store, and the countries supporting them. Only 2 of a total of 24 contact tracking apps are not EN-based, the latter totaling more than 38 million installs by European citizens.

TABLE 1. European contact tracing apps and their systems.

Contact Tracing App	System	# of Installs (Millions)	Country
Corona-Warn-App	EN	10+	Germany
NHS COVID-19	EN	10+	United Kingdom
Immuni	EN	5+	Italy
Radar Covid	EN	5+	Spain
Coronalert	EN	1+	Belgium
eRouška	EN	1+	Czechia
Koronavilkku	EN	1+	Finland
CoronalMelder	EN	1+	Netherlands
ProteGO Safe	EN	1+	Poland
STAYAWAY Covid	EN	1+	Portugal
SwissCovid	EN	1+	Switzerland
Smittestop	EN	0.5+	Denmark
COVID Tracker	EN	0.5+	Ireland
Stopp Corona	EN	0.1+	Austria
HOIA	EN	0.1+	Estonia
Apturi Covid	EN	0.1+	Latvia
Korona Stop LT	EN	0,1+	Lithuania
Smittestopp	EN	0.1+	Norway
#OstaniZdrav	EN	0.1+	Slovenia
Stop COVID-19	EN	0.05+	Croatia
COVIDAlert	EN	0.05+	Malta
CovTracer-EN	EN	0.01+	Cyprus
StopCovid	Other	10+	France
VírusRadar	Other	0.1+	Hungary
Total Installs (EN-based): 38M+			

For contact tracing Apps to operate with EN they require a mobile device with a compatible hardware and operating system, including EN and Bluetooth Low-Energy (BLE) support. Fig. 1 presents the interactions performed by an EN-based contact tracing App using Android OS.

According to [10], the core features of Contact Tracing Apps are the following:

- Show notifications and instructions to the user in case of a confirmed exposure to an infected user.
- Allow users to control when the Bluetooth functions (broadcast and scan) are active.
- Polling the server for new keys, receiving the files with the diagnosis keys and passing them to EN.
- In case of an infection, retrieving the keys for the last 14 days and upload them to the server.

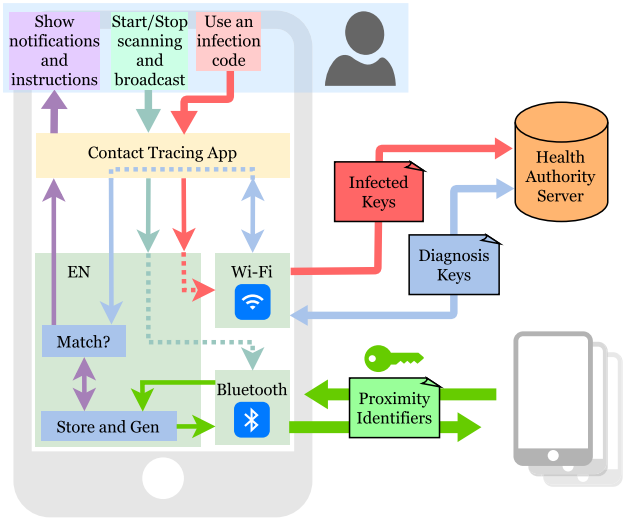

FIGURE 1. Schematic of contact tracing apps based on EN contact tracing apps and its interactions with the user, other smartphones, and the health authority.

The Contact Tracing Apps communicate with the EN and also handle external communications such as sending HyperText Transfer Protocol Secure (HTTPS) requests to the Health Authority server. In the event of an infection, the Contact Tracing App allows the usage of an infection code to request the last 14 days keys from the EN and upload them to the Health Authority via Wi-Fi, for example. It also requests Diagnosis Keys from the same server periodically and passes them to the EN to check for matches. The EN generates the Proximity Identifiers from the received keys and compares them with the ones already stored in the local storage. These identifiers are received via Bluetooth during the exchange process with other devices.

In the case of a positive match, the EN sends that information to the Contact Tracing App that is responsible to appropriately display that information and instructions to the user.

The EN and other similar protocols use the broadcast procedures such as advertisements that do not require the establishment of a connection to exchange identifiers. A device can periodically advertise a data block, and other devices are able to scan and process it. When a device broadcasts an advertising channel packet, the other devices need to be in scanning mode to receive it. Since both actions are not synchronized, there are no guarantees that the advertising packet will be received [11]. In the BLE standard, the maximum size for an advertising block is 31 bytes. The option adopted by EN is sending a payload with a 16-bit Universally Unique Identifier (UUID) to mark the payload as belonging to the EN and using 16 bytes for the Proximity Identifier and another 4 bytes for encrypted metadata that is sent together with the Proximity Identifier. If the user wants to disable the contact tracing operation, the Contact Tracing App sends an event to EN to immediately stop the BLE broadcast and scan operations, such as the Proximity Identifiers exchange.

The EN is responsible for the generation, storage and exchange of the proximity identifiers, where matches processing and key generation is accomplished in the background.

As detailed in [5] and [12], the key generation procedure of EN is as depicted in the Fig. 2. The system generates a daily random key named Temporary Exposure Key (TEK) used to derive a Rolling Proximity Identifier Key (RPIK) generated each 15 minutes, which in turn is used to derive an identifier named Rolling Proximity Identifier (RPI). In the Android OS, the EN is incorporated in the Google Play Services and it is responsible for:

- Managing the keys used by the system, including TEKs and RPIs.
- Managing Bluetooth functions, including scans, storage and analysis of exposure risk.
- Ask for user consent on the first activation of the system and before uploading the keys in case of an infection.

If the user has a confirmed infection, the health authority provides a code that allows him to report the infection, by uploading the TEKs from the last 14 days to the server. The other users later download these keys, and EN derives the RPIs and checks for matches in its contact records.

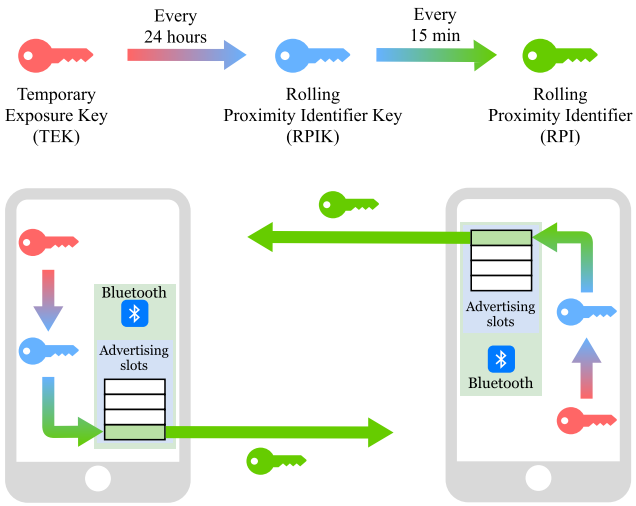

FIGURE 2. Schematic of how EN generates and exchanges the RPIs.

EN advertises over BLE each of the generated RPIs during several minutes. When a RPI rotates, the MAC address rotates as well to anonymize the user's device. According to [12] there is a tolerance (within 2 hours of its original time interval) for the RPI's validity, and it should be advertised using time intervals of 250 milliseconds. In the receiver, the App keeps the TEKs for 14 days and then deletes them.

Apart from advertising, EN also scans every 3 to 5 minutes for Bluetooth advertisements with the matching service UUID [12]. Whenever it receives a valid RPI, it stores it in the device to be later used to check for matches.

The advertising and scanning features are present in devices running Android OS from version 4.3 [13]. Given the

time since its introduction, these features are commonly seen in most modern smartphones [11].

III. RELATED WORK ON EXPOSURE NOTIFICATION SYSTEM ATTACKS

To collect the related work on existing attacks concerning the EN system, a systematic review was conducted. The systematic review process and results are presented in Fig. 3. The research question was defined as ''What are the known attacks of EN and other decentralized systems?''. Google Scholar was selected as search engine and the query presented in Fig. 3 was inserted in this search engine in March 4th, 2021, returning a total of 115 papers. Each paper had its abstract, content, and conclusions analysed to check for a match on a specific EN attack. After filtering, a total of 21 papers were relevant for further analysis, addressing a group of 19 unique attacks. Table 2 depicts each attack and their respective sources.

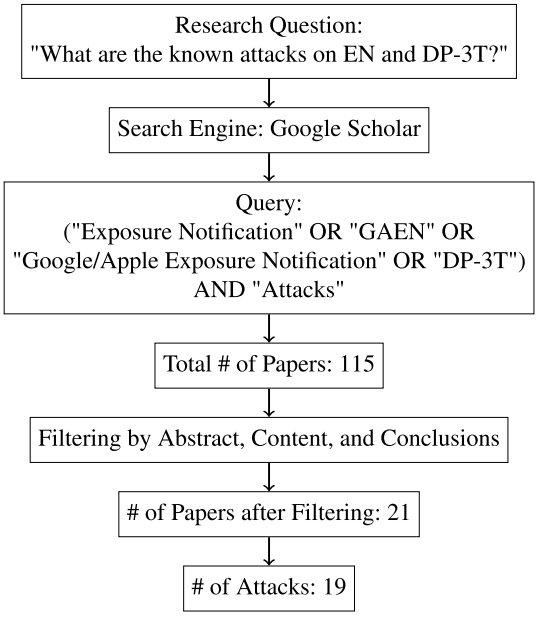

FIGURE 3. Stages, options and numbers of the systematic review performed.

Each disclosed attack is reviewed as follows:

1) Replay - This attack consists of sending RPIs already received by a device [16]. In [6], the author replayed released RPIs from the health authority and managed to trigger a exposure warning in the target. Others [18]–[20] defined scenarios such as advertising the received identifiers either immediately or at a later time. [22] presents an approach using a malicious Software Development Kit (SDK) injected in an App to replay received identifiers. [21] mentions the Contact Pollution attack that works similarly to the Replay attack mentioned in [6]. Mitigation strategies for this type of attack have been proposed using protocols and location of devices [6], [17]–[20].
2) Relay - This attack is similar to the Replay attacks; an attacker receives a RPI in one location and relays it to another device in a different location. The authors in [6], [18], and [24] present the attack exploring the fact that EN protocol do not registers location data in each encounter. In [25] a scenario for this attack is presented to use this attack for voter suppression during the pandemic of COVID-19. [23] successful implemented this attack with EN and devices such as Raspberry Pis were used to relay RPIs across a city. Acoording to [26] the mitigation strategies for this type of attack may introduce significant complexity to the system.
3) Linking - This attack is performed when an attacker associates two transmitted messages to the same device [22]. The author in [18] considers using the message timing and signal strength to link the received messages. It also suggested to use rejection sample techniques to add noise to the signal strength.
4) Battery Exhaustion - This attack is accomplished when an attacker sends large quantities of RPIs to a target exhausting its battery. This Denial of Service (DoS)-based attack is mentioned in [18], [28], and [29], and the authors highlight that the users tend to reject and uninstall the Contact Tracing Apps assuming that they will increase of the battery consumption rate. To mitigate these attacks, authors in [18] propose to use a proof-of-work to check if a received message is valid when under a high request load.
5) Profiling - in this attack, users movements are profiled by capturing their RPIs [23], [24], [30]. Authors in [23] test this attack by deploying devices to capture RPIs in critical points in a city. Whenever a key is reported, they check if there is a match, and this allows to trace the movements of a given user.
6) Singling-out - This attack is described in [18] and it is performed when an attacker records only 1 RPI. The attacker can then check if a given user is infected when the health authority publishes new TEKs. In [21], the authors executed a similar attack named Contact Isolation. They managed to associate a RPI to a device. This work also proposes other advanced techniques to link the RPI to a specific device, such as using Wi-Fi data or the smartphone's camera.
7) Storage Drainage - An attacker sends valid RPIs to the target and occupies storage space on the device [28], [29]. It is similar to the Battery Exhaustion attack but it is directed to the devices' storage [18].
8) Time Travel - In this attack, the attacker changes the time on the target device, setting the date to an earlier day, and injects RPIs that were valid for that specific time period. In [27] the authors managed to test multiple methods such as manually changing the time in the device's settings, and confirmed that they could indeed trigger alerts in the device with RPIs from previous days.

TABLE 2. Identified attacks and their sources.

Attack	[14]	[15]	[16] [17]	[6]	[18]	[19]	[20]	[21]	[22]	[23] [24]	[25] [26]	[5]	[27]	[28] [29]	[30]	[7]	[31]
1) Replay			•	•	•	•	•	•									•
2) Relay				•	•					•	•						
3) Linking					•				•								•
4) Battery Exhaustion					•									•			
5) Profiling										•				•			
6) Singling-Out					•			•									
7) Storage Drainage					•									•			
8) Time Travel												•	•				
9) Troll					•		•										
10) Belated Replay												•					
11) False Report				•													
12) Jamming	•																
13) Malicious SDK																	•
14) Nerd				•													
15) Paparazzi				•													
16) Passive Disruption		•															
17) Private Encounter Disclosure				•													
18) Simulated EN												•					
19) Terrorist Report																•	

9) Troll - This attack is performed when an adversary knows he is infected and attaches his device to a carrier to spread his RPIs around unsuspecting users [18]. When his TEKs are published, multiple exposure alerts will be triggered.
10) Belated Replay - In this attack, an attacker obtains TEKs published by the health authority and replays the derived RPIs [5]. This attack is also referred to as Keep It Simple Stupid (KISS) attack. The keys published are usually outdated, but some are still valid since life tolerance is set to 2 hours for a RPI. In [5] authors analyzed that with interoperability between countries, the health authorities do not publish the keys at the same time. The adversary can obtain keys from one country and use them in another.
11) False Report - This attack is performed when an attacker can falsely report their own keys as infected and generate alerts in other users [6].
12) Jamming - This attack consists in affecting the Bluetooth data transmission between two devices, through a device that is constantly transmitting thousands of invalid RPIs. In [14], the authors present a low-cost jamming attack that uses smartphones or Raspberry PIs to emit tokens faster than the target Apps. The authors found that distance estimation was affected and EN received less RPIs.
13) Malicious SDK - SDKs can be used to massively deploy attacks in mobile devices. In [31], the authors describe SDK-based attacks, such as the Biosurveillance attack which is a massive surveillance technique to infer the health status of the device's owner using a malicious SDK. At a large scale, the attacker can gather information in an entire region and what users might or not be infected.
14) Nerd - This attack is performed when an attacker's App collects specific information (such as GPS location, timestamps, or Wi-Fi networks) for each encounter simultaneously as the EN [6]. All the data can be inserted in a database and be further used to identify reported cases.
15) Paparazzi - Similar to the Singling-out attack, this attack is accomplished when an attacker, using a powerful antenna, captures the RPIs from a specific user and check them later for positive matches [6]. In [19], the authors analyze this attack in the context of a massive surveillance system. Variants of this attack are presented such as the Orwell attack, in which there is a collusion between the attacker and health authority's server.
16) Passive Disruption - Passive disruption attack can be performed by disabling the resources necessary for the EN to function. [15]. In the case of Android OS, disabling the GPS, halts also all the EN-based Apps.
17) Private Encounter Disclosure - Encounters between two users can be disclosed by an attacker if the attacker can successfully raise an alert on one of their devices [6]. When one of the users reports an alert, the attacker waits to see if the other user has an alert raised when the keys are reported to health authorities. The authors suggest adding a delay to the releases of keys, although this may decrease efficiency.
18) Simulated EN - This attack is performed by an App that operates similarly to EN. However, as explained in [5], the attacker can change the signal strength when advertising, meaning that it can be advertising from far away but it will be registered as if it was close. The target receives the RPI and EN falsely believes the attacker is closer than he is.
19) Terrorist Report - this attack is an improvement on the False Report attack. In [7], authors disclose a blockchain black market solution that allows safely selling and buying infected keys.

All the papers surveyed were published between 2020 and 2021 and, although the Battery Exhaustion and Storage Drainage attacks were already disclosed, there are no measurements to quantify their impact in Android devices. Also, none of the related works on bugs, attacks and vulnerabilities in EN targets the exhaustion of the Bluetooth advertisement slots.

IV. THE ADVERTISING OVERFLOW ATTACK

This section details the novel internal DoS-based attack - the Advertising Overflow attack. Fig. 4 presents the normal operation scenario (on the left) and an attack scenario (on the right).

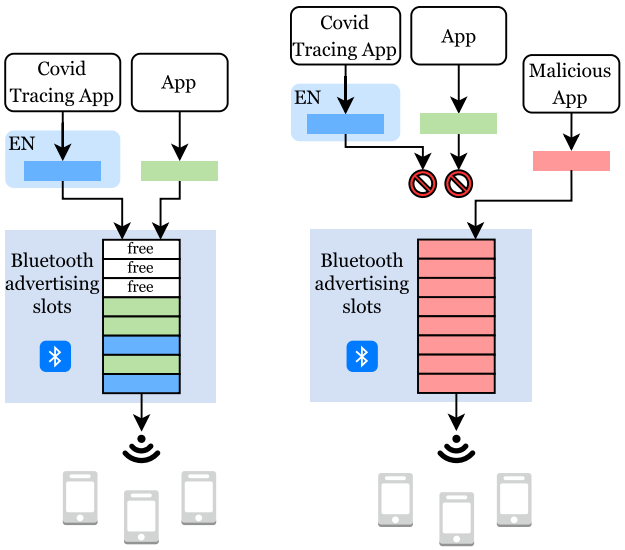

FIGURE 4. Normal operation scenario (left side) and attack scenario (right side).

In the normal operation scenario, each time EN or another App sends a Bluetooth advertisement, it requires the system to allocate it in an available advertising slot. The available slots are a global resource shared between all the Apps in the device, and the maximum capacity is dependent on the device's Bluetooth chip [32]. When an advertisement is placed by an App, the App is also responsible for removing it from the advertising slots; otherwise the Android OS will keep advertising/transmitting it until, for instance, the user disables the Bluetooth interface.

In the Android OS, an App can execute code when device events occur, such as advertising when a device reboots. To capture the event when it occurs, the App needs to implement a *BroadcastReceiver* class that will be executed and set a priority for the class ranging from −999 to 999. An higher priority means that this receiver will be executed by the OS before others capture the same event. However, even if an App (including EN) has a higher priority it will not be able to place an advertisement if the advertising slots are already occupied. The Apps are expected to handle the cases when there are no slots available, for example, by creating an internal buffer where advertisements will wait before trying to place them again in the advertisement slots. The EN default operation sets the priority to a given value different from the maximum one) and so, the EN does not have the highest priority access to these slots. The EN system needs, at least, one slot available in the Bluetooth advertisements slots.

In the attack scenario depicted in the right part of the Fig. 4, instead of having multiple Apps using advertisement slots, a malicious App occupies all the available slots in the device without releasing them, leaving no free slots for other apps to place their advertisements. Since the advertisement slots are a global resource shared between all the Apps in the device, an App can occupy all these slots whenever the device reboots, there is an EN restart event, or during EN normal operation. If an App occupies all the slots, according to the code publicly available in [33], the EN will throw the advertisement error *"ADVERTISE_FAILED_TOO_MANY_ADVERTISERS"* when trying to advertise, thrown by the *AdvertiseCallback onStartFailure()* method. This error is defined by Google in [34] as "not having any advertising instance available". After throwing the advertisement error, the EN will drop the advertisement and generate a new RPI to be sent. Since Android OS does not implement a timeout to automatically remove advertisements and the advertisement slots are only released by the App that occupied them, if a malicious App keeps occupying slots without releasing them, the advertisement error described above will repeat indefinitely. Using an already installed App with Bluetooth access, an attacker that controls an SDK can use the Bluetooth advertising feature of the device and start this attack unknowingly to the owner. This attack affects the EN and any other App depending on the Bluetooth advertisement slots.

V. IMPACT ASSESSMENT OF INTERNAL DOS-BASED ATTACKS

This section presents the impact assessment of a specific category of internal DoS-based attacks, namely, Battery Exhaustion, Storage Drain, and the Advertising Overflow attacks. The assessment on each attack was performed by using eight smartphones of three different models as presented in Table 3. Two devices are used as targets and the other six are used as attackers. The Target 1 (T_1) consists of a Nokia 8 model, featuring 4GB of RAM, a Snapdragron 835 CPU, and with an Android OS version 9 (Pie). The Target 2 (T_2) consists of a Samsung Tab A 2019 model, featuring 2GB of RAM, a Lassen O+ CPU, and with an Android OS version 9 (Pie). Initial tests conducted with one attacker towards both targets defined revealed no measurable impact regarding the Battery Exhaustion and the Storage Drain attacks. Thus, instead of using a one-to-one scenario, six attackers were used. The six attackers are labeled as Attacker 1 to Attacker 6 ($A_{1\text{-}6}$), and the set consist of Altice S23 models, featuring 1GB of RAM, a Cortex-A53 CPU, and with an Android OS version 8.1 (Oreo). All eight devices support Bluetooth version 5.

TABLE 3. Models and specifications for the smart phones used.

Device	Model	RAM	CPU	Android version
T_1	Nokia 8	4GB	Snapdragon 835	9
T_2	Samsung Tab A2019	2GB	Lassen O+	9
$A_{1\text{-}6}$	Altice S23	1GB	Cortex-A53	8.1

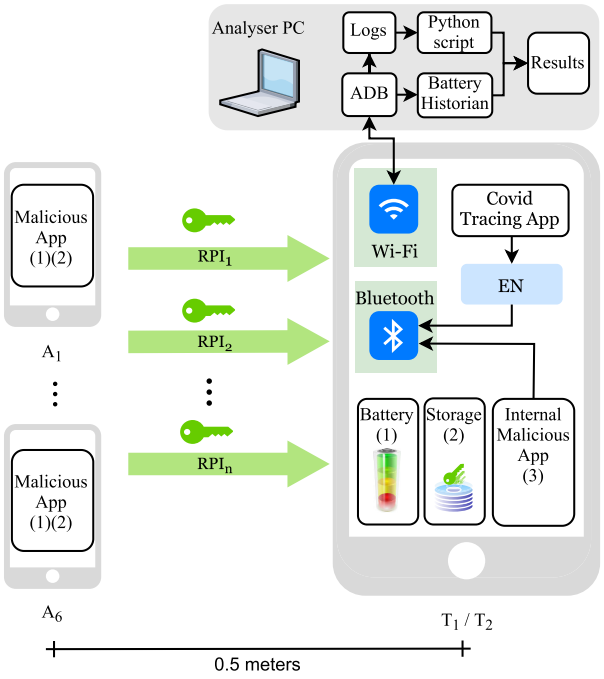

FIGURE 5. Testbed environment.

A testbed environment was set up according to Fig. 5. The attackers have a Malicious App installed, which is an Android App developed to send multiple simultaneous advertisements (15 advertisements is the maximum on all devices tested), each one containing an RPI. The generation of RPIs and advertising settings follows the publicly available source code for EN [33]. In both target devices the Portuguese Contact Tracing App (Stayaway Covid [35]) was installed and the EN was enabled in the settings. Three attacks were performed: (1) the reception and processing of RPIs are used to check battery discharge to measure the impact of battery exhaustion attack; in (2) the reception and processing of RPIs are used to monitor storage usage to measure the impact of storage drain attack, and in (3) the use of an Internal Malicious App allows to measure the impact of the Advertising Overflow attack.

To collect the results, an Analyser PC was connected via Wi-Fi to the target devices, and with the following tools installed: Android Debug Bridge (ADB) [36] and Battery Historian [37]. The ADB tool monitors in real time the targets and generates bug reports and logs, providing information about the state and tasks of the device during the tests such as battery and CPU usage, Apps on foreground, and wake-locks. The Battery Historian tool was used to present the information into readable graphs and charts. A Python script was developed to process the accumulated logs from ADB tool and to generate statistics about the number of advertisements scanned and other EN-related information such as scan's interval and frequency, and the errors thrown by EN. The results are statistic data from the Battery Historian and the Python script.

The tests are performed using a Baseline scenario, i.e. a scenario with no attackers or Malicious Apps, which is compared with an Attack scenario. Specific details and results are presented in the following subsections. Prior to each test, each target is rebooted, the Google Play Services and the Stayaway Covid App cache and storage is wiped, and the logs collection over ADB and the battery data collection is restarted.

A. BATTERY EXHAUSTION

According to [18], an adversary can advertise either a valid or invalid RPI to a target device. In turn, the device will wake up and process the received RPI, these actions require energy and imply a battery discharge.

In the testbed presented in the Fig 5, the T_1 and T_2 devices were set to a normal operation mode, using the default EN advertising interval, i.e. "*INTERVAL_MEDIUM*" (one advertisement around every 250ms) and "*TX_POWER_LOW*" power level. $A_{1\text{-}6}$ were set to generate the maximum number of advertisements (15), the advertising frequency was set to "*INTERVAL_LOW*" (one advertisement around every 100ms) and the power level was set to "*TX_POWER_HIGH*" (the largest visibility range for an advertising packet).

A round of 10 tests was performed for each target, where each test had a duration of 1 hour. Before executing each test, the Bluetooth setting was disabled and enabled again to trigger the first EN scan. After each test, Bluetooth is disabled, and a new bug report is generated with the last hour's statistics. In the Baseline scenario, the attackers behave with EN active in a regular operation. In the "Attack" scenario, the attackers were deployed with a Malicious App active around the target. For each scenario, the battery discharge for each target was collected and Battery Historian provided calculations for the device's battery consumption and the CPU usage time for the duration of the test.

The results for battery discharge (in percentage) and Bluetooth CPU time (in seconds) for each test, and for the T_1 and T_2 are presented in Table 4, where the Baseline scenario is represented as "B" and the Attack scenario, as "A". Average value, its standard deviation (σ), and the ratios between the Attack and the Baseline scenarios, were calculated. The results of each test were plotted in Fig. 6.

The results show that the average battery discharge for T_1 resulted in 0.918% for the Baseline scenario, and in 1.796% for the Attack scenario. For T_2, the average battery discharge was 0% for the Baseline scenario and 0.918% for the Attack scenario. The Baseline scenario for T_2 resulted in a 0% because the Android operating system could not measure the small amount of energy taken by Bluetooth in a regular

TABLE 4. Battery discharge tests results for the $T_{1\text{-}2}$ devices.

Test #	T_1 Battery Dis. (%)		T_1 CPU time (seconds)		T_2 Battery Dis. (%)		T_2 CPU time (seconds)	
	B	A	B	A	B	A	B	A
1	0.780	2.940	1.280	12.52	0	1.000	2.008	17.648
2	0.740	1.590	0.830	9.025	0	1.100	1.190	20.984
3	0.830	1.530	0.920	13.290	0	0.950	2.096	16.760
4	1.000	1.960	1.415	14.065	0	0.890	1.800	19.144
5	0.960	1.570	1.290	13.660	0	0.890	1.208	12.816
6	0.940	2.000	0.680	13.665	0	0.920	2.146	16.648
7	0.950	1.490	1.000	11.320	0	0.900	2.560	17.840
8	0.990	1.660	0.820	10.010	0	0.850	2.692	15.760
9	1.000	1.450	0.930	10.270	0	0.790	2.794	13.144
10	0.990	1.770	0.740	12.050	0	0.890	2.122	16.860
Avg	**0.918**	**1.796**	**0.991**	**11.988**	**0**	**0.918**	**2.062**	**16.760**
σ	**0.1**	**0.4**	**0.3**	**1.8**	**0**	**0.1**	**0.6**	**2.5**
Ratio	**1.956**		**12.102**		**n.a**		**8.130**	

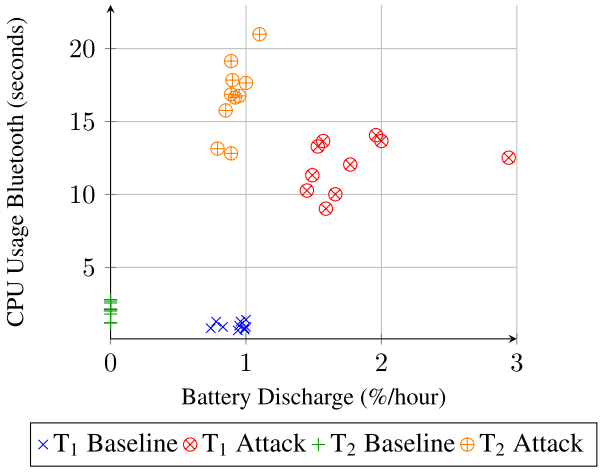

FIGURE 6. Battery consumption and bluetooth CPU usage for T_1 and T_2.

operation of this device. We assume that the differences of the Baseline scenario results between T_1 and T_2 can be explained by the fact that T_1 is 3 years older than the other device. The Attack/Baseline scenarios ratio for T_1 was 1.956 (for T_2 it could not be obtained since Baseline was 0%). Regarding the Bluetooth CPU usage time, the obtained ratios for T_1 and T_2 show an increase of 12.102 times and 8.130 times, respectively.

B. STORAGE DRAIN

According to [18], an adversary can generate and advertise valid RPIs that the target's device will process and store. In case a high number of RPIs are received, they will occupy storage space to an extent.

In the testbed presented in the Fig 5, the T_1 and T_2 devices were set to a normal operation mode, transmitting one RPI at a time. $A_{1\text{-}6}$ were set to send the maximum number of valid RPIs to be processed and stored by the targets.

A round of 10 tests was performed for each device with a duration of 15 minutes, which amounts to around 3 EN scans per test. After the tests ends, the total storage space occupied by the Google Play services folder in the device is analyzed with a Python script. For each scenario the logs of the targets were collected to count the total number of advertisements.

In the Baseline scenario, EN is active in all the devices in a normal operation mode. In the Attack scenario, the attackers have the malicious App active instead of EN, and sending multiple advertisements to $T_{1\text{-}2}$.

The results for occupied storage space (in kBytes) and total number of RPIs received for each test, and for the T_1 and T_2 are presented in Table 5, where the Baseline scenario is represented as "B" and the Attack scenario, as "A". Average value, its standard deviation (σ), and the ratios between the Attack and the Baseline scenarios, were calculated. The results of each test were plotted in Fig. 7.

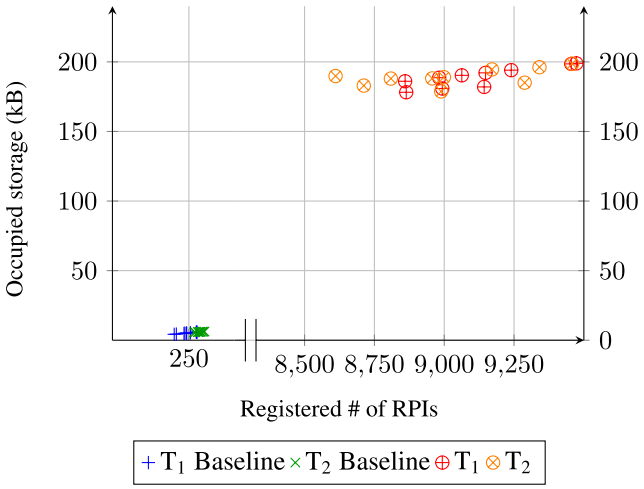

FIGURE 7. Storage drainage statistics for both T_1 and T_2 (15 minutes test).

The results show that the average storage space occupied for T_1 for the baseline was 5.14 kB, and 188.96 kB for the attack. For T_2, the occupied space was 5.96 kB for the baseline and 189.10 kB for the attack. The attack/baseline ratio for the storage occupation in T_1 was 36.758 and 31.726 in T_2. In both scenarios, the devices registered a similar total number of RPIs. The baseline for both devices registered

TABLE 5. Storage occupancy tests for the devices $T_{1\text{-}2}$.

Test #	T_1 Storage (kB)		T_1 # of RPIs		T_2 Storage (kB)		T_2 # of RPIs	
	B	A	B	A	B	A	B	A
1	5.78	190.32	275	9063	6.11	187.99	297	8809
2	4.39	198.93	209	9473	5.88	194.59	280	9171
3	4.91	180.85	234	8993	6.17	178.77	294	8989
4	5.82	178.14	277	8864	6.03	185.05	287	9288
5	5.10	182.00	243	9143	5.69	189.81	271	8611
6	5.36	192.11	255	9148	6.22	188.98	301	8999
7	4.24	198.56	202	9455	6.05	196.16	289	9341
8	4.98	188.62	237	8982	5.96	188.10	284	8957
9	5.75	186.06	274	8860	5.88	198.62	280	9458
10	5.08	194.04	242	9240	5.61	182.95	267	8712
Avg	**5.14**	**188.96**	**244.8**	**9122.1**	**5.96**	**189.10**	**285**	**9033.5**
σ	**0.55**	**7.22**	**26.3**	**217.1**	**0.2**	**6.1**	**10.9**	**278.8**
Ratio	**36.758**		**37.263**		**31.726**		**31.696**	

between 200 and 300 RPIs. In contrast, the attack scenario had between 8600 and 9600 registered RPIs. The total of RPIs presented similar ratios, of 37.263 for T_1 and 31.696 for T_2. When the total of received RPIs grows, the occupied storage also increases. Internally, EN removes stored duplicates and uses compression techniques to reduce the occupied storage.

C. ADVERTISING OVERFLOW

The Advertising Overflow attack can compromise the advertising feature of the EN system since it occupies all the advertising slots, for any given period of time.

In the testbed presented in the Fig 5, for the Baseline scenario, EN is set active in $T_{1\text{-}2}$ and in a normal operation mode. In the Attack scenario, $T_{1\text{-}2}$ ran the malicious App and EN at the same time.

The test had a duration of 1 hour for each of the two scenarios, baseline and attack. Before executing each test, each target device was rebooted and the Bluetooth was restarted to trigger the first EN scan. For each scenario the target's logs were collected to analyze errors and the successful advertisements.

The tests for the T_1 and T_2 are presented in Fig. 8. The figure displays the total number of placed advertisements during an interval of 60 minutes for the two scenarios. In both scenarios, the advertising attempts by EN are also displayed and have a different symbol if the attempt was successful or not. The maximum limit of advertisements for the devices $T_{1\text{-}2}$ is 15. The figure shows both scenarios had a regular number of advertisements throughout the test. The baseline for both devices had one advertisement placed in a steady frequency, about 4-5 minutes for T_2 and 8-9 minutes for T_1. The results also show that T_2 has a higher advertising frequency than T_1. During the attack scenario, both T_1 and T_2 had a total of 15 advertisements placed, and no advertisements were successfully placed by EN. The advertising attempts have a similar frequency to ones in the baseline scenario. The errors registered in the attack scenario did not change the frequency of advertising attempts by EN.

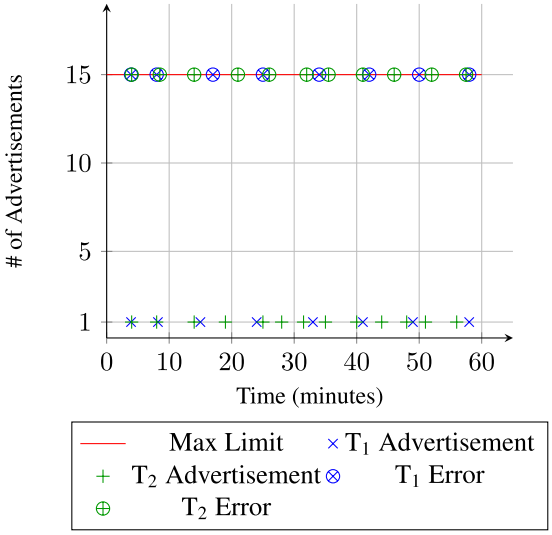

FIGURE 8. Advertisements placed by EN during the baseline and the attack tests.

VI. DISCUSSION

The tests for the selected attacks show that they have differing levels of impact and severity.

Battery Exhaustion attack imposed a maximum discharge of 3% during the test of one hour, using almost 2 times more battery than the baseline scenario. Although 3% of battery discharge in one hour can be considered as not relevant, if the number of attackers increase, higher amounts of RPIs will be sent, and this would increase the ratio between the baseline and attack results, thus presenting more impact Performing this attack with a larger volume of advertisements may generate a noticeable battery drain, especially in older devices.

The Storage Drainage attack imposed a low storage occupancy when compared to the total storage mounted in the test devices (ranging from 64 to 128 GBytes) which can be considered not relevant. Taking into account that this attack tries to fill the storage space of the target, the tests show

that, despite already sending 30 times more RPIs than the baseline, it still requires more RPIs to achieve that. Also, it is not possible to assure that the space occupied by the RPIs will remain the same throughout the day as the compression techniques are applied. They are not publicly available in the reviewed documentation, so the only assurances given are based on reverse engineering and behavior analysis.

Both Storage Drainage and Battery Exhaustion attacks require other devices to be close to their targets. These attacks can be successful even if their impact in software and hardware is not relevant, but if they are noticeable by the EN users, i.e. a user checking that the storage space is occupied by EN and he might uninstall the EN-based App.

In contrast to the previous attacks, the Advertising Overflow attack effectively blocked advertisements from EN during the entire test and neither the Android OS nor the EN showed any error notification. Thus, it can be assumed that this attack compromises the operation of any contact tracing Apps, for any given time period. In particular, this attack blocks the operation of any EN-based Apps such as the ones presented in Table 1, used by over 38 million European citizens.

The Advertising Overflow attack was reported to Google Security Team and recognized as a bug.

To mitigate the impact of this attack the following strategies can be considered:

- The advertisements from EN can be prioritized above all other Apps. In case of a device reboot, the EN will be able to advertise even if the attacker attempts to start advertising;
- EN can have a dedicated advertisement slots so that other Apps would not be able to interfere with its' behavior. This mitigation would also work in cases where an attacker occupies all the advertising slots before the user activates EN;
- EN can notify the user if it fails to advertise in, e.g. 3 consecutive intervals. Currently, the EN fails silently, and the user is not aware of this failure.

The Advertising Overflow attack silently affects the efficiency of digital contact tracing itself and renders the process of uploading the TEKs useless, since no other user will receive the respective RPIs generated. Deploying the attack at small or larger scale will always severely impact the efficiency of a digital contact tracing solution based on EN.

The Advertising Overflow attack was tested with a previously installed malicious App, but it can be deployed through a SDK used by other Apps, taking advantage of an already installed App that has Bluetooth permissions access, as described in [31].

VII. CONCLUSION

The digital contact tracing Apps based on EN are used to aid in the fight against COVID-19 rely on widespread usage and data integrity to monitor the exposure of an user.

In this paper, a novel Advertising Overflow attack is presented. Using a malicious App, an attacker can block the advertisements of EN by occupying the Bluetooth advertising slots of an Android OS device. This attack can be conveyed by a Malicious App or by an SDK.

This and other two DoS-based attacks (Battery Exhaustion and Storage Drain) were tested on Android OS devices to test and measure their impact. The Battery Exhaustion attack presented an increase of 1.956 times the usual battery discharge. Regarding Storage Drain the results show that, despite occupying 30 to 40 times more storage space, the impact of the Storage Drain is not relevant in overall Android OS.

Regarding the advertising overflow attack the tests show that it can effectively stop the expected advertising behavior of the EN and any Contact Tracing Apps depending on it. This attack is not limited to a interval of 1 hour, as it can disrupts the operation of EN for any given period of time and affects also any other Apps that use Bluetooth advertisements. The impact of Advertising Overflow attack can also be further tested on other BLE features and focus on testing this attack in the Apple iOS.

ACKNOWLEDGMENT

The Advertising Overflow attack was reported by the authors of this article to the Google security team, who confirmed it as a bug. They would like to thank the Google security team for their recognition and for including the authors of the disclosure and testing of this attack on Google Application Security honorable mentions board.

REFERENCES

[1] S. Altmann, L. Milsom, H. Zillessen, R. Blasone, F. Gerdon, R. Bach, F. Kreuter, D. Nosenzo, S. Toussaert, and J. Abeler, "Acceptability of app-based contact tracing for COVID-19: Cross-country survey study," *JMIR mHealth uHealth*, vol. 8, no. 8, Aug. 2020, Art. no. e19857.

[2] M. Leslie, "COVID-19 fight enlists digital technology: Contact tracing apps," *Engineering*, vol. 6, no. 10, pp. 1064–1066, Oct. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2095809920302484

[3] D. Skoll, J. C. Miller, and L. A. Saxon, "COVID-19 testing and infection surveillance: Is a combined digital contact-tracing and mass-testing solution feasible in the united states?" *Cardiovascular Digit. Health J.*, vol. 1, no. 3, pp. 149–159, Nov. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2666693620300360

[4] A. M. Wilson, N. Aviles, J. I. Petrie, P. I. Beamer, Z. Szabo, M. Xie, J. McIllece, Y. Chen, Y.-J. Son, S. Halai, T. White, K. C. Ernst, and J. Masel, "Quantifying SARS-CoV-2 infection risk within the Google/Apple exposure notification framework to inform quarantine recommendations," *medRxiv*, 2020. [Online]. Available: https://www.medrxiv.org/content/early/2020/09/17/2020.07.17.20156539, doi: 10.1101/2020.07.17.20156539.

[5] V. Iovino, S. Vaudenay, and M. Vuagnoux, "On the effectiveness of time travel to inject COVID-19 alerts," in *Topics in Cryptology—CT-RSA 2021*, K. G. Paterson, Ed. Cham, Switzerland: Springer, 2021, pp. 422–443.

[6] S. Vaudenay. (2020). *Analysis of DP3T—Between Scylla and Charybdis*. [Online]. Available: https://eprint.iacr.org/2020/399.pdf

[7] G. Avitabile, D. Friolo, and I. Visconti, "Terrorist attacks for fake exposure notifications in contact tracing systems," in *Applied Cryptography and Network Security*, K. Sako and N. O. Tippenhauer, Eds. Cham, Switzerland: Springer, 2021, pp. 220–247.

[8] S. Vaudenay, "Centralized or decentralized? The contact tracing dilemma," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 531, 2020.

[9] (Jun. 2021). *Mobile Contact Tracing Apps in EU Member States*. [Online]. Available: https://ec.europa.eu/info/live-work-travel-eu/coronavirus-response/travel-during-coronavirus-pandemic/mobile-contact-tracing-apps-eu-member-states_en

[10] Google. (2021). *Exposure Notifications API*. [Online]. Available: https://developers.google.com/android/exposure-notifications/exposure-notifications-api

[11] J. Siva, J. Yang, and C. Poellabauer, "Connection-less BLE performance evaluation on smartphones," *Procedia Comput. Sci.*, vol. 155, pp. 51–58, Jan. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919309238

[12] Google and Apple. (Apr. 2020). *Exposure Notification Bluetooth Specification*. [Online]. Available: https://blog.google/documents/70/Exposure_Notification_Bluetooth_Specification_v1.2.2.pdf

[13] Google. (Sep. 2020). *Bluetooth Low Energy Android Open Source Project*. [Online]. Available: https://source.android.com/devices/bluetooth/ble

[14] H.-C. Hsiao, C.-Y. Huang, B.-K. Hong, S.-M. Cheng, H.-Y. Hu, C.-C. Wu, J.-S. Lee, S.-H. Wang, and W. Jeng, "An empirical evaluation of Bluetooth-based decentralized contact tracing in crowds," 2020, *arXiv:2011.04322*. [Online]. Available: https://arxiv.org/abs/2011.04322

[15] F. Legendre, M. Humbert, A. Mermoud, and V. Lenders, "Contact tracing: An overview of technologies and cyber risks," Jul. 2020, *arXiv:2007.02806*. [Online]. Available: http://arxiv.org/abs/2007.02806

[16] E. Daw, "Component-based comparison of privacy-first exposure notification protocols," Cryptol. ePrint Arch., Tech. Rep. 2020/586, 2020. [Online]. Available: https://eprint.iacr.org/2020/586

[17] K. Pietrzak, "Delayed authentication: Preventing replay and relay attacks in private contact tracing," in *Progress in Cryptology—INDOCRYPT 2020* (Lecture Notes in Computer Science), K. Bhargavan, E. Oswald, and M. Prabhakaran, Eds. Cham, Switzerland: Springer, 2020, pp. 3–15.

[18] Y. Gvili, "Security analysis of the COVID-19 contact tracing specifications by Apple Inc. and Google Inc," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 428, 2020.

[19] G. Avitabile, V. Botta, V. Iovino, and I. Visconti, "Towards defeating mass surveillance and SARS-CoV-2: The pronto-C2 fully decentralized automatic contact tracing system," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 493, 2020.

[20] F. Buccafurri, V. De Angelis, and C. Labrini, "A privacy-preserving solution for proximity tracing avoiding identifier exchanging," May 2020, *arXiv:2005.10309*. [Online]. Available: http://arxiv.org/abs/2005.10309

[21] J. Huang, V. Yegneswaran, P. Porras, and G. Gu, "On the privacy and integrity risks of contact-tracing applications," Dec. 2020, *arXiv:2012.03283*. [Online]. Available: http://arxiv.org/abs/2012.03283

[22] P.-O. Dehaye and J. Reardon, "SwissCovid: A critical analysis of risk assessment by Swiss authorities," Jun. 2020, *arXiv:2006.10719*. [Online]. Available: http://arxiv.org/abs/2006.10719

[23] L. Baumgärtner, A. Dmitrienko, B. Freisleben, A. Gruler, J. Höchst, J. Kühlberg, M. Mezini, R. Mitev, M. Miettinen, A. Muhamedagic, T. D. Nguyen, A. Penning, D. Pustelnik, F. Roos, A.-R. Sadeghi, M. Schwarz, and C. Uhl, "Mind the GAP: Security & privacy risks of contact tracing apps," in *Proc. IEEE 19th Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, Dec. 2020, pp. 458–467.

[24] A. Gangavarapu, E. Daw, A. Singh, R. Iyer, G. Harp, S. Zimmerman, and R. Raskar, "Target privacy threat modeling for COVID-19 exposure notification systems," Sep. 2020, *arXiv:2009.13300*. [Online]. Available: http://arxiv.org/abs/2009.13300

[25] R. Gennaro, A. Krellenstein, and J. Krellenstein. (Aug. 2020). *Exposure Notification System May Allow for Large-Scale Voter Suppression*. [Online]. Available: https://static1.squarespace.com/static/5e937afbfd7a75746167b39c/t/5f47a87e58d3de0db3da91b2/1598531714869/Exposure_Notification.pdf

[26] S. Farrell and D. J. Leith. (May 2020). *A Coronavirus Contact Tracing App Replay Attack With Estimated Amplification Factors*. [Online]. Available: https://down.dsg.cs.tcd.ie/tact/replay.pdf

[27] A. Boutet, C. Castelluccia, M. Cunche, A. Dmitrienko, V. Iovino, M. Miettinen, T. D. Nguyen, V. Roca, A.-R. Sadeghi, S. Vaudenay, I. Visconti, and M. Vuagnoux, "Contact tracing by giant data collectors: Opening pandora's box of threats to privacy, sovereignty and national security," EPFL, Switzerland, Inria, France, JMU, Würzburg, Germany, Univ. Salerno, Italy, base23, Geneva, Switzerland, Tech. Univ. Darmstadt, Germany, Dec. 2020. [Online]. Available: https://hal.inria.fr/hal-03116024 and https://hal.inria.fr/hal-03116024/file/Digital_Contact_Tracing_2020-11.pdf

[28] B.-R. Chen and Y.-C. Hu, "Mitigating denial-of-service attacks on digital contact tracing," in *Proc. 18th Conf. Embedded Netw. Sensor Syst. (SenSys)* New York, NY, USA: Association for Computing Machinery, Nov. 2020, pp. 770–771, doi: 10.1145/3384419.3430599.

[29] M. R. Hussein, A. B. Shams, E. H. Apu, K. A. Al Mamun, and M. S. Rahman, "Digital surveillance systems for tracing COVID-19: Privacy and security challenges with recommendations," *CoRR*, vol. abs/2007.13182, 2020. [Online]. Available: https://arxiv.org/abs/2007.13182 and https://dblp.org/rec/journals/corr/abs-2007-13182.bib

[30] F. Rowe, O. Ngwenyama, and J.-L. Richet, "Contact-tracing apps and alienation in the age of COVID-19," *Eur. J. Inf. Syst.*, vol. 29, no. 5, pp. 545–562, 2020. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/0960085X.2020.1803155

[31] P.-O. Dehaye and J. Reardon, "Proximity tracing in an ecosystem of surveillance capitalism," Sep. 2020, *arXiv:2009.06077*. [Online]. Available: http://arxiv.org/abs/2009.06077

[32] D. Young. (Jun. 2020). *BLE Advertising Fails*. [Online]. Available: https://stackoverflow.com/a/62267146

[33] Google. (2020). *Exposure Notifications API Internals*. Accessed: Feb. 1, 2021. [Online]. Available: https://github.com/google/exposure-notifications-internals

[34] (2021). *Advertise Callback Android Developers*. [Online]. Available: https://developer.android.com/reference/android/bluetooth/le/AdvertiseCallback#ADVERTISE_FAILED_TOO_MANY_ADVERTISERS

[35] (Oct. 2020) *Stayaway Covid App*. Accessed: Feb. 1, 2021. [Online]. Available: https://stayawaycovid.pt/

[36] Google. *Android Debug Bridge (ADB)*. Accessed: Feb. 2, 2021. [Online]. Available: https://developer.android.com/studio/command-line/adb

[37] *Profile Battery Usage With Batterystats and Battery Historian*. Accessed: Feb. 2, 2021. [Online]. Available: https://developer.android.com/topic/performance/power/setup-battery-historian

HENRIQUE FARIA received the B.S. degree in computer engineering from the Instituto Politécnico de Viana do Castelo, Portugal, where he is currently pursuing the M.S. degree in cybersecurity. Since 2018, he has been a Software Engineer with Atlanse, Portugal. His research interests include analysis of vulnerabilities and exploits in mobile devices, and the development of safe practices in software engineering.

SARA PAIVA received the Ph.D. degree in computer science from Vigo University, Spain, in 2011. She is currently a Postdoctoral Researcher with the SMIOT Group, Oviedo University. She is an Assistant Professor with the Instituto Politécnico de Viana do Castelo, Portugal and a Researcher with Algoritmi. Her research interests include applied mobile computing and mobility in smart cities. She is currently the Vice-Chair of the IEEE Smart Cities Marketing Committee and the General Co-Chair of the EAI Convention on Smart Cities 360°, since the 2020 edition.

PEDRO PINTO received the M.S. degree in communication networks and services from the University of Porto, Portugal, in 2007, and the Ph.D. degree in telecommunications jointly from the Universities of Minho, Aveiro, and Porto, Portugal, in 2015. He is currently an Assistant Professor, the Director of the M.S. degree in cybersecurity, and the Data Protection Officer with the Instituto Politécnico de Viana do Castelo, Portugal. His research interests include areas of computer networks, data privacy, and cybersecurity. He is also a member of the INESC TEC Research Institution.