ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

A Artifact Appendix

A.1 Abstract

This artifact description contains information about a prototype implementation of PRIVANALYZER. The implementation is composed of (1) a policy parser; (2) a static analyzer; and (3) a set of function summaries. Code for performance evaluation on Parcel is not provided due to non-disclosure concern.

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

- **Algorithm: Parse policy strings in LEGALEASE to disjunctive normal form.**
- **Program: LEGALEASE parser + PRIVANALYZER + a set of function summaries.**
- **Run-time environment: Ubuntu 16.04 LTS.**
- **Execution: See below.**
- **Output: Residual policy.**
- **Experiments: See below.**
- **Publicly available?: Code and example test cases are publicly available.**
- **Code licenses (if publicly available)?: MIT License.**

A.3 Description

A.3.1 How to access

The codebase can be accessed from Github https://github.com/sunblaze-ucb/privguard-artifact with commit hash b1b5f3a16af6ab5f7cb0f0737aba27dd9d76c25b. We are still actively updating the codebase. To access the newest version, please use https://github.com/sunblaze-ucb/privguard-artifact.

A.3.2 Software dependencies

To run PRIVANALYZER, python3.6 and python3.6-venv are required. Additional Python package dependencies are as follow.

- pypandoc==1.6.4
- pyparsing==3.0.0rc2
- numpy==1.19.5

A.4 Installation

The statis analyzer has been tested in Ubuntu 16.04 system. To run the static analyzer, pleaes install python3.6 and python3.6-venv using the following lines.

```
sudo apt install python3.6
sudo apt install python3.6-venv
```

To download our codebase, run

```
git clone https://github.com/sunblaze-ucb/
    privguard-artifact.git
```

Then create and activate a python3.6 virtual environment, install python packages, and set environment variables by running

```
chmod u+x path-to-repo/setup.sh
path-to-repo/setup.sh
```

A.5 Evaluation and expected results

In this codebase, we provide test scripts for the policy parser and the static analyzer separately.

Policy Parser. To test the policy parser, run

```
python path-to-repo/src/parser/policy_parser
    .py
```

and input a valid policy string (e.g. "ALLOW FILTER age >= 18 AND SCHEMA NotPHI, h2 AND FILTER gender == 'M' ALLOW (FILTER gender == 'M' OR (FILTER gender == 'F' AND SCHEMA PHI))") in Legalease. The program will output the policy translated to Python objects.

To test converting a policy into its DNF form, run

```
python path-to-repo/src/parser/policy_tree.
    py
```

Static Analyzer. We provide 5 example programs to test the static analyzer. To run these examples, use the following script with correct flag values. Please make sure your environment variable is correctly set before testing the below functionality (see setup.sh for more information).

```
python path-to-repo/src/analyze.py --
    example_id XX
```

We provide details about two examples below.

ELECTRICAL HEALTH RECORD (0): The first example loads two data files: patients/data.csv and conditions/data.csv whose policies are as below.

```
ConjunctClause(redact: NAME(None:None),
    filter: AGE [e18, einf])
ConjunctClause(filter: CONSENT [eY, eY],
    filter: DESCRIPTION [
    eViralSinusitisDisorder,
    eViralSinusitisDisorder], privacy:
    Aggregation)
```

The residual policy is

```
ConjunctClause(UNSAT)
```

which means the policy is unsatisfiable. The reason is that in the last line of the program, the DataFrame calls its groupby method which prevents any further operation to satisfy the `PRIVACY Aggregation` attribute.

TRANSACTION PREDICTION (6): The second example loads one data file: train/data.csv whose policy is as below.

```
ConjunctClause(privacy: Aggregation, redact:
    ID(None:None))
```

The program drops the ID column and trains a model on the data, so the guard policy is fully satisfied and the residual policy is

```
ConjunctClause(SAT)
```