USENIX Security '25 Artifact Appendix: Sometimes Simpler is Better: A Comprehensive Analysis of State-of-the-Art Provenance-Based Intrusion Detection Systems

Tristan Bilot[123†], Baoxiang Jiang[4†], Zefeng Li[5], Nour El Madhoun[2],
Khaldoun Al Agha[1], Anis Zouaoui[3], Thomas Pasquier[5]
[1]Université Paris-Saclay, [2]LISITE, Isep,[3]Iriguard,
[4]Xi'an Jiaotong University, [5]University of British Columbia

A Artifact Appendix

A.1 Abstract

In this study [3], we developed a unified framework that consolidates eight state-of-the-art PIDSs [4–10, 12] into a single codebase. This framework supports extensive customization, allowing users to integrate components from different systems for complex analyses and ablation studies. Where possible, the original code was optimized with GPU-accelerated operations and refactored according to best coding practices to improve maintainability. Each system is configured through its own YAML file, specifying system-specific parameters. To reduce redundancy, the framework employs a pipeline system that automatically reuses previously computed components and leverages existing results when available. The framework is primarily implemented in PyTorch and PyTorch Geometric and comprises 82 Python files totaling 14,811 lines of code.

We intend this project as a living artifact and invite the community to submit new systems.

A.2 Description & Requirements

All training and evaluation were conducted on a server running Ubuntu 22.04, equipped with a 3.2GHz 16-core AMD EPYC 7343 CPU, 1024 GB of memory, and an NVIDIA GA100 GPU with 80GB of memory.

A.2.1 Security, privacy, and ethical concerns

To the best of our knowledge, this work does not raise any ethical issues. All experiments have been performed on publicly available datasets that have been acquired in an ethical manner and do not contain any sensitive information. No security mechanisms are disabled, nor are any destructive actions performed during the evaluation.

[†]Work partially completed while at the University of British Columbia.

A.2.2 How to access

Permanent link: The source code is available on Zenodo: https://zenodo.org/records/15603122.
Archived Git repository: The original code corresponding to the paper is available at https://github.com/ubc-provenance/PIDSMaker/tree/velox.
Maintained version of the framework: The updated and maintained version of the framework is available at https://github.com/ubc-provenance/PIDSMaker.
Documentation: A detailed documentation is available at https://ubc-provenance.github.io/PIDSMaker/.

A.2.3 Hardware dependencies

Access to a relatively high-end GPU is recommended to speed up experiment completion.

A.2.4 Software dependencies

The experiments were conducted in the following environment.
OS. 5.19.0-46-generic #47~22.04.1-Ubuntu
Docker. 28.1.1, build 4eba377
CUDA. 12.2
NVIDIA driver. 535.230.02

The following steps apply only if Docker is not already installed and working properly with CUDA.

1. Install Docker following the official instructions and configure it to run without sudo.
2. Install dependencies for CUDA support with Docker:

```
# Add the NVIDIA package repository
curl -fsSL https://nvidia.github.io/
    ↪ libnvidia-container/gpgkey | sudo
    ↪ gpg --dearmor -o /usr/share/keyrings
    ↪ /nvidia-container-toolkit-keyring.
    ↪ gpg
```

```
curl -s -L https://nvidia.github.io/
    ↪ libnvidia-container/stable/deb/
    ↪ nvidia-container-toolkit.list | \
  sed 's#deb␣https://#deb␣[signed-by=/usr/
      ↪ share/keyrings/nvidia-container-
      ↪ toolkit-keyring.gpg]␣https://#g' |
      ↪  \
  sudo tee /etc/apt/sources.list.d/nvidia-
      ↪ container-toolkit.list

# Update and install
sudo apt-get update
sudo apt-get install -y nvidia-container-
    ↪ toolkit

# Restart services
sudo systemctl restart docker
sudo nvidia-ctk runtime configure --runtime
    ↪ =docker
sudo systemctl restart docker
```

A.2.5 Benchmarks

We use the DARPA E3 [1], E5 [2], and OpTC [11] datasets.

A.3 Set Up

Note: If encountering issues during the installation or reproducibility of experiments, please refer to the guidelines in the archived Git repository and consider opening an issue.

A.3.1 Installation

1. Clone the Repo

```
git clone https://github.com/ubc-provenance/
    ↪ PIDSMaker.git -b velox velox
cd velox
```

2. Download Datasets

The DARPA Transparent Computing (TC) and Operationally Transparent Cyber (OpTC) datasets are large and require substantial computational resources to process. We provide pre-processed versions of these datasets, stored in a PostgreSQL database, with database dumps available for download.

Datasets Size. The sizes of each database dump are as follows: *raw* refers to the size of the dump after downloading and uncompressing the archive, while *loaded* indicates the size once loaded into the PostgreSQL table.

Downloadable Archives. We provide three files:

1. `optc_and_cadets_theia_clearscope_e3.tar`, containing all OpTC datasets, CADETS_E3, THEIA_E3, and CLEARSCOPE_E3.

Dataset	Raw (GB)	Loaded (GB)
CLEARSCOPE_E3	0.6	4.8
CADETS_E3	1.4	10.1
THEIA_E3	1.1	12
CLEARSCOPE_E5	6.2	49
CADETS_E5	36	276
THEIA_E5	5.8	36
OPTC_H051	1.7	7.7
OPTC_H_501	1.5	6.7
OPTC_H201	2.0	9.1

Table 1: Raw and loaded sizes of DARPA TC and OpTC dataset dumps.

2. `theia_clearscope_e5.tar`, containing `THEIA_E5` and `CLEARSCOPE_E5`.

3. `cadets_e5.dump` is the dump for CADETS_E5 (very large).

Steps.

1. Download the archive(s) into a new `data` folder from Google Drive. On the command line, use `curl` with an authorization token (see Stack Overflow for details):

   - Visit the OAuth 2.0 Playground at developers.google.com/oauthplayground.
   - In the `Select the Scope` box, paste `https://www.googleapis.com/auth/drive.readonly`.
   - Click `Authorize APIs`, then `Exchange authorization code for tokens`.
   - Copy the **Access token**.
   - Run the following in a terminal:

   Note. Each `curl` call may download only part of the file before pausing. When the loading bar stops, press `Ctrl+C` and rerun the command until the download completes.

```
mkdir data && cd data

# OpTC and E3 Datasets
curl -H "Authorization:␣Bearer␣
    ↪ ACCESS_TOKEN" -C - https://www.
    ↪ googleapis.com/drive/v3/files/11
    ↪ YVPAuWfeEqC_zV8KD0gNrnEPbHf2Y4M?alt=
    ↪ media -o
    ↪ optc_and_cadets_theia_clearscope_e3.
    ↪ tar

# THEIA and CLEARSCOPE E5 Datasets
```

```
curl -H "Authorization:␣Bearer␣
    ↪ ACCESS_TOKEN" -C - https://www.
    ↪ googleapis.com/drive/v3/files/1
    ↪ DfolzEa3PVz_6fGZUNEUm0sBP42LB7_1?alt
    ↪ =media -o theia_clearscope_e5.tar

# CADETS E5 Dataset
curl -H "Authorization:␣Bearer␣
    ↪ ACCESS_TOKEN" -C - https://www.
    ↪ googleapis.com/drive/v3/files/1
    ↪ Xiq7w0Ofz4jZG2PVFuNqi_i0fm28kRcT?alt
    ↪ =media -o cadets_e5.dump
```

2. Uncompress the archives (requires minimal additional space):

```
tar -xvf
    ↪ optc_and_cadets_theia_clearscope_e3.
    ↪ tar
tar -xvf theia_clearscope_e5.tar
```

3. Load Databases

We create two containers: one for the PostgreSQL database and another to execute the experiments.

1. Set your paths in `.env`:

```
cp .env.local .env
```

In `.env`, set `INPUT_DIR` to the `data` folder path. Optionally, set `ARTIFACTS_DIR` to the folder for generated files (multiple GBs). Then run:

```
source .env
```

2. Build and start the database container:

```
docker compose -p postgres -f compose-
    ↪ postgres.yml up -d --build
```

Note: Update environment variables using `source .env` after modifying `.env` before running `docker compose`.

3. In a terminal, access a shell in the container:

```
docker compose -p postgres exec postgres
    ↪ bash
```

4. If you have sufficient storage (135 GB), load all databases:

```
./scripts/load_dumps.sh
```

For limited storage, load databases individually:

```
pg_restore -U postgres -h localhost -p 5432
    ↪  -d DATASET /data/DATASET.dump
```

5. Once loaded, exit the container:

```
exit
```

4. Get into the PIDSMaker Container

The `pids` container is used for development and to run experiments.

1. For VSCode users, use the Dev Containers extension. Install the extension, then press `Ctrl+Shift+P` and select *Dev Containers: Open Folder in Container*.

2. Alternatively, load the container manually:

```
docker compose -f compose-pidsmaker.yml up
    ↪ -d --build
docker compose exec pids bash
```

The Python environment and framework are installed in this container. Now download the weights necessary to reproduce the Velox results.

```
pip install gdown
cd weights

# Neural net encoder weights
gdown https://drive.google.com/drive/folders/18
    ↪ vc5JuMwUY2TmrlONG3xyPdh9NfBXyAD --folder

# Word2vec weights
gdown https://drive.google.com/drive/folders/1w
    ↪ -QoUYsnAKVVMLnLfxQ2qS1W5--1kUaD --folder
```

A.3.2 Basic Test

Once you have a shell in the `pids` container, execute the following commands:

```
cd scripts
./run_local.sh velox THEIA_E3 --tuned
```

A.4 Evaluation workflow

A.4.1 Major Claims

The paper is experiment-heavy. It represents a high computational cost (a total of 453 days of computation on our hardware). Consequently, we do not expect you to reproduce all results presented in the paper. Instead we focus on the two following major claims:

(C1): A simpler ML architecture can achieve state-of-the-art performance on a number of common benchmark. This is demonstrated in Table 4, 5, and 6 via the ADP Best metric, and can be reproduced by the experiment E1.

(C2): We notice a high instability in the results. Training the system from scratch will lead to significantly different results across runs. This is discussed in §4, SC5. This is demonstrated in Table 4, 5, and 6 via the $\tilde{\sigma}_{\text{ADP}}$ metric, and can be reproduced by the experiment E2.

A.4.2 Experiments

Due to high computation cost, we focus on reproducing results from `velox`, `orthrus` and `nodlink` shown to have reasonable computational cost (see §5.1, Figure 13).

(E1): [5 human-minutes + few hours of compute + $\sim$ 300 GB disk]:

How to: We provide a single script (run_local.sh) that reproduces the final experimental results (Tables 4, 5, 6) presented in the paper. Based on claim C1, we only want to reproduce the best ADP metrics, which indicate the best-case detection capability of a system.

Preparation: Follow steps described in §A.3.

Execution: To reproduce best ADP scores, we provide the pre-computed weights, as training can take up to multiple days:

```
./run_local.sh velox DATASET --from_weights
    ↪ --tuned
```

- Replace DATASET with one of the following: `THEIA_E3` `CADETS_E3`, `CLEARSCOPE_E3`, `CLEARSCOPE_E5`, `CADETS_E5`, `THEIA_E5`, `optc_h201`, `optc_h501`, or `optc_h051`.

Results: Compare the `adp_score` metric with the associated value in Tables 4, 5, 6. Note that the ADP scores might be slightly different due to important instability, as demonstrated in experiment E2.

(E2): [5 human-minutes + several days of compute + $\sim$ 1TB disk]:

How to: Based on claim C2, each run spans for five iterations and the Mean, Min and Best metrics are printed, along with the standard deviation. For C2, we only focus on the standard deviation to measure the instability of systems. This is measured by the $\tilde{\sigma}_{\text{ADP}}$ metric.

Preparation: Follow steps described in §A.3.

Execution: The goal is to reach high $\tilde{\sigma}_{\text{ADP}}$ scores, as in Tables 4, 5, 6:

```
./run_local.sh {system} DATASET --
    ↪ experiment=run_n_times --tuned
```

- Replace `{system}` with one of the following: `velox`, `orthrus`, and `nodlink` (additionally `thretrace`, `kairos`, `rcaid`, and `flash` are also available).

Example with `velox`:

```
./run_local.sh velox THEIA_E3 --experiment=
    ↪ run_n_times --tuned
./run_local.sh velox CADETS_E3 --experiment
    ↪ =run_n_times --tuned
./run_local.sh velox CLEARSCOPE_E3 --
    ↪ experiment=run_n_times --tuned
```

Results: Compare the `adp_score_std_rel` metric with $\tilde{\sigma}_{\text{ADP}}$ in Tables 4, 5, 6. Here we show that all systems are subject to instability. This means the measured performance of a system differs significantly from one run to another. Consequently, **exact reproduction of the original results is very unlikely**. This is not a bug and is indeed the expected behavior (see §4, SC5).

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/.

References

[1] Transparent Computing Engagement 3 Data Release, Accessed 10th August 2025. https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md.

[2] Transparent Computing Engagement 5 Data Release, Accessed 10th August 2025. https://github.com/darpa-i2o/Transparent-Computing.

[3] Tristan Bilot, Baoxiang Jiang, Zefeng Li, Nour El Madhoun, Khaldoun Al Agha, Anis Zouaoui, and Thomas Pasquier. Sometimes Simpler is Better: A Comprehensive Analysis of State-of-the-Art Provenance-Based Intrusion Detection Systems. In *Security Symposium (USENIX Sec'25)*. USENIX, 2025.

[4] Zijun Cheng, Qiujian Lv, Jinyuan Liang, Yan Wang, Degang Sun, Thomas Pasquier, and Xueyuan Han. Kairos: Practical Intrusion Detection and Investigation using Whole-system Provenance. In *Symposium on Security and Privacy (S&P'24)*. IEEE, 2023.

[5] Akul Goyal, Gang Wang, and Adam Bates. R-CAID: Embedding Root Cause Analysis within Provenance-based Intrusion Detection. In *Symposium on Security and Privacy (S&P'24)*. IEEE, 2024.

[6] Xueyuan Han, Xiao Yu, Thomas Pasquier, Ding Li, Junghwan John Rhee, James W. Mickens, Margo I. Seltzer, and Haifeng Chen. SIGL: Securing Software

Installations Through Deep Graph Learning. In *Security Symposium (USENIX Sec'21)*. USENIX, 2021.

[7] Zian Jia, Yun Xiong, Yuhong Nan, Yao Zhang, Jinjing Zhao, and Mi Wen. MAGIC: Detecting Advanced Persistent Threats via Masked Graph Representation Learning. In *Security Symposium (USENIX Sec'24)*. USENIX, 2024.

[8] Baoxiang Jiang, Tristan Bilot, Nour El Madhoun, Khaldoun Al Agha, Anis Zouaoui, Shahrear Iqbal, Xueyuan Han, and Thomas Pasquier. ORTHRUS: Achieving High Quality of Attribution in Provenance-based Intrusion Detection Systems. In *Security Symposium (USENIX Sec'25)*. USENIX, 2025.

[9] Shaofei Li, Feng Dong, Xusheng Xiao, Haoyu Wang, Fei Shao, Jiedong Chen, Yao Guo, Xiangqun Chen, and Ding Li. NODLINK: An Online System for Fine-Grained APT Attack Detection and Investigation. In *Network and Distributed System Security Symposium (NDSS'24)*. The Internet Society, 2024.

[10] Mati Ur Rehman, Hadi Ahmadi, and Wajih Ul Hassan. Flash: A Comprehensive Approach to Intrusion Detection via Provenance Graph Representation Learning. In *Symposium on Security and Privacy (S&P'24)*. IEEE, 2024.

[11] Mike van Opstal and William Arbaugh. Operationally Transparent Cyber (OpTC) Data Release, 2019. https://github.com/FiveDirections/OpTC-data.

[12] Su Wang, Zhiliang Wang, Tao Zhou, Hongbin Sun, Xia Yin, Dongqi Han, Han Zhang, Xingang Shi, and Jiahai Yang. Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning. *IEEE Transactions on Information Forensics and Security*, 17:3972–3987, 2021.