USENIX Security '25 Artifact Appendix: GeCos Replacing Experts: Generalizable and Comprehensible Industrial Intrusion Detection

Konrad Wolsing[*,‡] Eric Wagner[*,‡] Luisa Lux[*,‡] Klaus Wehrle[‡] Martin Henze[‡,*]

[*]*Fraunhofer FKIE* [‡]*RWTH Aachen University*

A Artifact Appendix

A.1 Abstract

Protecting Industrial Control Systems (ICSs) against cyber-attacks is crucial to counter escalating threats to critical infrastructure. To this end, Industrial Intrusion Detection Systems (IIDSs) provide an easily retrofittable approach to uncover attacks quickly and before they can cause significant damage. Current research focuses either on maximizing automation, usually through heavy use of machine learning, or on expert systems that rely on detailed knowledge of the monitored systems. While the former hinders the interpretability of alarms, the latter is impractical in real deployments due to excessive manual work for each individual deployment.

To bridge the gap between maximizing automation and leveraging expert knowledge, we introduce GeCo, a novel IIDS based on automatically derived comprehensible models of benign system behavior. GeCo leverages state-space models mined from historical process data to minimize manual effort for operators while maintaining high detection performance and generalizability across diverse industrial domains. Our evaluation against state-of-the-art IIDSs and datasets demonstrates GeCo's superior performance while remaining comprehensible and performing on par with rule-based IIDS based on the manual effort of experts.

This artifact provides the implementation of GeCo and describes how to reproduce our main results regarding the detection performance of GeCo, as well as our comparison to related work from Section 6.1 of the publication.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Our artifact does not contain content that threatens security or privacy as we publish the code and the evaluation tools for a defensive intrusion detection system (GeCo). GeCo is evaluated purely on datasets of ICS process data and thus our evaluation process does not require malware. Due to privacy concerns, we do not make the results of our user study from Sec. 6.2.2 available.

A.2.2 How to access

Our artifact is available at `https://zenodo.org/records/15120036`. Besides the files serving as an online appendix for our publication, *code-and-results.zip* contains the files for this artifact. It includes the code of the proposed IIDS (GeCo) and the configuration files used for its evaluation. A permanently maintained version of GeCo is available at the IPAL repository `https://github.com/fkie-cad/ipal_ids_framework/tree/master/ids/geco`, yet it is not required for reproducing the results from our publication.

A.2.3 Hardware dependencies

GeCo does not require specific hardware architectures. Still, the hardware requirements (CPU cores and memory) depend on the dataset and impact the runtime performance. To use and evaluate GeCo with pre-trained models, a single CPU core with a minimum of about 2Gb RAM has proven sufficient. However, to retrain the models of GeCo, more memory must and more CPU cores should be spared.

The smaller datasets (BATADAL and TEP) can be trained in under one hour with 8 CPU cores and at least 2Gb of memory. The training of the SWaT, WADI, and HAI models is much more demanding (cf. Section 6.3 and Table 5 in our paper). Regarding memory, they require at least 16GB for SWaT and HAI and 64GB for WADI to load the datasets. The number of CPU cores impacts the training time. Note that depending on the chosen number of CPU cores, additional memory is required to enable parallel processing.

A.2.4 Software dependencies

GeCo and the evaluation tools are a collection of Python scripts that have been tested with Python 3.12. Other versions are expected to work as well. The implementation has been tested on Linux and macOS operating systems.

The implementation of GeCo is based on the open-source IPAL IDS framework[1] and uses its provided datasets and evaluation tools. Except for the datasets (cf. the following A.2.5), all necessary software is provided within the artifact.

[1]IPAL – `https://github.com/fkie-cad/ipal`

List of used software versions:

- https://github.com/fkie-cad/ipal_ids_framework v1.5.2
- https://github.com/fkie-cad/ipal_evaluate v1.2.9
- https://github.com/fkie-cad/ipal_datasets v1.3.7

A.2.5 Benchmarks

Note that we cannot publicly provide the dataset files (*datasets/*) for legal reasons. Still, the datasets can be obtained by the publishers and transcribed into the IPAL format following the instructions here (cf. *datasets/README.md*). These steps are necessary to reproduce any of our results:

1. Clone the https://github.com/fkie-cad/ipal_datasets repository.
2. Install the dependencies listed in the *requirements.txt* file.
3. Select a dataset, e.g., SWaT, for this example.
4. Obtain the raw dataset from the publishers. Links to the dataset publishers are listed in the *ipal_datasets/README.md* table. The precise version and files used in our paper are listed below.
5. Place the dataset files in the respective folder. For example, *ipal_datasets/SWaT/raw/[dataset files]*.
6. Execute the *transcribe.py/sh* script for the chosen dataset.
7. Copy the generated files and replace this artifact's placeholders *attack.json* and *ipal/** files.

List of used files and dataset versions:

- SWaT: Download all three XLSX files from the *SWaT.A1 & A2_Dec 2015/Physical* folder.
- WADI: Download all CSV files from the *WADI.A2_19 Nov 2019* folder.
- HAI: Download all **csv.gz* from the https://github.com/icsdataset/hai/tree/master/hai-21.03 repository.
- BATADAL: Download all three CSV files from https://www.batadal.net/data.html (*BATADAL_dataset04.csv*, *BATADAL_dataset03.csv*, and *BATADAL_test_dataset.csv*).
- TEP: Download all files from the https://github.com/mikeliturbe/pasad/tree/master/data repository.

A.2.6 Description of files and content

The artifact consists of various files and folders, some relevant for reproducing our results while others serve as additional material for our paper. In the following, we describe the structure of the artifact:

- *code/*: This folder contains the software necessary for conducting our evaluation, based on the IPAL IDS Framework (cf. A.2.4), and the implementation of GeCo (*code/ipal-ids-framework/ids/GeCo*).
- *config/*: For each dataset, the **.json* files define the hyperparameters for GeCo. Based on these configurations and the training data, GeCo learns a model stored within the respective **.model* files.
- *datasets/*: We evaluated GeCo on five datasets stored in this folder (cf. A.2.5 on how to prepare the datasets).
- *knowledge-based/*: This folder contains the artifact to reproduce our results of Sec. 6.1.3.
- *output/*: Once the datasets are labeled by GeCo, we store the IDSs output here.
- *related-work/*: This folder contains the alerts of related work (SIMPLE, TABOR, Invariant, Seq2SeqNN, PASAD), which we use as a comparison in Table 2. These results were reproduced with the help of the IPAL IDS framework. The **.state.gz* files contain the labeled datasets for each IDS, and the **.json* files the respective metrics.
- *results*: Based on the labeled dataset from *output/*, the **.json* files contain the calculated performance metrics for GeCo and the **.pdf* files depict the alerts of GeCo (black) compared to the dataset's attacks (red).
- *videos/*: This folder contains renderings of the dependency graph from Figure 7a for all datasets, which are complementary material for our publication.

A.3 Set-up

A.3.1 Preparation

Please prepare the dataset files as described in A.2.5. For each dataset, all the dummy files in the respective dataset folders (*datasets/SWaT,WADI,HAI,BATADAL/**) have to be replaced.

A.3.2 Installation

Execute the following commands to install the required software for GeCo:

```
python3 -m venv venv
source ./venv/bin/activate
pip3 install igraph==0.10.4
pip3 install code/ipal-ids-framework/
pip3 install code/ipal-evaluate/
```

More information can be found in the *README.md* file.

A.3.3 Basic Test

Execute the following command to apply GeCo to one of the selected datasets.

```
./run-ids.py {SWaT,WADI,HAI,BATADAL,TEP}
```

To test whether the installation was successful, select "n" in the *run-ids.py* script to skip the sometimes long training process. For testing the training, use one of the datasets for which training is fast, such as BATADAL or TEP.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): GeCo shows competitive detection performance, which is often better than related work. This is proven by the evaluation (E1) described in Section 6.1.1, whose results are reported in Table 2, Table 6, and Figure 12.

(C2): GeCo generalizes well to a new domain of the chemical Tennessee Eastman Process (TEP). This claim is supported by the evaluation (E2) described in Section 6.1.2, whose results are reported in Figure 6.

(C3): GeCo yields a detection performance that is on par with labor-intensively created knowledge-based IIDSs. This claim is assessed in evaluation (E3) described in Section 6.1.3 and Table 3.

(C4): GeCo's alerts are comprehensible as they closely correlate to the ICS's underlying physical process. This experiment was designed and conducted in Section 6.2.1 and the results are shown in Figure 7 and Figure 8.

A.4.2 Experiments

How to, preparation, and execution

To conduct the evaluation, execute the *run-ids.py [dataset]* script, providing one of the five datasets (SWaT, WADI, HAI, BATADAL, and TEP) as the argument one by one. If the goal is to reproduce our results, select "n" at the beginning of the *run-ids.sh* script. That way, GeCo uses the provided pre-trained models from the *config/* folder. These steps suffice to reproduce the results of our experiments (E1–E4). [5 human-minutes, 10 compute-minutes, 1GB disk space]

Optionally, the detection models of GeCo can also be retrained on the training data instead of using the pre-trained models. To this end, select "y" at the beginning of the *run-ids.sh* script. Since the training phase can be extensive depending on the dataset, it is advisable to leverage an appropriate amount of CPU cores for parallelization (cf. Section 6.3). The number of cores for training can be configured in the configuration files under *config/{SWaT,WADI,HAI,BATADAL,TEP}.json* with the *cpus* option (default is 7). We recommend training the BATADAL or TEP model first, as they are the least computationally demanding datasets [5 human-minutes, 1-2 compute-hours, 1GB disk space]. SWaT, HAI [5 human-minutes, 24 compute-hours] and WADI [5 human-minutes, 1+ compute-month] demand significantly more time for training even with 64 CPU cores and 236GB of RAM. Yet, this step is not mandatory for the success of the following experiments.

Results

(E1): *Detection Performance* [10 human-minutes]: GeCo's detection performance is measured by comparing its alerts to the datasets labels. The results can be found in the *results/* folder for each dataset, where the **.json* files list different detection performance metrics and the **.pdf* files visualize the alerts in relation to the attacks. To visualize and aggregate these results, execute the respective scripts (*table-2.py*, *fig-12.py*, and *table-6.py*) to obtain the results shown in our publication. These scripts either render an image or generate the LaTeX code for the respective figures and tables.

(E2): *Generalizability* [5 human-minutes]: We show that GeCo successfully transfers to a new industrial domain by applying GeCo to the TEP dataset. The detection performance stated in Section 6.1.2 (96.0 in F1 and 98.0 in eTaF1) can be validated with the *results/TEP.json* file. Additionally, the *fig-6.py* script renders GeCo's alerts compared to the PASAD IIDS.

(E3): *Expert Knowledge-based IIDS* [5 human-minutes]: We implemented an IIDS that leverages a collection of Invariants as listed in Table 8. Execute the following script located under *knowledge-based/run.sh* to apply this IIDS to the SWaT dataset. The results are stored in the *output.json* and *output.pdf* files. To reproduce Table 3 of our paper, execute the *table-3.py* script.

(E4): *Alerts Localization and Understandability* [5 human-minutes]: We analyze GeCo's detection model to infer which process values are responsible for the alerts by analyzing the dependency between the learned state-space models by means of a dependency graph in Figure 7a. The dependency graph can be recreated with the respective *fig-7a.py* script. Note that the graph is arranged differently with each execution of the script. Lastly, the *fig-8.py* script generates the results for Figure 8.

A.5 Notes on Reusability

If the goal is to examine GeCo's detection performance in more detail on the datasets evaluated so far, please refer to the files in the *output/* directory, as they contain the raw output of GeCo for each dataset. For other activities, such as applying GeCo to different datasets, we recommend working with the implementation provided by the IPAL IDS framework[1]. To this end, we direct the reader to the tutorial and documentation about IPAL[2]. For guidance on how to configure the hyperparameters of GeCo, please refer to Section 7.1 of our publication.

A.6 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/`.

[2]IPAL Tutorial – `https://github.com/fkie-cad/IPAL/blob/main/tutorial/Tutorial.md`