USENIX'23 Artifact Appendix: URET: Universal Robustness Evaluation Toolkit (for Evasion)

Kevin Eykholt* IBM Research
Taesung Lee* IBM Research
Douglas Schales IBM Research
Jiyong Jang IBM Research
Ian Molloy IBM Research
Masha Zorin University of Cambridge

A Artifact Appendix

A.1 Abstract

The provided artifact contains URET as described in the paper. The tools provided are sufficient to allow users to perform custom adversarial evaluations on machine learning classifiers regardless of input domain. Specifically, this version includes input transformer definitions for the common input types (e.g., numerical, text, and categorical) as well as the binary file input type described in the paper. With respect to results reproduction, it contains the model checkpoints, evaluation data, and notebooks used to generate most of the results in Table 6.

Some components described in the paper have been purposely left out of the provided artifact for proprietary reasons:

- No implementation of the "Model Guided" algorithm. This implementation was deemed proprietary.
- No data/notebooks for the Malware experiments. The malware samples used for evaluation are proprietary and may pose a risk if improperly handled.
- No data/notebooks for the DGA experiments. The training and evaluation data are proprietary. The model code is also proprietary.

Despite these limitations, the provided artifact can be used to perform the custom evaluations described in the paper.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

There should be no security, privacy, and ethical concerns.

A.2.2 How to access

URET is an evaluation toolkit for adversarial evasion attacks. The public URET repository is accessible at https://github.com/IBM/URET. It should contain the code necessary to perform an evaluation as well as notebook examples of how to use URET. The stable URL used for Artifact Evaluation is https://github.com/IBM/URET/tree/8bd1b4f4d78ac19f026e862b31ae933983c99551.

A.2.3 Hardware dependencies

Our artifact does not have any hardware requirement. Of note, a GPU is not required to run the evaluation notebooks and pre-trained model checkpoints have been provided. We tested our artifact on an Ubuntu 18.04 system with 8 CPU cores.

A.2.4 Software dependencies

The artifact repository contains a setup script for installing the required python libraries to use URET, independent of any machine learning libraries (e.g., Tensorflow, PyTorch, etc.). However, the example notebooks require a different setup script, which is included in the artifact, as the model checkpoints were trained using older libraries. The example notebooks were tested using Python 3.8. In Section A.3., we detail the necessary steps to install the required python libraries. We recommend evaluators create a virtual environment prior to running the setup script.

A.2.5 Benchmarks

The artifact requires the 2018 Home Mortgage Disclosure Act (HMDA) dataset to run the evaluation notebooks. We have already included a copy of the dataset in the artifact.

A.3 Set-up

A.3.1 Installation

Here, we provide instructions to deploy URET and run the evaluation notebooks. This was tested using Python 3.8.

1. Clone the artifact from the stable URL: https://github.com/IBM/URET/tree/8bd1b4f4d78ac19f026e862b31ae933983c99551

*These authors contributed equally.

2. In the artifact directory, replace the existing setup.py file with the version from from notebooks/setup.py.

3. Run the setup script in the top level directory (i.e. pip install -e .) to install the evaluation libraries. It is suggested you do this in a virtual environment.

After step 3, URET should be ready for use. To reproduce most of the results in Table 6, move in to the notebooks/ directory and run HMDA_results.ipynb. We have included pre-computed adversarial examples generated from running each of the exploration algorithms described in the paper. This samples are stored in notebooks/data/HMDA_adv_samples. Note that running a generation notebook (e.g., notebooks/HMDA_random.ipynb) will overwrite the saved samples by default.

A.3.2 Basic Test

After setting up URET, the easiest method to test functionality is to run one of the adversarial generation notebooks. We recommend running notebooks/HMDA_random.ipynb as it is the fastest algorithm to run. The notebook should run without errors, though you may get some warning messages. Cell 4 should display several progress bars and text related to the model being evaluated and the adversarial success rate of the generated samples.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): *URET can be used to perform generating adversarial evasion examples for a variety of input domains and formats. This claim proven by experiment E1 and Sections 6.2 and 6.3 in the paper. Experiment E1 is described in Section 6.1 of the main paper and its results are reported in Table 6.*

(C2): *URET can generate adversarial examples for inputs containing a multiple features in the input. This claim proven by experiment E1.*

(C3): *URET can generate adversarial examples for inputs containing a single feature. This claim proven by Sections 6.2 and 6.3 in the paper.*

(C4): *URET presents several different exploration confirmations that can be selected based on user needs. Experiment E1 shows results for several exploration configurations as well as a baseline (Random) to highlight the success rate/speed tradeoff.*

A.4.2 Experiments

(E1): *[HMDA Adversarial Examples][6 Human-hours]: Generate adversarial examples for five HMDA classification models using a baseline four exploration configurations.*

Preparation: Follow the installation instructions in Section A.3.1. Once URET has been installed along with its required libraries, change to the notebooks/ directory before beginning the experiment.
Execution: We have pre-computed adversarial examples for each of the generation notebooks. If the evaluator wants to re-generate the adversarial examples, then run the following notebooks:

1. notebooks/HMDA_random.ipynb - [25 Human-minutes] This generates adversarial examples where every transformation edge is selected randomly.
2. notebooks/HMDA_brute.ipynb - [1.5-2 Human-hours] This generates adversarial examples by exploring every edge.
3. notebooks/HMDA_lookup.ipynb - [1 Human-hour] This generates adversarial examples using the lookup table algorithm. Each generation process will first compute a transformation weight lookup table followed by generation.
4. notebooks/HMDA_simanneal.ipynb - [2.5-3 Human-hours] This generates adversarial examples using the simulated annealing algorithm. The default configuration file assigns 1 sec of attack time per sample.

Of the models, we found that the random forest and multi-layer perception models require the most amount of time to generate.
Results: To generate attack success rate and transformation count results on generated adversarial examples, run notebooks/HMDA_results.ipynb. It expects that there are adversarial examples for each exploration configuration and model.
To get the per sample generation times, we divide the generation time shown in the generation notebooks by the number of samples (2000). We have noticed that the simulated annealing generation time can be sometimes longer than the specified amount.

We have provide configuration files for each of the experiments shown in Table 6 of the paper. Due to randomness, there may be some slight variation in the success rate, transformation count, and per sample generation time between adversarial example generations. The configuration files can be found in notebooks/configs/HMDA. If interested, the evaluator can alter these configuration files to try different exploration settings. For the non-simulated annealing configuration files, consider modifying the beam width (i.e., how many potential transformation candidates are considered) and beam depth (i.e., how many transformations can be applied). For simulated annealing, consider modifying the transformation parameters:

- max_transform_i_sampled - Upper limit on feature transformation applied in a single transformation step

- global_max_transforms - how many transformations can be applied

or the attack time. We note that for simulated annealing, modifying the number of transformations without increasing attack time may result in decreasing the success rate given its random exploration process.

Note that we do not include the code, models, or data for the experiments show in sections 6.2 and 6.3 in main paper. We are unable to share the relevant material due to the proprietary nature of data. Section 6.1 is the only experiment that uses entirely non-proprietary data.

A.5 Notes on Reusability

URET is intended to be an evolving set of tools that can be used to evaluate adversarial robustness of classifiers with respect to evasion. If users find the current set of modules insufficient for their needs, they are encouraged to implement their own custom modules using the common interfaces exposed by URET. Specifically, we expect users may need to customize some or all of the following component:

- Input Transformers and Subtransformers (found in uret/transformers) - For data types beyond the basic and binary types we include in URET, users will need to provide new implementations, which the exploration algorithms can use.
- Custom Loss Functions (found in uret/transformers) - URET uses two common loss types: 1) classification loss based on ground truth labels or model predictions and 2) Distance based loss function. Users that require alerted or unique loss functions (e.g., a loss based on time series input data) can define their own function to provide to the explorer during initialization.
- Dependencies - Some feature relationships need to be handled outside of the transformation interface, such as normalization of a multi-feature vector input. These dependencies can be functionally defined and specified in the URET configuration file for the explorer to enforce during example generation.

The goal of URET was to provide a basic, but easily expandable set of tools to be used for adversarial evaluations. We hope that as users customize URET for their own needs, their implementations can be integrated into the public repository to expand URET's capabilities and help other users.

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/.