USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

How Transparent is Usable Privacy and Security Research? A Meta-Study on Current Research Transparency Practices

Jan H. Klemmer, Juliane Schmüser, Fabian Fischer, Jacques Suray, Jan-Ulrich Holtgrave, and Simon Lenau, *CISPA Helmholtz Center for Information Security;* Byron M. Lowens, *Indiana University Indianapolis;* Florian Schaub, *University of Michigan;* Sascha Fahl, *CISPA Helmholtz Center for Information Security*

https://www.usenix.org/conference/usenixsecurity25/presentation/klemmer

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium is sponsored by USENIX.

USENIX Security '25 Artifact Appendix: How Transparent is Usable Privacy and Security Research? A Meta-Study on Current Research Transparency Practices

Jan H. Klemmer [C] Juliane Schmüser [C] Fabian Fischer [C] Jacques Suray [C]
Jan-Ulrich Holtgrave [C] Simon Lenau [C] Byron M. Lowens [*] Florian Schaub [†]
Sascha Fahl [C]

[C]CISPA Helmholtz Center for Information Security, Germany
[*]Indiana University Indianapolis, USA
[†]University of Michigan, USA

A Artifact Appendix

A.1 Abstract

To support open science and transparency, we provide several artifacts that accompany our paper. The artifacts contain the data used for our analysis (including the raw analysis results, as well as computed data like the transparency score), analysis scripts to compute the transparency score, and results such as tables and figures for the paper, and the R code for the regression, as well as the generated outputs. Moreover, we include the PDFs that detail the results of our sample size estimation.

Please refer to the contained `README.md` for a more detailed overview of which artifacts are contained in which folders and files.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Except for installing dependencies, the software does not receive/send Internet data. All computations are done locally. The code does not execute any destructive steps. There are no security, privacy, or ethical concerns or risks for the evaluators.

A.2.2 How to access

The artifacts are hosted at Zenodo and accessible via the following link: https://doi.org/10.5281/zenodo.15532982.

A.2.3 Hardware dependencies

None.

A.2.4 Software dependencies

The contained source code can be executed on any system that fulfills the following requirements:

- Python with Poetry as dependency manager (Python dependencies are listed in `code/pyproject.toml`). Poetry can be downloaded here: https://python-poetry.org/.
- Docker (Installation: https://docs.docker.com/engine/install/).

A.2.5 Benchmarks

Our dataset with analysis results is contained in the artifacts and does not need to be installed separately.

A.3 Set-up

Please confirm that the required software packages are installed on the system as stated above (i.e., Python, Poetry, and Docker), before proceeding with the following steps.

A.3.1 Installation

1. Visit https://doi.org/10.5281/zenodo.15532982 and download the artifacts. Under the tab "Files" you can choose to download the `.zip` file.
2. Unarchive the `.zip` file on your local computer.
3. Run the following commands to install Python dependencies: `cd code/` and `poetry install`
4. For the regression that we implemented in R, we are using a Docker container. To build the docker container,

execute the following commands: `cd code/R/` and then `./build-container.sh`.

A.3.2 Basic Test

- For the Python scripts: Open a shell (from the `code/` directory) in the virtual environment with `poetry shell`. Then execute `jupyter notebook`. This should open Jupyter in your browser. (If that is not the case, you can manually navigate your browser to `http://localhost:8888/`). Try to open one of the notebooks (`.ipynb`).
- For the Docker container: `docker run -it --rm containr:latest /bin/bash` (opens a shell).[1]

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): *Transparency Criteria Availability:* For 52 transparency criteria, we quantify the transparency in terms of availability. On average, 65.0% ± 11.2 of the papers' applicable criteria are available (cf. Section 6.1); in total, 63.9% of all criteria were available, 5.2% partially available, and 30.9% unavailable. We present the detailed availability for all 52 transparency criteria in Table 3.

(C2): *Transparency Score:* To assess the overall transparency of papers, we calculate a transparency score (TS). The TS per paper is on average 0.677 ± 0.103 (cf. Section 6.3.2).

(C3): *Regression:* The exploratory regression on the TS reveals a significantly positive association of transparency with a paper's length and fewer methods. Most venues do not differ significantly from the baseline (SOUPS). The model indicates no significant relation with AE, a paper's primary method, and publication year. (Exact results contained in Table 4 and Section 6.4.2.)

A.4.2 Experiments

(E1): *[Transparency Analysis] [15 human-minutes + 0 compute-hour + <1GB disk]:* Process the result dataset obtained from the manual paper annotation and calculate availabilities, transparency scores, etc.
Preparation: Open a shell (from the `code/` directory) in the virtual environment with `poetry shell`. Then execute `jupyter notebook`. This should open Jupyter in your browser. (If that is not the case, you can manually try to navigate your browser to `http://localhost:8888/`).
Execution: First, open `transparency-score.ipynb` in Jupyter and execute all cells (*"Run"*, *"Run All Cells"*). This will transform the raw annotation results stored in `transparency-analysis.tsv`, calculate the transparency score and store all results in `results.tsv`. Next, open `main.ipynb` in Jupyter and execute all cells (*"Run"*, *"Run All Cells"*). Based on `results.tsv`, this notebook contains the actual analysis, e.g., to generate some of the figures and tables.
Results: The overall availability (C1) is presented in cells 19 and 20 of `main.ipynb`, and the table written to `output/tables/criteria.tex`. The results of the transparency score (C2) are written to `data/result.tsv` (column: "Transparency Score"), and the descriptive statistics are in cell 10.
Apart from that, all computed numbers are also stored (for usage in LaTeX) as key-value-pairs in `numbers-generated.tex`.

(E2): *[Regression] [15 human-minutes + 30 compute-minutes + 6GB disk]:* Run the regression model (and related other R code).
Preparation: Build the docker container as described above.
Execution: Run `docker run -it --rm -v $(pwd)/data:/data containr:latest /bin/bash` (from the `code/` directory) to open a shell in the docker container and mount your local `data/` directory. Once you entered the container environment, run `./main` to start the analysis.
Results: You can now retrieve all results in the `data/` folder on your local machine (or inside the container at `/data`). This also includes the regression table (C3).

A.5 Notes on Reusability

Our artifacts operate on the contained dataset. Therefore, the code is reusable on other datasets containing transparency analysis data (assuming that they are in the same format as our dataset). For example, future work might analyze the same transparency criteria, store the transparency analysis data in a similar `.tsv`/`.csv` file, and run our code to compute transparency scores.

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/usenixsec2025/`.

[1] **For Mac/Apple Silicon** it might be necessary to setup emulation first.
Option A): Enable Emulation in Docker Desktop (should be enabled by default): docker.desktop settings → general → virtual machine options. Choose *Apple Virtualization framework* and activate *Rosetta*.
Option B): Set Platform Explicitly When Pulling or Running: If Docker does not find an ARM version of an image, it may fail unless you explicitly request the `amd64` version: `docker run --platform linux/amd64 <image>`