USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Double-Edged Shield: On the Fingerprintability of Customized Ad Blockers

Saiid El Hajj Chehade, *EPFL;* Ben Stock, *CISPA Helmholtz Center for Information Security;* Carmela Troncoso, *EPFL and Max-Planck Institute for Security and Privacy* (*MPI-SP*)

https://www.usenix.org/conference/usenixsecurity25/presentation/el-hajj-chehade

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: Double-Edged Shield: On the Fingerprintability of Customized Ad Blockers

Saiid El Hajj Chehade[§], Ben Stock[‡], and Carmela Troncoso[§†]

§ EPFL, † Max-Planck Institute for Security and Privacy (MPI-SP) ‡ CISPA Helmholtz Center for Information Security

A Artifact Appendix

A.1 Abstract

In this work, we present three *scriptless* attacks to fingerprint ad-blocker configurations. We show that our attacks (1) identify 84% of the fitler-lists, (2) reduce the anonymity of *privacy-aware* users, in ad-blocker forum datasets, to a median of 48 users (0.2% of the population) using only 45 rules out of 577K, (3) capture fingerprints stable against constantly updated filter-lists, and (4) resist *SoTA* detectors and mitigations.

At a high level, the associated artifacts include

1. the *Main Fingerprinting Experiment [F]*, including fingerprint algorithms, forum dataset collection, our datasets, and evaluation and stats scripts. This component reproduces results in sections 5, 6, and 7.2 of the paper;
2. the *Web Measurement Study [W]*, including ad-blocker performance benchmarking and web-feature popularity measurement over the web. This component reproduces results in 7.1 and 7.3;
3. the *PoC Honey Page [H]*, an example page implementing a version of the attacks proposed. This component verifies the functionality of our attacks.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Executing experiments from the artifacts does not pose any security risk on the evaluators, as the code (1) only manipulates data associated with the experiments (included in the `./data` directory), and (2) sends GET requests to public services and APIs (namely, GitHub) at a reasonable rate, to prevent IP blocking. We also provide Docker containers to fully isolate the experimental setup. As for running the attacks in the Honey Page, they do not pose any security risk as they execute locally on the evaluator machine and only manipulate simple DOM elements and CSS.

Regarding ethical and privacy concerns, our experiments do not pose ethical or privacy risks. First, we scrape open-source (GPL-3) ad-blocker forum posts storing publicly-available information, which is standard practice in this line of work. On top of that, we do not store personal information of users and randomize their identifiers, to prevent privacy harms. Second, our web measurements implement precautions to not overload websites, like giving ample delay between visits (30 seconds). Our crawlers do not interact with the web pages beside the initial visit.

A.2.2 How to access

All components of the artifact including both code and datasets are accessible on zenodo.org/records/14736725. We also host the experiments' code (components [F] and [W]) on GitHub at spring-epfl/flfp. We also host the exemplary PoC honey pages (component [H]) on flfp-demo.github.io. The source code is available at flfp-demo/flfp-demo-builder.

A.2.3 Hardware dependencies

We initially ran the experiments on a Linux Ubuntu Server with 240 GB of RAM and 48 cores of Intel Xeon E5-2680. We verified the code's functionality on a MacBook M2 Pro with 32 GB RAM. However, on any desktop machine with fewer resources, running the fingerprinting algorithm on the full dataset size (18K users) will take considerably more time e.g., $\geq$ 16 hours.

A.2.4 Software dependencies

The main software required to run the artifacts is Docker (available at `https://docker.com`). Other libraries are bundled in the docker image we make available. All pieces of code can run on the Linux operating system with Python 3.10 or higher installed. The code has been tested on Ubuntu 20.04 LTS. The code relating to *main fingerprinting experiment* has also been tested on MacOS 14.2. We had several issues running code on Windows, so we recommend running the code on a Unix-based system.

A.2.5 Benchmarks

None.

A.3 Set-up

A.3.1 Installation

Detailed instructions are available in the `INSTRUCTIONS.md` included with the artifacts. We provide step-by-step instructions to set up the environments here:
[F] Main Fingerprinting Experiment. The instructions are available in the repo's `README.md`. In summary,

1. Download the code from Zenodo or clone the GitHub repository, then `cd` into the `filterlist-fingerprint` directory.
2. Create a GitHub API key and add it to the `.env` file (a `.env.example` file is provided.)
3. Create `./data` directory or extract provided data.
4. Pull the Docker image
   `docker pull saiidhc/flfp:usenix`
5. Run the docker container using the command
   ```
   docker container create -it
   -env-file .env
   -v $(pwd)/data:/flfp/data
   saiidhc/flfp:usenix
   ```
6. Run experiments according to `docs/REPRODUCING.md`

[W] Web Measurement Study. After downloading the code from Zenodo or cloning the GitHub repository, then `cd` into the `measurements-adblockers` directory. Set up a Python virtual environment with the libraries in `requirements.txt`. After installing Docker, you can run the experiments directly with the commands provided in its `README.md`.
[H] PoC Honey Page. To start a local version of the honey page, make sure you have Python 3.8 or above and run the command `bash watch.sh` from the source code of the honey page repository.

In all future sections, commands mentioned regarding components [F] and [W] are relative to their respective root directories mentioned above.

A.3.2 Basic Test

[F] Main Fingerprinting Experiment. From within the docker container, run any command from `README.md` while restricting the size of the dataset. For example, you can download forum issues with `python scripts/run/issues.py forum=adguard pages_limit=1`. If successful, you should find the dataset collected in `data/issues/adguard/<timestamp>/issues_confs.csv`.
[W] Web Measurement Study. Inside the `performance/docker` folder of this codebase, run
`bash run.sh logs/../../websites_inner_pages-demo.json cpu 1 chrome`
If successful, you should find the measurements stored in `performance/docker/chrome/data`
[H] PoC Honey Page. After running `bash watch.sh`, the honey page should be available by visiting `http://localhost:8000`.

A.4 Evaluation workflow

Prior to running experiments, especially for *Main Fingerprinting Experiment*, we require scraping the user forum datasets and the filter lists with the commands provided in the repository and described in `docs/REPRODUCING.md`. This step can be susceptible to network-related failures or API rate limiting; so, to avoid these challenges, we provide pre-scraped user forum datasets and filter-list datasets in Zenodo's `data-usenix.zip`.

The claims and experiments are distributed over the artifact components as follows: [F] C1 to C5 and E1 to E5. [W] C6 and E6.

A.4.1 Major Claims

(C1): A substantial decrease in filter-rule coverage does not lead to a strong reduction in filter-list coverage, across all attacks. This is proven by experiment (E1) described in section 6.1 of the paper whose results are reported in table 3.
(C2.1): The attacks extract an entropy comparable to prior work (figuring in Table 2), between 0.44 and 0.75. The attacks uniquely identify between 14% and 20% of privacy-aware users in the datasets. This is proven by experiment (E2) described in section 6.2 of the paper whose results are reported in table 4.
(C2.2): The higher the degree of ad-blocker customization, the smaller the anonymity set of the privacy-conscious user is. This is also proven by experiment (E2) whose results are illustrated in figure 2.
(C3): The attacks can generate stable fingerprints that resist the frequent updates of filter lists by maintainers. The majority of rules usable for *general fingerprinting* remain valid through a period of 1,400 days. This is proven by experiment (E3) described in section 6.3 of the paper whose results are reported in the same section and illustrated in figure 3.
(C4): The adversary needs to control few domains (around 13) to identify almost all filter lists using *domain-specific* rules. This is proven by experiment (E4) described in section 6.4 of the paper whose results are reported in table 5.
(C5): Increasing the robustness of filter-lists to detection through forcing global rules can reduce entropy to below 0.2. The number of rules added is greater for AdGuard

than uBlock Origin (e.g., 442K and 317K respectively for CSS Animation attack) – which is manageable for uBlock Origin according to claim (C6). (C5) is proven by experiment (E5) described in section 7.2 of the paper whose results are illustrated in figure 5.

(C6): Forcing ad-blockers to use all rules, AdGuard users will suffer more from performance drawbacks – page load time and CPU performance – than uBlock Origin users. This is proven by experiment (E6) described in section 7.1 of the paper whose results are in figure 4.

A.4.2 Experiments

(E1): [Filter-list Coverage Study] [30 human-minutes + 3 compute-hour]: We evaluate how many filter-list and equivalence sets each attack and subset of rules can re-identify. The expected result is a table similar to table 3 containing the number/proportion of identifiable filter lists by each attack.
Preparation: From the repository base folder, navigate with `cd filterlist-fingerprint/`. Download filter lists and parse them with the commands in the `docs/REPRODUCING.md` Section (I), or use the pre-scraped filter lists from the ZIP. Similarly download or extract user issue dataset.
Execution: Run commands in `docs/REPRODUCING.md` sections II.1 and II.2. This will execute the fingerprinting algorithm on user datasets and filter-list datasets.
Results: Run command `python scripts/paper_stats/6_1_rule_and_list_coverage.py`. The generated tables will be outputed to the terminal and stored in `scripts/paper_stats/figures/rule_list_coverage_*_*.csv`.

(E2): [User Anonymity Study] [30 human-minutes + 6 compute-hour]: We evaluate the entropy and uniqueness of user fingerprints (targeted and general fingerprints) given our attacks and their filter-list coverage.
Preparation: Same as (E1).
Execution: Same as (E1).
Results: Run command `python scripts/paper_stats/6_2_reducing_user_anonymity.py`. The generated tables will be outputed to the terminal and stored in `scripts/paper_stats/figures/attack_stats.csv`.

(E3): [Fingerprint Stability] [30 human-minutes + 4 compute-hour]: We study how stable are rules that are essential for uniquely identifying lists. We sample the history of the filter list at various time points and identify rules that no longer exist by that time, to estimate rule life-time. We generate a time plot of the proportion of important rules that remain for the attacks over ad-blockers.
Preparation: Same as (E1) and download or extract the "github commits" data in `docs/REPRODUCING.md` Section (I).
Execution: Same as (E1) and Section II.3 from `docs/REPRODUCING.md`.
Results: Run command `python scripts/paper_stats/6_3_stability.py`. The generated figure will be outputed to the terminal and stored in `scripts/paper_stats/figures/rule_stability.csv`.

(E4): [Domain Coverage] [30 human-minutes + 4 compute-hour]: We iterate over domains appearing in domain-specific rules to maximize the number of lists newly identifiable if this domain is covered. We repeat it iteratively until either all lists are covered or no domain is useful.
Preparation: Same as (E1).
Execution: Sections II.1 and II.5 from `docs/REPRODUCING.md`.
Results: Run command `python scripts/paper_stats/6_4_domain_coverage.py`. The generated table will be outputed to the terminal and stored in `scripts/paper_stats/figures/domain_coverage.csv`.

(E5): [Iterative Robustness] [30 human-minutes + 1 compute-hour]: Simulate a game between adversary and maintainer. The maintainer enables global rules that the adversary uses to uniquely identify lists, in turn the adversary tries to find new rules to use.
Preparation: Same as (E1).
Execution: Sections II.1 and II.4 from `docs/REPRODUCING.md`.
Results: Run command `python scripts/paper_stats/6_4_iterative_robustness.py`. The generated figures will be outputed to the terminal and stored in `scripts/paper_stats/figures/iterative_robustness_*.csv`.

(E6): [Ad-blocker Performance Benchmarking] [20 human-minutes + 6 compute-hour]: Increasing the number of lists and measuring the CPU performance and load time impact on popular websites.
Preparation: From the repository base folder, navigate with `cd measurement-adblockers/`
Execution: Follow instructions in `README.md` > "Data Collection" > "Ad-blocker CPU Performance".
Results: Follow instructions in `README.md` > "Data Processing" > "Ad-blocker CPU Performance".

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/.