

HomeScout: Anti-Stalking Mobile App for Bluetooth Low Energy Devices

Katharina O. E. Müller, Louis Bienz, Bruno Rodrigues, Chao Feng, Burkhard Stiller
Communication Systems Group CSG, Department of Informatics IfI, University of Zürich UZH
Binzmühlestrasse 14, CH—8050 Zürich, Switzerland

E-mail: [mueller;rodrigues;feng;stiller]@ifi.uzh.ch, louis.bienz@gmail.com

Abstract—Bluetooth Low Energy (BLE) personal trackers are affordable devices misused to track nonconsensual individuals. Due to the increased misuse, Apple implemented two detection applications. However, the Android application is limited to user-initiated scans with a fixed detection algorithm. This paper focuses on reducing the misuse of malicious trackers by examining current solutions, potential generic detection approaches, and improving tracker detection times.

HomeScout expands detection to the Tile and Samsung Galaxy SmartTag+, and examines the misuse potential of all BLE-enabled devices. HomeScout can reliably detect devices tracking the user as quickly as 1 minute once in motion by optimizing the parameters. The optimal parameter setting for distance is 200 m due to its high recall rate, for occurrence is 2, and for time is 1 minute. Furthermore, HomeScout applies the tracking algorithm to all BLE-enabled devices.

Index Terms—Internet-of-Things, Personal Trackers, Security, Sensing, Bluetooth Low Energy

I. INTRODUCTION

Bluetooth Low Energy (BLE) enables close-quarter communication between devices, thus facilitating indoor positioning. By detecting a specified BLE device, communication can be utilized to estimate its distance and direction from another device. Furthermore, with Apple's Find My app, it is possible to locate BLE devices not only indoors but across the globe. Consequently, allowing a plethora of misuse, as demonstrated by increasing reports, in which AirTags were used to track individuals, cars, and car owners, without their knowledge or consent [1], [2].

Apple quickly responded by implementing tracking protections on iOS to alert users of unknown AirTag in their vicinity. Additionally, the AirTag plays a sound once it is separated from its owner. However, other BLE-enabled Find My devices cannot play a sound; thus, additional tracking countermeasures are required to protect against arbitrary BLE-enabled devices. Similarly, Apple implemented the Tracker Detect application to protect Android users from potential AirTag misuse, covering all Find My compatible devices and providing instructions for AirTag deactivation. However, this solution required users to suspect and actively scan for AirTags. Thus, [3] developed AirGuard, allowing Android users to continuously scan for suspicious personal trackers in the background, representing a significant improvement over Tracker Detect.

Similarly, Samsung created the Galaxy Find network within its own ecosystem. In contrast, Tile and Chipolo create a user-based application network to locate trackers, which allows pairing third-party accessories like HP Laptops, FitBit Fitness-Trackers, and Skullcandy headphones. Hence, not only the mentioned trackers but BLE-capable devices, in general, can be misused as stalkerware.

This work investigates the trade-offs between tracker identification and privacy concerns. Thus, this paper's research question reads: Is it possible to find a single, universal, and generic way to identify personal trackers? In addition to exploring notification time improvements of the classification into malicious or non-malicious trackers, by examining the tags for additional potential evaluation parameters. Likewise, investigating whether identifying standard BLE devices, not intended but adaptable to misuse as trackers, can improve user protection against non-consensual tracking.

Thus, this work presents *HomeScout*, an Android tracker detection application that builds on the previous work accomplished by AirGuard. The *HomeScout*'s novelty lies in the extension of the tracking protection to the Tile, Galaxy SmartTag+, and all other BLE-enabled devices, the inclusion and evaluation of individually adjustable parameters to optimize the classification of trackers, the addition of scenario-based presets, the individual MAC randomization behavior analysis of tags in different states, and the respective RSSI value investigation as a further classification parameter. Thus, this work has the following contributions:

- Motion-activated tracker detection algorithm
- Adjustable parameters: distance, time, and occurrence
- User-friendly preset modes
- Samsung Galaxy SmartTag+, Tile, and general BLE device detection, identification, and classification
- Individual tracker RSSI evaluation and MAC randomization behavior analysis
- Open-source Code Repository ¹

This paper is organized as follows. Section II presents related work and background. *HomeScout*'s application design is presented in Section III. Section IV presents the prototype and parameter evaluation. Summary and future work are presented in Section V.

¹Available at: <https://github.com/LouisBienz/HomeScout>

II. RELATED WORK AND BACKGROUND

BLE can be used for localization and tracking purposes, as well as the underlying technology to facilitate them. Thus, related work on tracking BLE devices, finding networks, and detecting Apple's AirTags is complemented here with a short exploration of tracking protection and existing tracking protection solutions from AirGuard.

A. Tracking BLE Devices

BLE is used in various applications, including wildlife animal monitoring [4], smartphone-based indoor localization [5], [6], [7], indoor and outdoor localization [8], all relying on the detection and tracking of BLE devices. To prevent the misuse and involuntary tracking of persons through their devices, MAC randomization is employed, periodically changing devices' unique communication identifiers to ensure they cannot be linked to a specific device [9], [10]. Nevertheless, [11] was able to continuously single out and track two BLE devices out of 212 using only BLE advertisements. Hence, BLE localization can be reliably utilized for large tag networks, such as offline finding networks [12].

B. Finding Networks

Offline finding networks, such as Find My Network, are crowdsourced tracking systems that utilize Bluetooth to locate unpaired devices through advertisements. Consequently, [13] found that any Bluetooth-enabled device can be modified to be found within the Apple Find My Network. Nevertheless, Apple claims it cannot access the Find My Network information, deleting the content after 24 hours [14]. However, the analysis by [12] found that reporting an owner's different locations to the same finder device could allow Apple to create a social graph, and malicious macOS applications can retrieve a week of user location reports without consent.

In experiments by [3], tracking protections were tested against an AirTag, Chipolo ONE Spot, and a self-made OpenHaystack tag. The study evaluated the time-to-notification for each tracker in various scenarios, finding that iOS took significantly longer and did not consistently recognize the OpenHaystack tracker.

Existing applications do not allow users to customize tracking algorithm parameters, only targeting BLE-enabled personal trackers and not all potentially misused BLE devices, as summarized in Table I. In contrast, *HomeScout* introduces an application that detects any BLE device that could track users, and it further optimizes its tracking algorithm to provide notifications as early as possible.

C. AirTag Detection

[12] investigated the behavior of a lost Apple device, while AirGuard [3] focused on finding AirTag via the Find My Network. In both cases, localization starts with a lost Apple device without connectivity, upon which it broadcasts BLE advertisements. The advertisement shares the public advertisement key in the random address field [12]

TABLE I: Detection Application Comparison

App	OS	AirTag	Chipolo	OpenHaystack	Tile	SmartTag+	Find My Devices	All BLE Devices	Adjustable	Background
AirGuard	Android	●	●	●	○	○	○	○	○	●
Tracker Detect	Android	●	●	○	○	○	○	○	○	○
Find My	iOS	●	●	○	○	○	●	○	○	●
HomeScout	Android	●	●	○	●	●	●	●	●	●

to be compliant with the BLE format and shares the service subtype in the manufacturer-specific data field, such as AirDrop [15] or Find My Network [3]. The public key is changed every 15 minutes, as this is the recommended interval [16]. Since BLE advertisements are emitted in a two-second interval once disconnected from the Internet, the same public key can be recognized 450 times within those 15 minutes.

D. Tracking Protection

The Tile and Apple detection applications allow users to manually scan for malicious trackers, which requires the users to first, suspect they are being tracked. In contrast, *HomeScout* focuses on automatically scanning for malicious trackers in the background, currently only enabled by Apple's iOS and AirGuard's Android tracking protection. Thus, these applications and the utilized tracking algorithms are elaborated on in this subsection. [3] reverse-engineered Apple's tracking protection, finding that Apple stores all identified devices. Additionally, they identified two tracking algorithms:

1) *General Detection*: The Apple detection algorithm classifies BLE advertisements and decides whether a device is malicious in two to five minutes intervals. Criteria the device must fulfill are: a) followed for at least 10 minutes, and b) over a distance of 840 meters, considering only the advertisements received in the last 15 minutes.

2) *Visit-based detection*: Additionally, Apple utilizes its [17] objects to store a user's location for a specific period of time. However, the duration is not specified. Nevertheless, CLVisit objects contain information on the user's arrival and departure time. This combination of location and BLE advertisements facilitates detection.

Their first implementation considered all devices of the current and the last visited location, recording all associated devices. Hence, the received BLE advertisements are potentially malicious. Consequently, their second implementation introduced checks such as: a) received in the last five minutes and b) over a distance of over 420 meters. If both apply, the user is notified, speeding up detection.

E. AirGuard's Tracking Protection

AirGuard utilizes Android's WorkManager to detect the tracking. By default, these tasks can only be executed every 15 minutes. They scan for BLE devices and filter for Find

My advertisements, storing them with the current geolocation. After a scan, the app runs the detection algorithm by iterating over all devices based on their MAC address. The algorithm checks whether the advertisements of each MAC address have been received within the last 30 minutes and if this device has been fetched at least three times. Lastly, depending on the stored geolocations of the corresponding BLE advertisements, it checks if the device has been near the target user for 400 m. Their algorithm is based on the BLE MAC address, which depends on the public key, which changes once a day at 4 a.m. [18].

III. HomeScout DESIGN AND IMPLEMENTATION

HomeScout is designed to scan, identify and classify BLE trackers as unrecognized or recognized/authorized trackers. Android was selected as a platform since it offers greater control of the hardware (sensors available on the phone) via its APIs than iOS applications, which was a key requirement to enable *HomeScout*. *HomeScout*'s design followed the workflow illustrated in Figure 1 considering four element classes:

- **Services:** implements the tracking protection in three sub-services: location tracking, BLE scanning, and tracker classification.
- **Data layer:** comprises data sources from the phone's sensors and repositories
- **View models:** exposes data streams from repositories to the relevant screens *e.g.*, notifications and settings.
- **UI elements:** consists of a MainActivity and hosts the Welcome, Notifications, Settings, and Scan screens.

The first step in the workflow is to request the user on the startup screen for the necessary permissions to enable the use of the sensors, as illustrated in the Welcome fragment in Figure 1. Once the tracking protection is enabled, *HomeScout* determines whether the user is stationary in order to

initiate in parallel the tracking and classification tasks. The tracking algorithm (detailed in Subsection III-B) is activated when the user is in motion, which reduces the likelihood of false positives and battery consumption. Two parallel threads become active: BLE advertisements are scanned, identified, and stored in a local database. Secondly, the tracking algorithm classifies (detailed in Subsection III-C) the devices into malicious and non-malicious trackers. Then, the user is notified whether a malicious tracker is found, as illustrated in Figure 1 (right screenshot).

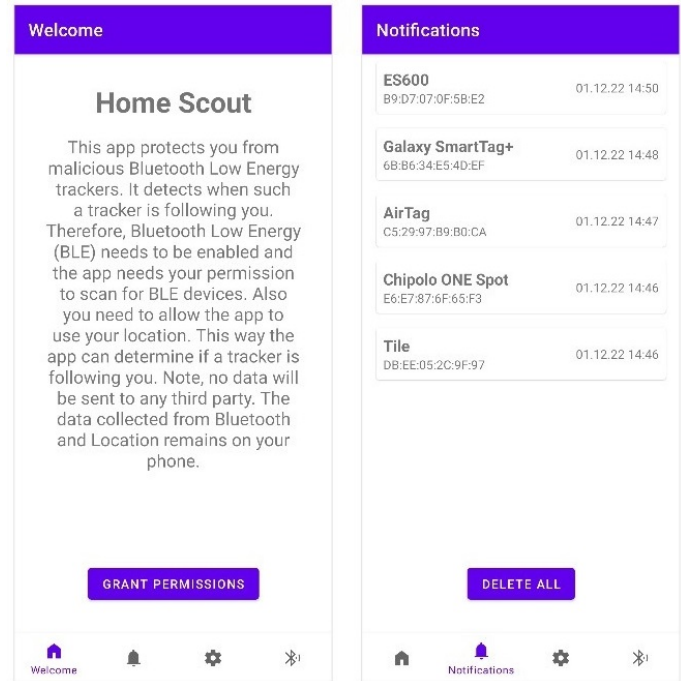


Fig. 2: *HomeScout* Screenshots: (left) Welcome Notice and Request for Permissions; (right) Notifications

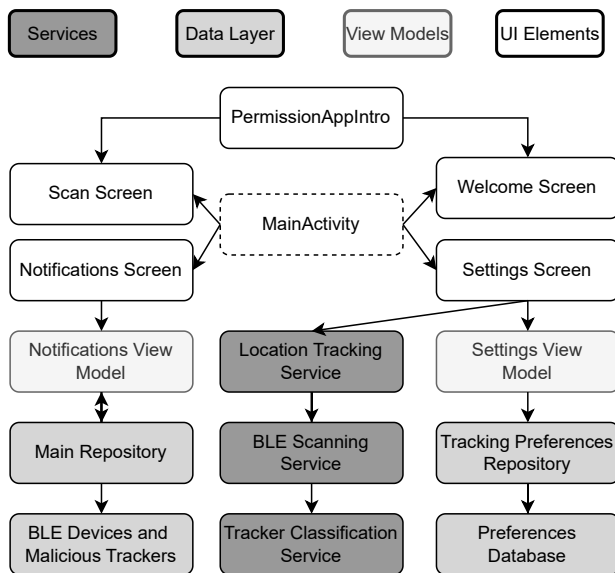


Fig. 1: *HomeScout* Basic Workflow and Classes

A. Scanning and Identification of BLE Devices

A *ScanResult* is returned from the *BluetoothLeScanner* and can be processed in the self-defined callback, handed to the *startScan()* function as a parameter. The *ScanResult* allows access to *ScanRecord* and *BluetoothDevice*. These three elements produce all the accessible information which an Android phone can receive from BLE advertisements. *HomeScout* operates as a passive scanner, so no Bond or GATT connection to any BLE device is established.

Based on the information returned by *ScanResult*, *HomeScout* assesses whether a holistic identification of BLE trackers is possible. The AirTag, Chipolo ONE Spot, Tile, and Galaxy SmartTag+ can be individually identified as trackers based on their BLE advertisements.

1) *AirTag*: The identification of AirTag was studied by [3], which identified four different device types, as listed in Table II. The interpretation and processing of the manufacturer-specific data are necessary to identify an AirTag as such. In this regard, *HomeScout* relies on

the study of AirGuard [3] to identify AirTags based on manufacturer-specific data retrieved via BLE.

2) *Chipolo ONE Spot*: The identification of the Chipolo ONE Spot works similarly to the one for the AirTag, as it is part of the Find My Network. The only difference is that the Chipolo ONE spot has a different device type according to Table II. If the bits are equal to *0b10*, the device is identified as a Chipolo ONE Spot.

3) *Galaxy SmartTag+*: Identifying a Galaxy SmartTag+ is fairly straightforward, since it adds its name to the BLE advertisement. It shows that the *scanRecord* of the Galaxy SmartTag+ contains the device's local name, "Smart Tag." If this name is found in a BLE signal's *scanRecord*, the device is identified as a Galaxy SmartTag+.

B. Tracking Algorithm

Algorithm 1 isUserStationary()

```

1: function ISUSERSTATIONARY
2:   create empty list allDistTraveled
3:   get ordered tail to head of userPosHist and store in list orderedPos
4:   secondLastIndex  $\leftarrow$  length of orderedPos - 2
5:   for i in range from 0 to secondLastIndex do
6:     create a new Location object currentLocation with latitude and longitude from orderedPos[i]
7:     create a new Location object nextLoc with latitude and longitude from orderedPos[i + 1]
8:     distLocs  $\leftarrow$  get distance between currentLocation and nextLoc
9:     add distLocs to allDistTraveled list
10:  end for
11:  totalDistTraveled  $\leftarrow$  0
12:  for all distance in allDistTraveled do
13:    totalDistTraveled  $\leftarrow$  totalDistTraveled + distance
14:  end for
15:  return  $\frac{\text{totalDistTraveled}}{\text{STATIONARY\_MOVING\_DISTANCE}} \leq 1$ 
16: end function

```

Algorithm 1 shows the function *isUserStationary()* utilized by the *LocationTrackingService* to check whether a user is stationary. It uses a pre-determined threshold defined in meters based on the path length the user has traveled in the given time, which determines whether the user is moving. *isUserStationary()* receives as inputs the user's position history and creates an empty list to store the distance traveled between each pair of positions. Then, it iterates through the list of ordered positions and calculates the distance between the current and next

TABLE II: BLE Manufacturer-specific Data [3]

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

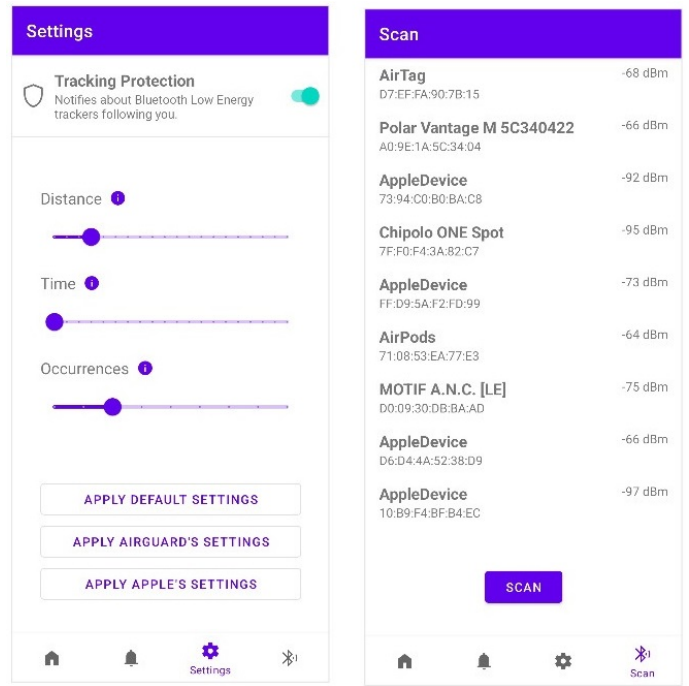


Fig. 3: HomeScout (left) Settings (right) Scan

positions using the *Location* object, which is then stored in the list created earlier (*allDistTraveled*). The function then calculates the total distance traveled by adding all the distances stored in the list. If the total distance traveled is less than or equal to the predefined constant *STATIONARY_MOVING_DISTANCE*, the function returns true, indicating that the user is stationary. Otherwise, it returns false, indicating that the user is moving.

The user path is created with a ring buffer (or circular buffer) data structure. Its size is determined to hold a user position path of approximately 2 minutes. Furthermore, the ring buffer provides an ordered trail of user positions, which returns the user's position history ordered from tail to head, which allows computing the distance between the current and consecutive position for every iteration. At the end of the loop, the computed distance is added to an array containing all distances calculated from tail to head. Finally, the function returns true or false, depending on whether the total distance traveled is smaller or equal to the constant *STATIONARY_MOVING_DISTANCE*, which is set to 50 meters.

This function optimizes energy consumption by activating and deactivating the BLE scanning services based on the user's movement. The scanning services are deactivated if the user is stationary, saving energy. This is relevant since an adversary can easily plant a tracker without detection if the user is stationary. However, if the user is moving, the scanning services are activated to detect any nearby trackers, thus preventing the adversary from tracking the user's movement.

C. Tracker Classification

The *TrackerClassificationService* is the *HomeScout*'s core, and it uses the *occurrences*, *timeInMin*, and *distance* parameters to classify a device as malicious or not (cf. Figure 3 left). The algorithm runs every 30 seconds to optimize battery consumption. The *startTrackerClassification()* function in the pseudocode is the core of the malicious tracker classification algorithm. It is designed to be adaptable based on user-defined parameters of *occurrences*, *time*, and *distance*. The function is executed every 30 seconds to classify all scanned BLE devices stored in the *hashMap-BleDevicesSortedByTime*.

Algorithm 2 *startTrackerClassification()*

```

1: function STARTTRACKERCLASSIFICATION
2:   if isServiceRunning then
3:     for each key in hashMapBLEDevices do
4:       deviceScans ← Specify MAC address scans
5:       if deviceScans.size < occurrences then
6:         continue
7:       end if
8:       newScT ← timestamp of the first scan
9:       oldScT ← timestamp of the last scans
10:      if newScT − oldScT < timeInMin × 60000 then
11:        continue
12:      end if
13:      distFollow ← 0.0
14:      for i in range from 0 to size of deviceScans − 2 do
15:        get distance between two locations
16:        distFollow ← distFollow + distBetweenLocs
17:      end for
18:      if distFollow < distance then
19:        continue
20:      end if
21:      create maliciousTracker object & add to database
22:    end for
23:    schedule startTrackerClassification to run after
    INTERVAL_TRACKER_CLASSIFICATION
24:  end if
25: end function

```

The algorithm iterates over the hashmap, where each key represents a unique MAC address, and the corresponding value is a list of all scanned BLE devices with the same MAC address, sorted by time. At least two occurrences of the same MAC address are required to state the distance and time parameters. Otherwise, the loop continues to the next key. Next, the algorithm checks whether the time criterion is fulfilled by comparing the difference in time between the youngest and oldest scans with the user-defined time parameter multiplied by 60,000 *i.e.*, converting minutes to milliseconds. If the time difference is smaller than the user-defined value, the loop continues; otherwise, the device is not classified as malicious.

Finally, the algorithm checks if the tracker followed the user for the defined distance parameter. The computation of the distance considers the previous distances between consecutive locations, and all those distances are summed up. If the summed distances are smaller than the distance value set by the user, the device is not classified as ma-

licious; otherwise, the current BLE device is stored in the database as a malicious tracker, and the user is notified instantly. If all the predetermined criteria are satisfied, the present Bluetooth Low Energy (BLE) device is classified as a malicious tracker and stored in the database. Instant notification of this classification is provided to the user, and the malicious tracker is presented on the notifications screen (cf. Figure 3 right).

IV. EXPERIMENTAL IN-FIELD EVALUATION

This Section presents the in-field experiments, which can be categorized by their focus: investigating device BLE advertisements and examining the tracking algorithm.

A. Advertisements of Tracking Devices

The analysis of the tracker's BLE advertisements was performed, while experiments regarding the tracker's states, MAC addresses, MAC randomization, and the Received Signal Strength Indicator (RSSI) accuracy had been run.

1) *Long Time Unpaired vs. Alternating Pairing State*: Two attack scenarios were tested: 1) a remote adversary with an unpaired BLE tracker, and 2) an active adversary causing the BLE tracker to alternate between paired and unpaired states. Each scenario was measured for one hour based on average Swiss commute [19] and working hours [20].

A tracker enters the lost state when it is unpaired and can be detected by anyone. To enter this state, the tracker must be paired with a smartphone and then turned off. Turning the smartphone back on transitions the tracker back into the paired state. Overall, only the Galaxy SmartTag+ randomizes its MAC address while unpaired. The AirTag, Chipolo ONE Spot, and Tile Tracker use the same MAC address. All trackers have an accuracy of about 30 m for alternating pairing states, which could place an attacker between 20 and 80 m from the victim. The Tile is the only tracker that does not randomize its MAC address, even when paired. The experiments used a distance of 50 m to observe the desired effect of alternating states.

2) *RSSI*: The experiment tested the reliability of RSSI for distance estimation in personal trackers. Despite RSSI being known as an unreliable estimator for distance, the experiment tested whether it could be useful as a rough estimation for input in tracker classification. The experiment involved placing each tracker at various distances ranging from 0 to 10 m from the scanner and measuring RSSI values in 10 iterations, then averaging them, as shown in Table III.

TABLE III: Tracker RSSI Values at Varying Distances

Results: RSSI Depending on Distance and Personal Tracker [dBm]							
Distance of Tracker from Smartphone [m]							
Personal Tracker	0 m	0.3 m	0.5 m	1.5 m	4 m	10 m	Range ([max] - [min])
AirTag	-40.7	-59.3	-65.8	-76.1	-77.7	-89.9	49.2
Chipolo ONE Spot	-45.9	-62.2	-61.0	-72.0	-82.0	-93.9	48.0
Galaxy SmartTag+	-40.0	-49.3	-51.5	-58.3	-73.0	-89.2	49.2
Tile	-32.1	-50.2	-54.0	-60.2	-64.6	-86.6	55.5
Mean [dBm]	-39.425	-55.25	-58.075	-66.65	-74.325	-89.9	50.475

Figure 4 shows the RSSI values and their mean as graphs, indicating that the signal strength decreases drastically

within the first 0.5 m and declines less with a larger distance. The results indicate that each tracker has an individual range of RSSI values, making it a disjointed contender for a fourth detection parameter.

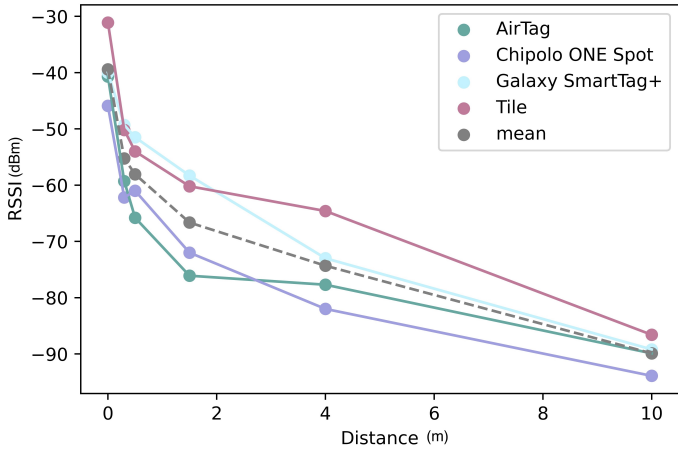


Fig. 4: Average RSSI Values per Distance

The experiment showed that RSSI is not a reliable measure for distance estimation, as it varies greatly between different trackers and does not follow a predictable decline. Therefore, it cannot be used to distinguish or identify trackers.

B. Tracking Algorithm

This experiment determines the optimal values for the three parameters that control the detection algorithm, namely the minimum time, distance, and the number of scans needed to trigger a notification for malicious trackers. The goal was to find the values that produce the most reliable results in detecting trackers early and notifying the user.

1) *Experimental Route:* Figure 5 depicts a 500 m long route divided into three parts - red, blue, and green, indicating areas with supermarkets, a bar, a public place, a train station path, a residential area with many apartments and a park, respectively. The play and flag icons depict the starting and ending points. The blue part has fewer objects blocking a BLE signal compared to the red and green parts.

2) *Experimental Procedure:* For each one of the 15 experiments, tracking protection was enabled at the starting point, carrying all four trackers in a pocket and resetting the local database. They were executed randomly, thus varying the BLE traffic and scanned devices throughout the day. The tracking protection automatically stopped once the LocationTrackingService evaluated that the user was stationary.

3) *BLE Devices in the Wild:* Figure 6 displays the run with the least scanned devices and the run with the most scanned devices, including the device types and frequency of appearance. Apple devices and devices of unknown type were the most frequent in both runs, while the Tile tracker and Galaxy SmartTag+ had the lowest occurrences. This



Fig. 5: Experimental Route in Zürich, Switzerland

suggests that foreign AirTags and Chipolo ONE Spots were scanned. Only the experimental trackers for the Galaxy SmartTag+ and Tile were observed during the tests.

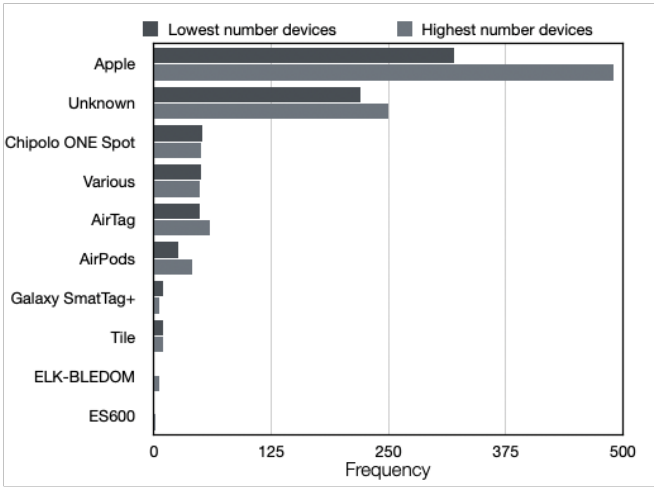


Fig. 6: Total Number of Devices per Type

4) *Testing of Tracking Preferences:* Determining whether a BLE device is a malicious tracker depends on three parameters, which are analyzed individually and iteratively to determine the appropriate values for each. In initial experiments, only the distance parameter was varied, while the number of occurrences and time preference remained constant.

In each experiment, True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN) were determined. While TP represents the four trackers that need to be identified. FP represents non-malicious BLE devices classified as malicious, and FN is when one of the four trackers is not identified. Additionally, TN represents

correctly classified as non-malicious. Lastly, precision and recall were calculated for each experiment based on these results.

Distance: *HomeScout* allows users to choose a minimum distance of 50 m and a maximum value of 1000 m. In the experiments, the distance parameter was incrementally increased by 50 m for each consecutive run, while the time and occurrences parameters remained at their lowest value of 1 minute and 2 occurrences, respectively. A total of 5 runs were conducted and presented in Table IV.

TABLE IV: Distance Variation Runs

Results: 50-250 m (Occurrence: 2, Time: 1 min)							
Distance	Results				Performance		Total Devices
	TP	FP	FN	TN	Precision	Recall	
50 m	4	23	0	470	14.8%	100%	497
100 m	4	9	0	497	30.7%	100%	510
150 m	3	3	1	470	50%	75%	477
200 m	4	0	0	471	100%	100%	475
250 m	4	0	0	520	100%	100%	524

The experiments revealed that a distance parameter of 200 m accurately distinguishes between malicious and non-malicious trackers without any FPs. Similarly, the 250 m run also produced satisfactory results. While it is possible that further increasing the distance parameter may yield similarly positive results, it would lead to an absence of detection on routes shorter than the set distance. However, reducing the distance parameter to 150 m resulted in Chipolo ONE Spot being an FN four times.

TABLE V: Distance and Time Variation Runs

Results: 200 m (Occurrence: 2, Time: 1 min)							
Run #	TP	FP	FN	TN	Precision	Recall	Total Devices
1	4	0	0	471	100%	100%	475
2	4	0	0	524	100%	100%	528
3	4	1	0	456	80%	100%	461
Results: 150 m (Occurrence: 2, Time: 2 min)							
1	4	3	0	637	57.1%	100%	644
2	4	1	0	525	80%	100%	530
3	4	3	0	437	57.1%	100%	444

Further, experiments on a 200 m distance showed a 100% recall rate, as presented in Table V. However, the last run had a precision of 80% due to an FP. An Apple device was incorrectly classified as a malicious device due to path intersection, indicating that devices can be wrongly detected as malicious even if they take a different route, as long as the number of path intersections, equals the algorithm's set number of occurrences.

Occurrences: To ensure early detection of malicious trackers, it is recommended to keep the number of occurrences as low as possible. Experiments showed that TP consistently had at least 4 occurrences. To further test this, two experiments were conducted with occurrence values of 4 and 5. Both experiments had a distance parameter set to 150 m and a time parameter set to 1 minute. These results,

with a distance parameter of 150 m and an occurrence value of 2, are summarized in Table VI.

TABLE VI: Occurrence Variation Runs

Results: 150 m (Occurrence: 2, 4, and 5, Time: 1 min)								
Run #	Occurrences	TP	FP	FN	TN	Precision	Recall	Total Devices
1	# ≥ 2	3	3	1	470	50%	75%	477
2	# ≥ 4	4	2	0	426	66%	100%	432
3	# ≥ 5	4	1	0	614	80%	100%	614
Results: 200 m (Occurrence: 4, Time: 1 min)								
1	# ≥ 4	4	0	0	461	100%	100%	465
2	# ≥ 4	4	0	0	471	100%	100%	475
3	# ≥ 4	4	0	0	567	100%	100%	571

Higher occurrences result in fewer FPs and increased precision, but setting it to 6 or more may cause delayed notification, taking up to three minutes. For better results, it is recommended to use a distance parameter of 200 m, 2 occurrences, and a 1-minute time parameter. The current design of *BluetoothScanningService* may limit higher occurrence values, but adapting its scan interval and scan period could be a possible solution.

Additionally, the recall rate decreases with a higher occurrence parameter, leading to FNs. Even when the distance is set to 200 m, increasing the occurrences parameter increases the risk of FNs, which pose a greater threat to victims than FPs. Two more experiments were conducted with 200 m for distance, 4 occurrences, and 1 minute for the time parameter.

Thus, setting the occurrences' parameter to 4 enhances the performance of the distance parameter set to 200 m, showing 100% precision and recall rates for all runs. This eliminates the FP observed when occurrences were set to 2. Moreover, the current runs scanned slightly more devices than earlier experiments, indicating that the occurrences' parameter improves the robustness of the results. However, increasing the occurrences' parameter further may result in FNs, as observed for the Tile tracker.

Time: The time parameter must strike a balance, as its increase delays user notification, however, it improves the distinction of devices passing by versus truly tracking and potentially reduce the distance parameter. Thus, three experiments with a 2-minute time parameter and a 150 m distance parameter were conducted, as shown in Table V.

All runs had a recall rate of 100%, but none had a precision rate of 100%, due to the seven FPs per run. Hence, increasing time does not sufficiently increase accuracy to allow for a lower distance parameter. Additionally, the number of Tile occurrences was at least four in all runs, which strengthens the choice of setting the occurrences' parameter to 4.

Additionally, there appears to be an order of significance to the parameters. The distance parameter on its own provides good classification results. Further, these results are fine-tuned and strengthened by the additional consideration of the occurrences' parameter. While the time parameter serves more as a restricting parameter by delaying user notification.

C. Comparison to AirGuard and iOS Results

Overall, the optimal *HomeScout* parameter settings for the pocket scenario are 200 m for distance, 4 occurrences, and 1 minute for the time parameter. Hence, allowing for notification within 1 minute if already in motion and 3 minutes from a stationary position. In comparison, AirGuard required 35 minutes and 20 seconds, and iOS detection required 1 hour 45 minutes 14 seconds for the same scenario [3]. Thus, showcasing a substantial improvement to previous solutions. Furthermore, this approach reaches the limit of BLE device advertisement intervals with a scanning period of 15 s, as BLE device advertisements are sent at varying intervals between 20 ms and 10.24 s [21], thus requiring a scanning period of at least 10.25 s to detect all potential devices.

V. SUMMARY AND FUTURE WORK

The *HomeScout* App developed improves the detection time of BLE stalkerware by 31-34 min compared to previous approaches. However, no universal method for identifying BLE trackers was found; each device must be analyzed separately. Nevertheless, this might be desirable, since a universal identification method could lead to security and privacy concerns. While easily identifiable trackers offer protection from stalking attacks, adversaries can also more easily target them, as they make locating and stealing attached items easier. Additionally, *HomeScout* shows that it is possible to discern specific BLE trackers from general BLE devices. While additional testing has to be done on detecting non-tracking BLE devices as trackers, *HomeScout* presented an adaptable tracking algorithm to detect malicious devices and notify users. The investigation of the algorithm's parameters revealed promising results, and RSSI values could potentially enhance tracking protection, even though further analysis can be conducted to further optimize detection thresholds. In addition, it was found that the AirTag, Chipolo ONE Spot, and Tile Tracker do not randomize their MAC address when out of range of their owner device for at least an hour. In contrast, the Galaxy SmartTag+ randomizes its MAC address regardless of its proximity to the owner's device. Thus, since malicious tracker detection is based on MAC addresses, BLE devices, such as the Galaxy SmartTag+, will be regarded as new devices with each randomization, leading to increased FPs. Overall, *HomeScout* contributes to protecting Android users from stalking attacks using BLE devices and provides a basis for future work to enhance tracking protection.

HomeScout has shown promising results, despite needing further improvements with its tracking algorithm. While some scenarios still need to be adapted, such as traveling with friends or using public transportation, this can be addressed in the future. Additionally, a trade-off between tracker identification and security concerns has been identified. However, one potential solution is using data fingerprinting to identify patterns in BLE advertisements for different devices. Although gathering labeled

training and test data requires time and knowledge, this determines a potential step toward better security and privacy. Additionally, the experiments with different RSSI values showed their potential use as a shield, with the RSSI value decreasing with distance from the receiver. However, further experiments are needed to determine an appropriate threshold.

REFERENCES

- [1] D. Hoffmeyer, "Airtag-Stalking: Hat Apple die Gefahr unterschätzt?" *Neue Zürcher Zeitung*, 2022, [Accessed: 21.04.2023]. [Online]: <https://www.nzz.ch/panorama/airtag-stalking-hat-apple-die-gefahr-unterschaetzt-ld.1688765?reduced=true>
- [2] S. Ulrich, "Stalking-Werkzeug: Bernerin (19) findet fremden Airtag unter ihrem Beifahrersitz - 'Ich geriet in Panik'," *20 minuten*, 2022. [Online]: <https://t.ly/29dd>
- [3] A. Heinrich, N. Bittner, and M. Hollick, "AirGuard-Protecting Android Users from Stalking Attacks by Apple Find My Devices," *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2022, pp. 26–38.
- [4] E. D. Ayele, N. Meratnia, and P. J. M. Havinga, "An asynchronous dual radio opportunistic beacon network protocol for wildlife monitoring system," *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2019, pp. 1–7.
- [5] Y. Zhuang, J. Yang, Y. Li, L. Qi, and N. El-Sheimy, "Smartphone-Based Indoor Localization with Bluetooth Low Energy Beacons," *Sensors* 2016, Vol. 16, p. 596, 4 2016. [Online]: <https://www.mdpi.com/1424-8220/16/5/596/htm>
- [6] P. Kriz, F. Maly, and T. Kozel, "Improving Indoor Localization Using Bluetooth Low Energy Beacons," *Mobile Information Systems*, Vol. 2016, 2016.
- [7] R. Giuliano, G. C. Cardarilli, C. Cesarini, L. Di Nunzio, F. Fallucchi, R. Fazzolari, F. Mazzenza, M. Re, and A. Vizzarri, "Indoor Localization System Based on Bluetooth Low Energy for Museum Applications," *Electronics* 2020, Vol. 9, p. 1055, 6 2020. [Online]: <https://www.mdpi.com/2079-9292/9/6/1055/htm>
- [8] X. Li, D. Wei, Q. Lai, Y. Xu, and H. Yuan, "Smartphone-based integrated PDR/GPS/Bluetooth pedestrian location," *Advances in Space Research*, Vol. 59, No. 3, pp. 877–887, 2 2017.
- [9] G. Kalantar, A. Mohammadi, and S. N. Sadrieh, "Analyzing the effect of Bluetooth low energy (BLE) with randomized MAC addresses in IoT applications," *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018, pp. 27–34.
- [10] J. K. Becker, D. Li, and D. Starobinski, "Tracking Anonymized Bluetooth Devices," *Proceedings on Privacy Enhancing Technologies*, Vol. 2019, No. 3, pp. 50–65, 7 2019.
- [11] M. Nikodem and M. Bawiec, "Experimental evaluation of advertisement-based bluetooth low energy communication," *Sensors*, Vol. 20, No. 1, p. 107, 2019.
- [12] A. Heinrich, M. Stute, T. Kornhuber, and M. Hollick, "Who Can Find My Devices? Security and Privacy of Apple's Crowd-Sourced Bluetooth Location Tracking System," *Proceedings on Privacy Enhancing Technologies*, pp. 227–245, 7 2021.
- [13] A. Heinrich, M. Stute, and M. Hollick, "OpenHaystack: A Framework for Tracking Personal Bluetooth Devices via Apple's Massive Find My Network," *WiSec 2021 - Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pp. 374–376, 6 2021.
- [14] Apple Incorporated, "iCloud and Find My," [Accessed: 28.03.2023]. [Online]: <https://www.apple.com/icloud/find-my/>
- [15] G. Celosia, M. Cunche, and U. Lyon, "Discontinued Privacy: Personal Data Leaks in Apple Bluetooth-Low-Energy Continuity Protocols," *Proceedings on Privacy Enhancing Technologies*, Vol. 2020, No. 1, pp. 26–46, 1 2020.
- [16] M. Woolley, "Bluetooth Core Specification v5.1," *Bluetooth*, 2019.
- [17] Apple Incorporated, "CLVisit," *Apple Developer Documentation*, [Accessed: 28.03.2023]. [Online]: <https://developer.apple.com/documentation/corelocation/clvisit>

- [18] —, “Find My Network Accessory Specification,” Vol. 1, No. 1, pp. 1–50, 2020, [Accessed: 28.03.2023]. [Online]: <https://bit.ly/FindMyNetworkAccessoryProtocolSpecification>
- [19] Swiss Federal Statistical Office, “Structural Survey Commuter mobility,” Neuchatel, Switzerland, 2021, [Accessed: 28.03.2023]. [Online]: <https://www.bfs.admin.ch/bfs/en/home/statistics/mobility-transport/passenger-transport/commuting.html>
- [20] —, “Work Volume Statistics (WV),” Neuchatel, Switzerland, p. 10, 11 2021, [Accessed: 28.03.2023]. [Online]: <https://www.bfs.admin.ch/bfs/en/home/statistics/mobility-transport/passenger-transport/commuting.html>
- [21] U. Philipp, “Ellisys Bluetooth Seminar Series,” 2020. [Online]: https://www.ellisys.com/seminars/bt_2020_bluetooth_le_rf_testing_why_not_over_the_air.pdf