USENIX'23 Artifact Appendix: Is Your Wallet Snitching On You? An Analysis on the Privacy Implications of Web3

Christof Ferreira Torres
ETH Zurich

Fiona Willi
ETH Zurich

Shweta Shinde
ETH Zurich

A Artifact Appendix

This work explores the privacy implications that Web3 technologies such as decentralized applications and wallets have on users. To this end, we build a framework that measures exposure of wallet information. First, we study whether information about installed wallets is being used to track users online. We analyze Tranco's top 100K websites and find evidence that 1,325 websites run scripts to probe whether users have wallets installed in their browser. Second, we measure whether decentralized applications and wallets leak the user's unique wallet address to third-parties. We intercept the traffic of decentralized applications and wallets and find over 2000 leaks across 211 applications and more than 300 leaks across 13 wallets. This appendix details how to access our artifact (implementation of framework and our dataset) and to reproduce our results.

A.1 Abstract

Our artifact consists of source code, datasets, and scripts to generate the results of our paper. We aim for Artifacts Available, Artifacts Functional, and Results Reproduced badges. In more detail, we open-source the implementation of our framework via GitHub. We also provide our dataset of collected site snapshots on the Top 100K websites, DApps and wallet extensions, which can be utilized to reproduce the figures and tables included in our paper.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

At its core, our framework visits websites and interacts with wallet extensions automatically while recording any outgoing traffic such as HTTP requests, WebSocket requests and cookies. Hence, there may be security issues if the user provides a malicious website or wallet extension to interact with. We advise testing our framework on websites and wallet extensions that the user trusts. However, as part of our reproducibility experiments, our framework tries to crawl some of the top websites provided by Tranco. Hence, it might be that the users visit illegal websites or websites with adult content, depending on which country they reside. In terms of privacy, websites may fingerprint or track the utilization of our framework.

A.2.2 How to access

Code: The code of our artifact is available via the following GitHub repository: https://github.com/christoftorres/Web3-Privacy/commit/d5884c73dba5783ea3dc419433680596ea90e882. The repository provides a detailed `README.md` file on how to set up our framework and how to use it to reproduce our results. For artifact evaluation, please checkout the branch "artifact-review".
Data: The GitHub repository contains mainly the code. Most of the data that is necessary to reproduce our results needs to be downloaded via Zenodo: https://zenodo.org/record/8071006.

A.2.3 Hardware dependencies

Our framework has been evaluated using an Apple MacBook Pro with an Apple M1 Pro chip containing 10 cores and 32 GiB of memory. However, we also tested our framework on a machine with a 12th Gen Intel(R) Core(TM) i9-12900K containing 16 cores. We recommend using something similar. Going as low as 16 GiB of memory and 30 GiB of storage should work as well.

A.2.4 Software dependencies

Our framework has been tested on MacOS Monterey version 12.6 and on 64 bit Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-67-generic). The framework leverages Node.js, Python, and MongoDB.

A.2.5 Benchmarks

This artifact already provides all the necessary data (e.g., Tranco top 100K websites, blocklists, etc.) that is required to test its functionality and reproduce the results from our paper.

A.3 Set-up

For more details and easy to use copy and paste commands, we refer to the README.md of https://github.com/christoftorres/Web3-Privacy.

A.3.1 Installation

1. Git clone our repository: https://github.com/christoftorres/Web3-Privacy. The rest of the instructions assume you are in the project directory using a terminal window.
2. For artifact reviewers: "git checkout artifact-review"
3. Install Python3 and its dependencies:
   (a) apt-get update -q && apt-get install -y wget curl unzip software-properties-common python3-distutils python3-pip python3-apt python3-dev
   (b) python3 –version
   (c) pip3 install -r requirements.txt
4. Install Node.js and its dependencies:
   (a) curl -sL https://deb.nodesource.com/setup_18.x | bash -
   (b) apt-get update -q && apt-get install -y nodejs
   (c) node –version && npm –version
   (d) cd framework/tracker-radar-collector && npm install
   (e) cd framework/request-interceptor && npm install
5. Install MongoDB:
   (a) wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | apt-key add && echo "deb [arch=amd64,arm64] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.4 multiverse" | tee /etc/apt/sources.list.d/mongodb-org-4.4.list && apt-get update && apt-get install -y mongodb-org

A.3.2 Basic Test

We can perform the following two basic tests to test whether our framework is installed properly:

1. **Test Web3-based browser fingerprinting detection:**
   (a) cd framework/tracker-radar-collector
   (b) npm run crawl – -u "https://www.nytimes.com" -o ./data/ -f -v -d "requests,targets,apis,screenshots"
   (c) cat data/www.nytimes.com_89db.json | grep ethereum -C 10
   (d) The terminal should display "window.ethereum" along with other JavaScript properties.
   (e) In case nytimes.com does not return any results, it might be that they updated their script. In this case you can try "https://xhamster.com", however, be aware that this is a website with adult content.
2. **Test wallet address leakage detection:**
   (a) cd framework/request-interceptor
   (b) node run –interactive -u https://notional.finance/portfolio –debug verbose -w metamask-chrome-10.22.2 -t 30
   (c) cat notional.finance.json | grep 7e4abd63a7c8314cc28d388303472353d884f292
   (d) The terminal should display several entries which highlight that the wallet address is being leaked by the DApp to third-parties.

A.4 Evaluation workflow

Disclaimer. The web is constantly changing, websites may remove or add scripts from one day to the other. Hence, it might be that some websites that were found to be probing user's wallets in the past, might not be doing so anymore. Moreover, our framework is only as good as the components that is uses (e.g., TRC, Puppeteer, etc.). Thus, if Puppeteer is not able to intercept a request or if TRC is not able to load a website properly, then our framework might not be able to detect JavaScript calls to wallet APIs or detect leaks.

A.4.1 Major Claims

(C1): *There are at least 10 websites among Tranco's top 1K websites that probe whether their users have a wallet extension installed in their browser. This is proven by the experiment (E1) described in Section 4.2 whose results are reported in Table 3.*

(C2): *While most websites only probe for the window.ethereum object, there are also websites that probe for different combinations of wallet APIs. This is proven by the experiment (E1) described in Section 4.2 whose results are reported in Table 4.*

(C3): *Wallet extension probing is being performed mostly by websites categorized as adult content. This is proven by the experiment (E1) described in Section 4.2 whose results are reported in Table 5.*

(C4): *The top 10 third-party scripts that probe for wallet APIs also collect other information that is required to perform browser fingerprinting. This is proven by the experiment (E1) described in Section 4.2 whose results are reported in Table 6.*

(C5): *The combination of five popular blocklists results in 56% of the third-party scripts being blocked. This is proven by the experiment (E1) described in Section 4.2 whose results are depicted in Figure 5.*

(C6): *We detect more leaks than Winter et al. [66] due to the fact that we also analyze HTTP POST requests and WebSocket requests. This is proven by the experiment (E2) described in Section 4.3.1 whose results are reported in Table 7.*

(C7): *Infura is the most widespread third-party towards where wallet addresses are leaked. This is proven by the experiment (E3) described in Section 4.3.1 whose results are reported in Table 8.*

(C8): *Exchanges leak the most often the user's wallet address to third-parties. This is proven by the experiment (E3) described in Section 4.3.1 whose results are reported in Table 9.*

(C9): *13 out of 100 wallet extensions leak the user's wallet address to third-parties. This is proven by the experiment (E4) described in Section 4.3.2 whose results are reported in Table 10.*

A.4.2 Experiments

(E1): *[Analyze Web3-Based Browser Fingerprinting] [5 human-minutes + 5 compute-minutes]: This experiment analyses the data that was gathered through our crawl on the top 100K websites in November 2022 and parsed via our browser fingerprinting detection script.*

How to: *Performing the entire crawl from scratch on the top 100K websites would take very long and result in different results as the web keeps on changing. Therefore, we provide a dump of our MongoDB collection which already contains the data processed by our "detect_fingerprinting.py" script. The dump can be imported to analyze our findings. However, for reproducibility purposes we also provide a raw snapshot of all the requests and JavaScript calls that were collected via our crawl in November 2022.*

Preparation: *Download the browser fingerprinting datasets using "wget https://zenodo.org/record/8071006/files/browser-fingerprinting-datasets.zip && unzip browser-fingerprinting-datasets.zip && mv datasets browser-fingerprinting/ && rm browser-fingerprinting-datasets.zip" and the browser fingerprinting results using "wget https://zenodo.org/record/8071006/files/browser-fingerprinting-results.zip && unzip browser-fingerprinting-results.zip && mv results browser-fingerprinting/ && rm browser-fingerprinting-results.zip". Change the working directory using "cd browser-fingerprinting/results". Import the MongoDB dump by first creating a temporary directory using "mkdir db". Afterwards, run MongoDB locally using the temporary directory: "mongod –dbpath db" and import the collection using "mongoimport –uri="mongodb://localhost:27017/web3_privacy" –collection fingerprinting_results –type json –file fingerprinting_results.json.*

Execution: *After having imported the MongoDB dump and making sure that MongoDB is running, we can run the analysis script by first chaining our working directory using "cd browser-fingerprinting/analysis" and running the analysis script using "python3 analyze_detected_fingerprinting.py".*

Results: *The terminal will display Tables 3, 4, 5, and 6, which should be equivalent to the tables included in the paper. Moreover, the script will also output in the same directory as the analysis script a PDF file named "blocklists.pdf" which should be equivalent to Figure 5 in the paper. Please note, in order to be able to plot the file "blocklists.pdf" you are required to have LaTeX installed on your system.*

(E2): *[Analyze Wallet Address Leakage] [5 human-minutes + 5 compute-minutes]: This experiment analyzes the requests collected via our interceptor on the 66 DApps by Winter et al. and compares them to the results of Winter et al.*

How to: *Performing the entire crawl from scratch on 66 websites would take very long and result in different results as the web keeps on changing. Therefore, we provide a snapshot of all the requests that we intercepted during our crawl.*

Preparation: *Download the wallet address leakage datasets using "wget https://zenodo.org/record/8071006/files/wallet-address-leakage-datasets.zip && unzip wallet-address-leakage-datasets.zip && mv datasets wallet-address-leakage/ && rm wallet-address-leakage-datasets.zip" and the wallet address leakage results using "wget https://zenodo.org/record/8071006/files/wallet-address-leakage-results.zip && unzip wallet-address-leakage-results.zip && mv results wallet-address-leakage/ && rm wallet-address-leakage-results.zip". Change the working directory using "cd wallet-address-leakage/analysis".*

Execution: *Run the comparison script using "python3 find-leaks-and-scripts-winter-et-al.py ../results/whats_in_your_wallet/crawl ../datasets/whats_in_your_wallet".*

Results: *The terminal will display at the end Table 7, which should be equivalent to Table 7 included in the paper.*

(E3): *[Analyze Wallet Address Leakage] [5 human-minutes + 60 compute-minutes]: This experiment analyzes the requests collected via our interceptor on the DAppRadar.com dataset.*

How to: *Performing the entire crawl from scratch on DAppRadar.com dataset would take very long and result in different results as the web keeps on changing. Therefore, we provide a snapshot of all the requests that we intercepted during our crawl.*
Preparation: *Change the working directory using "cd wallet-address-leakage/analysis".*
Execution: *Run the analysis script using "python3 find-leaks-and-scripts-dapps.py".*
Results: *The terminal will display at the end Tables 8 and 9, which should be equivalent to Tables 8 and 9 included in the paper.*

(E4): *[Analyze Wallet Address Leakage] [5 human-minutes + 5 compute-minutes]: This experiment analyzes the requests collected via our interceptor on 100 popular wallet extensions.*
How to: *Performing the entire crawl from scratch on the wallet extensions dataset would take very long as it requires a large amount of manual interaction with each wallet extension. Therefore, we provide a snapshot of all the requests that we intercepted during our crawl.*
Preparation: *Change the working directory using "cd wallet-address-leakage/analysis".*
Execution: *Run the analysis script using "python3 find-leaks-and-scripts-wallet-extensions.py".*
Results: *The terminal will display at the end Table 10, which should be equivalent to Table 10 included in the paper.*

A.5 Notes on Reusability

The folder "browser-fingerprinting/datasets/tranco" contains the list of websites that have been crawled during our study. Researchers can reuse this list to try to reproduce the results or perform followup studies. The folder "browser-fingerprinting/results/crawl" contains snapshots of all the JavaScript calls and requests that we collected during our study. Researchers can reuse these snapshots to compare to reproduce our results and compare their own results to ours. Researchers can easily extend our framework to analyze other wallet APIs by modifying the files "walletSimulator.js" and "walletSimulatorWithAntiBotDetection.js" contained in "framework/tracker-radar-collector/helpers".

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20220926. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2023/.