USENIX Security '25 Artifact Appendix: Current Affairs: A Security Measurement Study of CCS EV Charging Deployments

Marcell Szakály
University of Oxford

Sebastian Köhler
University of Oxford

Ivan Martinovic
University of Oxford

A Artifact Appendix

A.1 Abstract

Our paper investigates via measurement study the security of DC CCS EV charging stations across Europe. Our artifacts are divided into three parts: The data collection system consists of specialized hardware that connect to a CCS charging station, and software to emulate a vehicle and collect data about charging sessions. Our dataset contains both machine and human readable information about the devices we tested. Finally, we provide the data analysis code used to view and extract statistics from the dataset.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

The code of our artifacts is not expected to harm any host system, it reads and writes to files inside of artifact related directories.

Building and using our data collection hardware must be done with care, to ensure that unsafe voltages coming from EV chargers are appropriately contained and that the device does not harm chargers. Our system is designed to never enable the high voltage output of EV chargers, and it does not require connecting to any of the high voltage pins of the connector. However, for practical reasons, using the data collection system is not part of this artifact evaluation.

A.2.2 How to access

The evaluated version of our artifacts can be found permanently on Zenodo at https://zenodo.org/records/14712106. The code repositories are also available on GitHub, the data collection system at https://github.com/ssloxford/current-affairs/, and the data analysis system at https://github.com/ssloxford/current-affairs-analysis.

A.2.3 Hardware dependencies

Our dataset and analysis code use basic portable Python features, and require minimal resources (<1GB disk, <1GB RAM).

The data collection system requires specific hardware: A Raspberry Pi, a Homeplug Green PHY modem, a custom PCB for electrical signalling, and a CCS inlet. In addition, to collect data, the device must be connected to a DC CCS EV charger. It is possible to activate test mode, which removes requirements for any special hardware, but in this mode no useful experiments can be performed.

It is not possible for us to provide remote access to this hardware throughout the artifact evaluation period, since for any meaningful results it would have to be connected to an EV charger and require interacting with buttons on the charger.

A.2.4 Software dependencies

The data analysis was tested using Python 3.9 on Windows 10 and Python 3.12.3 on Kubuntu 24.04.1 LTS, but we expect it to work on other mainstream modern Linux and Windows versions. It requires Python, Jupyter, and common Python packages.

The data collection system runs on Raspbian, and requires low level hardware access. The installation process requires changing core system settings and requires root access, but the application itself does not require root. It requires Python, Java, Maven, Git, and a C compiler. The main code is written in Python 3.9, and relies on standard packages. In addition, it requires a compiled C Python module, the source code and build instructions are provided for this. Finally, a modified version of the V2Gdecoder https://github.com/FlUxIuS/V2Gdecoder application is needed to convert binary EXI messages used by CCS into XML. For this we provide a "diff" file with our modifications. The application further requires XML schema files for the protocols, which are available (after purchase) as part of the DIN 70121, ISO 15118-2 and ISO 15118-20 standards. We provide instructions on where to place these files once obtained. As part of these AE process, it is possible to start the data collection system without any of these schema files, as they are only required in later stages of the CCS protocol.

A.2.5 Benchmarks

None.

A.3 Set-up

A.3.1 Installation

Dataset For the dataset alone, download and extract `data.zip` from Zenodo. The setup script for the data analysis (see below) automatically downloads and places this in the `data` subfolder of the analysis repository.

Data analysis For the data analysis, our instructions here assume x64 Kubuntu 24.04.1 LTS and Python 3.12, see the readme file for more general instructions. The ISO image to install the operating system can be found at https://cdimage.ubuntu.com/kubuntu/releases/24.04.1/release/, and it can be used in a Virtual Machine. First, download the files: the artifact evaluated version is available on Zenodo, and will be updated with the final version.

```
# wget https://zenodo.org/records/15119591/files/
analysis.zip?download=1 -O analysis.zip
# unzip analysis.zip -d analysis
# cd analysis
```

We provide an install script to download and install all dependencies. It requires root in order to run `apt install` commands.

```
# chmod +x install.sh
# ./install.sh
```

All Python dependencies are installed in a virtual environment, this command is necessary to activate it in any new shell instance.

```
# source venv/bin/activate
```

Data collector For the data collection system, we do not expect reviewers to have the necessary hardware and time to create the full system. Instructions for doing this are described in the readme file, while here we outline minimal steps to verify compilation and partial execution, using no special hardware, Kubuntu 24.04.1 LTS and Python 3.12. First, download the files. The latest version is available on GitHub, and Zenodo will be updated with the final version.

```
# wget https://zenodo.org/records/15119591/files/
collector.zip?download=1 -O collector.zip
# unzip collector.zip -d collector
# cd collector
```

We provide an install script to download and install all dependencies. It requires root in order to run `apt install` commands.

```
# chmod +x install.sh
# ./install.sh
```

This downloads the necessary Java app, C module code, and compiles them. All Python dependencies are installed in a virtual environment, this command is necessary to activate it in any new shell instance:

```
# source venv/bin/activate
```

If running without the real hardware, it is important to disable hardware access. To do this, edit `code/utils/settings.py`, and change:

```
SKIP_BASIC = True
SKIP_SLAC = True
```

Further, edit `code/main_ev.py`, and replace `"eth0"` on line 37 with the name of an Ethernet interface to bind to (you can obtain a list from `# ifconfig`)

A.3.2 Basic Test

Dataset To test the dataset, in the data folder open `index.html` and browse the data.

Data analysis Make sure the virtual environment has been activated. To test the analysis system, in the root of the analysis folder run the file `plots.ipynb`, which should output the plots found in the paper into the `plots` folder. To run it as an application, and output plots into the `plots` folder, use:

```
# python3 -m jupyter execute plots.ipynb
```

Data collector The install script should succeed without error, indicating that everything has been compiled and downloaded successfully. Make sure the virtual environment has been activated. Start the data collector by running:

```
# python3 main_harness.py
```

and open https://localhost:8000. The server uses a self signed certificate, which most browsers prompt the user to accept. If the UI prompts "Click here to activate WS certificate" follow this link and accept the self signed certificate a second time, then return to the original page. The UI should now show the "Start" button, this confirms that the webserver is working as expected.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): We publish detailed information about our dataset, including location, serial number and photos of each charger (where known) as well as the test results for each. This is validated by E1.

(C2): Our dataset contains 325 chargers, with the composition described in Section 3.3 of the paper. These values were derived from our dataset using E2.

(C3): Our data indicates that many CCS deployments do not use TLS and use old standard versions, with exact numeric results throughout Section 4. These values were derived from our dataset using E2.

A.4.2 Experiments

(E1): *[30 human-minutes + negligible compute resources]: Validate availability of dataset*

Preparation: Install the dataset.
Execution: Open `index.html` in the dataset folder with a browser. Click onto various charger entries and look at them.
Results: Confirm that our dataset contains information about tested chargers, and results.

(E2): *[30 human-minutes + negligible compute resources]: Validate results derived from the dataset*
Preparation: Install the data analysis system following the instructions above (or using the readme file in the analysis repo).
Execution: Activate the virtual environment of the analysis repo and open the notebook:

```
# python3 -m jupyter notebook plots.ipynb
```

Run all cells in the notebook.
Results: The notebook should run without error. The first few cells of the notebook calculate the statistics claimed in Section 3.3. The next few cells calculate the results claimed in 4.2, 4.1 and 4.3, as well as generate the plots. Table 2 is saved to `output/all_data.tex`

A.5 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/usenixsec2025/.