ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

USENIX Security '25 Artifact Appendix: VAPD: An Anomaly Detection Model for PDF Malware Forensics with Adversarial Robustness

Side Liu
Wuhan University

Jiang Ming
Tulane University

Yilin Zhou
Wuhan University

Jianming Fu
Wuhan University

Guojun Peng
Wuhan University

A Artifact Appendix

A.1 Abstract

In this work, we propose VAPD, a reconstruction-based anomaly detection model designed with dual forensic objectives: (1) identifying PDF malware through reconstruction errors between the input and output, and (2) localizing anomalous regions within the document. We evaluate VAPD on multiple datasets, samples, achieving an accuracy of 99.54%, outperforming existing anomaly detection methods. Furthermore, we assess the robustness of VAPD using four adversarial attack frameworks in both feature space and problem space, demonstrating its strong resilience against adversarial manipulations.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

This artifact does not involve steps such as vulnerability reproduction and malicious code execution. All data has been processed, and there will be no risks to computer security and privacy.

A.2.2 How to access

The evaluation preprocessed datasets and code used in this artifact are available at `https://zenodo.org/records/15603773` and `https://zenodo.org/records/16732430`. This artifact can be accessed publicly.

A.2.3 Hardware dependencies

1. At least 250 GB of available disk space;
2. At least 32 GB of RAM;
3. A GPU of 1080 Ti or higher is recommended, with more than 11 GB of VRAM (Optional).

A.2.4 Software dependencies

1. Python 3.10 or higher;
2. CUDA 11.8 or higher (Optional);
3. Other Python package dependencies and versions can be found in requirements.txt.

A.2.5 Benchmarks

1. Contagio Baseline Dataset;
2. Baseline models such as RTM, KNN, and DeepSVDD have been placed in the artifact.

A.3 Set-up

A.3.1 Installation

Running the artifact requires installing Python, along with the corresponding Python packages and a compatible CUDA version (CUDA is optional — required for training, but GPU and CUDA may not be necessary for inference/testing).

Python First, please download Python 3.10 or a later version from the official Python website: Python 3.10.17.

Python Package Dependencies) We recommend using Virtualenv to create a virtual environment. First, install Virtualenv by running:

```
> pip3 install virtualenv
```

Then, create a virtual environment:

```
> virtualenv venv
```

Activate the virtual environment:

```
> source venv/bin/activate
```

Finally, install the required packages provided in `https://zenodo.org/records/16732430`:

```
> pip3 install –r requirements.txt
```

CUDA (Optional) If a GPU is available, you may optionally install the CUDA Toolkit from https://developer.nvidia.com/cuda-toolkit to enable GPU acceleration during training, which can significantly speed up the process.

If no GPU is available, the artifact can still be run using the CPU for testing or inference purposes.

A.3.2 Basic Test

Navigate to the *vapd_src* directory first, and execute `test.py` like the following:

```
> python test.py
```

And, it will ouput:

```
usage: test.py [-h] -dataset DATASET -label
LABEL -model MODEL -model_path MODEL_PATH
```

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): VAPD demonstrates outstanding classification performance across multiple PDF datasets (E1).

(C2): VAPD exhibits strong adversarial robustness against various adversarial attack settings (E2).

(C3): VAPD is capable of localizing anomalous objects within PDF files. (E3).

A.4.2 Experiments

The experiments are divided into three parts:

(E1): Detection on D1, D2, and D3.

(E2): Adversarial experiments against adversarial attacks.

(E3): Localization experiments within PDF.

E1. Detection on baseline and extended dataset
[5 human-minutes + 5 compute-minutes]

Navigate to the *vapd_src* directory:

```
> cd vapd_src
```

The evaluation script is test.py, and the usage of test.py is as follows:

```
usage: test.py [-h] -dataset DATASET -label
LABEL -model MODEL -model_path MODEL_PATH
```

- -dataset: the path to the preprocessed dataset file
- -label: lable file
- -model: the model to be evaluated, VAPD | KNN | AE | DeepSVDD | RTM

Please run the following command to evaluate VAPD on the D1 dataset and it will output the result on console:

```
> python test.py −−dataset ./data/D1_test.npy −−label
./data/D1_label.npy −−model VAPD −−model_path
./models/vapd_weights.pkl
```

As for D2, please run the following command:

```
> python test.py −−dataset ./data/D2_test.npy −−label
./data/D2_label.npy −−model VAPD −−model_path
./models/vapd_weights.pkl
```

As for D3, please run the following command:

```
> python test.py −−dataset ./data/D3_test.npy −−label
./data/D3_label.npy −−model VAPD −−model_path
./models/vapd_weights.pkl
```

E2. Adversarial experiments against adversarial attacks.
[2 human-minutes + 60 compute-minutes]

For the gradient descent attack, first navigate to the adversarial attack directory:

```
> cd adversarial\ attack/
```

Then simply run GradAttack.py as following:

```
> python GradAttack.py
```

This script will print the process in the console and generate a CSV file to save the final results.

For MalGAN, first navigate to the malgan directory

```
> cd adversarial\ attack/malgan
```

Then run the following command. The console will display the progress, and a figure will be generated at the end as the result.

```
> python cus_malgan.py
```

Since the evaluation of EvadeML and reverse mimicry involves setting up a distributed Cuckoo sandbox cluster, we are currently unable to provide ready-to-run containers or virtual machines. Moreover, the dynamic verification process is extremely time-consuming, with a full run potentially taking over four weeks. Therefore, we have only implemented adversarial evaluations in the feature space, which are much easier to run and reproduce.

E3. Localization experiments within PDF
[5 human-minutes + 5 compute-minutes]

We have prepared a subset of approximately 100 samples from the test set for evaluating localization. To use it, run the following command. The value after –index should be an integer between 0 and 100. After execution, the top-5 localization results will be printed to the console.

```
> python locate.py −−index 12
```

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/.