A Artifact Appendix

A.1 Abstract

This artifact provides the steps for demonstrating the functionality of our system, minTAP, and reproducing the main results in the paper. It includes proof-of-concept implementations of both the client and the server based on the minTAP protocol. The client is implemented as a Chrome extension, while the server hosts a minTAP-compatible service in a docker container.

A.2 Artifact check-list (meta-information)

- **Data set:** We use a non-public dataset that takes about 300 MB.
- **Run-time environment:** The client requires a Chrome browser with developer mode enabled. The server requires Ubuntu 20.04 with Docker installed.
- **Metrics:** Privacy benefits (in terms of data minimized) and execution time.
- **Output:** Graph and console outputs that should closely match the results given in the original paper.
- **Experiments:** this artifact consists of three experiments: verifying minTAP functionality, replicating privacy benefits, and replicating execution time.
- **How much disk space required (approximately)?:** 500 MB.
- **How much time is needed to prepare workflow (approximately)?:** 1-2 hours.
- **How much time is needed to complete experiments (approximately)?:** 1-2 hours.
- **Publicly available (explicitly provide evolving version reference)?:** The code will be publicly available in `https://github.com/EarlMadSec/minTAP`.
- **Archived (explicitly provide DOI or stable reference)?:** `https://doi.org/10.5281/zenodo.6523010`.

A.3 Description

A.3.1 How to access

An archived version of the code is available at `https://doi.org/10.5281/zenodo.6523010`.

A.3.2 Hardware dependencies

We use an AWS EC2 `t3.large` instance, but any machine with similar hardware specifications should also work.

A.3.3 Software dependencies

The following are the software dependencies for minTAP. The provided Docker setup file will manage other dependencies.

- **Client:** Chrome 98
- **Server:** Ubuntu 20.04 with Docker 20.10 installed

A.3.4 Data sets

We did not publish the dataset.

A.3.5 Models

N/A

A.3.6 Security, privacy, and ethical concerns

N/A

A.4 Installation

Server installation. Please follow the instructions located under `Server/README.md` for building and running the docker container. The docker will set up a minTAP-compatible service on the host machine's port 5000. Make sure both inbound and outbound network traffics are allowed on this port.

Test account setup. You need to create a developer account at `https://ifttt.com/developers` and follow the steps below to register the minTAP-compatible service with IFTTT.

1. Create a new service named `mintap_service` in `https://ifttt.com/services/new` and add a new trigger based on the instructions in `Server/README.md`.
2. Go to `https://platform.ifttt.com/services/mintap_service/api` and fill the `IFTTT API URL` field with the URL path to the minTAP-compatible service (i.e., port 5000 on the server host machine).
3. Go to `https://platform.ifttt.com/services/mintap_service/api/authentication` and fill the `Authorization URL` field with `[IFTTT API URL]/mintap/auth/authorize` and the `Token URL` field with `[IFTTT API URL]/mintap/auth/token`.

Once the server information is set up, go to the following links to run the IFTTT's built-in sanity checks. If the server is set up correctly, all tests should pass.

- `https://platform.ifttt.com/services/mintap_service/api/endpoint_tests`
- `https://platform.ifttt.com/services/mintap_service/api/authentication_test`

Client installation. Please follow *Step 2* in this Chrome help page to install minTAP's browser extension. The client's source code is located inside the `Client/` folder. Note that the client does not need to be installed on the same machine as the server.

A.5 Experiment workflow

The experiment comprises 3 components:

1. **Functionality test.** The first workflow shows how our minTAP client and server integrate with IFTTT. It requires using the test account to create and modify rules in IFTTT. Click the `Personal Applets` link in IFTTT Developer Dashboard and create a new rule that uses `mintap_service` as the trigger service. You may modify the filter code inside the rule to change its behavior. Once you hover over the save button, minTAP's client will automatically fill the minTAP's related fields. See the Usage section in `Server/README.md` on how to manually trigger the rule.

2. **Analysis of privacy benefit.** The second workflow describes the procedure to reproduce the analysis in Section 7.2. Refer to the description under `rule_analysis/README.md` for detailed instructions. Due to potential privacy concerns, we did not publish the original dataset we used in the paper.

3. **Execution time.** We provide a test API to measure the latency overhead of minTAP in terms of the execution time of its additional operations. The test API can be accessed using the curl command:

   ```
   $ curl "[IFTTT API URL]/ifttt/v1/triggers/bench"
   ```

A.6 Evaluation and expected results

Following is a description of the expected results after running each workflow.

1. **Functionality test.** After the rule is run, check the rule's activity logs on IFTTT's website (by clicking the `view activity` button in the rule's page) to confirm that all unneeded trigger attributes are removed.

2. **Analysis of privacy benefit.** A plot similar to Figure 9 should be generated.

3. **Execution time.** The test API will return the average latency overhead (in seconds) over 20 runs. The value should be less than 0.03 seconds.

A.7 Version

Based on the LaTeX template for Artifact Evaluation V20220119.