

USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

BLE Theft Auto: Evaluating the Security of Aftermarket BLE-based Automotive Remote Control Systems

Jerry Yu, Yibo Wei, Sumanth Rao, Mohak Vaswani, Jefferson Chien, Christian Dameff, Nishant Bhaskar, and Aaron Schulman, *UC San Diego*

https://www.usenix.org/conference/usenixsecurity26/presentation/yu-jerry

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
BLE Theft Auto: Evaluating the Security of Aftermarket BLE-based Automotive Remote Control Systems

Jerry Yu* Yibo Wei* Sumanth Rao Mohak Vaswani Jefferson Chien
Christian Dameff† Nishant Bhaskar Aaron Schulman
UC San Diego

A Artifact Appendix

A.1 Abstract

This artifact package accompanies the paper *"BLE Theft Auto: Evaluating the Security of Aftermarket BLE-based Automotive Remote Control Systems"*. It includes marimo notebooks, data analysis scripts, a formal proof, a scrubbed Python KARR attack implementation, a pre-built Android BLE scanner APK, and a WiGLE data collection script. The artifacts are designed to reproduce our analyses where data sharing permits, with synthetic data used when real-world datasets cannot be released.

A.2 Artifact Availability

Two artifacts are provided: a public package (under embargo until August 12, 2026) and a restricted package containing sensitive data and analysis materials that will remain access-controlled.

A.3 Description & Requirements

The public artifact includes the notebooks `ad_cluster.py` and `scale_estimation.py`, the `wigledl.py` WiGLE collection script, the `proof.lean` formal proof, the pre-built `cluetooth.apk` Android app, and the scrubbed `karr-attack/karr/` Python implementation with secret keys removed. The restricted artifact contains sensitive data and analysis materials that cannot be distributed publicly. The notebooks are intended for artifact evaluation. `proof.lean` formalizes the algebraic core of the scale-estimation argument: under independent Bernoulli sampling with per-address inclusion probability p, the probability of an observed interval of size k is $(1-p)^{k-1}p$, so $P(X=1)=p$. It does not formalize the full empirical estimator or WiGLE dataset.

The restricted artifact contains two main components. The `data/` folder includes raw advertisement data from our wardriving collection and is required to run `ad_cluster.py` in the public artifact; copy the entire `data/` folder into the same directory as the notebook. The `data-analysis/` folder contains the original scripts used to produce figures and tables in the paper; these are not intended to be reproducible because of WiGLE EULA restrictions, but can serve as a reference for the analysis process. Reproducible versions of our data analysis algorithms are provided in the public artifact as marimo notebooks.

*These authors contributed equally to this work.
†Primarily affiliated with UC San Diego Health.

A.3.1 Security, privacy, and ethical concerns

Evaluators must not attempt unauthorized access or real-world attacks. The `karr-attack/karr/` Python code in the public artifact is provided for inspection only; secret keys are removed, and we ask reviewers not to attempt to reproduce the attack or use the artifact against any vehicle or deployed system. The `ad_cluster.py` notebook requires sensitive data: raw BLE advertisement packets collected with our Cluetooth scanner app. We removed all location, timestamp, and MAC address data before inclusion; please keep this data private and do not redistribute it. The `scale_estimation.py` notebook uses synthetic data because the original wardriving dataset is restricted by WiGLE's EULA. The Android BLE scanner app uploads scanning data to our database. **Warning: if you run the app with an Internet connection, we will learn your precise location and see identifiers from nearby devices (e.g., your phone or laptop Bluetooth name, or smart TV model).** Install and run it only if you are willing to share such data. Sensitive data and analysis materials are shared only via the restricted Zenodo record, while the public artifact is also access-controlled until August 12, 2026.

A.3.2 How to access

Download the public and restricted artifact packages from Zenodo:

- **Public artifact (embargoed until August 12, 2026):** https://doi.org/10.5281/zenodo.21405664. This package contains the notebooks, pre-built Android

APK, WiGLE script, formal proof, and scrubbed Python KARR attack implementation intended for public release when the embargo lifts on August 12, 2026.
- **Restricted artifact (permanently restricted):** https://doi.org/10.5281/zenodo.20656858. This package contains sensitive data and analysis materials that will remain restricted.

A.3.3 Hardware dependencies

A 64-bit machine is required; 8+ GB of RAM is recommended for running the notebooks smoothly. The BLE scanner app requires an Android device running Android 7.0 (API 24) or newer.

A.3.4 Software dependencies

None beyond the items listed in the Set-up section below. We recommend a Linux environment for running the artifacts.

A.3.5 Benchmarks

None.

A.4 Set-up

For a more readable walkthrough, you can also consult the `public/README.md` file included in the package.

After access is granted, download the public and restricted artifact packages and unzip both archives into the same parent directory. The archives contain top-level `public/` and `restricted/` directories. Navigate to the `public` directory; all subsequent setup and execution steps assume the current working directory is `public/` unless otherwise noted.

We use `uv` (https://docs.astral.sh/uv/) for dependency and Python version management. Install it with:

```
curl -LsSf https://astral.sh/uv/install.sh
| sh
```

The pre-built Android APK is included, so Android Studio is not required for artifact evaluation. The WiGLE data collection script requires a WiGLE API key from https://api.wigle.net/; set it in the `WIGLE_API_KEY` environment variable.

A.4.1 Installation

Set up the environment from the `public` directory:

```
uv sync
```

This automatically installs the correct Python version (3.13) and all dependencies, including the marimo notebook framework. No manual virtual environment setup is needed.

Alternatively, the public artifact includes a `Dockerfile` and `docker-compose.yml`. After copying the restricted `data/` directory as described below, start the containerized notebook environment with:

```
docker compose up -build
```

Then open http://localhost:2718 and select either notebook. The scale estimation notebook can run without the restricted data directory.

Run notebooks: Open a notebook in your browser with:

```
uv run marimo edit ad_cluster.py
uv run marimo edit scale_estimation.py
```

Android app: Install the pre-built `cluetooth.apk` on your device. See the privacy notice in the Security, privacy, and ethical concerns section before running the app.

WiGLE script (optional): If you have a WiGLE account, set `WIGLE_API_KEY` in your environment to download real data. This is not required for evaluation because the notebooks already demonstrate the algorithms using synthetic data that mimics real-world MAC address allocation patterns. If you do not have a key, skip this step.

```
export WIGLE_API_KEY="your_api_key"
uv run python wigledl.py karr.jsonl
-country=US -namelike="QT %"
```

We provide this utility to collect Bluetooth Search data from WiGLE and write raw API responses as JSONL; this is a convenience for reviewers to obtain WiGLE data in the format used by our analysis, since we cannot distribute the original dataset.

Restricted data for clustering: To run `ad_cluster.py`, copy the `data` directory from the restricted artifact into the current directory:

```
cp -r ../restricted/data .
```

This makes `data/raw_ads.txt` available alongside the notebook. If the data is missing, the notebook will display an error message with instructions.

A.4.2 Basic Test

From the `public` directory, run:

```
uv run marimo edit scale_estimation.py
```

Execute all cells. Successful execution produces the expected figures and tables without errors. For the Android app, install and run `cluetooth.apk` on an Android device and verify that BLE advertisements are detected and logged by the app.

A.5 Evaluation workflow

A.5.1 Major Claims

(C1): The advertisement clustering notebook reproduces the cluster counts and BLE name-pattern summaries re-

ported in Table 1 and supports the discovery methodology described in Section 3.1 of the paper. This claim is evaluated by Experiment E1.

(C2): The scale-estimation notebook exercises the methodology in Section 6, including the sequential address-allocation behavior illustrated in Figure 6 and the estimator used to derive the deployment estimates in Table 5. The original WiGLE-derived dataset cannot be redistributed, so Experiment E2 validates the estimator workflow with synthetic data rather than reproducing the exact values in Table 5.

(C3): The Android BLE scanner app implements the BLE advertisement collection workflow described in Section 3.1 of the paper. Experiment E3 validates that the app detects and logs nearby BLE advertisements.

A.6 Experiments

(E1): *Advertisement Clustering [30 human-minutes + standard PC]:* Reproduce the clustering results for Section 3.1 using `ad_cluster.py`.
Preparation: Run `uv sync` as described in Set-up. Copy the restricted data:

```
cp -r ../restricted/data .
```

Execution: Run:

```
uv run marimo edit ad_cluster.py
```

Execute all cells in the notebook.
Results: Compare the notebook's terminal-cluster summary with Table 1 of the paper. Confirm that the cluster counts and BLE name-pattern summaries correspond to the rows reported in that table.

(E2): *Scale Estimation with Synthetic Data [30 human-minutes + standard PC]:* Validate the scale estimation workflow for Section 6 using `scale_estimation.py`.
Preparation: Run `uv sync` as described in Set-up.
Execution: Run:

```
uv run marimo edit
scale_estimation.py
```

Execute all cells. The notebook demonstrates the full workflow using synthetic data. The last few cells optionally load real WiGLE data; if you do not have a WiGLE API key, simply skip them. The synthetic data cells are sufficient to evaluate the workflow, but not to reproduce the exact WiGLE-derived scale estimates in the paper.
Results: Verify that the notebook outputs the expected synthetic scale estimates and sequential-allocation plots consistent with Figure 6 and the Section 6 methodology. These results validate the estimator used for Table 5 but do not reproduce Table 5's exact WiGLE-derived values.

(E3): *BLE Data Collection with Android App [30 human-minutes + Android device]:* Validate the BLE scanning workflow used in the paper.
Preparation: Install `cluetooth.apk` on an Android device.
Execution: Launch the app and perform BLE scanning in your environment. **Warning: running the app with Internet access will share your precise location and nearby device identifiers with us.** If you do not wish to share nearby BLE data, stop after installing the app and do not run it, per the privacy notice in the Security, privacy, and ethical concerns section.
Results: Verify that BLE advertisements are detected and logged by the app as expected.

A.7 Notes on Reusability

The notebooks use marimo, a reactive Python notebook framework that stores notebooks as plain `.py` files. This makes them easy to version-control, diff, and integrate into existing workflows. The notebooks are designed with functional programming in mind: core routines are pure, have no side effects, and avoid context dependencies to maximize reusability on new datasets. In `scale_estimation.py`, the estimator function takes a sequence of serial numbers (integers) and returns an estimated count; any incrementing identifiers representable as integers can be adapted to this estimator, not just MAC addresses. In `ad_cluster.py`, the pipeline accepts raw advertisement packets encoded as hex strings (one per line) and runs the clustering algorithm. The hierarchical clustering function is pure and composable: future researchers can provide any distance function that takes two advertisement packets and returns a float distance, then compose new clustering pipelines from those building blocks.

A.8 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/.