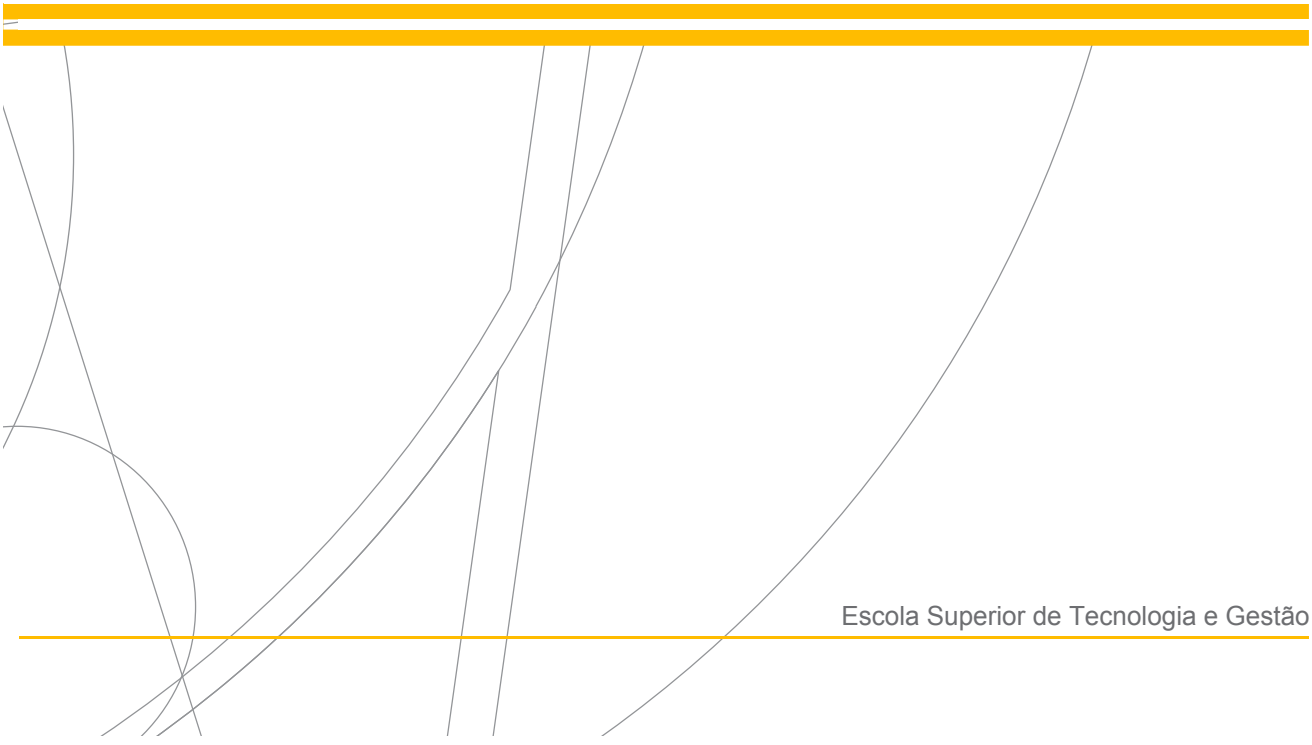
Instituto Politécnico
de Viana do Castelo

DETECTION AND EXPLOITATION OF VULNERABILITIES IN THE COVID-19 EXPOSURE NOTIFICATION SYSTEM

Henrique Faria



Escola Superior de Tecnologia e Gestão

Instituto Politécnico
de Viana do Castelo

Henrique Bacelar da Silva e Faria

DETECTION AND EXPLOITATION OF VULNERABILITIES IN THE COVID-19 EXPOSURE NOTIFICATION SYSTEM

Nome do curso de Mestrado

Mestrado em Cibersegurança

Trabalho efetuado sob a supervisão de

Professora Sara Paiva

Professor Pedro Pinto

Março de 2022

Mestrado em Cibersegurança
Master in Cybersecurity

Detection and Exploitation of Vulnerabilities in the COVID-19 Exposure Notification System

a master's thesis authored by

Henrique Bacelar da Silva e Faria

and supervised by

Sara Maria da Cruz Maia de Oliveira Paiva
Professora Adjunta, IPVC

Pedro Filipe Cruz Pinto
Professor Adjunto, IPVC

This thesis was submitted in partial fulfilment of the requirements for the Master's degree in Cybersecurity at the Instituto Politécnico de Viana do Castelo on

ipvc

24 March, 2022

Abstract

The contact tracing mobile Apps are one of the many initiatives to fight the COVID-19 virus. These Apps use the Exposure Notification (EN) system available on Google and Apple's operating systems. However, contact tracing applications depend on the availability of Bluetooth interfaces to exchange proximity identifiers that, if compromised, directly impact the effectiveness of the apps.

This thesis discloses the Advertising Overflow attack, a novel internal Denial of Service (DoS) attack targeting the EN system on Android Operating System (OS) 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.

The impacts of Advertising Overflow and other known DoS attacks, Battery Exhaustion and Storage Drainage, were evaluated using two smartphones as targets and another six smartphones as attackers. The attacks scenarios were compared against a baseline scenario. The results show that the Battery Exhaustion attack imposes a battery discharge rate 1.95 times superior to the baseline. Regarding the Storage Drain, the storage usage increased more than 30 times the baseline results. The results of the novel attack reveal that a malicious App is able to block the usage of Bluetooth advertising by any other App by any chosen time period, canceling the operation of the EN system and compromising the efficiency of any contact tracing App based on EN.

The macro analysis of the EN-related attacks and their categorizations also enabled the proposal of a novel taxonomy for EN-based attacks. This taxonomy identifies and categorizes the wide range of existing attacks according to their particularities and characteristics, with a granular multi-level approach, and it includes the most recent attacks. The proposed taxonomy allows a better understanding of EN's current attack vectors and procedures, and highlights areas or vulnerabilities that can be further explored, analyzed, and fixed.

Keywords: Exposure Notification. Android. Bluetooth. Covid-19.

Resumo

As aplicações móveis de rastreio de contactos são uma das muitas iniciativas para combater o vírus COVID-19. Estas Apps utilizam o sistema Exposure Notification (EN) disponível nos sistemas operativos da Google e da Apple. No entanto, as aplicações de rastreio de contactos dependem das interfaces Bluetooth do dispositivo móvel para transmitir identificadores que, se comprometidas, impactam diretamente a eficácia destas Apps.

Esta tese apresenta o Advertising Overflow, um novo ataque de Denial of Service (DoS), que afeta o sistema EN em dispositivos Android. O ataque é executado por uma aplicação maliciosa que ocupa todos os slots de Bluetooth Advertising num dispositivo Android, bloqueando qualquer tentativa de transmissão do sistema EN.

Os impactos do Advertising Overflow e de outros ataques DoS já conhecidos, o Battery Exhaustion e o Storage Drainage, foram avaliados utilizando dois smartphones como alvo e outros seis smartphones como atacantes. Os cenários de ataque foram comparados com um cenário base. Os resultados mostram que o ataque de Battery Exhaustion impõe uma taxa de descarga da bateria 1,95 vezes superior à descarga no cenário base. No ataque de Storage Drainage, a utilização de armazenamento aumentou mais de 30 vezes que o cenário base. Os resultados do novo ataque revelam que uma aplicação maliciosa é capaz de bloquear a utilização de Bluetooth Advertising por qualquer outra aplicação durante um período de tempo escolhido, bloqueando a operação do sistema EN e comprometendo a eficácia de qualquer aplicação de rastreio de contactos baseada no sistema EN.

A análise macro dos ataques relacionados com o EN e as suas categorizações permitiu a criação de uma nova taxonomia para ataques baseados no sistema EN. Esta taxonomia identifica e categoriza a vasta gama de ataques existentes de acordo com as seus particularidades e características, com uma abordagem granular, e inclui os ataques mais recentes. A taxonomia proposta permite uma melhor compreensão dos atuais vectores e procedimentos de ataque, e destaca áreas ou vulnerabilidades que podem ser mais exploradas, analisadas, e corrigidas.

Palavras-chave: Exposure Notification. Android. Bluetooth. Covid-19.

Contents

List of Figures

List of Tables

Acronyms

ADB Android Debug Bridge

API API

BLE Bluetooth Low-Energy

DoS Denial of Service

DP-3T Decentralized Privacy-Preserving Proximity Tracing

EN Exposure Notification

GAEN Google/Apple Exposure Notification

HTTPS HyperText Transfer Protocol Secure

KISS Keep It Simple Stupid

MCyber Master in Cybersecurity

OS Operating System

PACT Privacy Automated Contact Tracing

PEPP-PT Pan-European Privacy-Preserving Proximity Tracing

POC Proof of Concept

RPI Rolling Proximity Identifier

RPIK Rolling Proximity Identifier Key

SDK Software Development Kit

TCN Temporary Contact Numbers Protocol

TEK Temporary Exposure Key

UUID Universally Unique Identifier

WN Wireless Network

Chapter 1

Introduction

This chapter presents the context of this work concerning the worldwide COVID-19 pandemic and the technological measures implemented to mitigate the risks. Section 1.1 delineates the expectations for the investigation results and the main focus of the research and development. Section 1.2 expands on the current needs of the mobile contact tracing Apps and EN specifically. Section 1.3 details the objectives of this paper and its research. Section 1.4 presents the three contributions of this paper. Finally, Section 1.5 presents the organization and chapters of this paper.

1.1 Context

The spread of the pandemic virus COVID-19 motivated the development of technology to mitigate and fight the virus' spread [1]. Since it is a global issue, large corporations such as Google and Apple and governments have presented technological solutions for automated contact tracing [31].

The authors in [13] mention that using mobile contact tracing Apps which take advantage of the built-in sensors available in the devices is a valuable tool to control the COVID-19 virus spread. Using GPS or Bluetooth, mobile Apps can track users' encounters and the possibility of an infection. With the data collected, the system warns the users of the infection risk when another user reports an infection to the health authority [31]. The risk calculation of an infection can also be quantified using algorithms and techniques as analyzed in [40].

EN system, also known as Google/Apple Exposure Notification (GAEN), was released by Google and Apple, with similarities to an already existing protocol named Decentralized Privacy-Preserving Proximity Tracing (DP-3T), and consists of a decentralized architecture that, according to [37], is privacy-oriented. This architecture ensures interoperability between Apple and Android devices, allows the execution of the contact tracing tasks at the OS level, and prevents the OS from terminating or interfering with a task running on background [31]. The study in [40] 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. In comparison, the centralized approach handles the matching process of the anonymous identifiers in a central server controlled by the Health Authority [37].

1.2 Motivation

Since the success of a mobile contact tracing App lays in wide adoption amongst users [1], it is necessary to validate if the security and privacy of the users is assured. Contact tracing Apps have seen a rise in the Android and Apple stores' downloads and there have been multiple systems for automated mobile contact tracing as well. These Apps claim to be helpful in contact tracing and there have been recent statewide adoptions of specific applications as the official contact tracing Apps. The Apps also claim to consider privacy and anonymity concerns, but as shown by [1, 31, 38], authors have identified issues related to security incidents or insecure frameworks. EN is one of the systems that was widely adopted in Europe as the official contact tracing system [33].

A study of the existing attacks on the EN system is of the upmost importance to contextualize users concerns with privacy and security, and also measure the impact of existing attacks on the EN system's efficiency.

1.3 Objectives

Given the importance of the analysis and the impact measurement of existing attacks on the EN system, this thesis has the following objectives:

- Verify if there are more attacks on EN that have not been disclosed;
- Measure the impact of existing attacks on contact tracing efficiency and the EN system;
- Propose a taxonomy to classify the existing attacks and provide an overview of the current attack vectors and vulnerabilities.

1.4 Contributions

This work presents three contributions: the novel attack Advertising Overflow, an analysis of the impact of internal DoS-based attacks such as Advertising Overflow, Battery Exhaustion and Storage Drainage, and a taxonomy proposal for EN attacks.

The contributions are the following:

- Advertising Overflow - Attack that allows a malicious mobile App to exhaust all the advertisements slots, thus blocking other Apps from placing a Bluetooth advertisement. This contribution lead to the journal publication "An Advertising Overflow Attack Against Android Exposure Notification System Impacting COVID-19 Contact Tracing Applications" [14];
- Impact of DoS-based attacks - Analysis of the impact of Battery Exhaustion, Storage Drainage and Advertising Overflow on Android devices.
- Taxonomy proposal - Novel taxonomy for EN attacks that identifies and categorizes the wide range of existing attacks according to their particularities and characteristics, with a granular and multi-level approach, and it includes the most recent attacks. This contribution lead to a submission of a paper currently in review to the journal "IEEE Security and Privacy";

1.5 Organization

This paper is organized as follows. In Chapter 2, the EN system and its inner workings are analyzed, what features it has and how it uses them. Chapter 3 explores the related work with a review on existing attacks for EN and also presents a state-of-art on the

categorization of these attacks. Chapter 4 details the novel advertising overflow attack. Chapter 5 presents the tests done to assess the efficiency of the attacks analyzed and how they compare to the novel Advertising Overflow attack. Chapter 6 presents the taxonomy proposal and details the reasoning for each new category and the attacks related to them. Finally, Chapter 7 draws conclusions and points to future work.

Chapter 2

Contact Tracing Apps and the Exposure Notification System

COVID contact tracing Apps have been developed worldwide to assist in the detection of people infected by COVID-19 and in preventing the increase of transmission chains.

For the Apps based on EN, Google and Apple defined that each country can only use one digital contact tracing App. Table 2 compiles information from [33] and lists the currently available Contact Tracing Apps in Europe, the system they use (EN or other), ordered by 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 39 million installs by European citizens. Table 2 compiles information about other known contact tracing Apps used in the rest of the world. The Apps of the countries presented in this table use EN systems but also alternatives such as the BlueTrace protocol or other using GPS, QR codes or both Bluetooth and GPS alternatives. Fig. 2.1 presents the comparison of the total of downloads for each of the contact tracing systems.

EN-based contact tracing Apps require a mobile device with a compatible hardware and Android OS version, including EN and Bluetooth Low-Energy (BLE) support. Fig. 2.2 presents the interactions performed by an EN-based contact tracing App using Android OS.

Table 2.1: European contact tracing Apps and their systems.

Contact Tracing App	**System**	**# of Installs**	**Country**
Corona-Warn-App	EN	+10M	Germany
NHS COVID-19	EN	+10M	United Kingdom
StopCovid	Other	+10M	France
Immuni	EN	+5M	Italy
Radar Covid	EN	+5M	Spain
Coronalert	EN	+1M	Belgium
CoronalMelder	EN	+1M	Netherlands
eRouška	EN	+1M	Czechia
Koronavilkku	EN	+1M	Finland
ProteGO Safe	EN	+1M	Poland
STAYAWAY Covid	EN	+1M	Portugal
SwissCovid	EN	+1M	Switzerland
COVID Tracker	EN	+500k	Ireland
Smittestop	EN	+500k	Denmark
Apturi Covid	EN	+100k	Latvia
HOIA	EN	+100k	Estonia
Korona Stop LT	EN	+100k	Lithuania
#OstaniZdrav	EN	+100k	Slovenia
Smittestopp	EN	+100k	Norway
Stopp Corona	EN	+100k	Austria
VírusRadar	Other	+100k	Hungary
COVIDAlert	EN	+50k	Malta
Stop COVID-19	EN	+50k	Croatia
CovTracer-EN	EN	+10k	Cyprus

Table 2.2: Non-european contact tracing Apps and their system.

Contact Tracing App	**System**	**# of Installs**	**Country**
COVIDSafe	BlueTrace	+1M	Australia
HaMagen	Other (using GPS)	+1M	Israel
NZ COVID Tracer	Other (using QR Codes)	+1M	New Zealand
TraceTogether	BlueTrace	+1M	Singapore
Tabaud	EN	+1M	Saudi Arabia
BeAware Bahrain	Other (using GPS)	+100k	Bahrain
GH Covid-19 Tracker App	Other (using GPS)	+5k	Ghana
Jersey COVID Alert	EN	+5k	US (Jersey)

According to [23], 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 sending

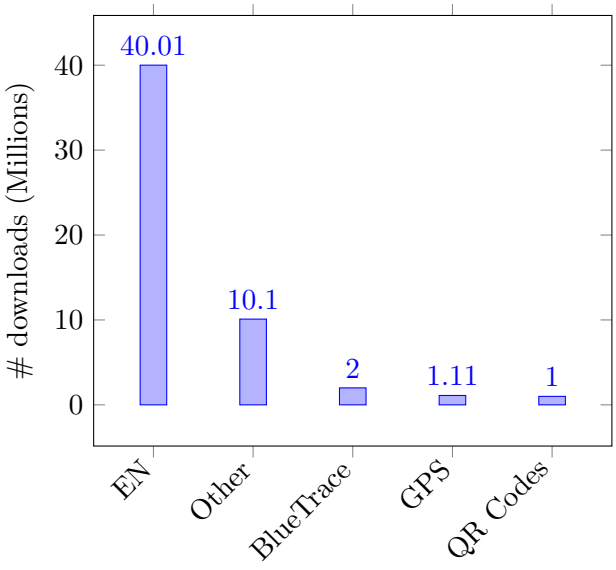


Figure 2.1: Comparison of total users of each contact tracing system identified in Table 2 and Table 2.

them to EN.

- In case of an infection, retrieving the keys for the last 14 days and upload them to the server.

The Contact Tracing Apps interact with the EN system and handle communications with external entities, for example, HyperText Transfer Protocol Secure (HTTPS) requests to the Health Authority server. The Contact Tracing App also allows the usage of an infection code to request the last 14 days keys from the EN and uploads them to the Health Authority. This event is possible when a user is diagnosed as infected. The App also requests Diagnosis Keys from the Health Authority periodically and passes them to the EN to check for matches.

The EN derives the Proximity Identifiers from the received keys and compares them with the ones already stored in the local storage. These stored identifiers were received via Bluetooth during the exchange process with other devices.

When an identifier marked as infected is matched with one stored in the user's device, EN sends that information to the Contact Tracing App that is responsible for informing and instructing the user.

The EN uses the broadcast procedures such as advertisements that do not need a connection to exchange identifiers. A device periodically advertises a data block, and

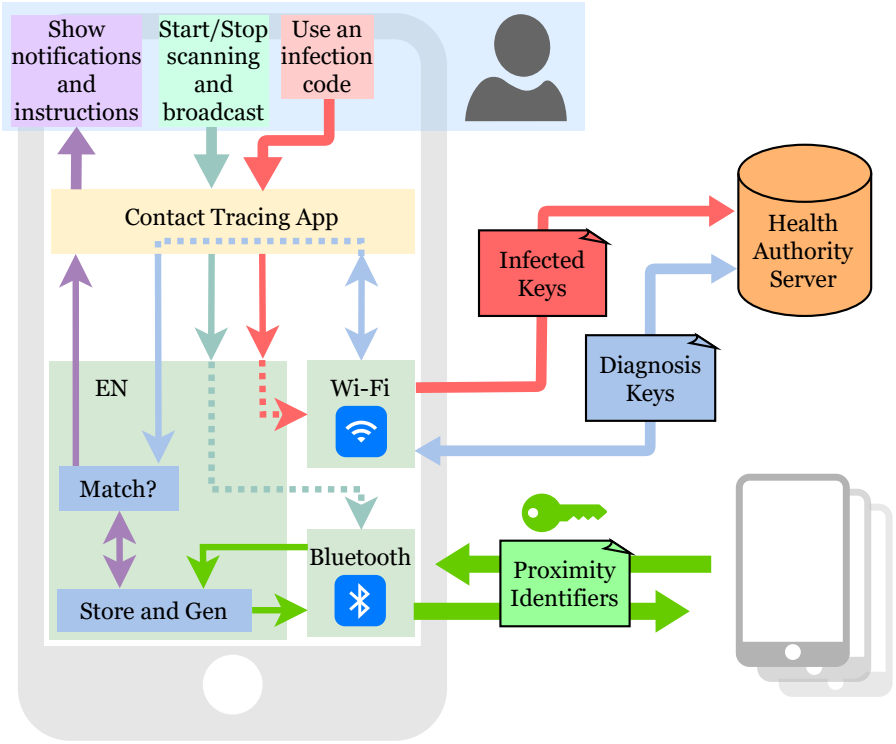


Figure 2.2: Schematic of how an EN contact tracing App works and its interactions with the user, other smartphones, and the Health Authority.

surrounding devices can scan and process it. When a device broadcasts an advertising channel packet, other devices in scanning mode are able to receive it. Since both actions are not synchronized, there are no guarantees that the advertising packet will be received [36]. In the BLE standard, the maximum size for a advertising block is 31 bytes. The option adopted by EN is sending a payload with a 16-bit Service 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 sent together with the Proximity Identifier. If the user disables the contact tracing operation, the Contact Tracing App sends an event to EN to stop the BLE broadcast and scan operations.

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

As detailed in [2, 29], the key generation procedure of EN is depicted in the Fig. 2.3. The system generates a daily random key named Temporary Exposure Key (TEK) used to derive a Rolling Proximity Identifier Key (RPIK) each 15 minutes, which is further used to derive a RPIs identifier. 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.

The health authority provides a code that allows an infected user 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.

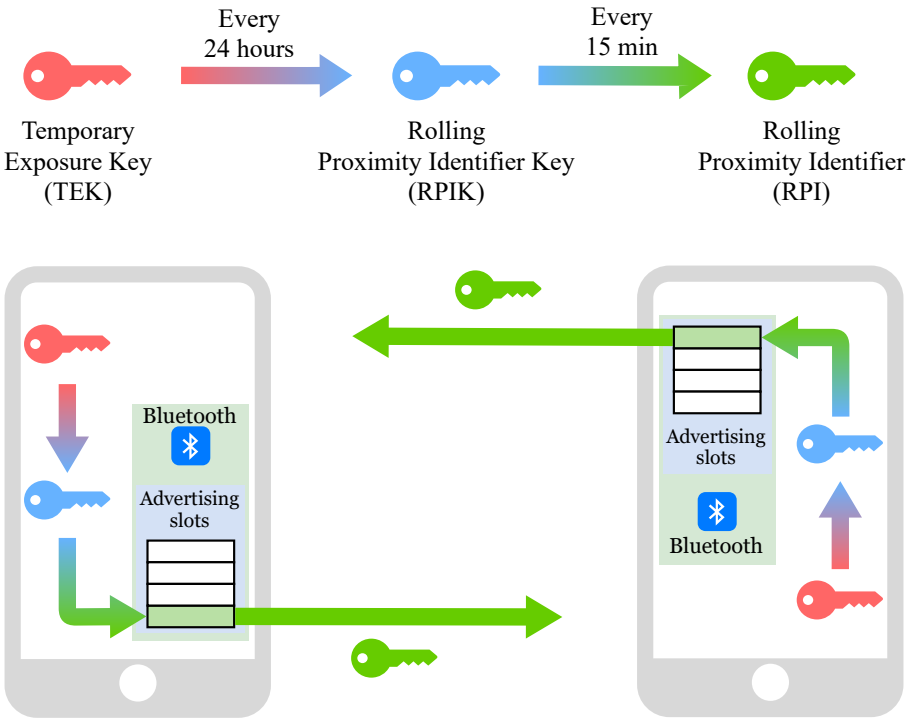


Figure 2.3: Schematic of how EN generates and exchanges the RPIs.

EN advertises over BLE the RPIs during several minutes. When a RPI rotates, the MAC address rotates as well to anonymize the user's device. According to [2] 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 [2]. Whenever it receives a valid RPI, it stores it to be used to check for matches.

The advertising and scanning features are present in devices running Android OS from version 4.3 [21]. These features are commonly seen in most modern smartphones [36].

Chapter 3

Related Work

This chapter is divided into two sections, a review on existing attacks (3.1) and a state-of-art on EN attacks categorization (3.2).

3.1 Review on EN Existing Attacks

A review was conducted to collect the related work on existing attacks concerning the EN system. It was divided into two processes: an analysis of Google's public documentation [22] and a systematic review of the existing attacks on EN.

Fig. 3.1 presents the review process and results. The research question was defined as "What are the known attacks of EN and other decentralized systems?". The selected search engine was Google Scholar and the query presented in Fig. 3.1 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 with a total of 40 attacks. The public documentation review obtained 1 document that contained 13 attacks. While analyzing the attacks in each paper, it was concluded that some of the attacks shared many similarities with another previous attack and the differences were minute details. Therefore, these similar attacks were grouped under the first published attack (main attack) and declared a sub-attack. The main attack's name is also used as the group's name. After analyzing the attacks, the following results were obtained: 23 Main attacks and 20 Sub-attacks.

Table 3.1 depicts each attack and their respective sources.

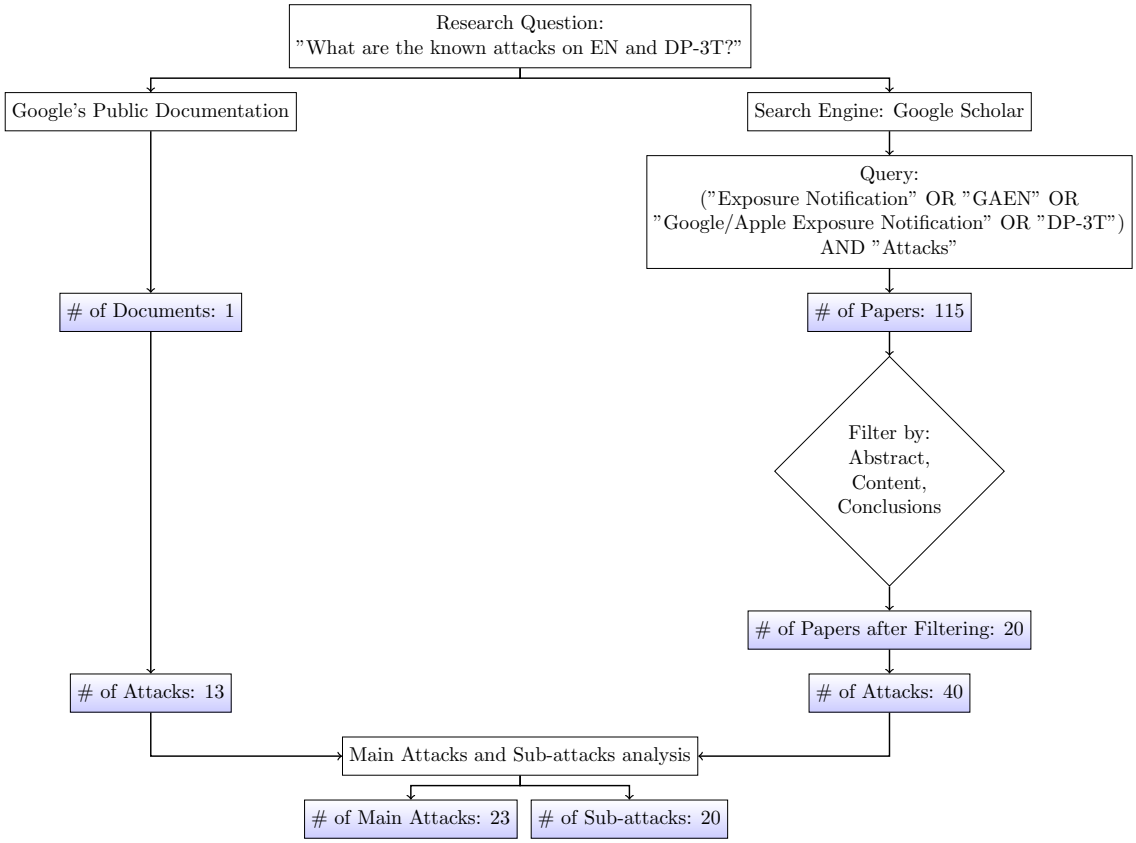


Figure 3.1: Stages, options and numbers of the review performed.

Table 3.1: Main Attacks and their sources.

Main Attack	[26]	[30]	[10]	[34]	[39]	[25]	[4]	[7]	[27]	[12]	[5]	[16]	[17]	[15]	[29]	[6]	[8]	[28]	[3]	[11]	[22]
1) Replay		•	•	•	•	•	•	•	•	•		•	•	•	•		•	•		•	•
2) Relay		•	•	•	•	•					•		•			•	•	•			•
3) Re-identification					•	•	•	•	•	•											•
4) False Report			•		•											•			•		•
5) Bluetooth-based Tracking					•							•						•			•
6) Compromised Mobile Device										•			•							•	•
7) Linking			•			•														•	•
8) Backend Impersonation					•		•	•													
9) Diagnosis Server Compromise							•	•													•
10) Profiling					•						•					•					
11) Storage Drainage						•											•	•			
12) Battery Exhaustion						•												•			
13) Coercion Threat					•							•									
14) Jamming	•	•																			
15) Network Traffic Analysis												•									•
16) Time Machine															•	•					
17) Forensics and Physical Access to Devices																					•
18) Linking Diagnosis Keys																					•
19) Occasional Disclosure					•																
20) Passive Disruption		•																			
21) Private Encounter Disclosure					•																
22) Simulated EN															•						
23) Troll						•															

Each disclosed attack is reviewed as follows:

1. Replay - An attacker sends RPIs that were received from another device [10]. In

[39], the author replayed released RPIs from the health authority and managed to trigger a exposure warning in the target. [30] identifies this attack as a cybersecurity risk for contact tracing systems. Others [4, 7, 25] defined scenarios such as tweaking the advertising interval from receiving and sending the RPIs. [12] presents an approach using a malicious Software Development Kit (SDK) injected in an App to replay received identifiers. [7] named this attack as Fregoli. This attack can also be considered the same as the Disruption risk in [22]. Mitigation strategies for this type of attack have been proposed using protocols and location of devices [4, 7, 25, 34, 39]. [27] mentions the Contact Pollution attack that is identical to the Replay in [39].

1.1. Belated Replay - Attacker derives RPIs from publicly published TEKs and replays them. It is also known as RPI Spoofing using reported diagnosis keys in [22]. In [29] this attack is presented as Still-Valid Keys (Kiss attack). The keys published are usually outdated, but some are still valid since RPI's life tolerance is set to 2 hours. In [29] 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.

1.2. False Positive - The attacker uses a SDK to obtain RPIs from a server and then replays them to the surrounding users [11].

1.3. Missile - The attacker uses a Bluetooth amplifier transmitter to broadcast infected RPIs to users far away [7].

2. Relay - Similar to the Replay; an attacker receives a RPI in one location and replays it using a different device in another location [30]. The authors in [16, 25, 39] present the attack exploring the fact that EN protocol do not register each encounter's location data. In [17] the authors propose that this attack can be used for voter suppression during the pandemic of COVID-19. [5] implemented this attack using devices such as Raspberry Pis to relay RPIs across a city. [15] suggests that the mitigation strategies for this type of attack may introduce significant complexity to the system. Similarly to the Replay attack, [30] proposes a new protocol to mitigate this vulnerability.

3. Re-identification - An attacker collects a target's RPIs and links them to diagnosis keys published by the Health Authority [22].

    3.1. Paparazzi - Similar to the Singling-out, an attacker, using passive BLE devices, captures the RPIs from a specific user and checks them later for positive matches [39]. In [4, 7], the authors analyze this attack in the context of a massive surveillance system and added two attacks, Orwell and Matrix. These attacks only differ from the Paparazzi in the attacker's capabilities and, as such, they are not considered as different attacks from the Paparazzi.

    3.2. Singling-out - An attacker records only 1 RPI, and can then check if a given user is infected when the health authority publishes new TEKs [25]. In [27], the authors executed a similar attack named Contact Isolation. They managed to link a RPI to a device. This work also proposes other advanced techniques to link the RPI to a device, such as using Wi-Fi data or the smartphone's camera. This attack is also mentioned in [7] as the Gossip attack.

4. False Report - The attackers can falsely report their own keys as infected and generate alerts in other users [39]. This attack is also mentioned in [22] as Server Data Pollution.

    4.1. Impersonation - A malicious user tests positive and reports another user's reporting data [10].

    4.2. Terrorist Report - In [3], authors disclose a blockchain black market solution that allows safely selling and buying infected keys. This attack is mentioned in [6] as Tracing Forgery.

5. Bluetooth-based tracking - According to [22], switching on Bluetooth leaves the user at risk of any risk already present in the Bluetooth stack.

    5.1. Bluetooth Beacon - Attacker uses a Bluetooth beacon to collect information about active Bluetooth devices. [39].

    5.2. Eavesdropping - Attacker captures RPIs with sniffers to build movement profiles for the users [16, 28].

6. Compromised Mobile Device - EN stores information on the device that can be used to determine the infection status of a user [22]. An attacker can potentially verify if the user has uploaded any TEKs. SDKs can be used to massively deploy attacks in mobile devices [11]. In [12, 17] it is detailed that an attacker can covertly use a malicious SDK to capture RPIs and store their geolocation, or start replaying them with other devices using the same SDK.

    6.1. Biosurveillance - Massive surveillance technique to harvest information about users' infection status [11]. At a large scale, the attacker can gather information in a region and what users might be infected.

7. Linking - An attacker associates two transmitted RPIs to the same device. The author in [25] 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.

    7.1. Address Carryover - Bluetooth MAC rotation and RPIs rotation are not aligned and an attacker links two anonymous identifiers to the same device [10]. This is also referred to as Tracking COVID-positive using the TEK in [22].

    7.2. De-anonymization - Authors in [11] suggest using SDKs to store received RPIs and more information such as GPS location to link a device to them.

8. Backend Impersonation - Attacker impersonates the backend server and sends specific a TEK as an infected key to the target [39]. This attack is also referred to as Matteotti in [4, 7].

9. Diagnosis Server Compromise - An attacker with access to the diagnosis server can potentially de-anonymize users [22].

    9.1. Bombolo - The attacker tries to use the data uploaded to the Health Authority to analyze personal contacts between users [4].

    9.2. Brutus - The attacker, controlling the server and the Health Authority, tries to map the real identity of an infected user to her uploaded data using the authentication mechanism [4, 7].

10. Profiling - An attacker captures a user's RPIs and tracks them [39]. Authors in [5] 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 can then trace the movements of a specific user.

    10.1. Nerd - A malicious App that collects specific information (such as GPS location, timestamps, or Wi-Fi networks) for each encounter simultaneously as the EN [39]. All the data can be inserted in a database and be further used to identify reported cases.

11. Storage Drainage - An attacker sends valid RPIs to the target and occupies storage space on the device [8, 25, 28].

12. Battery Exhaustion - An attacker sends large quantities of RPIs to a target exhausting its battery. This DoS-based attack is mentioned in [25, 28], 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. Authors in [25] propose to use a proof-of-work to check if a received message is valid when under a high request load.

13. Coercion Threat - Since EN stores collected data on the device, the attacker can force the targets to reveal their infection status [16, 39].

    13.1. Militia - A community of people with access to data collected by other attacks such as the Nerd attack can use that information to threaten infected people and isolate them [39].

14. Jamming - This attack consists in affecting the Bluetooth data transmission between two devices, through a device that is constantly transmitting thousands of invalid RPIs. Jamming is an inherent risk to systems reliant on Bluetooth [30]. In [26], 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.

15. Network Traffic Analysis - Attacker infers the target's infection status by analyzing

the network traffic and communications with the Health Authority's server [22].

16. Time Machine - EN uses the device's time to validate if a received RPI is expired or not. If an attacker changes the time, EN will accept an already expired RPI and generate exposure alerts if it is confirmed as infected later [6].

    16.1. Set Clock Manually - Attacker manually sets the time in the target's device to the acceptance window of the RPIs [29].

    16.2. Master of Time - 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 [29].

    16.3. Rogue NTP Server - If a device is connected to the Internet, it may use a Network Time Protocol (NTP) server to synchronize the time settings. An attacker can use a rogue NTP server to change the target's device time [29].

17. Forensics and Physical Access to Devices - Since EN stores information on the device, an attacker with physical access can extract the list of received RPIs or TEKs.

18. Linking Diagnosis Key - An attacker analyses Diagnosis Keys published by the Health Authority and links them to the same individual. [22] states that this attack is feasible only when there are very few people with a positive diagnostic. They suggest padding the published keys with random keys to mitigate this risk.

19. Occasional Disclosure - A user that receives an exposure alert may recall which encounter may have lead to this potential infection [39].

20. Passive Disruption - Passive disruption attack can be performed by disabling the resources necessary for the EN to function. [30]. In the case of Android OS, disabling the GPS stops all the EN-based Apps.

21. 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 [39]. When one of the users reports an alert, the attacker waits to check 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.

22. Simulated EN - Malicious App that operates similarly to EN. However, as explained in [29], 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.

23. Troll - An infected attacker attaches his device to a carrier to spread his RPIs around unsuspecting users [25]. When his TEKs are published by the Health Authority, multiple exposure alerts will be triggered.

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. Fig. 3.2 presents the number of sources that the Main Attacks have. Replay and Relay, with 17 and 11 sources respectively, are the most referenced attacks. Meanwhile, 7 attacks have a single source and 6 have 2 sources.

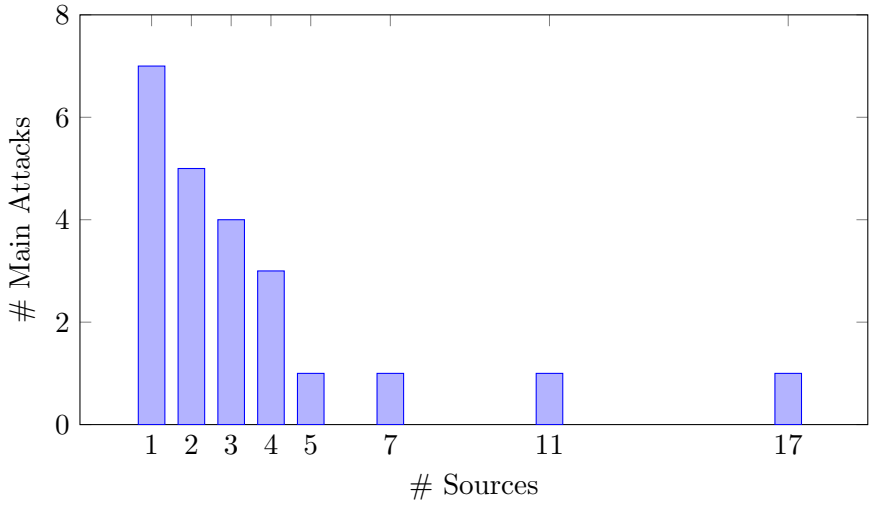


Figure 3.2: Total of sources per Main Attack.

3.2 State-of-Art on EN Attacks Categorization

The state-of-art on existing taxonomies related to EN attacks was obtained using the following sources of information:

1. Review on Google and Apple's public and official documentation regarding attacks on EN system;

2. Survey on taxonomies and groups of attacks on previous attacks based on the EN attacks review in Section 3.1;

3. Review of existing EN and decentralized systems attacks taxonomies: papers were collected from a search using Google Scholar search engine with the following query:

    - ("GAEN" OR "Google/Apple Exposure Notification" OR "Exposure Notification") AND "taxonomy" AND "attack".

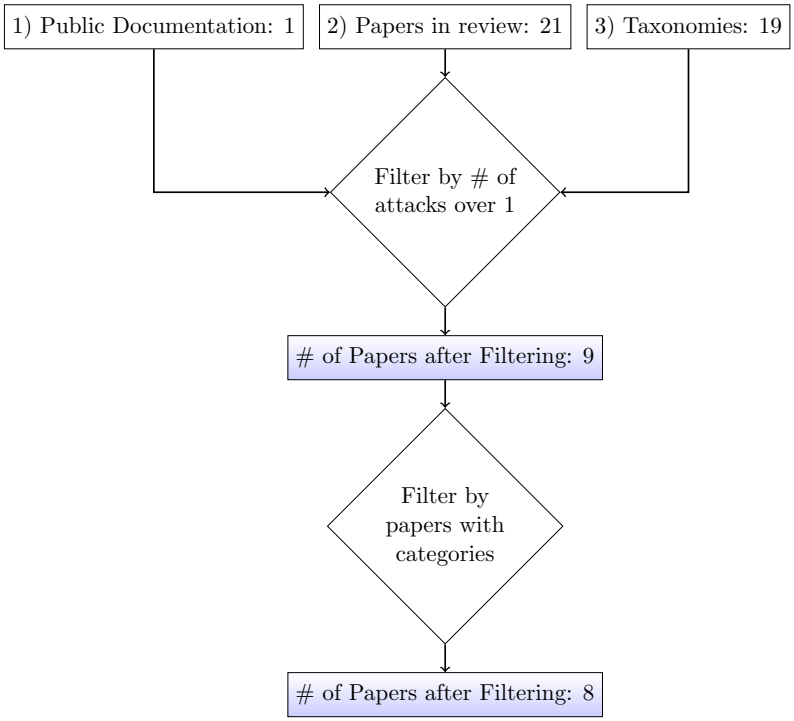


Figure 3.3: Stages and numbers of the selection process.

From the analysis of the sources above, the source (1) returned 1 public and official document focusing attacks on the EN, the source (2) returned 21 papers, and the source (3) returned 19 papers. Then, these results were filtered to obtain the papers that had at least more than 1 attack described. After this stage, the total number of papers was 9. Then, a filter was applied in order to exclude papers with no characterizations or taxonomies. The final number of papers was 8.

In the public document [22], Google provides a list of security and privacy risks on EN and the measures taken in the design of EN to mitigate those risks. Fig. 3.4 presents a graphical representation of this list. These attacks are categorized according to their main objective: tracking, learning social interactions, learning COVID-positive status, or disruption of the system.

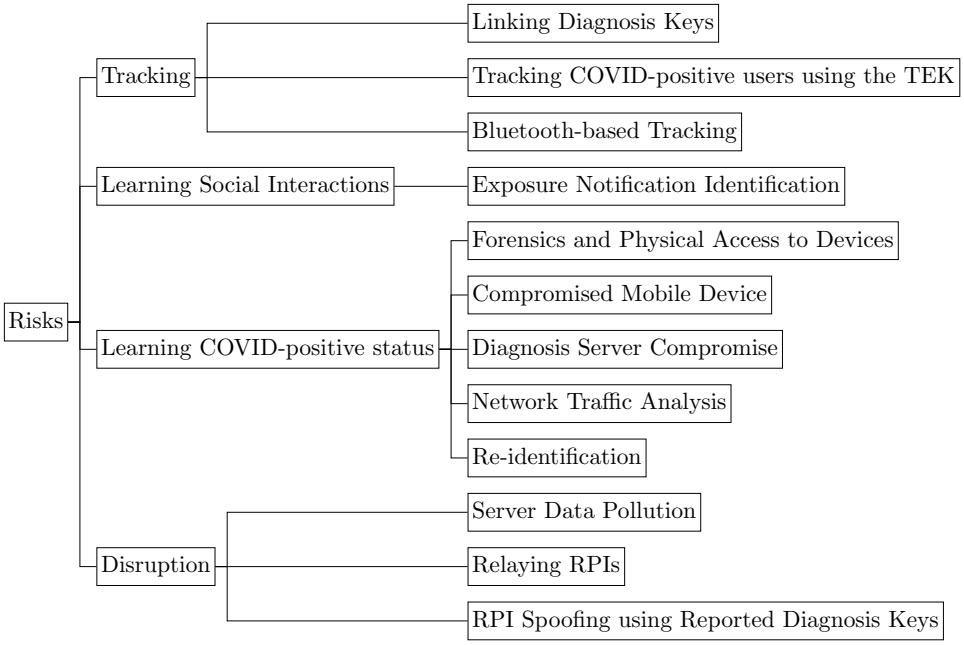


Figure 3.4: EN risks identified in [22].

The Tracking category includes attacks that use features of the EN system to track a user infection status like the Bluetooth Tracking attack or tracking users by TEKs. The Learning social interactions category includes the attacks where an attacker tries to infer if an user has met with the infected person by analyzing if the exposure warning is raised or not. The Learning Covid-positive status category involves a more direct approach to ascertain the infection status of a user. Either by compromising the diagnosis server, the mobile device or performing a forensics analysis on the device, the attacker can validate the status of the user. In the other attacks, network traffic analysis and re-identification, the attacker has to analyze the HTTPS communications between the user and servers or analyze the RPIs and compare them with the infected ones. The Disruption category involves actively injecting false RPIs and compromising the integrity of the data stored in the devices. These RPIs can be spoofed from a diagnosis key reported, relayed to another location as the relay attack in [25].

In [39], the author introduces a large number of attacks and categorizations. This paper is one of the first presenting vulnerabilities of the EN system. Fig. 3.5 is a representation of these categorizations. The author groups the attacks into two main categories: False Alert Injection and Tracking. There is a subcategory in Tracking for Deanonymizing Known Reported Users. The Tracking attacks involve obtaining information about a user and his infection status. The attacks range from using Bluetooth beacons to capture RPIs to coercion and physical threats. A subset of attacks for deanonymization use techniques to capture RPIs of a specific user and then compare them with the ones released as infected. False Alert Injection attacks involve sending either false or stolen RPIs and TEKs to other users and cause an exposure alert. This can be done with either Replay and Relay attacks or even falsely reporting a TEK as infected. This categorization provides a one level categorization for the attacks with the false alert injection and tracking categories. Given the characteristics of the analyzed attacks, this categorization could have more granularity in their differentiation.

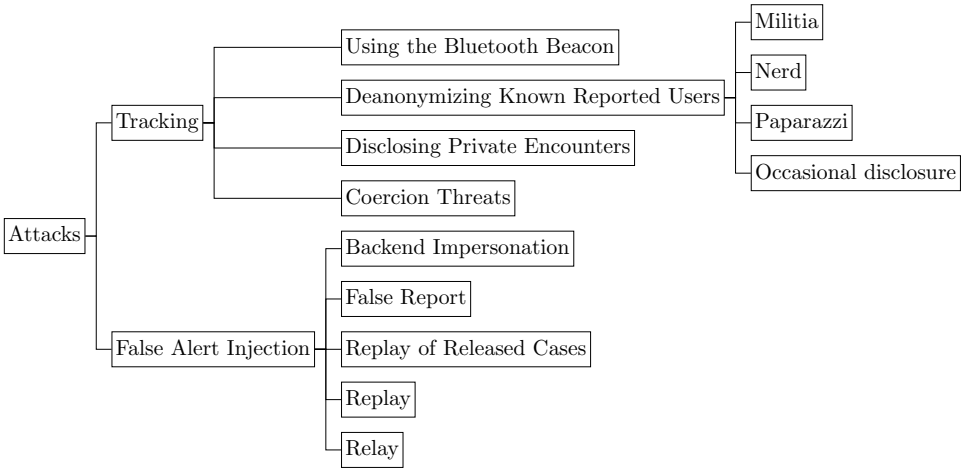


Figure 3.5: Attacks identified in [39].

In [16], the authors present a privacy threat model that includes multiple attacks on the EN system. This model associates specific goals to each of the identified attacks, so this taxonomy in Fig. 3.6 is a graphical representation of the attacks and the attacker's goals. The attacks are similar to the ones disclosed in [22] and [39]. Eavesdropping involves capturing the RPIs sent by EN. The data disclosure, coercion, and spoofing, tracking and replay have already been explained previously. The attacks were previously analyzed in Section 3.1. This taxonomy differs from the others presented in this state-of-art review

since it focuses on the goals of the attack. It presents a multiple scenarios and variants for each attack and it provides one level of categorization.

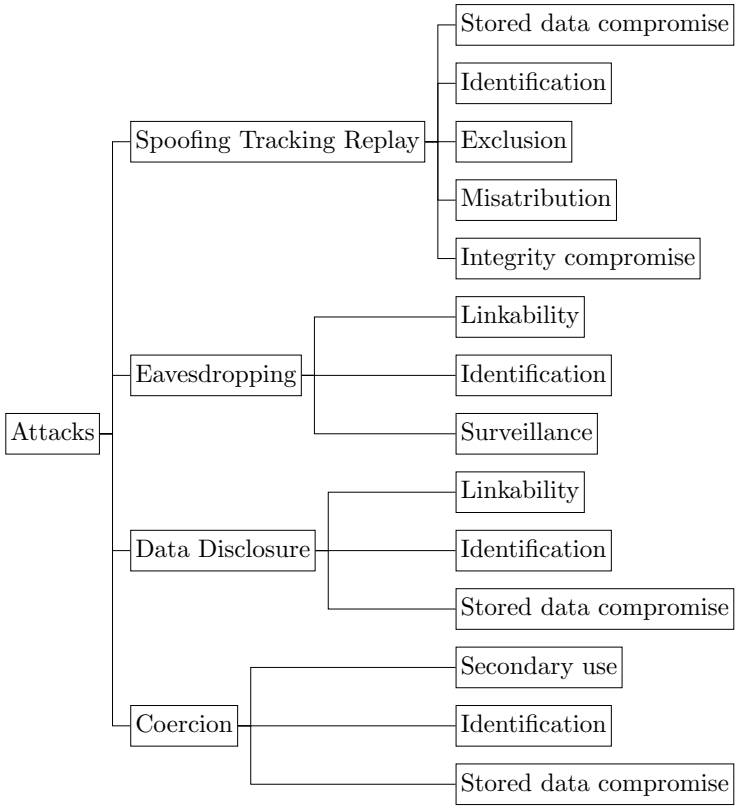


Figure 3.6: Attacks identified in [16].

In [4], authors present an alternative decentralized contact tracing system that they consider better prepared for Mass Surveillance attacks. Multiple attacks, including the ones in [39], are presented and the authors categorize them as Mass Surveillance or as Other. The attacks presented as other are either variants of the Replay attack in [39] or examples of attacks such as the Singling-out [39]. The Mass Surveillance attacks are variants of compromised scenarios, such as compromising the backend, the EN App or the health authority. It focuses in checking the infection status of the users or tracking their contacts with other people. Fig. 3.7 is a graphical representation of this classification. This categorization focuses particularly on the Mass Surveillance type attacks and is not a general classification. Other attacks can also be added and more classification levels can also be included.

In [7], the authors present an alternative to contact tracing system to EN and compare the vulnerabilities on the exchange of ephemeral identifiers that are present in the de-

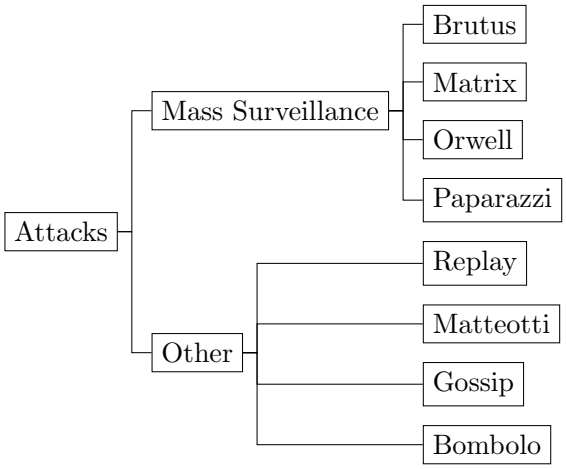


Figure 3.7: Attacks identified in [4].

centralized system with their own. The paper mentions linkability and low-cost as terms related to the attacks, and despite not presenting a specific category, we consider that the paper's attacks are classified as a linking related attack type. The categorization is presented in Fig. 3.8. This classification is focused only on attacks that target the exchange of identifiers. Since it does not mention other existing attacks at the time identified in [39], it is not taken as a general taxonomy for EN attacks. All the mentioned attacks are related with linkability and each explores different parts of the EN system with scenarios such as collusion between attackers and the Health Authority.

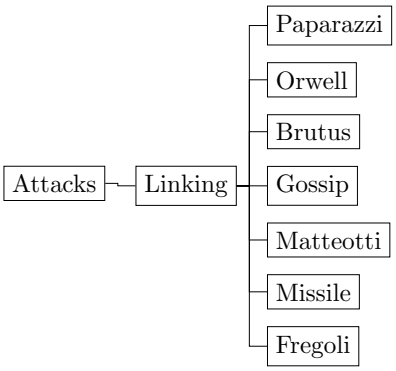


Figure 3.8: Attacks identified in [7].

In [27], the authors present two attacks, Contact Pollution and Contact Isolation, respectively identifying them as Data Poisoning and Privacy type attacks. Fig. 3.9 represents the mentioned classification. This classification is provided only for two attacks discovered by the authors. The categories divide the attacks by how they affect the data integrity and

the privacy of the user. The attacks are similar to the Replay and Singling-out attacks.

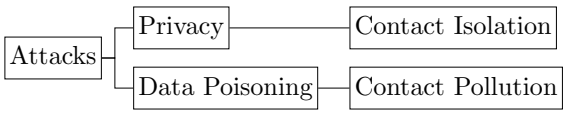


Figure 3.9: Attacks and categories extracted from [27].

In [5], the authors analyze 2 types of attacks: Security and Privacy. They also test those attacks in real scenarios. Fig. 3.10 depicts the classifications provided for the attacks. As the previous papers, this classification is only provided for the attacks tested or discovered by the authors. Both attacks can be included in the definition provided in [39].

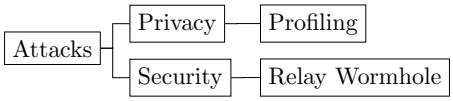


Figure 3.10: Attacks and categories extracted from [5].

The authors in [29] explore multiple attacks that falsely inject an identifier in the target. As in [39], this category is named False Alert Injection. Similar to previous papers, this is not a general taxonomy or classification for EN attacks. However, it provides information about different False Alert Injection attacks and how they are executed. In this paper, the attacks related to Time Travelling are introduced for the first time with which an attacker manipulates the time in a device so EN accepts an outdated key. This attacks are the Master of Time and Belated Replay. Another type of attack for injecting false alerts is using an App that simulates the EN's behaviour or using infected TEKs from other countries that are still valid in other countries.

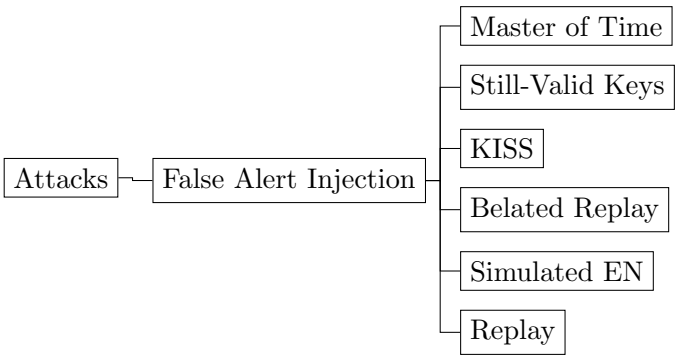


Figure 3.11: Attacks and categories extracted from [29].

Fig. 3.12 is the taxonomy presented in [9] for general attacks on contact tracing. The

authors divided the attacks into three main categories, distinguishing them by what technology they use: Bluetooth, GPS, or generic for both systems. The attacks on Bluetooth based Solutions Bluesnarfing and Bluebugging, involve establishing a connection with the target, which is impossible with the Bluetooth advertising feature. Bluejacking allows the attacker to send unsolicited messages to the target. Most Generic attacks have been explained in previous classifications and in Section 3.1, except for Resource Drain and Screen Lock. The first involves sending messages to waste resources such as the battery, and the latter is a ransomware attack that locks the target's screen when installing a fake contact tracing App.

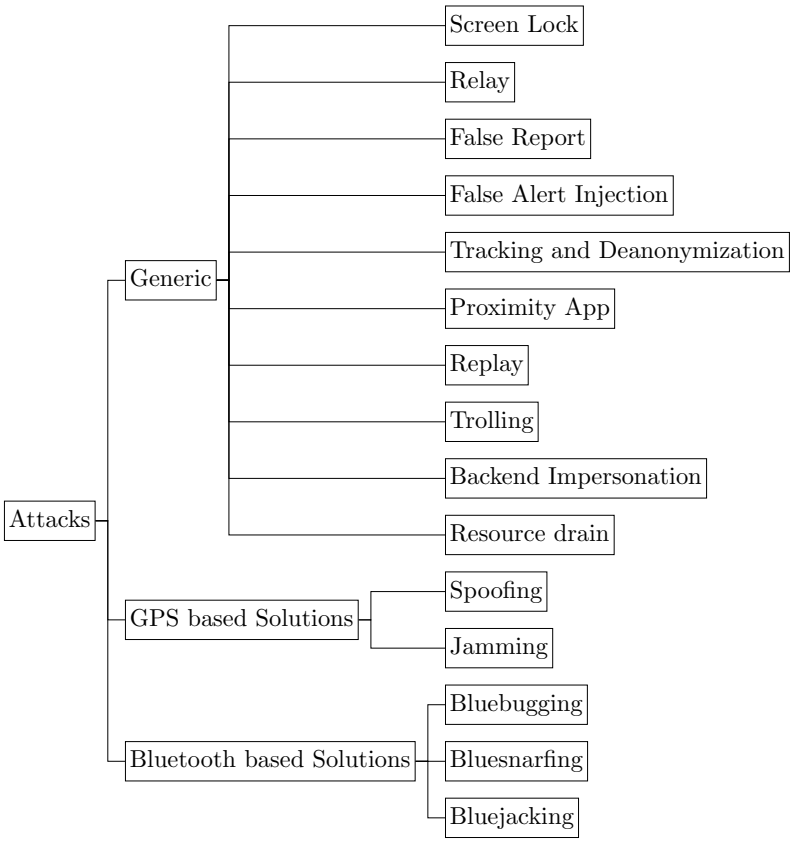


Figure 3.12: Contact tracing attacks taxonomy from [9].

Fig. 3.13 displays the selected papers grouped by the month of their publication date in a timeline starting from the launch of EN, 2020, to the present. From this information is is possible to check that these research works were proposed from April 2020 to March 2021, using categorizations to incorporate the disclosed EN attacks.

As a final remark regarding this state-of-art review, there is no general and updated

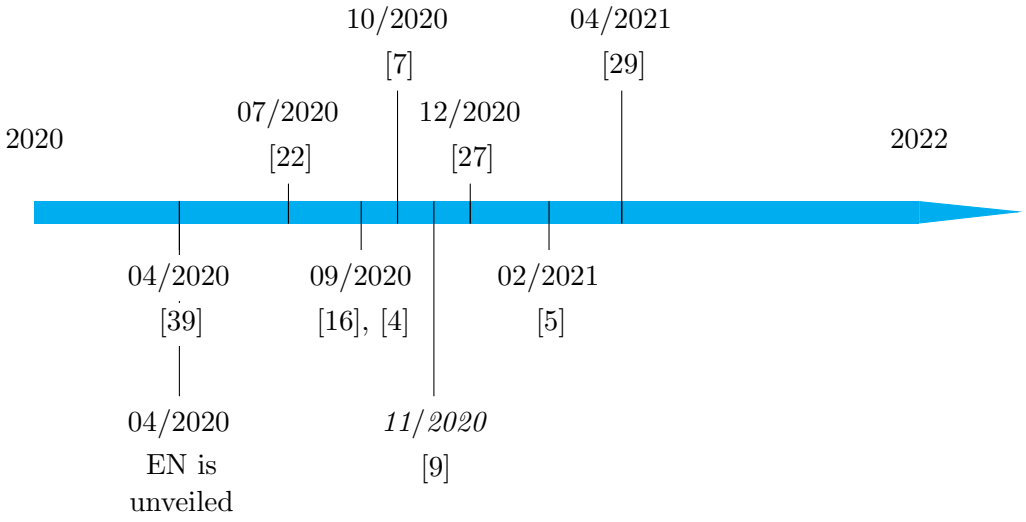


Figure 3.13: Timeline of the reviewed papers for the attacks on EN grouped by the month of their publication date.

taxonomy for EN-based attacks. Google's categorization concerns specific attacks that are common to decentralized contact tracing systems and if the EN is vulnerable to them or not. It does not include the latest known attacks. Likewise, the taxonomy found in [9] is not specific to the EN system, and its' objective is distinguishing attacks that target GPS, Bluetooth, or that are generic for both system types. The taxonomies analyzed in the papers mentioned in the systematic review of Section 3.1, are not general taxonomies, can be extended in their granularity, since they use only one level classification, and they do not include all the known attacks. The focus of the categorizations found is usually classifying a subset of attacks discovered or tested by the authors. Some of the classifications, such as the one in [39], have been adopted by more authors in recent papers.

Chapter 4

Advertising Overflow Attack

This chapter details the novel internal DoS-based attack - the Advertising Overflow attack. Fig.4.1 presents the normal operation scenario (on the left) and an attack scenario (on the right).

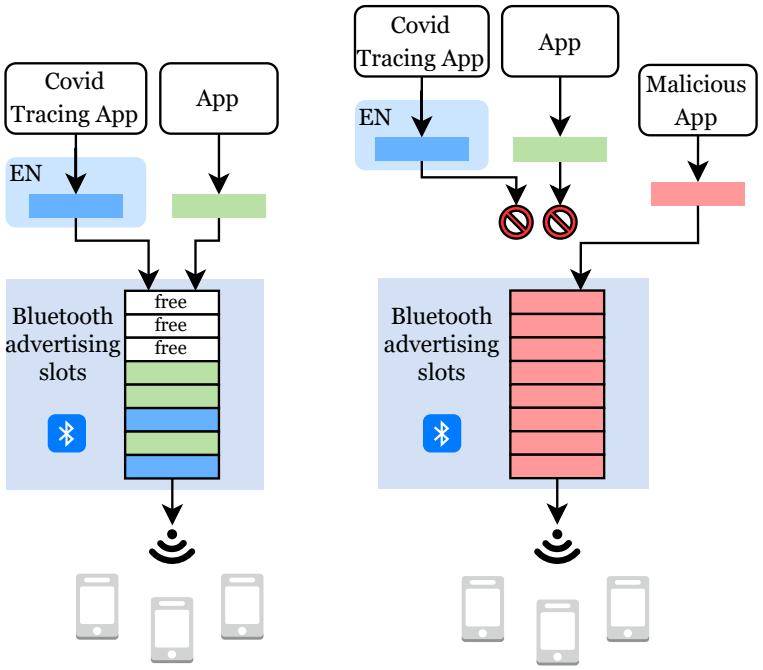


Figure 4.1: Normal operation scenario (left side) and attack scenario (right side).

In the normal operation scenario, each time EN or another App sends a Bluetooth advertisement, it requires the Android OS to place it in an available advertising slot. The advertising 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 [41]. 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. A higher priority means that this receiver will be executed by the OS before others capture the same event. However, an App (including EN) with a higher priority 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. 4.1, instead of having multiple Apps using advertisement slots, a malicious App occupies all the available slots in the device without ever releasing them, leaving no free slots for other apps to place their advertisements. Since the advertisement slots are shared between all the Apps in the device, an App can occupy all these slots when, for example, the device reboots. If an App occupies all the slots, according to the code publicly available in [24], 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 [18] 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.

This attack was first found during the tests for the Battery Exhaustion and Storage Drainage. An Android App was developed to use the Bluetooth Advertising feature like EN to measure the efficiency of these attacks. This malicious App's objective was to send thousands of RPIs in a short time interval and store the transmitted data for further analysis. However, during the tests, the App could never place more than 15 simultaneous Bluetooth advertisements. Upon further investigation of the App's logs, the error was found, and it was always displayed when there were more than 15 advertisements. The malicious App was installed on other Android devices, and it showed the same error at 15 placed advertisements.

After finding a maximum limit, it was analyzed if the malicious App kept the same number of advertisements during an extended period instead of a short interval. As a result, the execution period of the malicious App was extended to 1 hour, and the advertisements were never removed. This finding concluded that the advertisements could be kept longer than EN's rotation period for RPIs.

Then, the malicious App was designed to start advertising as soon as one of the triggering events of EN advertising happened, e.g. by device reboot or enabling Bluetooth interface. Since it is possible for any App to place a listener for those events, it was also possible for the malicious App's code to be executed before the EN's code if it had a higher priority. After researching the available listener priorities, the malicious App was modified to include a listener for the device's reboot and enabling Bluetooth. The App was deployed on a smartphone with Stayaway Covid installed, and then the device rebooted. After it turned on, the device's logs showed that the EN system was throwing errors because it could not advertise any RPIs.

After validating this new attack, research was conducted to validate if any attacks resembled the Advertising Overflow or if it was indeed a novel attack. As seen in the review in Section 3.1, the closest existing attacks are the Battery Exhaustion and Storage Drainage, both DoS attacks on the EN system. However, both attacks do not block the EN's advertisements but focus instead on depleting the internal device's resources such as the battery or storage. In addition, they need an external device to transmit the data, while Advertising Overflow requires an App on the same smartphone.

Chapter 5

Impact Assessment of DoS-based Attacks

This chapter presents the impact assessment of a subset of DoS-based attacks, namely, Battery Exhaustion, Storage Drain, and the Advertising Overflow attack.

5.1 Testbed

The assessment on each attack was performed by using eight smartphones of three different models as presented in Table 5.1. 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 have Bluetooth version 5.

A testbed environment was set up according to Fig. 5.1. The attackers have a Malicious App installed, which is an Android App developed to send simultaneous advertisements (15 advertisements is the maximum on all devices tested), each one containing an RPI.

Table 5.1: 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

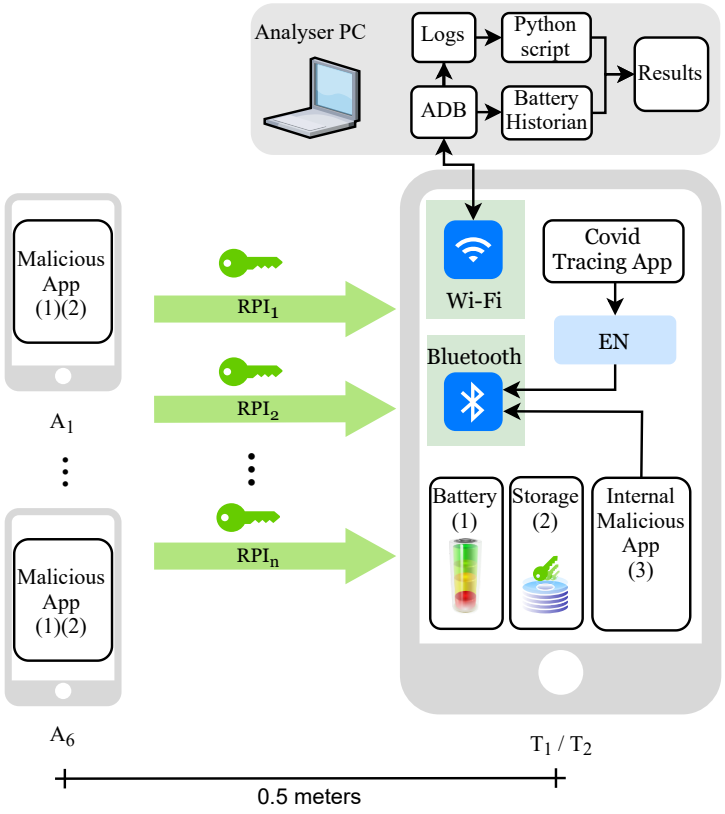


Figure 5.1: Testbed Environment.

The generation of RPIs and advertising settings follows the publicly available source code for EN [24]. In both target devices the a Contact Tracing App (Stayaway Covid) 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) [20] and Battery Historian [35]. 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 wiped, the logs collection over ADB reset, and the battery collection reset.

5.2 Battery Exhaustion

According to [25], an attacker can advertise either valid or invalid RPI to a target device. In turn, the device will wakeup and process the received RPI. These actions require energy and discharge the battery. The attack is depicted in Fig. 5.2.

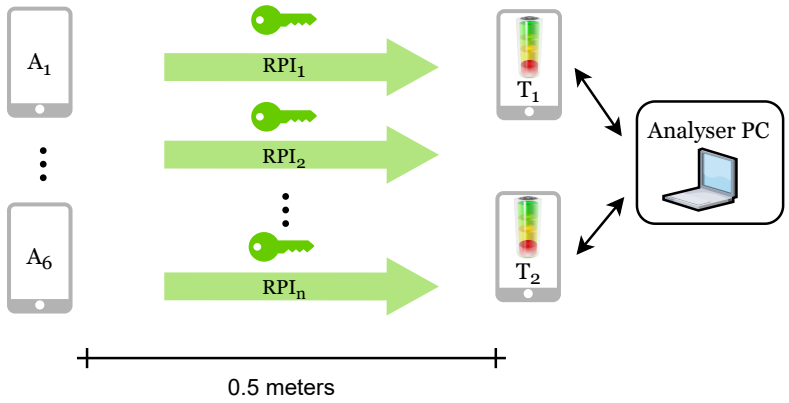


Figure 5.2: Battery Exhaustion attack in which attackers send RPIs to targets to and wastes their battery.

Android OS Bluetooth advertisements can be configured according to parameters defined in [19]. These configurations include the advertising interval ("INTERVAL" config-

urations) and the range of the Bluetooth transmissions ("TX_POWER" configurations). By default, EN uses *"INTERVAL_MEDIUM"* (one advertisement around every 250ms) and *"TX_POWER_LOW"* power level. However, for the attackers, the Android advertising procedure was setup 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 were performed for each target, where each test has a duration of 1 hour. Before executing each test, the Bluetooth setting was disabled and enabled again to trigger the first EN scan. After an hour has passed, 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 5.2. Average value, its standard deviation (σ), and the ratio between Attack (A) and Baseline (B), was calculated. Each test result was plotted in Fig. 5.3.

Table 5.2: 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**	

The results show that the average battery discharge for T_1 resulted in 0.918% for the Baseline, and in 1.796% for the attack. For T_2, the average battery discharge was 0% for the Baseline and 0.918% for the Attack. The Baseline 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 operation of this device. We assume that the differences of the baseline 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 ratio for T_1 was 1.956 and for T_2 could not be calculated 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.

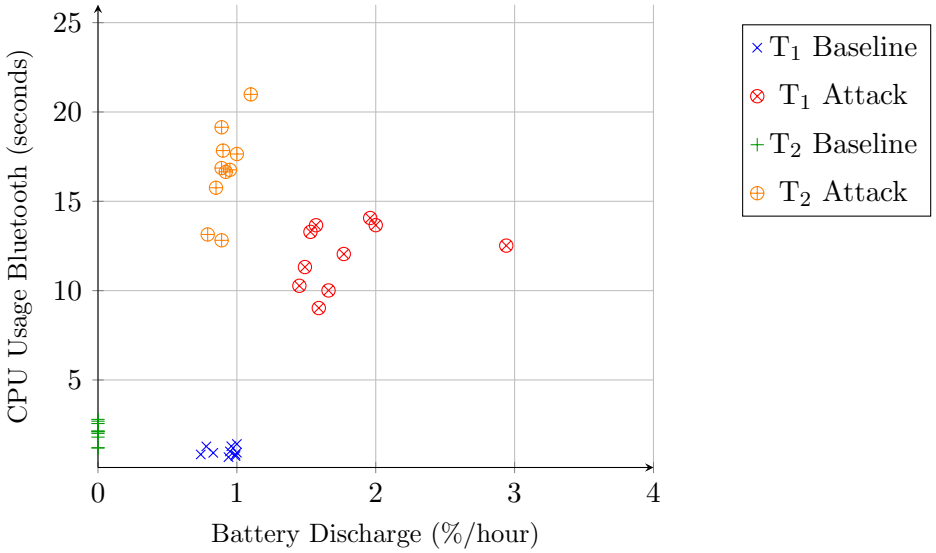


Figure 5.3: Battery consumption and Bluetooth CPU usage for T_1 and T_2.

5.3 Storage Drain

According to [25], 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. Fig. 5.4 depicts the storage drain attack.

The T_1 and T_2 devices are in a normal operation mode and transmitting one RPI at a time. However, $A_{1\text{-}6}$ sends valid RPIs in the order of thousands that will be processed and stored by the targets.

A round of 10 tests was done 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

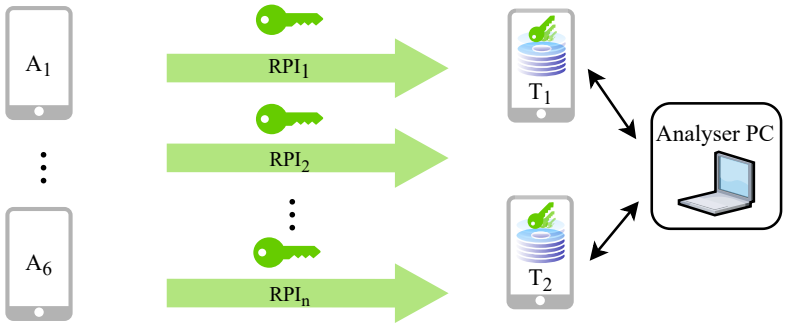


Figure 5.4: Storage drain attack in which attackers send RPIs to targets to occupy their storage space.

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 "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 5.3. Average value, its standard deviation (σ), and the ratio between Attack (A) and Baseline (B), was calculated. Each test result was plotted in Fig. 5.5.

Table 5.3: 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**	

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 between 200 and 300 RPIs. In contrast, the attack scenario has 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.

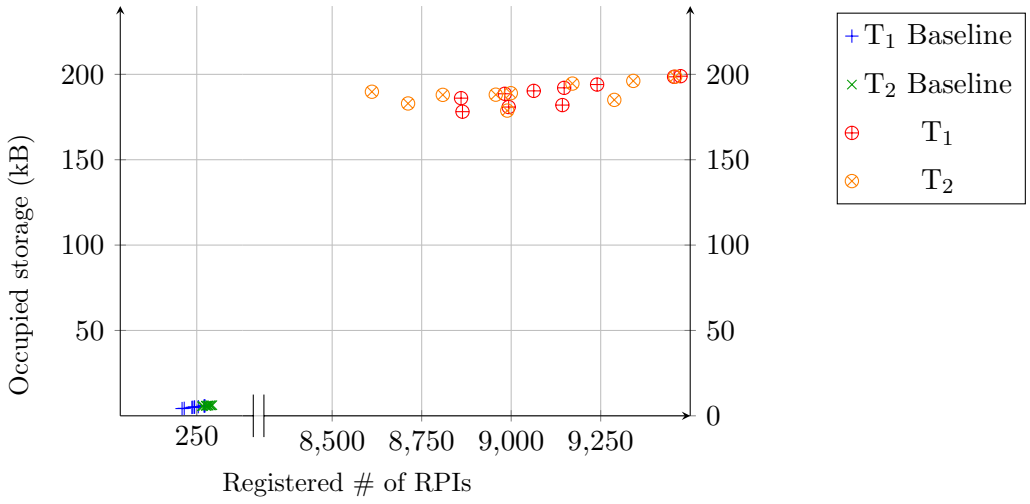


Figure 5.5: Storage drainage statistics for both T_1 and T_2.

5.4 Advertising Overflow

The advertising overflow attack, when implemented correctly, can compromise the advertising feature of the EN system since it occupies all the advertising slots. This attack was presented in Chapter 4 and depicted in Fig. 4.1.

A test with a duration of 1 hour was performed for the two targets and two scenarios, baseline and attack. In "Baseline" scenario, EN is active in $T_{1\text{-}2}$ in a normal operation mode. In the "Attack" scenario, $T_{1\text{-}2}$ ran the malicious App and EN at the same time.

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. 5.6.

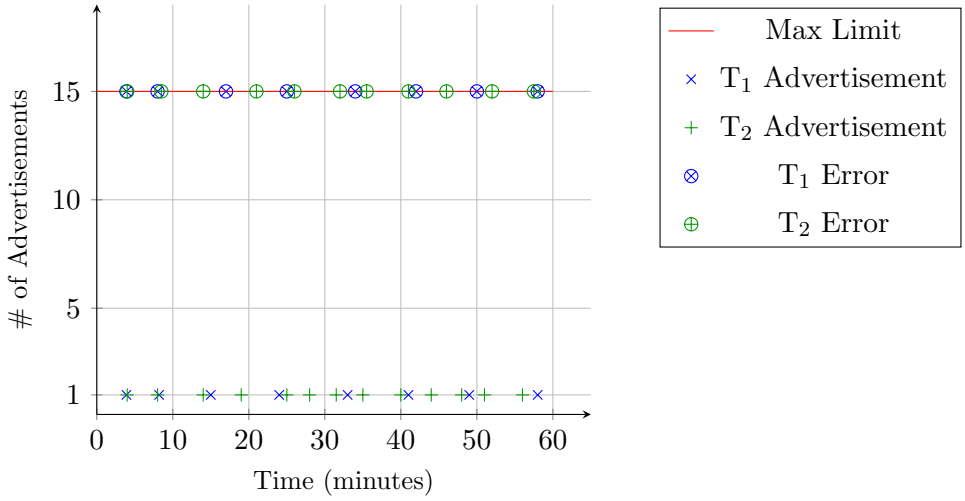


Figure 5.6: Advertisements placed by EN during the baseline and the attack tests.

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.

5.5 Discussion

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

Battery Exhaustion attack presented 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 very noticeable battery drain, especially in older devices.

The Storage Drainage attack presented 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 requires other devices 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 effectively blocked advertisements from EN during the entire test and neither the Android OS or 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. This attack blocks the operation of any EN-based Apps such as the ones presented in the Table 2, used by over 39 million European users.

The Advertising Overflow attack was reported to Google Security Team in April 19th, 2020. This disclosure lead to Google's confirmation of the bug and they rewarded the authors with the inclusion in the Google Application Security honorable mentions board.

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, EN fails silently, and the user is not aware of the failure.

The Advertising Overflow instead of alarming the users about EN usage, it silently affects the efficiency of digital contact tracing itself and renders the process of uploading the TEKs useless, since no other user will have received the corresponding RPIs. Deploying the attack at small or larger scale will always severely impact the efficiency of a digital contact tracing solution based on EN.

Chapter 6

The Proposed Taxonomy

The taxonomies analysis in Section 3.2 shows that there is no up-to-date taxonomies for EN attacks. Besides the lack of recent attacks, the categorizations have also focused in attacks discovered in the same paper they were presented in, and they usually have a single level classification. Therefore, we propose a new taxonomy that includes all the attacks and classifies them with a multi-level approach.

The proposed tree-like taxonomy divides all the identified attacks into three main categories: Denial of Service, False Alert Injection, and Information Disclosure. In [12, 25, 39], authors have commonly identified novel attacks and used these categories to label them. This proposal expands these known categorizations and adds more granularity and levels instead of the single level classification currently used. As such, the first level of classification is these 3 known attack classes from which more granular classifications expand.

The second level of the taxonomy tree has 2 subcategories for each main category. This level focuses on analyzing the outcome or target of every attack and classifies them by their likely outcome. The definition of each subcategory takes into account the attack's execution method and the outcome. Each of the new subcategories is painted with the green color in the figures.

The attacks are divided into Main Attacks and Sub-attacks according to the review in Section 3.1. Main Attacks are in the third level and Sub-attacks in the fourth level connected to their Main Attack.

The proposed taxonomy is designed to incorporate novel attacks. Recently, researchers

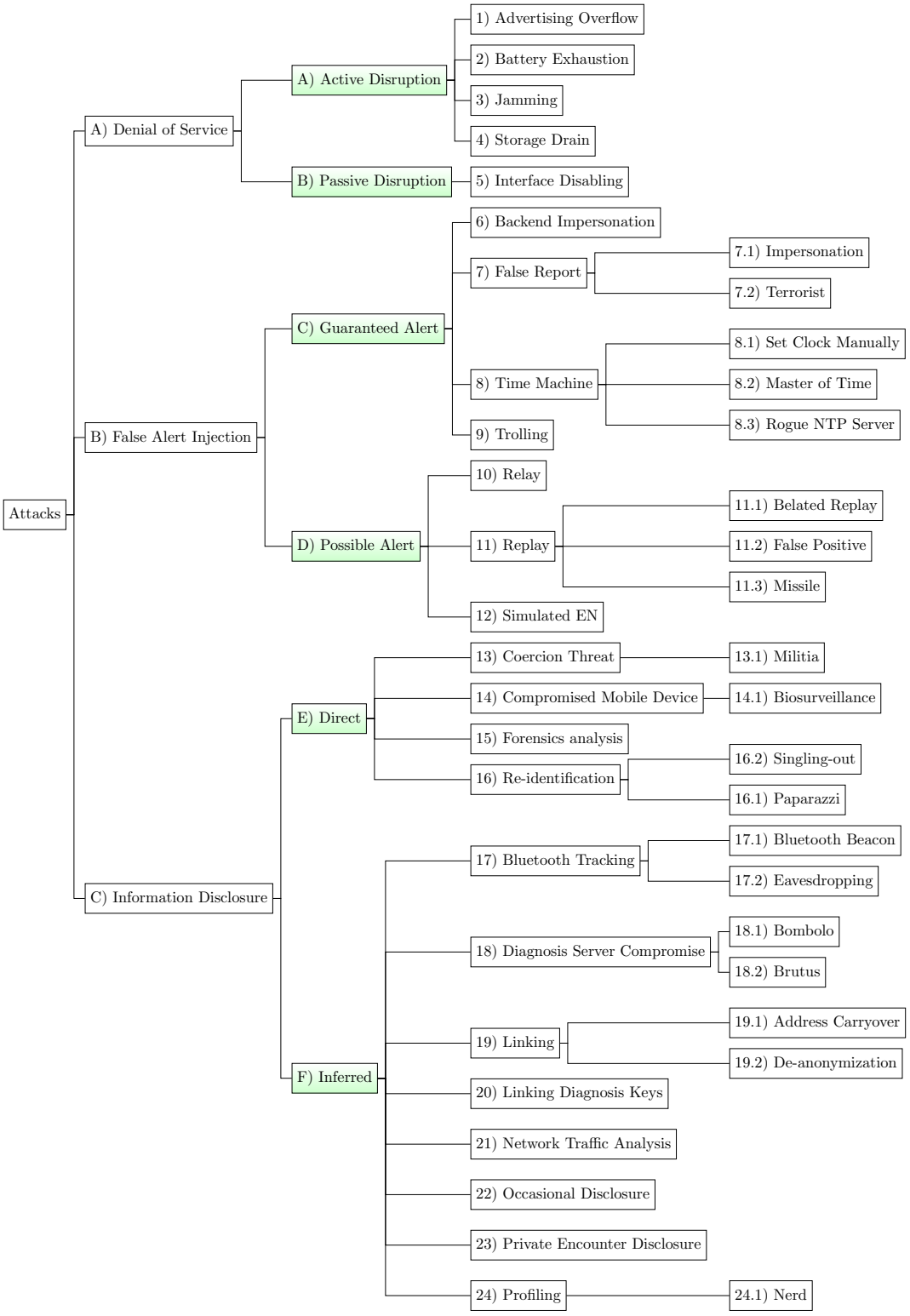


Figure 6.1: Proposed taxonomy for EN attacks.

in [29] have uncovered a type of attack that despite belonging to the same class of the False Alert Injection, it does not have the same execution method or likely outcome as most of

the others. This taxonomy allows a differentiation of these attacks and does not group them under general categories. In this document, the Main Attacks and their Sub-attacks in the proposed taxonomy are referenced using unique cardinal numbers as codes. Fig. 6.1 presents the proposed taxonomy with all the attack groups and the classified attacks. Table 6.1 links the Main Attack numbers in the proposed taxonomy to the codes using during the review on existing EN attacks. Advertising Overflow (Main Attack 1) review code is not included because it was not part of the review.

Table 6.1: Mapping between taxonomy attack codes and the Main Attacks in Section 3.1

Taxonomy Code	Review Code
1	-
2	12
3	14
4	11
5	20
6	8
7	4
8	16
9	23
10	2
11	1
12	22
13	13
14	6
15	17
16	3
17	5
18	9
19	7
20	18
21	15
22	19
23	21
24	10

6.1 Denial of Service Attacks Group

A DoS attack is an attempt at thwarting the legitimate use of a service [32]. Regarding Bluetooth, this attack can stop communication between two devices by affecting the resources in one of them. The resources range from the Bluetooth chipset's capacity to

the device's battery and storage. These category's attacks are not limited to Jamming with a large volume of requests and include Apps that can disrupt the expected behaviour of the EN system.

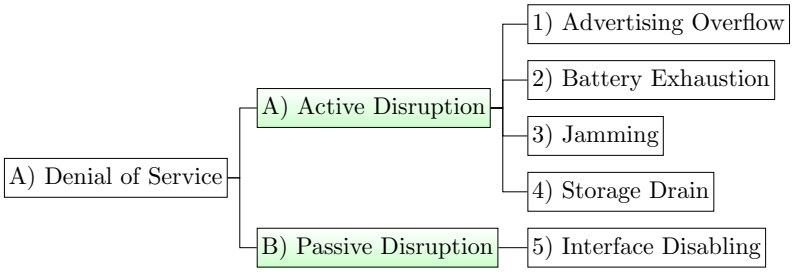


Figure 6.2: Denial of Service attacks group.

Two categories were proposed for this type of attacks: Passive Disruption and Active Disruption. A Passive Disruption attack thwarts the EN system by stopping OS features that are essential to its working, for example, Bluetooth and GPS in an Android device. By disabling Bluetooth, either manually or programmatically, the attacker is not using any EN features to attack the system, but it is only disabling a core feature with an unrelated measure. An Active Disruption attack affects the EN system by using features of the EN such as flooding an area with Bluetooth messages with the UUID of EN or discharging a device's battery by sending fake EN Bluetooth messages. Each attack focuses on a device's resource, such as the battery, storage or Bluetooth interface.

The attacks categorized as active disruption are Jamming, Advertising Overflow, Requests Overflow, Battery Exhaustion and Storage Drainage. Last, Interface Disabling is a Passive Disruption technique that involves disabling the Bluetooth feature in the device, for example.

Fig. 6.2 presents the Denial of Service attacks group in the proposed taxonomy.

6.2 False Alert Injection Attacks Group

False alert injections raise alerts of infection in an unsuspecting target that was not exposed to an infected person. Classified by [39], having a false infection contact may lead to discrimination or social accountability. It can be used to affect targets like football players before a match. An attacker has to inject a specific RPI in the target's device to raise the false alert. The means of delivery and RPI generation vary from attack to attack.

This category of attacks is also described as fake contact events [39].

This type of attack can be a Guaranteed Alert or a Possible Alert. These subcategories separate the attacks by their efficacy in raising an exposure alert on the target. The Guaranteed Alert types will always raise an alert in the target's device. Meanwhile, the other category is not guaranteed to reach the target or raise an exposure alert.

Replay, Relay and Simulated EN attacks use randomly obtained RPIs that may or may not be later identified as infected, so they are of the possible alert type. On the other hand Backend Impersonation, Time Machine, False report and Trolling attack all use RPIs that are confirmed as infected. The Time Machine attack has also multiple variants that execute the spoofing by abusing the validity of the RPIs by deriving identifiers from released TEKs and changing the time on target's device. Fig. 6.3 shows the False Alert Injection attacks group of the proposed taxonomy.

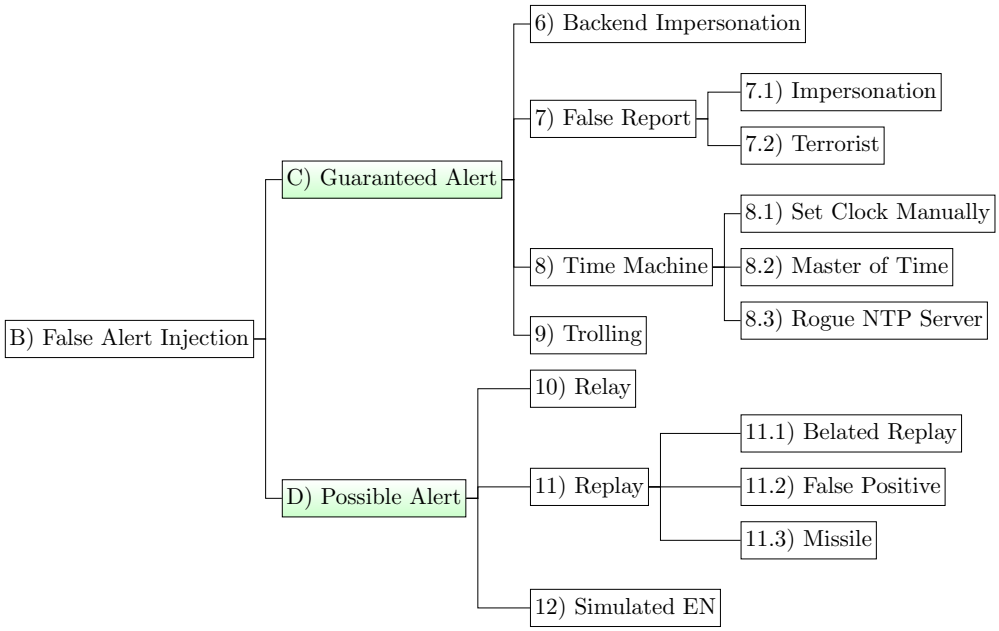


Figure 6.3: False alert injection attacks group.

6.3 Information Disclosure Attacks Group

Information Disclosure attacks compromise the anonymity assurance of the EN system and reveal if an infected RPI comes from a certain user or even if two users have been in close proximity or not. This category's attacks are divided by the reliability and method

of obtaining private data. Inferred type attacks use algorithms that analyze distance, GPS positioning, or even EN copycat Apps (mobile Apps mimicking EN) to accumulate information. On the other hand, Direct attacks obtain information directly from the target and do not need to calculate/process any data.

All the Inferred attacks resort to either creating a similar App as EN to obtain more information or using algorithms and profiling to analyze if two RPIs can be from the same user like the Profiling attack. To validate their proximity, Private Encounter Disclosure and Occasional Disclosure rely on checking if two users receive an exposure warning in a similar time interval. Other advanced methods involve Compromising the Diagnosis Server, Network Traffic Analysis between the target's device and the server, Bluetooth Tracking of the device and Linking either the Diagnosis Keys to a specific user or Linking RPIs to the same device. Direct attacks are less versatile as they usually target a single individual and can not be reused for more than one encounter. All the attacks obtain the information directly from the target's device by either capturing a single RPI or manually checking the infection status on the EN-based App. Examples of these attacks range from directly forcing the target to reveal his contact tracing App infection status (Coercion attack), forensic analysis to the mobile device, or Re-identification techniques such as the Singling-out or Paparazzi attacks.

Fig. 6.4 displays the Information Disclosure attacks group of the proposed taxonomy.

6.4 Discussion

The proposed taxonomy uses different categories that merges or arranges multiple categories in previous documents and focuses on distinguishing the attacks based on their type. A total of 43 Main Attacks and Sub-attacks were incorporated in the proposed taxonomy, including the most recent disclosed attacks. This taxonomy highlights possible attack vectors in the EN system, useful to enhance the defense strategies of the EN system. By analyzing how many attacks focus on a specific feature or vulnerability of EN, such as False Alert Injection or a DoS, the researchers community can focus on specific vulnerabilities that should be prioritized.

Table 6.2 groups the Main Attacks presented in the taxonomy by their publication date

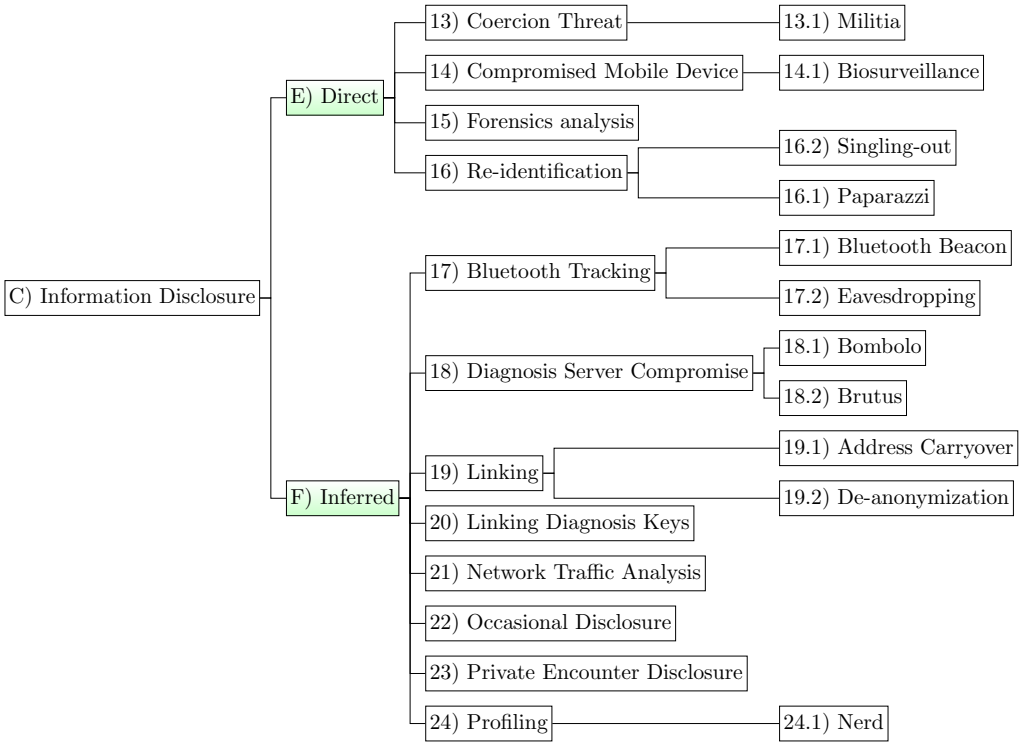


Figure 6.4: Information disclosure attacks group.

and Fig. 6.5 presents the evolution of the cumulative number of Main Attacks. From these results, in the initial month on April 2020 there was an atypical number of disclosed attacks since it matched the official release of the EN system, and Google's documentation [22] and Vaudenay [39] present a large number of Main Attacks for EN in Information Disclosure and False Alert Injection. After the first month, the uncovered Main Attacks never exceed the total of 3 in a single month. It is also relevant to notice that Denial of Service attacks started to be disclosed on June 2020.

Table 6.2: Taxonomy's Main Attacks grouped by the first time they were presented in a paper.

Date	Main Attacks
April, 2020	6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24
July, 2020	5
September, 2020	2, 4, 19
November, 2020	3, 8
July, 2021	1

Fig. 6.6 shows the attacks distribution by each main category (False Alert Injection, Information Disclosure and Denial of Service). From this data the Information Disclosure

attacks represent the majority of the attacks with 53% and the Denial of Service attacks represent 14% of the total attacks.

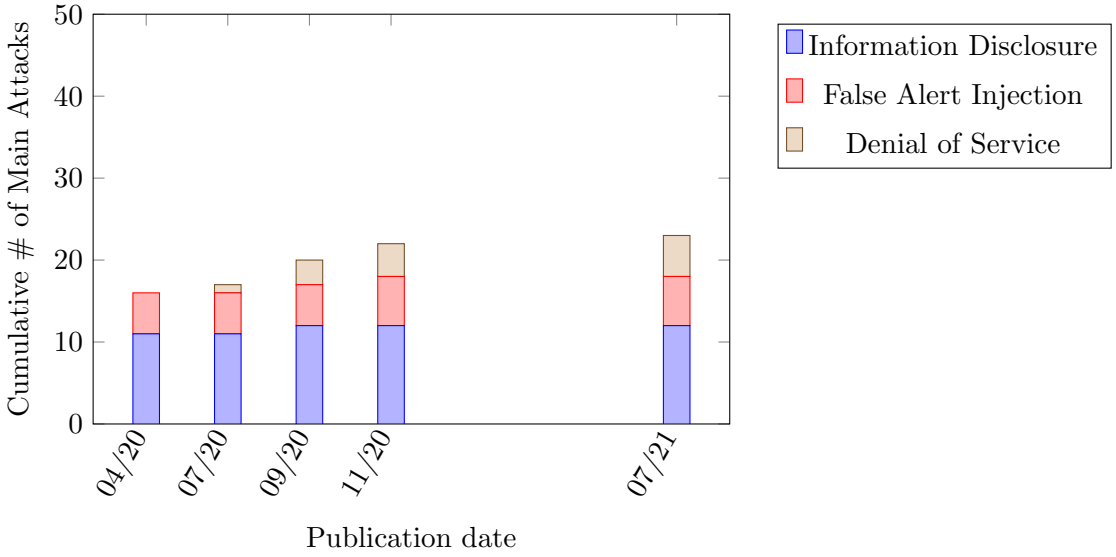


Figure 6.5: Evolution of the cumulative number of EN Main Attacks grouped by category.

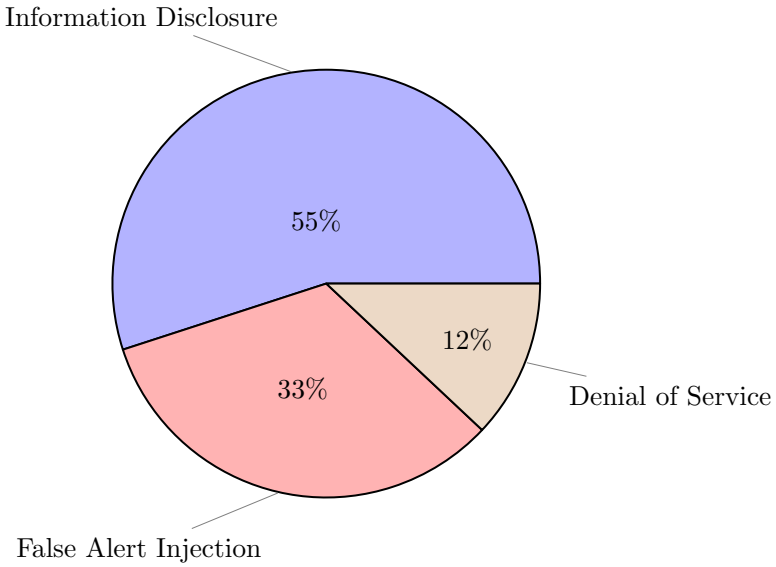


Figure 6.6: Distribution of Main Attacks by main category.

Chapter 7

Conclusions

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. The mobile Apps using the built-in sensor in the devices, Bluetooth and GPS, track the infection exposure of a user and can then warn him if he has been in at risk.

Despite the claims of concerns related to security and privacy, multiple attacks have been identified on the EN system and they target the integrity of the data and the privacy of an infected user.

A novel Advertising Overflow attack was also presented following the taxonomy proposal. Using a malicious App, an attacker can block the advertisements of EN by occupying the Bluetooth advertising slots of an Android OS device.

Advertising Overflow compromises the contact tracing expected behavior which has a severe impact if deployed in any scale. The attack can be conveyed by a Malicious or an SDK. Besides impacting EN, this attack will also affect any other Apps that use Bluetooth advertisements.

This and other two tested DoS attacks (Battery Exhaustion and Storage Drain) were tested on Android OS devices to verify 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 from EN. This attack is also not limited to a interval

of 1 hour, as it can disrupts the operation of EN for longer periods of time.

A new taxonomy for EN-based attacks is also presented. This taxonomy uses a granular and multi-level approach in a tree-like structure. Besides the granularity, all the known EN attacks have been included in this review. All the currently known attacks are separated according to three main categories, DoS, False Alert Injection, and Information Disclosure. This paper introduces 6 new subcategories for the attacks. With these new categories, the taxonomy allows a better understanding of EN's current attack vectors and highlights areas that can be further explored, analyzed and fixed.

The impact of Advertising Overflow attack can also be further tested on other BLE features on Apple's iOS. Regarding the taxonomy, it can also be further improved by testing the attacks in each OS and then divide them by the OS that they target.

References

[1] Samuel Altmann et al. "Acceptability of App-Based Contact Tracing for COVID-19: Cross-Country Survey Study". eng. In: *JMIR mHealth and uHealth* 8.8 (2020), e19857. ISSN: 2291-5222. DOI: 10.2196/19857.

[2] Google Inc. Apple Inc. *Exposure Notification Bluetooth Specification*. Apr. 2020. URL: https://blog.google/documents/70/Exposure%5C_Notification%5C_Bluetooth%5C_Specification%5C_v1.2.2.pdf (visited on 01/31/2021).

[3] Gennaro Avitabile, Daniele Friolo, and Ivan Visconti. *TEnK-U: Terrorist Attacks for Fake Exposure Notifications in Contact Tracing Systems*. Tech. rep. 1150. 2020. URL: http://eprint.iacr.org/2020/1150 (visited on 01/31/2021).

[4] Gennaro Avitabile et al. *Towards Defeating Mass Surveillance and SARS-CoV-2: The Pronto-C2 Fully Decentralized Automatic Contact Tracing System*. Tech. rep. 493. 2020. URL: http://eprint.iacr.org/2020/493 (visited on 02/24/2021).

[5] L. Baumgärtner et al. "Mind the GAP: Security Privacy Risks of Contact Tracing Apps". In: *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. ISSN: 2324-9013. Dec. 2020, pp. 458–467. DOI: 10.1109/TrustCom50675.2020.00069.

[6] Antoine Boutet et al. *Contact Tracing by Giant Data Collectors: Opening Pandora's Box of Threats to Privacy, Sovereignty and National Security*. University works. EPFL, Switzerland; Inria, France; JMU Würzburg, Germany; University of Salerno, Italy; base23, Geneva, Switzerland; Technical University of Darmstadt, Germany, Dec. 2020. URL: https://hal.inria.fr/hal-03116024.

[7] Francesco Buccafurri, Vincenzo De Angelis, and Cecilia Labrini. "A Privacy-Preserving Solution for Proximity Tracing Avoiding Identifier Exchanging". In: *arXiv:2005.10309 [cs]* (May 2020). arXiv: 2005.10309. URL: http://arxiv.org/abs/2005.10309 (visited on 02/24/2021).

[8] Bo-Rong Chen and Yih-Chun Hu. "Mitigating denial-of-service attacks on digital contact tracing". In: *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*. SenSys '20. New York, NY, USA: Association for Computing Machinery, Nov. 2020, pp. 770–771. ISBN: 978-1-4503-7590-0. DOI: 10.1145/3384419.3430599. URL: https://doi.org/10.1145/3384419.3430599 (visited on 02/24/2021).

[9] Aaqib Bashir Dar et al. "Applicability of mobile contact tracing in fighting pandemic (COVID-19): Issues, challenges and solutions". In: *Computer Science Review* 38 (2020), p. 100307. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2020.100307. URL: https://www.sciencedirect.com/science/article/pii/S157401372030407X.

[10] Ellie Daw. *Component-Based Comparison of Privacy-First Exposure Notification Protocols*. Cryptology ePrint Archive, Report 2020/586. https://ia.cr/2020/586. 2020.

[11] Paul-Olivier Dehaye and Joel Reardon. "Proximity Tracing in an Ecosystem of Surveillance Capitalism". In: *arXiv:2009.06077 [cs]* (Sept. 2020). arXiv: 2009.06077. DOI: 10.1145/3411497.3420219. URL: http://arxiv.org/abs/2009.06077 (visited on 01/22/2021).

[12] Paul-Olivier Dehaye and Joel Reardon. "SwissCovid: a critical analysis of risk assessment by Swiss authorities". In: *arXiv:2006.10719 [cs]* (June 2020). arXiv: 2006.10719. URL: http://arxiv.org/abs/2006.10719 (visited on 02/22/2021).

[13] Neeltje van Doremalen et al. "Aerosol and Surface Stability of SARS-CoV-2 as Compared with SARS-CoV-1". en. In: *New England Journal of Medicine* 382.16 (Apr. 2020), pp. 1564–1567. ISSN: 0028-4793, 1533-4406. DOI: 10.1056/NEJMc2004973. URL: http://www.nejm.org/doi/10.1056/NEJMc2004973 (visited on 10/19/2020).

[14] Henrique Faria, Sara Paiva, and Pedro Pinto. "An Advertising Overflow Attack Against Android Exposure Notification System Impacting COVID-19 Contact Tracing Applications". In: *IEEE Access* 9 (2021), pp. 103365–103375. DOI: 10.1109/ACCESS.2021.3099017.

[15] Stephen Farrell and Douglas J Leith. "A Coronavirus Contact Tracing App Replay Attack with Estimated Amplification Factors". en. In: (May 2020), p. 4. URL: https://down.dsg.cs.tcd.ie/tact/replay.pdf.

[16] Ananya Gangavarapu et al. "Target Privacy Threat Modeling for COVID-19 Exposure Notification Systems". In: *arXiv:2009.13300 [cs]* (Sept. 2020). arXiv: 2009.13300. URL: http://arxiv.org/abs/2009.13300 (visited on 03/04/2021).

[17] Rosario Gennaro, Adam Krellenstein, and James Krellenstein. "Exposure Notification System May Allow for Large-Scale Voter Suppression". In: Aug. 2020. URL: https://static1.squarespace.com/static/5e937afbfd7a75746167b39c/t/5f47a87e58d3de0db3da91b2/1598531714869/Exposure_Notification.pdf.

[18] Google. *Advertise Callback — Android Developers*. en. Feb. 2021. URL: https://developer.android.com/reference/android/bluetooth/le/AdvertiseCallback#ADVERTISE_FAILED_TOO_MANY_ADVERTISERS (visited on 02/24/2021).

[19] Google. *AdvertiseSettings*. en. Sept. 2020. URL: https://developer.android.com/reference/android/bluetooth/le/AdvertiseSettings (visited on 02/24/2021).

[20] Google. *Android Debug Bridge (ADB)*. en. URL: https://developer.android.com/studio/command-line/adb (visited on 03/15/2021).

[21] Google. *Bluetooth Low Energy — Android Open Source Project*. en. Sept. 2020. URL: https://source.android.com/devices/bluetooth/ble (visited on 02/28/2021).

[22] Google. *Exposure Notification: Risks and Mitigations FAQ*. en. Web site. (accessed on 1 February 2021). 2020. URL: https://github.com/google/exposure-notifications-internals/blob/main/en-risks-and-mitigations-faq.md (visited on 02/18/2021).

[23] Google. *Exposure Notifications API*. en. 2021. URL: https://developers.google.com/android/exposure-notifications/exposure-notifications-api (visited on 03/22/2021).

[24] Google. *Exposure Notifications API Internals*. en. Web site. (accessed on 1 February 2021). 2020. URL: https://github.com/google/exposure-notifications-internals (visited on 01/21/2021).

[25] Yaron Gvili. "Security Analysis of the COVID-19 Contact Tracing Specifications by Apple Inc. and Google Inc". In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 428.

[26] Hsu-Chun Hsiao et al. "An Empirical Evaluation of Bluetooth-based Decentralized Contact Tracing in Crowds". In: *CoRR* abs/2011.04322 (2020). arXiv: 2011.04322. URL: https://arxiv.org/abs/2011.04322.

[27] Jianwei Huang et al. "On the Privacy and Integrity Risks of Contact-Tracing Applications". In: *arXiv:2012.03283 [cs]* (Dec. 2020). arXiv: 2012.03283. URL: http://arxiv.org/abs/2012.03283 (visited on 02/25/2021).

[28] Molla Rashied Hussein et al. "Digital Surveillance Systems for Tracing COVID-19: Privacy and Security Challenges with Recommendations". In: *CoRR* abs/2007.13182 (2020). arXiv: 2007.13182. URL: https://arxiv.org/abs/2007.13182.

[29] Vincenzo Iovino, Serge Vaudenay, and Martin Vuagnoux. *On the Effectiveness of Time Travel to Inject COVID-19 Alerts*. Tech. rep. 1393. 2020. URL: http://eprint.iacr.org/2020/1393 (visited on 01/21/2021).

[30] Franck Legendre et al. "Contact Tracing: An Overview of Technologies and Cyber Risks". In: *arXiv:2007.02806 [cs]* (July 2020). arXiv: 2007.02806. URL: http://arxiv.org/abs/2007.02806 (visited on 01/31/2021).

[31] Mitch Leslie. "COVID-19 Fight Enlists Digital Technology: Contact Tracing Apps". en. In: *Engineering* (Sept. 2020), S2095809920302484. ISSN: 20958099. DOI: 10.1016/j.eng.2020.09.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S2095809920302484 (visited on 10/19/2020).

[32] Jelena Mirkovic and Peter Reiher. “A taxonomy of DDoS attack and DDoS defense mechanisms”. en. In: *ACM SIGCOMM Computer Communication Review* 34.2 (Apr. 2004), pp. 39–53. ISSN: 0146-4833. DOI: 10.1145/997150.997156. URL: https://dl.acm.org/doi/10.1145/997150.997156 (visited on 01/31/2021).

[33] *Mobile contact tracing apps in EU Member States*. June 2021. URL: https://ec.europa.eu/info/live-work-travel-eu/coronavirus-response/travel-during-coronavirus-pandemic/mobile-contact-tracing-apps-eu-member-states_en.

[34] Krzysztof Pietrzak. “Delayed Authentication: Preventing Replay and Relay Attacks in Private Contact Tracing”. en. In: *Progress in Cryptology – INDOCRYPT 2020*. Ed. by Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 3–15. ISBN: 978-3-030-65277-7. DOI: 10.1007/978-3-030-65277-7_1.

[35] *Profile battery usage with Batterystats and Battery Historian*. en. URL: https://developer.android.com/topic/performance/power/setup-battery-historian (visited on 01/20/2021).

[36] Joshua Siva, Jian Yang, and Christian Poellabauer. “Connection-less BLE Performance Evaluation on Smartphones”. In: *Procedia Computer Science* 155 (2019), pp. 51–58. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2019.08.011. URL: https://www.sciencedirect.com/science/article/pii/S1877050919309238.

[37] 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?” en. In: *Cardiovascular Digital Health Journal* (Oct. 2020), S2666693620300360. ISSN: 26666936. DOI: 10.1016/j.cvdhj.2020.09.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S2666693620300360 (visited on 10/19/2020).

[38] Ruoxi Sun et al. “Vetting Security and Privacy of Global COVID-19 Contact Tracing Applications”. en. In: *arXiv:2006.10933 [cs]* (July 2020). arXiv: 2006.10933. URL: http://arxiv.org/abs/2006.10933 (visited on 11/02/2020).

[39] Serge Vaudenay. *Analysis of DP3T - Between Scylla and Charybdis*. 2020. URL: https://eprint.iacr.org/2020/399.pdf (visited on 01/22/2021).

[40] Amanda M Wilson et al. *Quantifying SARS-CoV-2 infection risk within the Google/Apple exposure notification framework to inform quarantine recommendations*. en. preprint. Epidemiology, July 2020. DOI: 10.1101/2020.07.17.20156539. URL: http://medrxiv.org/lookup/doi/10.1101/2020.07.17.20156539 (visited on 10/19/2020).

[41] David Young. *BLE advertising fails*. June 2020. URL: https://stackoverflow.com/a/62267146 (visited on 02/03/2021).