

AirGuard - Protecting Android Users From Stalking Attacks By Apple Find My Devices

Alexander Heinrich
Secure Mobile Networking Lab
Department of Computer Science
TU Darmstadt, Germany
aheinrich@seemoo.de

Niklas Bittner
Secure Mobile Networking Lab
Department of Computer Science
TU Darmstadt, Germany
nbittner@seemoo.de

Matthias Hollick
Secure Mobile Networking Lab
Department of Computer Science
TU Darmstadt, Germany
mhollick@seemoo.de

ABSTRACT

Finder networks in general, and Apple's Find My network in particular, can pose a grave threat to users' privacy and even health if these networks are abused for stalking. Apple's release of the AirTag—a very affordable tracker covered by the nearly ubiquitous Find My network—amplified this issue. While Apple provides a stalking detection feature within its ecosystem, billions of Android users are still left in the dark. Apple recently released the Android app "Tracker Detect," which does not deliver a convincing feature set for stalking protection. We reverse-engineer Apple's tracking protection in iOS and discuss its features regarding stalking detection. We design "*AirGuard*" and release it as an Android app to protect against abuse by Apple tracking devices. We compare the performance of our solution with the Apple-provided one in iOS and study the use of *AirGuard* in the wild over multiple weeks using data contributed by tens of thousands of active users.

CCS CONCEPTS

• **Security and privacy** → Software reverse engineering; **Privacy protections**.

KEYWORDS

Privacy, location tracking, stalking, reverse engineering, Bluetooth, Find My, AirTag

ACM Reference Format:
Alexander Heinrich, Niklas Bittner, and Matthias Hollick. 2022. AirGuard - Protecting Android Users From Stalking Attacks By Apple Find My Devices. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '22), May 16–19, 2022, San Antonio, TX, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3507657.3528546

1 INTRODUCTION

Apple's release of the AirTag in April 2021 drastically changed the application domain of so-called "key finders" or "trackers," i.e., battery-operated devices that can be attached to personal items of importance and allow to locate these items if lost, misplaced, or stolen.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WiSec '22, May 16–19, 2022, San Antonio, TX, USA
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9216-7/22/05…$15.00
https://doi.org/10.1145/3507657.3528546

While the first generation of finders by different manufacturers was typically directly linked to the user's smartphone and, hence, limited in range and capabilities, the current finders operate using large-scale finder networks such as Apple's Find My network [5, 42].

In the case of loss or theft, the owner can receive a detailed location report of the device. AirTags are linked to Apple IDs and the owner can then view the location in the Apple Find My app on an iPhone, Mac, or Apple Watch (i.e., linked to the same Apple ID). The accuracy of these reports in practice is around 30 m in urban environments [25].

Apple's privacy-first approach to the Find My network delivers leading security and privacy properties for finder networks. It was found to protect the location privacy of the legitimate AirTag owner against outsiders as well as against Apple's ecosystem [25]. The security and privacy of many other finder systems was in a dismal state prior to the release of the AirTag [41].

However, AirTags can also be used for nefarious purposes. Given the diminutive size of only 3 cm, they can be easily hidden, thus allowing for tracking or stalking unsuspecting victims. The ubiquitous nature of the Find My network, combined with its high accuracy and low entry cost, lowers the bar for abuse. The media reacted promptly with articles linking AirTags with domestic abuse and stalking attempts [13, 20, 32]. Moreover, exposed individuals such as celebrities, activists, or critical journalists might further be targeted by paparazzi, secret services, or oppressive regimes, respectively. AirTag abuse is not limited to stalking or domestic abuse, but they are also used to identify the parking location of valuable cars, allowing to steal the cars thereafter [29, 32]. In March 2022, the Ukrainian government has also issued a warning that AirTags are used to track humanitarian aid by "saboteurs" [21]. Some of the known abuse cases were uncovered due to the automatic tracking protection feature offered by Apple devices such as iPhones. The tracking protection identifies suspicious AirTags in the user's surroundings and notifies the user when a device has been following them for a prolonged period [32].

AirTags can potentially endanger all non-Apple users in the aforementioned abuse scenarios because other smartphone ecosystems like Google's Android do not include compatible and mandatory tracking protection. It took Apple more than six months before releasing an Android app called "Tracker Detect" on December 11, 2021, to remedy this issue partially. Unfortunately, this app is ill-suited for the purpose and not usable in practice, as it requires the user to perform repeated manual scans to find a tracking device hopefully (see Section 5.1). As a result, Apple's current efforts leave all non-Apple users wide open for abuse.

Our goal is to provide non-Apple users with comprehensive anti-tracking protection. We emphasize automatic operation and prioritize user interface design to support non-experts. Lastly, our solution eschews including features that could be used for abuse.

Our key contributions are:

(1) We reverse-engineer Apple's tracking protection in iOS.
(2) We design, build and release the open-source[1] Android app *AirGuard* to protect people from AirTag abuse.
(3) We evaluate *AirGuard* and compare it against the tracking protection implemented in iOS.
(4) We analyze an anonymous dataset generated by *AirGuard* users to analyze tracking attacks in the wild.

This paper is structured as follows: Section 2 introduces the Find My network that powers AirTags and other Find My devices. In Section 3, we reverse-engineer the tracking detection in iOS. We explain how trackers are detected and when the system notifies the user. In Section 4, we introduce our app *AirGuard*. We describe its features, highlighting the user interface design and the tracking detection algorithm. Section 5 evaluates *AirGuard* in three scenarios and compares it to the iOS tracking detection algorithm. In Section 6, we analyze an anonymous dataset retrieved from users of our app. We present how many Find My devices are in use and how many users of our app have been notified about a potential tracker. We conclude our work in Section 7.

2 BACKGROUND AND RELATED WORK

This section describes the essential operation of Bluetooth item finders with a finder network, discusses details of Apple's Find My network, characterizes several stalking protections in place, and introduces work on analyzing Apple wireless services and other Bluetooth item finders.

2.1 Bluetooth Device Finders

Many Bluetooth-based item finders, or key finders, are now using finder networks aiding to find lost or misplaced trackers [7, 40]. Those networks are generally based on apps from item finder manufacturers. Any user of the app helps to find lost or stolen item finders. In principle, whenever an item finder is discovered via Bluetooth with the app, the app automatically reports the current location to the manufacturer, who then sends a notification of the discovered device to the item finder's owner. The ability to locate lost devices and the accuracy of reported locations mostly depends on the number of Android app users. iOS apps cannot scan for Bluetooth devices in the background, limiting the functionality of the finder network.

This area has gained much new attention since Samsung and Apple have created their Bluetooth item finders. Both companies created enormous finder networks that utilize active smartphones as finder devices [35].

Several researchers analyzed key finders from manufacturers like Tile, TrackR, and Nut for privacy and security. Two new privacy-preserving and end-to-end encrypted key finder protocols have been proposed [22, 41].

[1] https://github.com/seemoo-lab/AirGuard

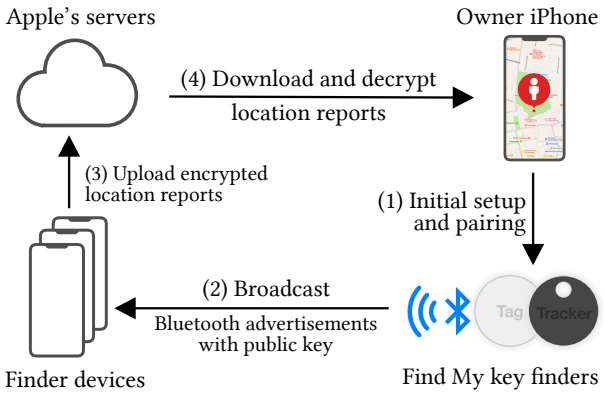


Figure 1: Simplified Find My network workflow. Adapted from [25].

2.2 The Find My Network

We summarize the features of the Find My network in this section. We describe its custom Bluetooth Low Energy (BLE) advertisement format and the working of the device finders available. For additional details on the Find My network and its cryptography, we refer to [25]. The authors also created OpenHaystack to build custom AirTag-like key finders [24]. Selected details about the Find My network have also been published by Apple [6].

We define the terminology used throughout this paper as follows:

- *Finder* devices (e.g., iPhones, iPads, Macs) aid in finding lost or stolen devices.
- The *Find My network* is a network of *finder* devices.
- *Find My accessories* are small devices that can be found through Apple's Find My network, including AirTags, AirPods, the Chipolo Spot ONE, and other third-party devices certified by Apple.
- *Find My capable Apple devices* are mainly devices with a screen, including iPhones, iPads, MacBooks, and other Apple devices, which also participate in the Find My network.

Since iOS 15, iPhones integrate a low-power mode allowing them to participate in the Find My Network even when they are turned off or their battery is empty. This low-power mode has been reverse-engineered and analyzed in [18].

Figure 1 shows a simplified representation of the Find My network workflow. The following steps are performed to recover misplaced or stolen devices: (1) The Find My accessories get initialized with an elliptic curve P-224 private-public key pair and a random secret. Those initial keys are called master beacon keys. Beginning with the private-public key pair, the devices can create an infinite number of rotating key pairs by utilizing a key derivation function and the known random secret. (2) When the accessory loses its BLE connection to the owner's iPhone, it emits the current public key using BLE advertisements. For Find My devices: these devices emit their current public key when they lose the internet connection (i.e., a MacBook without WiFi) (3) Finder devices, which discover such a Find My accessory, extract the public key, generate an ephemeral private-public key pair, and perform a one-sided key exchange using elliptic curve Diffie-Hellman. The generated shared secret

Table 1: Find My network advertisement format (with zero-index bytes). Adapted from [25].

Bytes	Content
0–5	BLE address $((p_i[0] \mid (0b11 \ll 6)) \mid\mid p_i[1..5])$
6	Payload length in bytes (30)
7	Advertisement type (0xFF for manufacturer-specific data)
8–9	Company ID (0x004C)
10	Find My network type (0x12)
11	Data length in bytes (25)
12	Status (e.g. battery level)
13–34	Public key bytes $p_i[6..27]$
35	Public key bits $p_i[0] \gg 6$
36	Hint (0x00 on iOS reports)

is then used to encrypt the finder's geolocation. The encrypted location and the finder's public key are uploaded to Apple's servers. (4) The owner can now use the Find My application to download and decrypt the location reports. For this, the owner device performs the other side of the key exchange by using the ephemeral public key of the finder and the private key of the Find My accessory, resulting in the same shared secret and allowing the owner to decrypt the location report.

For decryption purposes, the master beacon keys of all devices are synchronized in an encrypted form using iCloud. Utilizing the end-to-end encrypted iCloud keychain, any device signed in with the same Apple ID can decrypt the master beacon keys and generate the same private and public keys that were used in the BLE advertisements [25]. So far, it has not been detected that Apple violates the promise not to access the users' private keys [25]. Therefore, Apple should not be able to decrypt location reports.

2.2.1 BLE advertisement format. A public key on the elliptic curve P-224 can be compressed to 28 bytes length. Apple used some tricks to pack all bytes in one standard BLE advertisement while maintaining their common Type Length Values (TLV) based advertisement structure [25].

The first six bytes of the public key form the BLE address. Since the address needs to be identified as a static address, the first two bits must be set to `0b11`. The 22 remaining bytes are stored in the advertisement's manufacturer data. The manufacturer data follows Apple's standard encoding: The first two bytes are set to the company ID of Apple, the next byte is fixed to `0x12`, which identifies that this advertisement is used for Find My. The next byte defines the length of the message. Then, the remaining bytes are filled with the public key, a status byte, and a hint byte resulting in the structure shown in Table 1.

2.2.2 AirTag and Find My accessories. Apple opened up the Find My network such that accessories by other manufacturers can be located by it. Those accessories are available in as key finders [16], bikes, or headphones [7]. The AirTag was the first device to integrate Ultra-Wide Band (UWB) to allow short distance ranging and directional finding.So far, other accessories are not allowed by Apple to implement similar functionalities [4]. Many details about the internals of the Apple AirTag, how to modify the AirTags Firmware, and how to interact with it via Bluetooth have been revealed in [19]. Find My accessories have four states in which they operate [7]: (1) Unpaired (2) Connected (3) Nearby (4) Separated. In the unpaired state, the accessory is waiting to be paired and initialized by an iPhone, iPad, or iPod touch. When the accessory is paired, it stays connected with one of the owner devices (i.e., linked to the same Apple ID). If the accessory disconnects from the owner iPhone, it switches into the nearby state. In this state, it advertises the first part of the current public key in the BLE address, but it does not advertise the second part in the manufacturer data. Therefore, nearby finders cannot forward the accessory's location. After 15 min, the accessory transitions from the nearby state into the separated state. Now, it starts advertising the payload described in Section 2.2.1, and finder devices can report the accessory's location. The accessory can always transition back to the connected state from the nearby state and the separated state if an owner device is in proximity.

2.3 Find My Stalking Protection

2.3.1 Accessories. All small Find My accessories have to implement simple stalking protections. If the accessories stay separated for three days, they start playing a sound when they recognize movement. After a sound is played, the accessories will remain silent for at least six hours [4]. There are already manuals on the internet showing how an AirTag can be modified to disable the sound feature altogether [1]. In addition, those modified versions are now sold online as well, which lowers the bar for abusive use [27].

2.3.2 iOS. Since the release of the AirTags, iOS has contained a tracking detection that identifies when a Find My accessory has followed the user and sends a tracking alert. The user can view a route of the tracking device that has followed him, play a sound on the device, and get information on how to deactivate it. This system has been evaluated through extensive testing, and several ways to bypass it have been discovered in [12, 34]. We reverse-engineer the implementation in Section 3 and evaluate *AirGuard* against it in Section 5.

2.3.3 Android. In December 2021, Apple released an app called "Tracker Detect" on the Google Play Store [9]. This app was Apple's answer to numerous concerns about location tracking attacks on Android users by AirTags [13, 29, 32]. It allows a user to perform a manual BLE scan for Find My accessories. If the app finds such a device nearby, the user has to wait 10 min before they can play a sound on the device to find it. Section 5.1 discusses the features in greater detail and compares them to our implementation *AirGuard*.

2.4 Apple Wireless Services

Researchers have analyzed multiple wireless services used by Apple devices. AirDrop and Apple's custom WiFi communication layer have been analyzed for security and privacy in [37, 38]. Many of Apple's wireless services are summarized under the term *Continuity*. Those heavily rely on BLE and have been analyzed for privacy issues in [14, 33, 36]. The security of Bluetooth on iOS and the cellular baseband have been analyzed using fuzzing techniques in [26, 30]. In 2019, UWB was added as a new wireless technology in iPhones. The usage of UWB in combination with the U1 chip has been analyzed in [17]. UWB provides accurate distance measurements that are

secure against relay attacks. The technology is planned to be used to unlock cars [8]. Nevertheless, a first attack resulting in a distance reduction has been published [31].

3 IOS TRACKING DETECTION

iOS 14.5 added support for the Apple AirTag. This update also included an automatic tracking detection [10]. According to experiments by Mayberry et al., the tracking detection is triggered by an AirTag that sends BLE Find My advertisements while the user is moving, and the AirTag is in proximity to the user's iPhone. When an AirTag or another Find My accessory is marked as "*suspicious*," the user will receive a notification. The reception of such a notification can take up to several hours [34]. We reverse-engineer the tracking detection implemented in iOS and demonstrate our results in this section. Our analysis is based on iOS 15.2, released on December 13, 2021. This detection is available on iPhone, iPad, and iPod Touch.

3.1 Methodology

We use a combined dynamic and static analysis method to reverse-engineer the tracking detection. All these analyses are performed manually, and we rely on tools to facilitate them.

Dynamic analysis. Our goal was to reverse-engineer the current implementation of the automatic tracking detection in iOS. At the time of writing, no jailbreak supports iOS 15.2 or newer. Therefore, our dynamic analysis is based on system logs.

The tracking detection is executed by the `locationd` daemon linking the `TrackingAvoidance` framework. Daemons are UI-less processes that the operating system runs to handle tasks, including location processing, Bluetooth, WiFi, and more. Frameworks in iOS are dynamically linked, extending the functionality of processes. Both binaries use the system logs to output detailed information: Advertisements received, MAC addresses of Find My accessories, reasons for classifying a device as *suspicious*, and locations where a tracker has been detected. Apple provides several debug profiles to increase the log verbosity [11]. We installed the Bluetooth, Location Services, and AirTag profile for this analysis. We use the logs to understand the general behavior of the `TrackingAvoidance` framework and to identify entry points for a static analysis.

Static analysis. We base the static analysis on disassembling the binary of the `TrackingAvoidance` framework. As the strings in the system logs must be part of the binary, we use those to identify relevant functions in the `TrackingAvoidance` framework. The `locationd` and the `TrackingAvoidance` framework are written in Objective-C, which leaves all method names readable for our inspection. Our static analysis mainly focuses on the algorithms that classify a Find My accessory as a tracker.

3.2 System Overview

The tracking detection on iOS is a complex combination of several system services that communicate through XPC, Apple's cross-process communication. In Figure 2, we highlight the components interacting with each other.

Many Apple services, e.g., Continuity [33, 36], the COVID-19 exposure notification framework [3], and the Find My network require

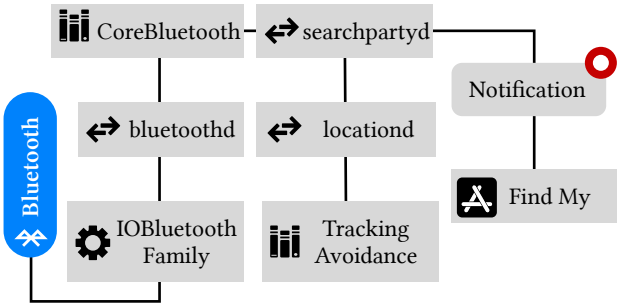


Figure 2: Simplified representation of the iOS tracking detection using components such as apps (), daemons (), frameworks (), and drivers ()

periodic scans for BLE advertisements. Therefore, `bluetoothd` performs these scans and forwards the results to different system daemons. All BLE advertisements using the Find My format are first forwarded to the `searchpartyd`. This daemon is handling the discovery of lost devices and sends the encrypted location reports to Apple's iCloud servers such that the owner can locate and find their devices [25]. After processing the advertisement, the `searchpartyd` forwards it to the `locationd`. This daemon utilizes the `TrackingAvoidance` framework, which handles the detection of trackers following the owner of the iPhone/iPad. If this framework has classified a device as *suspicious*, it tells the `searchpartyd` to emit a notification. After opening the notification, the user is forwarded to the Find My app to view a detailed report about the discovered tracker.

All detection mechanisms are based on BLE advertisements and do not incorporate more accurate distance measurements using UWB.

3.3 Collecting Events for Tracking Detection

In the `TrackingAvoidance` framework, all Find My advertisements received are checked if they relate to the current user's devices. All devices logged in with the same Apple ID pre-compute the private and public keys of all devices linked to this account. Those keys are then stored in a read-protected directory on disk for quick look-ups [25].

Non-owner advertisements are added to a `TAStore`. Besides BLE advertisements, the store receives all kinds of events that are partially used to classify if the person is being tracked. These events include regular location updates, user activity events, system state events, and vehicular states. Table 2 shows a list of all known event states.

3.4 Classification of Tracking Devices

We identified three algorithms implemented by the `TrackingAvoidance` used to identify *suspicious* devices. The first one is the *general detection*, which is executed regularly and iterates over all devices saved in the `TAStore`. Significant location changes, so-called visits, set off the other two. We identified the default values for many parameters in the framework. However, the actual values

Table 2: Different event states added to TAStore.

System States	User Activities	Vehicluar States
Display on	Static	Vehicular
DeviceUnlockedSinceBoot	Pedestrian	Non Vehicluar
HasKoreaCountryCode	Vehicular	
Wifi	unknown	
LocationServices		
BatterySaver		
HighThermal		
AP		

may change during execution, and as they are not logged, we cannot inspect them dynamically.

3.4.1 General detection. Every $2-5$ min, the `locationd` runs a classification on all devices from which advertisements have been received in a recent time frame (default: 15 min). The algorithm defines a threshold duration (default: 10 min) and a threshold distance (default: 840 m) for which the devices had to follow the person.

The classification of all devices follows the pseudo-code shown in Algorithm 1. Essentially, it checks if any device exceeds both thresholds, if the recent user activity is somehow plausible and if, additionally, a people density scan measurement is used. We could not find clear indications if the people density scan was performed during our dynamic analysis. In iOS 15.2, the people density scan does not seem to be activated. For every *suspicious* device, iOS sends a notification to the user.

Algorithm 1 General Filter that classifies suspicious devices.

```
devices ← TAStore.devices
for device in devices do
    isInVehicle ← TAStore.lastVehicularState.isInVehicle
    peopleDensity ← TAStore.lastPeopleDensity
    userActivity ← TAStore.dominantUserActivity
    walkingSpeed ← TAStore.walkingSpeed
    dist ← device.distanceTravelled
    dur ← device.durationTravelled
    isDriving ← isInVehicle | peopleDensity
    isDriving ← isDriving & (userActivity = vehicular)

    isWalking ← (userActivity = pedestrian)
    isWalking ← isWalking&(walkingSpeed < MAX_SPEED)
    validMovement ← isDriving | isWalking
    minDistTravelled ← dist > THRESHOLD_DISTANCE
    minDurTravelled ← dur > THRESHOLD_DURATION
    result ← validMovement
    result ← result & minDistTravelled
    result ← result & minDurTravelled
    if result = 1 then
        mark device as suspicious
    end if
end for
```

Table 3: Different device types in iOS TrackingAvoidance

Device Type	Bits	Category	Example
Other	$0b00$	Apple devices	iPhone, Mac, iPads
D (Durian)	$0b01$	AirTags	AirTag
H (Hawkeye)	$0b10$	3rd Party	Chipolo ONE Spot
HELE	$0b11$	Headphones	AirPods Pro

3.4.2 Visit-based detection. The public `CoreLocation` framework implements the `CLVisit` objects, informing an app that a person is currently *visiting* a specific location. Those objects contain an arrival and a departure time, and they are generated when the user arrives or leaves a location. Apple does not specify the duration a person needs to be at a location to generate a `CLVisit` [2]. Based on these objects, the `TrackingAvoidance` framework observes user visits and combines this information with BLE advertisements saved to the `TAStore`.

The first algorithm, `TADetectionTypeVisit` is simple. It creates an intersection between the devices seen at the last visit and the current visit. All devices in this intersection are marked as *suspicious* devices. No other checks are performed. The distance and the duration that the device has traveled with the user are stored.

The second algorithm, `TADetectionTypeSingleVisit` uses the intersection created before and performs extra checks on the devices. If all checks evaluate to *true*, the device will be marked as a *suspicious* device. For each device, the algorithm checks:

(1) Has the last advertisement been received recently (default: in the last 5 min)?
(2) Is the device traveling with the user for a longer distance than the threshold distance (default: 420 m)?
(3) Is the device traveling with the user longer than the threshold duration (default: 5 min)?

The dynamic analysis has shown that both algorithms are in active use to classify a device as a *suspicious* device. Interestingly, the threshold distance and duration in the `TADetectionTypeSingleVisit` are only half the size compared to the general detection.

3.5 Tracking Device Types

In iOS 15.2, the system identifies four different device types named `Other`, `D`, `H`, and `HELE`. Bit 2 and 3 of the status byte in the BLE advertisement categorize the devices. We identified categories for each type and listed them in Table 3.

Devices that are marked as "Other" are, in most cases, Apple devices such as MacBooks that lost their internet connection and are now sending Find My advertisements [25]. Apple sets their status byte to `0x00`. Note that such devices are marked as "irrelevant" for tracking in the system logs. Previously Heinrich et al. created a tag that copied the behavior of a MacBook with Find My, which essentially creates a tracker that evades Apple's tracking detection, an issue initially discovered and reported by Mayberry et al. [24, 34]. Using our reverse engineering approach, we can confirm that this issue persists in iOS 15.2.

3.6 Notifications

At what time a notification is delivered is defined by several factors. When the user has returned home and and the *suspicious* devices continue to appear during the BLE scans, the system sends out a notification promptly. If the user is not returning home, the notifications are delayed. In this case, the `TrackingAvoidance` marks *suspicious* devices as "staging." Every staging phase contains an end date. After the staging has ended, the `locationd` can either send a notification immediately or prolong the staging phase. We could not identify the trigger that determines when a notification is sent if the user does not return home. The evaluation in Section 5.3 shows that it can take several hours.

3.7 Discussion

Apple's `TrackingAvoidance` framework implements a tracking detection that leverages the information from many different sources and frequent (every $1-5$ min) BLE scan intervals. Many of those features are not available to third-party applications. The downside of this framework is that Apple neglects the fact that any BLE capable device can be transformed into a Find My network tracker by mimicking the behavior of an iPhone as demonstrated by the OpenHaystack framework [24]. Users will not get a notification for those trackers.

Apple's tracking detection identifies trackers early on. However, it also delays the delivery of notifications. The only way to reliably get a notification is by returning to the home location, which essentially gives the home location away to the attacker. A possible reason why notifications are delayed is that Apple wanted to minimize the risk of false positives and guarantee that the trackers are travelling with the user.

At last, Apple's Bluetooth API disallows access to Bluetooth scans in the background. Removing those limitations could allow third-party developers to create apps that detect trackers from other manufacturers.

4 AIRGUARD

After the AirTag release in April 2021, we started with the design and implementation of AirGuard. The first public version was released in August about four months before Apple's Tracker Detect app was released in December 2021. The automatic tracking protection is intended to work similarly to the one implemented in iOS. Using regular BLE background scans, *AirGuard* detects all kinds of Find My capable tracking devices, including self-made ones. In less than 45 min[2] users receive a notification about the tracker and guidance on finding it. In this section, we discuss ethics and privacy considerations, we introduce *AirGuard* and its tracking detection algorithm. We then explain how we designed the user interface to cater to non-expert users.

4.1 IRB Approval and Ethics Considerations

The features of *AirGuard* require that the app has access to several sensitive Android permissions. For example, *AirGuard* can access to the user's location in the background to provide accurate tracking detection. We are ensuring that no location data leaves the Android device, and make the source code of *AirGuard* publicly available on GitHub[3] for independent reviews.

All users of the app will be asked if they are willing to share anonymous data to participate in our study. We follow TU Darmstadt's process for determining whether our study is subject to review by the institutional review board (IRB). Due to the voluntary nature and the fully anonymous data, our study was declared exempt from further IRB review [39].

To ensure high privacy standards and ethical uncritical research, we implement the following measures: users can participate voluntarily, they have to confirm that they are 18 years or older, they can stop data sharing at any time in the app, and we fully anonymize all data before it leaves the participant's smartphone.

To anonymize, our implementation will remove all location data at which tracking devices have been found, all MAC addresses of tracking devices, and all data shared by devices, such as the public key in Find My BLE advertisements. Therefore, we can neither identify individual participants nor trackers used against or by participants.

Centrally on our server, we can obtain the number of devices, beacons, notifications that a user has received. These objects contain metadata like the date the devices and beacons were received and a basic device type. A database schema for all tables that contain user-generated data is shown in Appendix A.3.

4.2 Tracking Detection

The tracking detection of *AirGuard* is a combination of scheduled BLE background scans and a detection algorithm that classifies malicious devices as location trackers.

4.2.1 Bluetooth scanning. The app uses the Android Work Manager to schedule recurring background tasks to perform the tracking detection. In these tasks, the app performs a BLE scan and runs our tracking detection. In Android, such tasks can be executed at maximum every 15 min, which limits the time to detection.

During a scan, the app filters the results for Find My advertisements in the separated state (see Section 2.2). *AirGuard* fetches the current geolocation and stores all advertisements received together with the location in a local database. When Find My accessories are in the nearby state, they cannot be located by finders, and therefore *AirGuard* ignores them.

To achieve high reliability on detecting trackers, the scan duration is set to 8 s and the Android scan mode to *low latency*. More details on these parameters are listed in Appendix A.1.

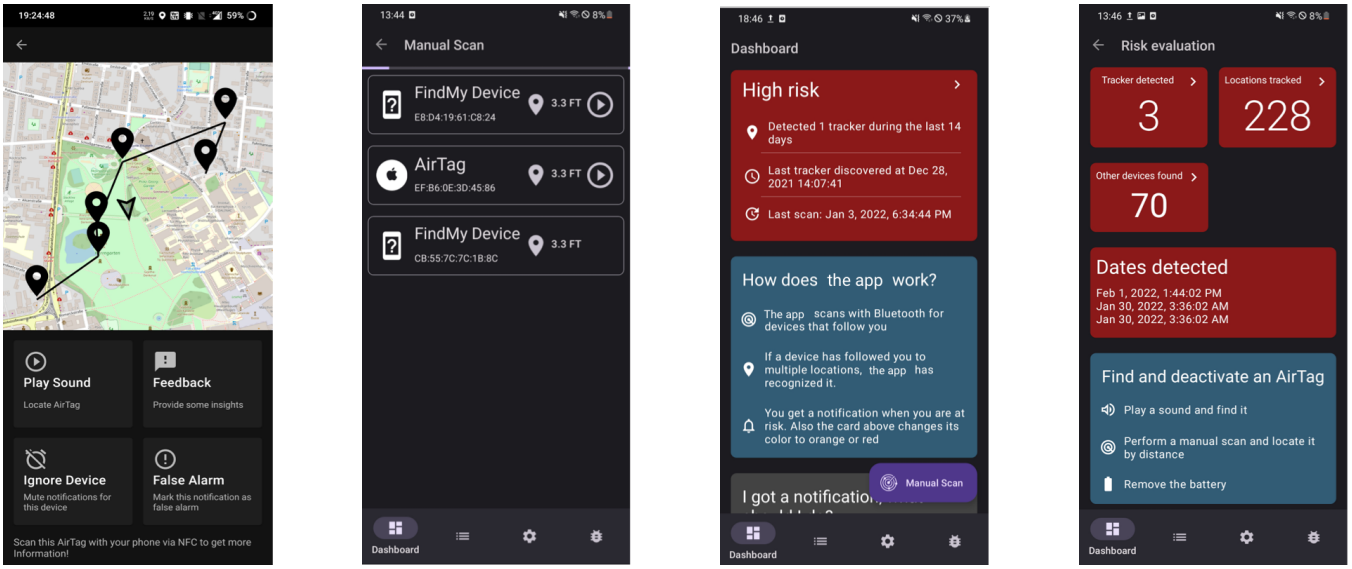
4.2.2 Tracker classification algorithm. After every scan, the app runs our detection algorithm, which iterates over all devices in the database. Generally, *AirGuard* identifies different devices based on their BLE MAC address. As the MAC address is formed by using the current public key, it does not change until the public key changes.

For every device, the algorithm performs the following checks:

(1) Have the device's advertisements been received for longer than the minimum duration of 30 min.
(2) Has the device been discovered at least three times.
(3) Has the device traveled with the user for a minimum distance of 400 m.

[2]Since version 1.2 AirGuard uses a minimum tracking detection of 2.5 h for Apple devices and 30 min for Find My accessories

[3]https://github.com/seemoo-lab/AirGuard



(a) Information after a tracking alert.

(b) User performing a manual scan to find trackers.

(c) User with a high risk of being tracked.

(d) Details on how trackers follow the user.

Figure 3: Screenshots of *AirGuard*.

(4) Has there already been a tracking alert in the last seven hours? *AirGuard* limits the number of notifications to one per device every for this duration.
(5) If all checks evaluate to `true`, a notification will be delivered to the user immediately.

In general, our algorithm is a simplified version of Apple's general tracking detection as described in Section 3.4.1. We use a minimum distance and a minimum duration to limit the number of false positives when warning a user. Therefore, an AirTag of a neighbor or a family member does not result in a notification.

Since we identify devices based on their BLE MAC address, they will be recognized as a new device when the public key changes. Our main targets are Apple AirTags, and other Find My accessories. According to the specification, Find My accessories update their public key once a day around 4 a.m. local time [4].

4.2.3 Notifications. *AirGuard* delivers tracking alerts as Android notifications. In the current implementation, the notifications are delivered early. Early notifications should help users to identify a potential tracker quickly and even before they return home and might reveal their home location to someone else.

When users open the notification, they get a detailed overview of the locations where the tracking device has been detected. Figure 3a shows a screenshot of the user interface presented to the user when a tracking notification is opened. Here, the user can play a sound on the device, ignore it, send feedback, and mark this notification as a false alarm. The feedback is basic information on where the tracking device has been hidden (car, bike, backpack, clothes), and it will be sent to the developers if the user has agreed to share data.

4.3 Manual Scanning

Finding a tracking device can be difficult, especially if they are self-made or modified such that the AirTags cannot play a sound [27]. In this case, a user can manually initiate a scan process and see an automatically updating estimated distance of all potential tracking devices in the vicinity. Figure 3b shows a screenshot of this process. *AirGuard* estimates a distance based on the received signal strength indicator (RSSI) and sample measurements with an AirTag. Precision varies due to the design of Bluetooth. If the device does support playing a sound, this can also be performed from here immediately.

4.4 User Experience

The app's main view is a dashboard, as shown in Figure 3c. This dashboard should allow non-experts to identify their personal risk at one glance. The main element is the risk level card, which represents the user's risk of being tracked in three colors:

(1) Green (Low): no tracking device following the user has been detected in the last 14 days.
(2) Orange (Medium): one or multiple devices have been detected that followed the user for less than 24 hours.
(3) Red (High): one device or multiple devices have been detected that followed the user for more than 24 hours.

Additional details on how many trackers were found and when the last one has been detected are displayed as well. When the user taps on the card, they get more details (see Figure 3d): the number of locations that have been recorded with the tracker and the number of Find My devices that have been found in total.

From here on, users can get more information by viewing a list of all devices that have been marked as tracking the users' location. They can view all locations where the device has been detected on a map and initiate the sound on Find My accessories. At last,

users can also view all Find My devices found by the app even if they have not followed them. Expert users who want to validate the tracking detection behind *AirGuard* highly requested this feature.

5 EVALUATION

This section evaluates and compares *AirGuard* to the iOS automatic tracking detection by testing them in three scenarios. The focus is set on: trackers that can be detected, placements of the trackers, the time it takes until the user is notified, and the number of locations at which the tracker has been detected until a notification was sent.

Attacker model. We define our attackers as people with basic technical knowledge who have access to an Apple device, buy an off-the-shelf tracker by Apple or a Find My certified manufacturer. We include attackers who are capable of using the open-source framework of OpenHaystack to create a self-made tracker. Using this device, they try to track the location of their victim. Neither the iOS tracking detection nor *AirGuard* can protect against advanced tracking attacks with skillfully modified Find My devices. For example attackers could create stealth trackers by making the tracker appear as a different device during every BLE scan [12, 34]. We consider detection of such stealth tracker to be future work.

5.1 Tracker Detect app

We did not evaluate *AirGuard*'s automatic tracking detection against Apple's Tracker Detect app because the latter does not implement such features. Tracker Detect (v1.1) cannot automatically detect tracking devices in the background and, hence, cannot warn the user about such a device. The app can only perform manual scans allowing users to search for trackers, if they suspect a potential threat.

Compared to the manual scan of *AirGuard*, Tracker Detect does not provide an estimation on the distance to a device and it cannot detect Find My capable Apple devices and therefore no OpenHaystack based trackers. If *AirGuard* detects a tracker the app is able to play a sound on it immediately, while Tracker Detect has an artificial waiting time of 10 min. Due to its Apple integration, Tracker Detect can exactly identify which type of tracker has been found and deliver detailed information on how to disable the tracker and how it can be returned to the owner if it is a lost device. This information is retrieved by setting up a BLE connection to the tracker and reading bytes from a BLE characteristic. The characteristic contains a (potentially encrypted) byte sequence which has not been fully reverse-engineered yet. Based on this sequence the app identifies the tracker type and fetches information on how to disable it.

5.2 Setup of the Experiments

To evaluate our app and the iOS tracking detection, we use the following devices: for the iOS tracking detection, we use an iPhone 12 mini that runs iOS 15.2. Find My and the Find My Network are activated on the iPhone. For the *AirGuard* tracking detection, we use a Samsung S21 Ultra 5G with *AirGuard* version 1.1.2 installed. *AirGuard* uses the following settings: location access in the background is granted, battery optimization is disabled. Before each test, we clear the database of *AirGuard*.

We test the app against three different devices: an Apple AirTag [5], a Chipolo ONE Spot [16], and a self-made OpenHaystack tag [24]. The AirTag and the Chipolo tag follow Apple's reference implementation of the Find My accessories [7]. During the evaluation experiments, all tags were kept in a separate state. The self-made OpenHaystack tags essentially copy the BLE messages of a lost iPhone. They keep a static public key and do not update it.

We use the following placements for our testing: In a pocket of a victim's coat, inside a backpack, and attached to a car. The first two placements are ideal for tracking a person's movement. The currently available devices can be easily hidden inside a backpack, making it difficult for a victim to discover them. Tagging vehicles has been chosen because of recent media coverage of thieves attaching AirTags to high-end cars to track vehicles back to their parking location [29, 32].

5.3 Results

We divide the results of our evaluation into sections based on the placements of the trackers and summarize the results in Table 4. We utilize all three trackers in all scenarios to identify how the different systems recognize different tracking devices.

Pocket. For the trackers' pocket position we were able to receive tracking notifications in 35 minutes with *AirGuard*, which is much faster than it took iOS to send a notification. Even though iOS did recognize the tracking devices early on and continued to scan for them in frequent intervals, the system decided to delay the tracking notification. In this evaluation the notification was sent exactly when we returned to the iPhone's *home* location.

Backpack. For the backpack position of the trackers, *AirGuard* took 31 minutes to locate the trackers and send a notification. For iOS, it took more than 4 hours to send a notification. Both devices were recognized early on and linked to over 70 locations. Again, the main reason for iOS to delay notifications is the user's current location. We did stay away from the home location for 6 hours in this scenario.

Car. News reports mention that the fuel cap is a commonly used position by attackers who do not have access to the inside of the car [15]. We glued an AirTag and a Chipolo ONE Spot to the inside of the fuel cap. For space reasons, we placed the OpenHaystack tag inside the car. We took two 30 minute rides with the prepared car and stayed away from the car for 3 hours and 30 minutes in between. *AirGuard* sent a tracking notification during the second ride, when the trackers were detected for the third time. The iOS tracking detection never sent a notification, not even after returning to the iPhone's home location.

5.4 Discussion

Our evaluation shows that *AirGuard* detected all three trackers in all scenarios. The app uses an aggressive detection algorithm, which resulted in a notification after three discoveries for all scenarios.

Car. Unfortunately, the iOS tracking detection failed to detect trackers that were not permanently close to the user in our car scenario. Based on our results in Section 3, we try to identify the reason for these missed notifications. The general detection algorithm only

Table 4: Tracking Detection Results

Tracker position	Tracker	Time to notification	Locations with tracker until notification
iOS			
Pocket	AirTag	1h 45m 14s	35
Pocket	Chipolo	1h 45m 14s	28
Pocket	OpenHaystack	-	-
Backpack	AirTag	4h 14m 23s	87
Backpack	Chipolo	4h 14m 23s	74
Backpack	OpenHaystack	-	-
Car	AirTag	-	-
Car	Chipolo	-	-
Car	OpenHaystack	-	-
AirGuard			
Pocket	AirTag	35m 20s	3
Pocket	Chipolo	35m 20s	3
Pocket	OpenHaystack	35m 20s	3
Backpack	AirTag	30m 49s	3
Backpack	Chipolo	30m 49s	3
Backpack	OpenHaystack	30m 49s	3
Car	AirTag	4h 18m 11s	3
Car	Chipolo	4h 18m 11s	3
Car	OpenHaystack	4h 18m 11s	3

uses the advertisements received in the last 15 min. If the user now stays out of the vicinity for more than 15 min, the trackers will be forgotten by iOS. However, the default threshold duration of 10 min and the threshold distance of 840 m have been exceeded in our test twice. We expect the default values to change during execution and based on user behavior. If this has been the case, it would explain why no notification has been delivered. Media reports have mentioned that people have been notified of AirTags on the car [32]. A straightforward mitigation would be to integrate all received advertisements of the current day in the tracking detection.

Notifications. The iOS' tracking detection delivers notifications much later, depending on the user's location. Apple has announced to mitigate this issue in an update [28]. Furthermore, the evaluation has confirmed that self-made tags with OpenHaystack will not be detected [34].

Scan opportunities. One weakness of *AirGuard* is the limited scan opportunities the Android operating system grants it. Therefore, *AirGuard* finds fewer locations with a tracker. As a comparison, iOS detected the AirTag at 35 locations in 1 h 45 min in the pocket scenario and *AirGuard* detected it at seven locations.

False positives. The limited scan opportunities and the intentional early notifications can also lead to false positives. During our evaluation, we did not experience any false notification, but we can imagine several scenarios that would trigger one: (1) Travelling in a train or airplane next to an AirTag in separated state (e.g., due to Airplane mode) *AirGuard* will issue a notification. (2) GPS drifts. When using *AirGuard* indoors and the smartphone fails to get a good GPS signal or fails to receive a location update based on WiFi signals, the location might drift away for more than our minimum distance. If an AirTag is close in this case, it will appear to be following the user, and the app may send a notification.

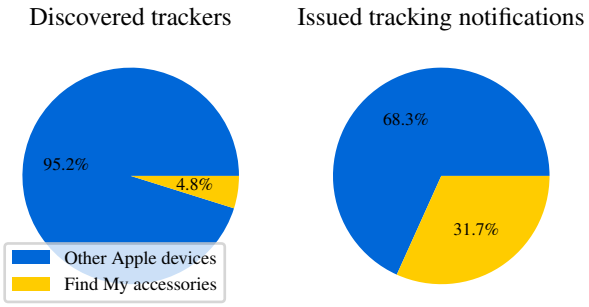


Figure 4: Device type distribution for discovered trackers as well as issued tracking notifications.

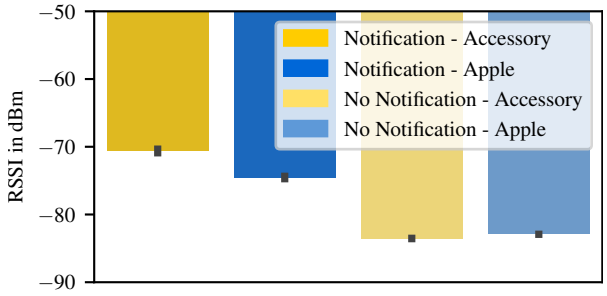


Figure 5: Mean RSSI comparing tracking devices with normal devices.

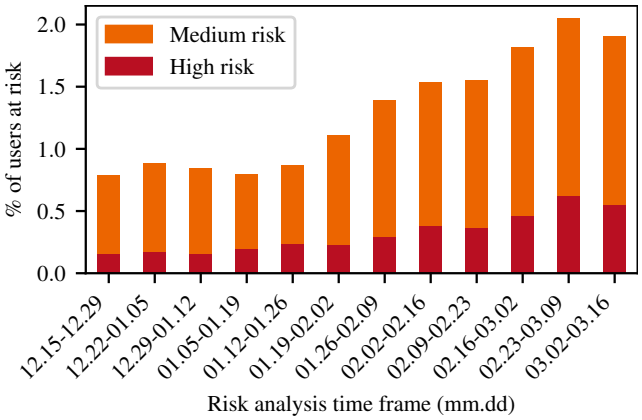
Those false positives may not be harmful to most users because the user can then match the reported locations to a specific event based on the provided map. However, the risk score of the app will increase to medium if the user receives one false positive in 14 days. We keep possible solutions for this for future work.

Fast rotating trackers. *AirGuard* and the iOS tracking detection, do not detect skillfully modified tracking devices, which rotate their public key more than once an hour [34]. As the public keys are unlikable, the device would be detected as a new device after a key rotation. Until a recent publication an attacker would require advanced technical knowledge to create such a tracking device [12]. Now, as the source code of such a device is partially open-source, attackers with basic programming knowledge can create a tracker that cannot be easily detected automatically.

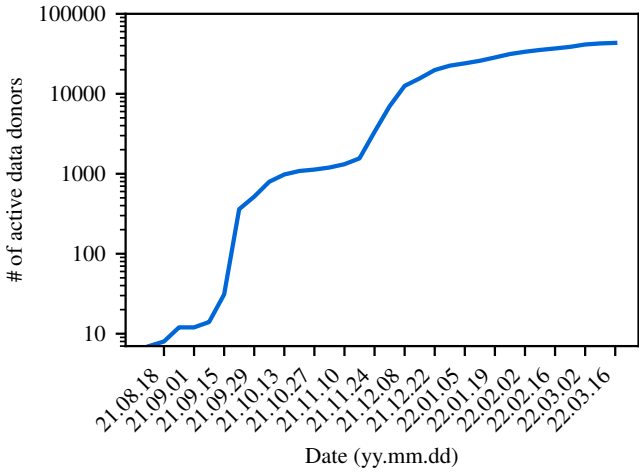
AirGuard can detect such trackers using the manual scan feature. Since they are mostly masking themselves as Find My capable Apple devices, the manual scanning opportunity in iOS and in Apple's Android App Tracker Detect do not discover them.

6 REAL WORLD OBSERVATIONS

Since the first release of *AirGuard*, users have been able to share anonymized data. The shared data allows identifying potential problems of the app and getting hints on the prevalence of tracking attacks in the wild.



(a) Percentage of data donors at a medium and high risk levels from December 15, 2021 to March 16, 2022.



(b) Overall active data donors since the release of the app.

Figure 6: Statistics from data donors

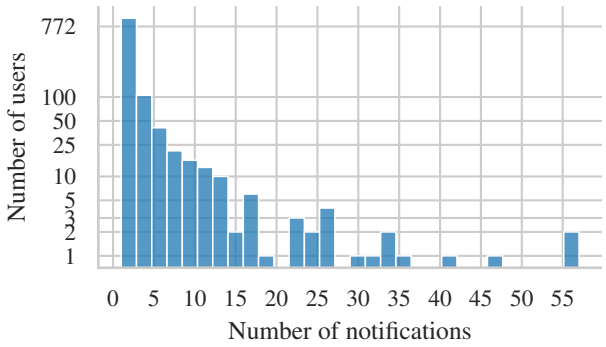


Figure 7: Density distribution over number of notifications received per user.

Provided data. We ask all users if they are willing to share anonymous data with us during the app setup. If they agree, the app will regularly upload received advertisements, devices, tracking notifications sent, and optional user feedback. With over 120 000 active users, of which over 38 000 donated anonymized data, we created a dataset with more than 11 million Find My advertisements received.

Limitation. We limit the evaluation of our data to a 13-week time frame starting on December 15, 2021, and ending on March 16, 2022, as we changed our data collection framework with an update on December 11, 2021.

6.1 Discovered Devices

We can separate the devices that *AirGuard* found into two groups: Find My accessories (e.g., AirTags) and Find My capable Apple devices (e.g., MacBooks) as defined in Section 2.2. Self-made tags copy the behavior of Apple devices and are therefore identified as Apple devices [24]. In total, users have discovered over 9 million devices that use Find My in our defined time frame. Only about 467 000 were Find My accessories. All devices together caused 9728 notifications for 3680 users. As shown in Figure 4, 32% of all notifications were caused by Find My accessories, while they represent only 5% of all devices found. Since Apple devices are expected to rotate their public key every 15 min [18, 25], which results in detecting a new device after every rotation, the distribution of discovered trackers can be expected.

Figure 5 shows the mean RSSI based on the device type and if the device issued a tracking notification. The app captures the RSSI for all advertisements received by *AirGuard* users. It hints at how close (lower means closer) a tracker is to the user during the reception, but it does not allow precise distance measurements. The dataset has two takeaways: (1) Devices that trigger a notification are closer to the user. (2) Tracking accessories are closer to the user than Apple devices.

Limitations. From the current dataset, it is impossible to state if notifications have been initiated on purpose by users who wanted to test *AirGuard*, and how many Apple devices were falsely classified as malicious trackers.

To further evaluate the number of false alarms, we tried to identify how many Apple devices could have been an OpenHaystack tag and how many notifications were marked as false alarms by participants. We detected 17 trackers in our database which follow the behavior of OpenHaystack tags by being detected for over one day without changing their public key. Those devices were identified by *AirGuard* as an Apple device even though Apple devices generally should update their public key every 15 min [25]. Through our whole evaluation period, these OpenHaystack based tags make up only 0.26% of all Find My capable Apple devices. During the same time, 308 notifications were reported as false alarms and 80% of those were caused by Apple devices. We see a slight tendency that they cause more false alarms than AirTags if compared to Figure 4. Altogether, we are currently expecting that many notifications caused by Find My capable Apple devices are false alarms.

As *AirGuard* needs to detect a device for at least 30 min, many notifications from Apple devices signify that not all devices stick to the 15 min public key update interval identified in [25]. In fact, the average detection duration of Apple devices that caused a notification is 90 min. If Apple devices do not permanently keep that interval, it would also explain why Apple refrained from warning users if such a device follows them. To limit the number of false positives, version 1.2 of *AirGuard* introduces a minimum detection time of 2.5 h for Apple devices. It has been released in a phased roll-out from March 4 to March 14.

6.2 Location Tracking Prevalence

The main goal of *AirGuard* is to serve as a tool that protects its users from abuse through stalking by using an AirTag or a similar Find My accessory. This section analyzes the data to discover hints on the prevalence of location tracking in the wild. We limit the data to only Find My accessories, such as AirTags and the Chipolo ONE Spot. We do not cover self-made trackers and lost Apple devices because they can cause more false positives (see Section 6.1).

As described in Section 4.4, the app calculates a user-based risk level which is based on the time a user has been tracked. The risk level always accounts for trackers detected in the last two weeks, and it can be low, medium, or high.

Figure 6a shows the percentage of data donors marked with a medium or a high-risk level. The number of data donors in a specific risk level results from calculating every user's risk level at the time frames shown on the horizontal axis using a two-week sliding window. We then calculate the percentage by dividing through all active data donors. An active data donor is defined as a user who has submitted any data during the two-week time frame. It shows that in the beginning, around 0.2% of data donors are constantly ranked at a high risk of being tracked. Starting at the end of January, the percentage increases up to 0.6% for users in high risk and 1.5% for users in medium risk. We assume that this increase results from optimized scanning parameters (see Appendix A.1) delivered in an update on January 30, 2022.

Figure 6b shows the number of active data donors since the release of *AirGuard*. The first steep increase of new data donors is related to a hacker news post[4]. After a woman who has been a victim of a tracking attack mentioned our app in a TikTok video[5] *AirGuard* received a second steep increase.

Figure 7 shows a histogram over the number of notifications a user has received. Most users receive only one notification, while some individuals received up to 57 notifications in our 13-week observation period.

6.2.1 User feedback. Users can give feedback on identified trackers. This can be that a tracker is marked as a false alarm or that the user provides us with basic information where a tracker has been hidden. During our evaluation, 14 users reported the finding place of identified trackers: one user declared that the tracking device was placed on their bike, one was on or in their clothes, four were found in a backpack and eight times a tracker was reported to be in or on their car. Besides our feedback through the app, we also received reports from multiple users through reviews on the Google Play Store or email. One user who wants to stay anonymous messaged us via email: "*I have found 2 Airtags attached to my car . . . I don't really know what else to do or say. I just wanted to say thank you for your app.*" Several screenshots of the app showing tracking devices found have been attached to this email.

[4]https://news.ycombinator.com/item?id=28577951

[5]https://www.tiktok.com/@angel.edge95/video/7035276471237217541

7 CONCLUSIONS

We reverse-engineered the iOS tracking detection to analyze its limits. Furthermore, we confirmed a way to bypass the detection using self-made trackers in iOS 15.2, initially discovered in iOS 14.5. We designed, implemented, and released the Android app *AirGuard*, an open-source anti-tracking solution against Find My accessories. More than 120 000 active users show a high demand for automatic tracking detection. A first evaluation showed that *AirGuard* found more trackers in different scenarios compared to the iOS tracking detection. We analyzed a dataset generated by over 38 000 *AirGuard* users. The dataset showed that the tracking detection is triggered in practice and several user reports confirmed this. We hope our work leads to better tracking protection mechanisms implemented by tracker and smartphone manufacturers directly.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers and our shepherd, Dan Wallach, for their constructive feedback. We thank the participants of our the study. This work has been funded by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE.

REFERENCES

[1] AirtagAlex. 2021. How to remove the speaker coil from the Airtag (make your Airtag very silent!) - YouTube. https://www.youtube.com/
[2] Apple Inc. 2014. CLVisit | Apple Developer Documentation. https://developer.apple.com/documentation/corelocation/clvisit
[3] Apple Inc. 2020. Exposure Notification | Apple Developer Documentation. https://developer.apple.com/documentation/exposurenotification
[4] Apple Inc. 2020. Find My Network Accessory Specification - Release R1. https://images.frandroid.com/wp-content/uploads/2020/06/Find_My_network_accessory_protocol_specification.pdf
[5] Apple Inc. 2021. AirTag. https://www.apple.com/airtag/
[6] Apple Inc. 2021. *Apple Platform Security*. Technical Report. 189 pages.
[7] Apple Inc. 2021. Apple's Find My network now offers new third-party finding experiences. https://www.apple.com/newsroom/2021/04/apples-find-my-network-now-offers-new-third-party-finding-experiences/
[8] Apple Inc. 2021. Explore UWB-based car keys - WWDC21 - Videos - Apple Developer. https://developer.apple.com/videos/play/wwdc2021/10084/
[9] Apple Inc. 2021. Tracker Detect - Apps on Google Play. https://play.google.com/store/apps/details?id=com.apple.trackerdetect&hl=en&gl=US
[10] Apple Inc. 2021. What to do if you get an alert that an AirTag or Find My network accessory is with you. https://support.apple.com/en-us/HT212227
[11] Apple Inc. 2022. Profiles and Logs - Bug Reporting - Apple Developer. https://developer.apple.com/bug-reporting/profiles-and-logs/
[12] Felix Bräunlein. 2022. Find You: Building a stealth AirTag clone | Positive Security. https://positive.security/blog/find-you
[13] Albert Fox Cahn. 2021. Apple's AirTags Are a Gift to Stalkers. *Wired* (May 2021). https://www.wired.com/story/opinion-apples-air-tags-are-a-gift-to-stalkers/ Section: tags.
[14] Guillaume Celosia and Mathieu Cunche. 2020. Discontinued Privacy: Personal Data Leaks in Apple Bluetooth-Low-Energy Continuity Protocols. *Proceedings on Privacy Enhancing Technologies* 2020, 1 (Jan. 2020), 26–46. https://doi.org/10.2478/popets-2020-0003
[15] Hartley Charlton. 2021. Apple's AirTag Item Trackers Increasingly Linked to Criminal Activity. https://www.macrumors.com/2021/12/31/airtag-increasingly-linked-to-crime/
[16] Chipolo d.o.o. 2021. Chipolo ONE Spot. https://chipolo.net/en/pages/chipolo-one-spot

[17] Jiska Classen and Alexander Heinrich. 2021. Wibbly Wobbly, Timey Wimey – What's Really Inside Apple's U1 Chip. Presentation at Black Hat USA 2021.
[18] Jiska Classen, Robert Reith, Alexander Heinrich, and Matthias Hollick. 2022. Evil Never Sleeps: When Wireless Malware Stays On After Turning Off iPhones. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, San Antonio, TX, USA.
[19] Jiska Classen, Thomas Roth, and Fabian Freyer. 2021. Hardwear.Io NL 2021: Over The Air-Tag: Shenanigans With A Keyfinder by Jiska , Fabian And Thomas. https://www.youtube.com/watch?v=nrs0kNG7jcA
[20] Geoffrey A. Fowler. 2021. Review | Apple's AirTag trackers made it frighteningly easy to 'stalk' me in a test. *Washington Post* (May 2021). https://www.washingtonpost.com/technology/2021/05/05/apple-airtags-stalking/
[21] Geoffrey A. Fowler. 2022. Review | Am I Being Tracked? Anti-stalking Tech from Apple, Tile Falls Short. *Washington Post* (March 2022). https://www.washingtonpost.com/technology/2022/03/31/airtags-stalking/
[22] Chinmay Garg, Aravind Machiry, Andrea Continella, Christopher Kruegel, and Giovanni Vigna. 2021. Toward a Secure Crowdsourced Location Tracking System. *arXiv:2106.00217 [cs]* (June 2021). https://doi.org/10.1145/3448300.3467821 arXiv: 2106.00217.
[23] Google. 2017. Battery Historian. https://github.com/google/battery-historian
[24] Alexander Heinrich, Milan Stute, and Matthias Hollick. 2021. OpenHaystack: a framework for tracking personal bluetooth devices via Apple's massive find my network. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '21)*. Association for Computing Machinery, New York, NY, USA, 374–376. https://doi.org/10.1145/3448300.3468251
[25] Alexander Heinrich, Milan Stute, Tim Kornhuber, and Matthias Hollick. 2021. Who Can Find My Devices? Security and Privacy of Apple's Crowd-Sourced Bluetooth Location Tracking System. *Proceedings on Privacy Enhancing Technologies* 2021, 3 (July 2021), 227–245. https://doi.org/10.2478/popets-2021-0045
[26] Dennis Heinze, Jiska Classen, and Matthias Hollick. 2020. ToothPicker: Apple Picking in the iOS Bluetooth Stack. In *14th USENIX Workshop on Offensive Technologies (WOOT 20)*. USENIX Association. https://www.usenix.org/conference/woot20/presentation/heinze
[27] Michael Kan. 2022. 'Silent AirTags' With Speakers Removed Pop Up on Etsy, eBay. https://uk.pcmag.com/mobile-phone-accessories/138509/silent-airtags-with-speakers-removed-pop-up-on-etsy-ebay Section: Mobile Phone Accessories.
[28] Alex Kirschner and Apple Inc. 2022. An update on AirTag and unwanted tracking. https://www.apple.com/newsroom/2022/02/an-update-on-airtag-and-unwanted-tracking/
[29] Rudy Koski. 2021. Discovery of AirTag tracking device prevents double theft of truck. *FOX 7 Austin* (Dec. 2021). https://www.fox7austin.com/news/discovery-of-airtag-tracking-device-prevents-double-theft-of-truck#95
[30] Tobias Kröll, Stephan Kleber, Frank Kargl, Matthias Hollick, and Jiska Classen. 2021. ARIstoteles – Dissecting Apple's Baseband Interface. In *Computer Security – ESORICS 2021 (Lecture Notes in Computer Science)*. Springer International Publishing, Cham, 133–151. https://doi.org/10.1007/978-3-030-88418-5_7
[31] Patrick Leu, Giovanni Camurati, Alexander Heinrich, Marc Roeschlin, Claudio Anliker, Matthias Hollick, Srdjan Capkun, and Jiska Classen. 2021. Ghost Peak: Practical Distance Reduction Attacks Against HRP UWB Ranging. *CoRR* abs/2111.05313 (2021). https://arxiv.org/abs/2111.05313 arXiv: 2111.05313.
[32] Ryan Mac and Kashmir Hill. 2021. Are Apple AirTags Being Used to Track People and Steal Cars? *The New York Times* (Dec. 2021). https://www.nytimes.com/2021/12/30/technology/apple-airtags-tracking-stalking.html
[33] Jeremy Martin, Douglas Alpuche, Kristina Bodeman, Lamont Brown, Ellis Fenske, Lucas Foppe, Travis Mayberry, Erik C. Rye, Brandon Sipes, and Sam Teplov. 2019. Handoff All Your Privacy: A Review of Apple's Bluetooth Low Energy Continuity Protocol. *arXiv:1904.10600 [cs]* (June 2019). http://arxiv.org/abs/1904.10600 arXiv: 1904.10600.
[34] Travis Mayberry, Ellis Fenske, Dane Brown, Jeremy Martin, Christine Fossaceca, Erik C. Rye, Sam Teplov, and Lucas Foppe. 2021. Who Tracks the Trackers? Circumventing Apple's Anti-Tracking Alerts in the Find My Network. In *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society (WPES '21)*. Association for Computing Machinery, New York, NY, USA, 181–186. https://doi.org/10.1145/3463676.3485616
[35] Samsung Electronics Inc. 2020. Find My Mobile | Apps & Services. https://www.samsung.com/my/apps/find-my-mobile/
[36] Milan Stute, Alexander Heinrich, Jannik Lorenz, and Matthias Hollick. 2021. Disrupting Continuity of Apple's Wireless Ecosystem Security: New Tracking, DoS, and MitM Attacks on iOS and macOS Through Bluetooth Low Energy, AWDL, and Wi-Fi. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 3917–3934. https://www.usenix.org/conference/usenixsecurity21/presentation/stute
[37] Milan Stute, David Kreitschmann, and Matthias Hollick. 2018. One Billion Apples' Secret Sauce: Recipe for the Apple Wireless Direct Link Ad hoc Protocol. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18)*. Association for Computing Machinery, New York, NY, USA, 529–543. https://doi.org/10.1145/3241539.3241566

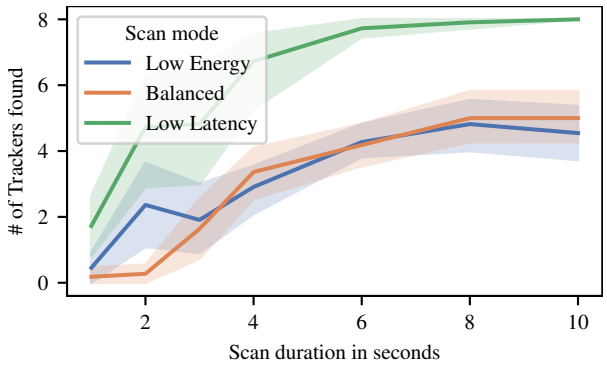


Figure 8: Evaluation of different scan parameters

[38] Milan Stute, Sashank Narain, Alex Mariotto, Alexander Heinrich, David Kreitschmann, Guevara Noubir, and Matthias Hollick. 2019. A Billion Open Interfaces for Eve and Mallory: MitM, DoS, and Tracking Attacks on iOS and macOS Through Apple Wireless Direct Link. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, USA, 37–54. https://www.usenix.org/conference/usenixsecurity19/presentation/stute
[39] The Ethics Commission of TU Darmstadt. 2019. Ethics Application TU Darmstadt. https://www.intern.tu-darmstadt.de/gremien/ethikkommisson/antragstellung/index.en.jsp
[40] Tile Inc. 2020. How Does the Tile Network Work? | Tile. https://www.thetileapp.com/en-eu/blog/what-is-tile-network-community-find-lost-stolen-far-away
[41] Mira Weller, Jiska Classen, Fabian Ullrich, Denis Waßmann, and Erik Tews. 2020. Lost and found: stopping bluetooth finders from leaking private information. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, Linz Austria, 184–194. https://doi.org/10.1145/3395351.3399422
[42] XY Labs. 2014. XY - The secure iOS/Android Bluetooth tracking tag. https://www.kickstarter.com/projects/xyfindit/xy-the-secure-tracking-tag

A *AIRGUARD* - ADDITIONAL DETAILS

A.1 BLE Scan Parameters

Android's BLE scans use two parameters that can enhance the number of scan results or save energy: The scan duration and the scan mode. To optimize our systems configuration, we tested all combinations of those parameters with ten consecutive scans. We tested a scan duration from 1 s to 10 s against all available scan modes. Figure 8 shows the results of this evaluation with eight tracking devices in range. A scan duration of 8 s using the *low latency* scan mode delivered good results. In version 1.1.2, we updated the app to use these optimized parameters at the end of January 2022. Before this update we relied on a *balanced* scan mode and 15 s duration. This combination was finding fewer devices and was keeping the app active longer, which resulted in more energy consumption. Our changes reduced the energy consumption from 0.23% to 0.13%, evaluated with the Google Battery Historian [23].

A.2 Sound on Accessories

Whenever a tracker has been detected by the iOS tracking detection or the Android App Tracker Detect, the user can play a sound on the accessory. Playing a sound on a tracker significantly simplifies the discovery of the device. The same functionality has been adopted in *AirGuard*.

Apple has not documented how this action can be triggered. The Find My network accessory specification defined a BLE Generic Attribute Profile (GATT) protocol to play and stop a sound [4].

Table 5: Characteristics to start and stop playing a sound on Find My accessories

Device	Behavior	Service	Characteristic	Value to write
AirTag	Play sound	7DFC9000-7D1C-4951-86AA-8D9728F8D66C	7DFC9001-7D1C-4951-86AA-8D9728F8D66	0xAF
Accessories & AirPods	Play sound	FD44	4F860003-943B-49EF-BED4-2F730304427A	0x01 00 03
Accessories & AirPods	Stop sound	FD44	4F860003-943B-49EF-BED4-2F730304427A	0x01 01 03

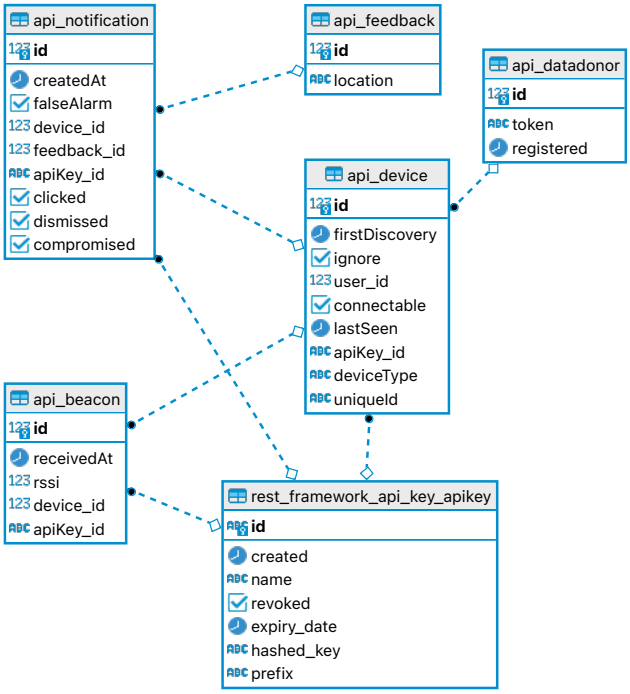


Figure 9: Database Schema for all tables that contain user generated data

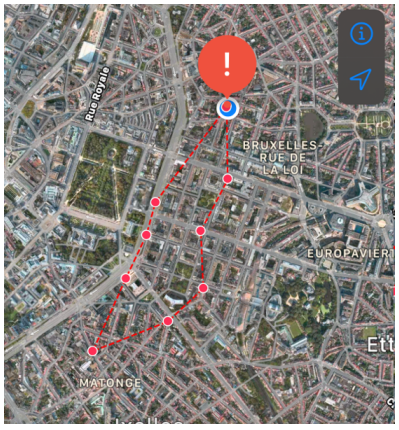


Figure 10: Screenshot of an AirTag tracking an author displayed in the Find My App.

GATT protocols consist of services which can contain multiple characteristics. In Apple's specification, two opcodes are defined to start and stop a sound if they are written to a specific characteristic. The characteristic is still available on sold Find My accessories, like the Chipolo ONE Spot, but the opcodes have changed. Also, the AirTag supports the same functionality, but it uses an entirely different BLE characteristic.

To identify the actual values of these sound characteristics, we used two methods of reverse engineering: iOS system logs and Android Bluetooth Host Controller Interface (HCI) snoop logs.

We used an AirTag that triggered a tracking notification on iOS, then we played a sound on the device and retrieved the system logs of the `locationd` daemon at the same time. Fortunately, the logs print the characteristic that needs to be accessed and the value that needs to be written. In iOS 15.2, the AirTag only supports playing a sound but not stopping it.

On Android, we activated the Bluetooth HCI snoop logs and played a sound on the Chipolo ONE Spot using the Tracker Detect app. The HCI logs contain all commands that are sent to the Bluetooth chip from the host (i.e., Android) system and the responses from the chip (e.g., GATT characteristics used). We extracted the values needed to start a sound and stop the sound. The discovered values to play and stop sounds are listed in Table 5.

A.3 Database Schema

In Figure 9 we printed our complete database schema that is used on our study server. It contains all tables, their columns and relations.

B IOS TRACKING DETECTION

Figure 10 shows a screenshot from Apple's Find My app. It shows a path constructed by several points where the iPhone has seen the AirTag as a tracking device.