

USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

PrivacyShield: Relaying BLE Beacons to Counter Unsolicited Tracking

Florian Hofhammer, *EPFL;* Daniele Antonioli, *EURECOM;* Mathias Payer, *EPFL*

https://www.usenix.org/conference/usenixsecurity26/presentation/hofhammer

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 35th USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 35th USENIX Security Symposium.

August 12–14, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-58-8

Open access to the Artifact Appendices to the Proceedings of the 35th USENIX Security Symposium is sponsored by



جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology



USENIX Security '26 Artifact Appendix: PRIVACYSHIELD: Relaying BLE Beacons to Counter Unsolicited Tracking

Florian Hofhammer
EPFL

Daniele Antonioli
EURECOM

Mathias Payer
EPFL

A Artifact Appendix

A.1 Abstract

PRIVACYSHIELD is a relay-based system to protect users against unsolicited tracking via offline finding trackers such as Apple AirTags. These offline finding trackers emit Bluetooth Low Energy (BLE) beacons that are then picked up by nearby devices such as smartphones. These devices then upload their current location as an approximation of the trackers location to the vendor's servers, from which the owner of the tracker can then retrieve it.

As such trackers are small, easy to hide, and available at low cost, they are frequently abused for malicious purposes such as stalking. PRIVACYSHIELD aims to prevent such misuse by masking a potential victim's location. To this end, PRIVACYSHIELD relays AirTag beacons via a network of relay stations that reemit the beacons at different locations than where they were originally. Consequently, third-party devices detect the beacons at these locations and report the wrong location to the vendor's servers. The AirTag's real location is thus hidden from the owner of the tracker, which in this malicious case is the perpetrator.

This artifact appendix describes our artifact for PRIVACYSHIELD. The artifact contains the source code and documentation for *(i)* the relay application submitting AirTag beacons to the PRIVACYSHIELD server, *(ii)* the relay server collecting and redistributing AirTag beacons, *(iii)* the relay firmware reemitting beacons received from the server, and *(iv)* scripts to collect location records from Apple's servers for evaluation purposes.

A.2 Description & Requirements

This section describes the software and hardware requirements to reproduce a deployment of PRIVACYSHIELD.

A.2.1 Security, privacy, and ethical concerns

As described in the Ethical Considerations section of our paper, PRIVACYSHIELD selectively hides the location of trackers from their owners. While this is intended to protect potential victims from stalking, it may also temporarily and inadvertently prevent legitimate object tracking use cases. Therefore, we recommend evaluators to take precautonary measures when recording and relaying AirTag beacons, such as notifying people in the vicinity about the ongoing experiment and adjusting the minimal signal strength threshold in the AirGuard application to only capture beacons from AirTags in the direct vicinity, i.e., AirTags that are part of the experiment.

A.2.2 How to access

PRIVACYSHIELD is available at `https://doi.org/10.5281/zenodo.17964520` and `https://github.com/HexHive/privacyshield`. We use the (stable) Zenodo reference to point to the version of the artifact as it passed the Artifact Evaluation process. The (floating) GitHub reference points to the latest version of the artifact, which at the time of writing is identical to the Zenodo version and identified by the tag sec26-artifact-evaluated, commit 11a509cfcab20feaa12e735ef701366ba88f6066.

A.2.3 Hardware dependencies

The minimal hardware requirements for evaluating general functionality of PRIVACYSHIELD are as follows:

- an AirTag,
- an Android device with Bluetooth Low Energy (BLE) support to run the AirGuard application for recording and uploading beacons,
- an Apple device (iPhone, iPad, or Mac) to pick up the beacons and upload the location records to Apple's servers,
- a second Apple device to retrieve the location records and show the AirTag's location in the Find My application,
- a computer (physical or virtual) to run the PRIVACYSHIELD server (must be accessible from the Android device running the application as well as the relay station), and
- an ESP32-C3-based development board to act as a relay station. Other ESP32 variants may also work but have not been tested.

If one of the Apple devices used is a Mac, the PRIVACYSHIELD server can also be run on the same device. This does not accurately reflect a real-world deployment but is sufficient for the sake of the experiments.

Note that the above hardware requirements are minimal and can only verify the general functionality of PRIVACYSHIELD. In order to retrieve location reports from Apple's servers for evaluation purposes outside of the Find My application, a Mac is required as one of the Apple devices. This requirement stems from the need to extract the AirTag's private key from the macOS keychain, which is not possible on iOS or iPadOS devices.

A.2.4 Software dependencies

All the steps for building binary artifacts and running the relay server are containerized. The only software requirement is therefore a working Podman or Docker installation.

Note that the (optional) evaluation scripts based on Findmy.Py require a working Python3 installation and extracting AirTag's private keys from a macOS keychain first. This step is a prerequisite for the scripts and is not part of our artifact. We point the evaluator to upstream tutorials on how to perform this step, e.g., `https://docs.mikealmel.ooo/FindMy.py/getstarted/02-fetching.html`.

A.2.5 Benchmarks

None.

A.3 Set-up

Make sure to have the aforementioned software dependencies installed.

Furthermore, if you aim to retrieve location reports from Apple's servers outside of the Find My application, make sure to correctly extract the AirTag's private keys from a macOS keychain first. We refer to upstream tutorials such as `https://docs.mikealmel.ooo/FindMy.py/getstarted/02-fetching.html` for this step. This is not required for general functionality tests but was used to collect statistical information for our evaluation.

A.3.1 Installation

First, download the artifact, e.g., via `git clone https://github.com/HexHive/privacyshield`.

Set up and run the PRIVACYSHIELD server, the modified AirGuard application, and the relay firmware as described in the `README.md` in the repository. This includes installing required Python packages, applying our patches to upstream projects where necessary, and building and running binary artifacts and scripts.

A.3.2 Basic Test

If all the components are installed and running correctly, you can perform some basic tests to verify the general functionality of PRIVACYSHIELD.

To test whether the relay application is able to pick up AirTag beacons and upload them to the PRIVACYSHIELD server, start the application on the Android device and run `adb logcat --pid=$(adb shell ps | grep seemoo | cut -d ' ' -f 8) | grep insertScanResult` on a device that has `adb` access to the Android device, e.g., the machine you built the application on. Then, bring an AirTag in the vicinity of the Android device. If everything works correctly, you should see log messages indicating that AirTag beacons were picked up.

To test whether the server can receive beacons, run it with increased logging output, e.g., via `python3 server.py -v`. To test whether the server correctly redistributes beacons to relays, you can manually submit a beacon via `curl`: `curl -L --json '"data": "tyPkKWDIHv9MABIZEE2KEXyvZ+Vfq1kYAXoD35MRrlERI44CAA==", "valid_to": "2026-02-28 00:00:00"' "http://<server address here>/api/v1/airtag"`. This should successfully add the beacon to the server's database, which you can then retrieve via `curl -vL "http://<server address here>/api/v1/airtag?valid=true"`.

Once you configured and installed the relay firmware on the microcontroller development board, you can check its log outputs via a serial connection, e.g., using `screen /dev/ttyUSB0 115200` (assuming the serial device shows up as `/dev/ttyUSB0`, replace if necessary).

Optionally, as an end-to-end test, you can verify that relaying beacons works by bringing the AirTag close to the Android device running the application, and then removing the battery from the AirTag to stop it from emitting beacons. If everything works correctly, the AirTag should still show up as providing updates in the Find My application, as the relay station now effectively turns into a clone of the original AirTag.

A.4 Evaluation workflow

This section describes the steps required to evaluate the masking functionality of PRIVACYSHIELD based on a working system as set up in the previous section.

[Mandatory for Artifacts Functional & Results Reproduced, optional for Artifact Available] This section should include all the operational steps and experiments which must be performed to evaluate if your your artifact is functional and to validate your paper's key results and claims. For that purpose, we ask you to use the two following subsections and cross-reference the items therein as explained next.

A.4.1 Major Claims

(C1): PRIVACYSHIELD effectively masks the location of AirTags from their owner by relaying their beacons at a different location. We demonstrate this in the experiments described in Section 5.3 of the paper.

(C2): PRIVACYSHIELD can mask the location successfully even when there are significantly fewer devices picking up the relayed beacons compared to the original beacons. We demonstrate that this behavior can be achieved by adjusting the frequency at which the relays emit beacons. This is shown in the experiments described in Sections 5.4 and 5.5 of the paper.

A.4.2 Experiments

(E1): [Functionality Verification] [30 human-minutes]: This experiment verifies the functionality according to claim (C1).

Preparation: Make sure the Android application, the PRIVACYSHIELD server, and the relay firmware are correctly installed and running as described in Section A.3.

Execution: Bring the experiment AirTag close to the Android device running the sniffing application. With the relay station powered on and being located physically seperate from the AirTag, check whether the AirTag shows up in the Find My application at the relay stations position instead of its real position.

Results: The AirTag should show up in the Find My application at the relay station's position, demonstrating that PRIVACYSHIELD successfully masked the AirTag's real location.

(E2): [Verification of Resilience to Adversarial Environments] [30 human-minutes]: This experiment verifies that PRIVACYSHIELD can successfully mask the location of an AirTag even when there are significantly fewer devices picking up the relayed beacons compared to the original beacons, as described in claim (C2).

Preparation: Make sure the Android application, the PRIVACYSHIELD server, and the relay firmware are correctly installed and running as described in Section A.3. Ensure that the AirTag's beacons are picked up by multiple devices, e.g., by placing it in a busy area. Ensure that the relay station is placed in an area with significantly fewer devices picking up beacons, e.g., a controlled office environment.

Execution: Bring the experiment AirTag close to the Android device running the sniffing application. With the relay station powered on and being located physically seperate from the AirTag, check whether the AirTag shows up in the Find My application at the relay stations position instead of its real position. If this is not the case, adjust the frequency at which the relay emits the relayed beacons by reducing the `BLE_ADVERTISEMENT_INTERVAL` value for the relay firmware and repeat the experiment.

Results: The AirTag should show up in the Find My application at the relay station's position after adjusting the advertisement frequency. The results should mirror those presented in Table 2 in the paper, if the environment is similar to the one used in our evaluation.

Note that the above experiments are sufficient to validate the major claims of the paper. To get more fine-grained results that back up the claims made in the paper, the evaluator can optionally retrieve the history of location reports from Apple's servers using the scripts provided in the `findmy/` subdirectory of the repository. These scripts allow precomputing the AirTag's public keys for a configurable time frame, retrieving location reports, and plotting them on a map.

Warning

Interacting with Apple's servers with an unofficial client may get your account banned. If you choose to do those fully optional steps, make sure to use a secondary Apple ID!

A.5 Notes on Reusability

Our server implementation can easily be scaled horizontally by deploying it on multiple machines. These can then either serve disjoint sets of AirTag beacons, or share a common database by replacing the current SQLite backend with a networked database such as PostgreSQL, MySQL, or similar. Furthermore, the server can be federated by submitting beacons to other servers via the provided REST API.

The relay firmware can be adapted to other microcontroller platforms leveraging their built-in BLE capabilities and APIs. The current implementation is based on the Espressif SDK, and consequently should run successfully on any ESP32 variant that supports BLE and WiFi.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2026/.