Toward Practical and Usable Provenance-based Intrusion Detection Systems

Tristan Bilot, Thomas Pasquier

In recent years, researchers have turned to provenance-based intrusion detection systems (PIDSs) as a promising way to spot attacks that evade past traditional defenses. At their core, these systems build provenance graphs, which act like detailed maps of how information flows through a computer. Such graphs treat system entities (e.g., processes, files, and network connections) as nodes, and the interactions between them (e.g., system calls) as edges. By analyzing these graphs, anomaly-based PIDSs learn what "normal" behavior looks like and then flag unusual activity, making them well-suited for catching stealthy attacks such as advanced persistent threats (APTs) or previously unknown zero-day exploits.

Despite claims of near-perfect detection rates, today's PIDSs are nowhere near ready for real-world use. Their biggest flaw is how they report results: most state-of-the-art PIDSs generate coarse-grained alerts with tens of thousands of nodes or events, burying analysts under mountains of noise. This is not just an engineering oversight; it is the direct result of evaluation practices. By optimizing for specific evaluation metrics instead of usable outputs, the community has built detectors that are impressive on paper but fall short of being helpful to a security team.

Our research also exposes deeper flaws in how current systems are designed and evaluated. This comes in the way of industry adoption [6]. Even when working from the same datasets, different studies use their own training and testing splits, preprocessing pipelines, and labeling strategies. These inconsistencies, common across many ML-driven fields [9], fragment evaluation and make it impossible to compare competing methods fairly.

This article summarizes our recent efforts to address these challenges, drawing on two papers [3, 4] presented at USENIX Security 2025.

The Need for High Quality of Attribution

Evaluating a PIDS typically involves measuring detection quality in terms of true and false positives. However, we find that many state-of-the-art systems do not optimize for Quality of Attribution (QoA): the effort a human analyst must expend to interpret alerts, trace an attack’s root cause, assess its impact, and dismiss false positives. In practice, prior work often produces massive volumes of low-QoA output, much of it irrelevant to actual attack activity. The result is predictable: noisy reports that overwhelm analysts, fueling alert fatigue [1] and, ultimately, burnout [2].

We identified three main evaluation strategies used in assessing state-of-the-art systems.

Neighborhood Approach: Labels all nodes within two hops of the ground truth nodes as malicious. This pulls in benign nodes that only share dependencies with attack events, leading to overestimation.

Batch Approach: Groups events into fixed-size or time-based batches and assigns anomaly scores per batch. Benign events occurring alongside attacks are often flagged, inflating results.

Source Approach: Labels all descendants of identified source nodes as malicious. This exaggerates attack scope: for example, normal activity after a process hijack is also marked malicious.

Although systems built under these evaluation strategies may seem effective at flagging anomalies associated with attacks, they fall short when it comes to accurate node-level attribution. Prior work [7, 8] has acknowledged this gap, emphasizing the challenge posed by the overwhelming volumes of data that state-of-the-art PIDSs generate and the burden this places on security analysts.

Our approach: We undertook a painstaking manual analysis of the datasets, cross-checking the textual documentation against the actual data to pinpoint the specific nodes involved in each attack. Whereas prior work labeled thousands of nodes as malicious, our analysis revealed that the true number is far smaller—between 41 and 123 nodes, as shown in Figure 1.

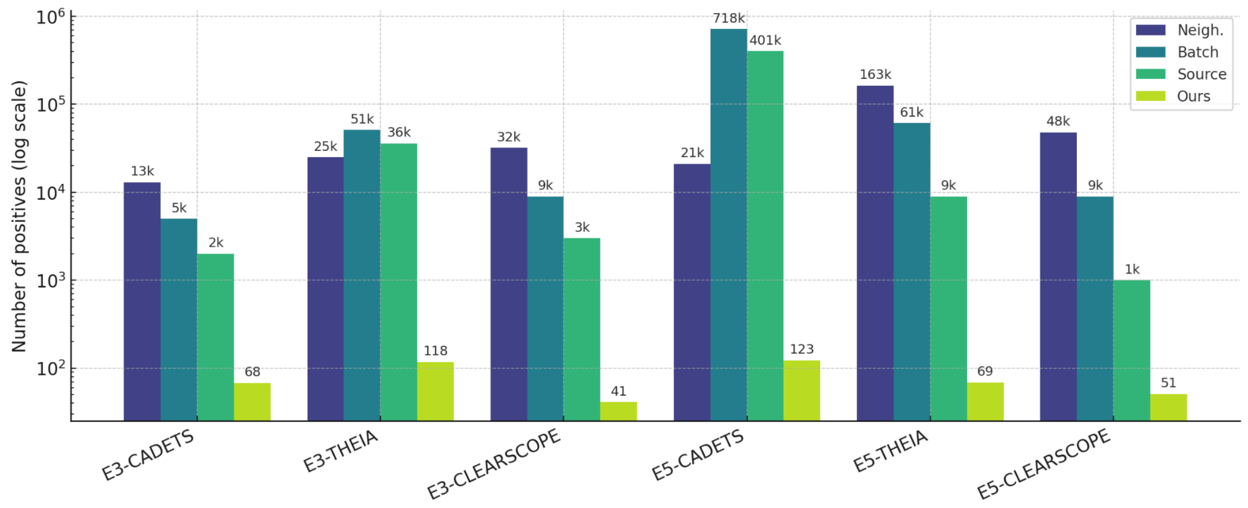


Figure 1. Number of malicious nodes reported by each evaluation strategy, on DARPA TC datasets.

By shifting the evaluation focus to the quality of attribution, we encourage the development of systems that deliver far more helpful results. Instead of rewarding detectors that flood analysts with thousands of noisy alerts, this approach promotes systems that highlight a small set of high-quality attack nodes containing the actual malicious payloads, outputs that analysts can act on directly.

Achieving High Quality of Attribution

Most existing systems look good on paper but fall short when analysts need fine-grained answers. Under our node-level evaluation, they simply aren't designed to deliver the precision required. This gap led us to build **Orthrus**, a PIDS tailored for the job. Orthrus combines temporal graph learning with causality analysis to do two things analysts care about most: (1) detect node-level anomalies without relying on prior attack knowledge, and (2) reconstruct clear attack scenarios. Instead of drowning analysts in noise, Orthrus generates concise, high-quality summary graphs that spotlight the malicious activity. The result is less time wasted on irrelevant alerts and more time spent understanding real threats. As Figure 2 shows, Orthrus achieves this through five coordinated components.

Graph Construction: Provenance graphs are built from raw system logs, with redundant edges pruned while preserving event order.

Edge Featurization: Graphs are transformed into sequences of vectorized edges by transforming the textual attributes of entities into text embeddings using word2vec. These embeddings aim to embed entities with similar semantics closer in embedding space, and are used as node features in the graph.

Temporal Graph Learning: An encoder-decoder architecture captures structural and temporal patterns. The encoder computes node embeddings by gathering the last activity of nodes and aggregating neighbor information with an attention-based graph neural network (GNN). The decoder passes node embeddings in an additional neural network and the model is trained end-to-end to predict system call types between connected nodes from benign activity. The resulting reconstruction loss serves as an anomaly score to assess malicious activity.

Anomaly Detection: Malicious nodes are detected by thresholding and clustering. The threshold is automatically computed as the maximum loss encountered on the validation set. The flagged nodes are subsequently separated in two clusters with K-Means, with the top-cluster selected as reported attack nodes.

Attack Reconstruction: Detected nodes are used in an attack reconstruction step, aiming to uncover more nodes by causality analysis from the graph structure and predicted scores [10]. Orthrus finally reports concise summary graphs representing the predicted attack path.

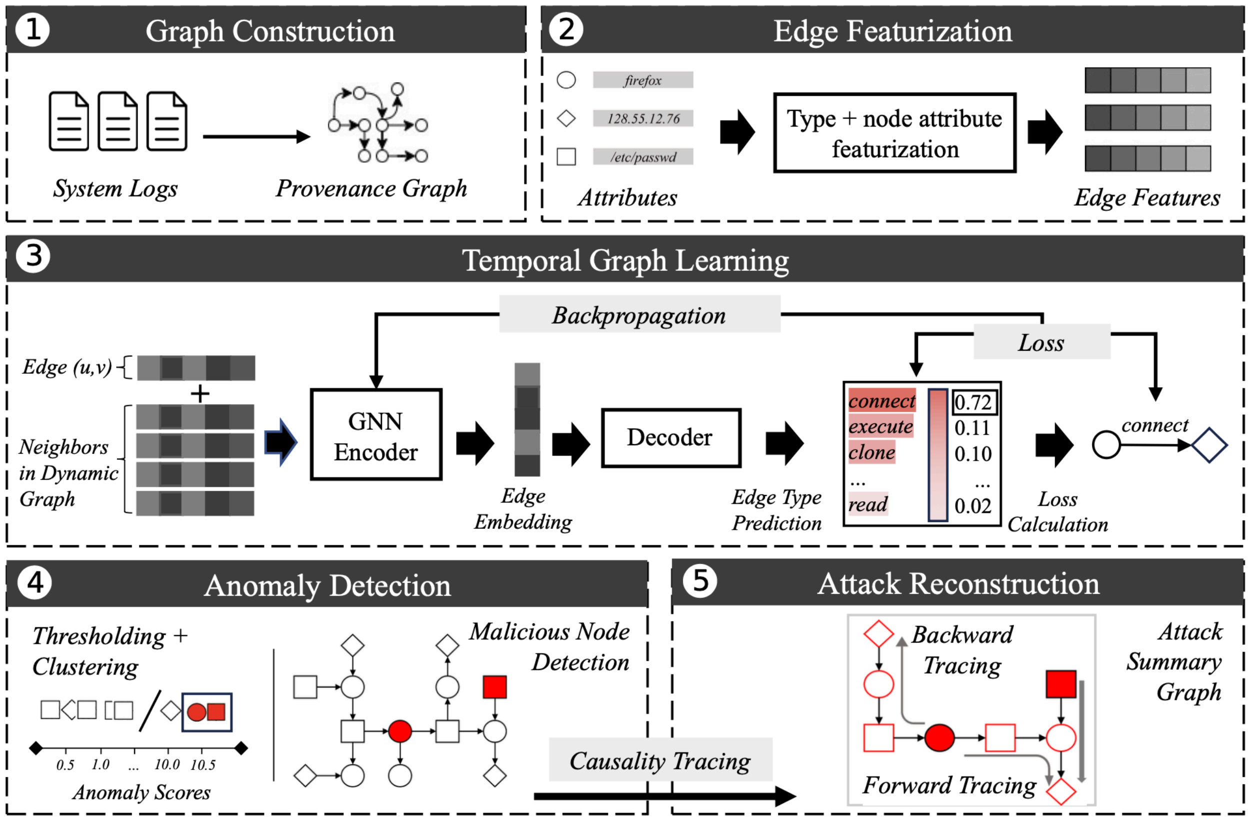


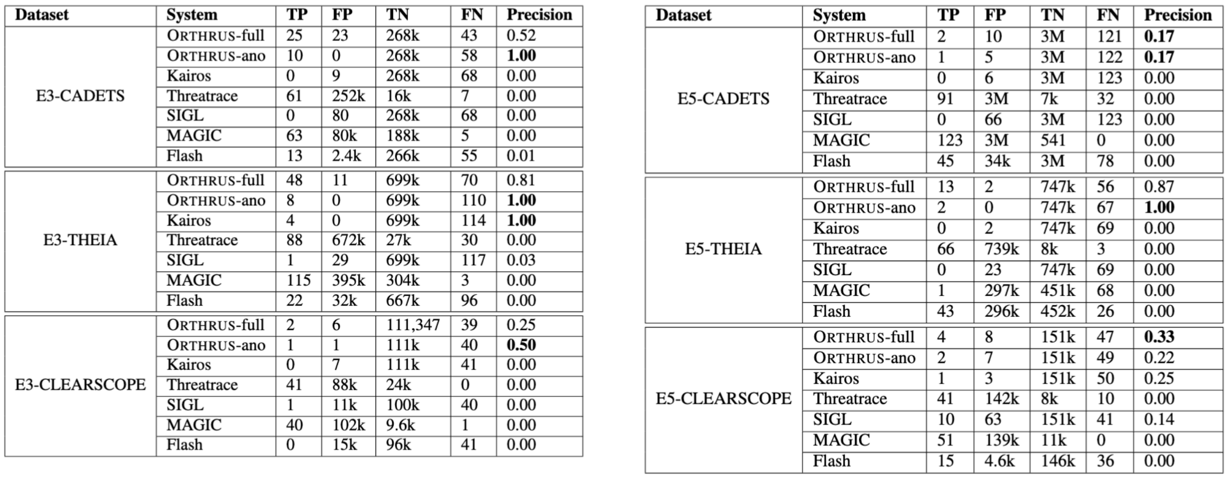
Figure 2. Architecture of Orthrus.

Evaluation of Orthrus

We evaluate Orthrus (Figure 3) against five state-of-the-art baselines using our node-level strategy on six DARPA TC datasets, tuning all baselines consistently through grid search on key hyperparameters.

We benchmark two variants of Orthrus:

- **Orthrus-full**: full Orthrus pipeline as shown in Figure 2.
- **Orthrus-ano**: Orthrus pipeline without the attack reconstruction step.



Dataset	System	TP	FP	TN	FN	Precision
E3-CADETS	ORTHRUS-full	25	23	268k	43	0.52
	ORTHRUS-ano	10	0	268k	58	**1.00**
	Kairos	0	9	268k	68	0.00
	Threatrace	61	252k	16k	7	0.00
	SIGL	0	80	268k	68	0.00
	MAGIC	63	80k	188k	5	0.00
	Flash	13	2.4k	266k	55	0.01
E3-THEIA	ORTHRUS-full	48	11	699k	70	0.81
	ORTHRUS-ano	8	0	699k	110	**1.00**
	Kairos	4	0	699k	114	**1.00**
	Threatrace	88	672k	27k	30	0.00
	SIGL	1	29	699k	117	0.03
	MAGIC	115	395k	304k	3	0.00
	Flash	22	32k	667k	96	0.00
E3-CLEARSCOPE	ORTHRUS-full	2	6	111,347	39	0.25
	ORTHRUS-ano	1	1	111k	40	**0.50**
	Kairos	0	7	111k	41	0.00
	Threatrace	41	88k	24k	0	0.00
	SIGL	1	11k	100k	40	0.00
	MAGIC	40	102k	9.6k	1	0.00
	Flash	0	15k	96k	41	0.00

Dataset	System	TP	FP	TN	FN	Precision
E5-CADETS	ORTHRUS-full	2	10	3M	121	**0.17**
	ORTHRUS-ano	1	5	3M	122	**0.17**
	Kairos	0	6	3M	123	0.00
	Threatrace	91	3M	7k	32	0.00
	SIGL	0	66	3M	123	0.00
	MAGIC	123	3M	541	0	0.00
	Flash	45	34k	3M	78	0.00
E5-THEIA	ORTHRUS-full	13	2	747k	56	0.87
	ORTHRUS-ano	2	0	747k	67	**1.00**
	Kairos	0	2	747k	69	0.00
	Threatrace	66	739k	8k	3	0.00
	SIGL	0	23	747k	69	0.00
	MAGIC	1	297k	451k	68	0.00
	Flash	43	296k	452k	26	0.00
E5-CLEARSCOPE	ORTHRUS-full	4	8	151k	47	**0.33**
	ORTHRUS-ano	2	7	151k	49	0.22
	Kairos	1	3	151k	50	0.25
	Threatrace	41	142k	8k	10	0.00
	SIGL	10	63	151k	41	0.14
	MAGIC	51	139k	11k	0	0.00
	Flash	15	4.6k	146k	36	0.00

Figure 3. Node-level detection results of Orthrus and baselines.

Overall, **Orthrus-ano** reliably detects at least one node per attack across all datasets, while producing fewer false positives than baseline methods. Indeed, a practical system must minimize false positives (i.e., ensure high precision) as long as all attacks are detected, rather than focusing on identifying every single node involved in an attack (i.e., high recall) [6].

Orthrus-full builds upon Orthrus-ano's output to recover additional true positives. It successfully enhances attribution while maintaining a low false positive rate, providing a complementary output that analysts can leverage for deeper investigation.

Toward More Practical Systems

Although Orthrus marks progress toward practical detection by substantially reducing analyst workload through concise reports, this system and other recent approaches still face limitations that hinder real-world deployment. The practicality and usability of PIDSs should matter as much as detection accuracy for real-world adoption. To move the field forward, system design must prioritize analyst needs so that research prototypes evolve into tools that are truly deployable, rather than remaining academic exercises.

In particular, real-time detection remains challenging, as it would require running inference on the provenance graph after every incoming edge, an approach that does not scale with increasing numbers of hosts. One way to address this challenge is to significantly simplify the architecture, avoiding heavy models such as GNNs. However, many recent works neglect to compare their systems against simpler baselines, instead favoring evaluations only against equally complex, state-of-the-art approaches.

Motivated by this gap, we developed a unified framework that re-implements eight state-of-the-art PIDSs published in top-tier security venues. The framework is modularized to support combinatorial exploration of architectural components, enabling systematic benchmarking of performance across models of varying complexity.

Through this exploration, we experimented with each PIDS and identified several limitations in their design and evaluation strategies. We distill these into **nine key shortcomings** that, in our view, must be addressed for the community to develop and publish more practical systems.

1. **Insufficient Detection Granularity**: As demonstrated in Orthrus, most systems detect attacks at the graph or neighborhood levels, leading to thousands elements to analyze.
2. **Missing Metric to Measure Attack Detection**: Traditional metrics don't account for individual attacks and are biased toward thresholding: Systems rely on fixed, arbitrary and manually set thresholds that fail to adapt dynamically.
3. **Impractical Thresholding Methods**: Systems rely on fixed, arbitrary and manually set thresholds that fail to adapt dynamically.
4. **Unfair Comparison with Baselines**: Evaluation baselines are left untuned, while proposed systems are typically extensively tuned.
5. **Not Measuring Instability**: The detection performance of systems is extremely unstable under identical configurations.
6. **Featurization Methods Trained on Test Data**: Some systems train on test data, leading to data snooping by exploiting information that would be unavailable in practice.
7. **Overly Complex Architectures**: Systems are usually complex, yet they are rarely compared to much simpler models.
8. **Insufficient Scalability**: Systems do not meet scalability and overhead requirements for a practical deployment.
9. **Lacking Real-Time Detection**: Systems are poorly fitted for real-time setting due to their design and overhead.

Some shortcomings (1–6) can be addressed within existing PIDSs through more thorough experimentation or simple architecture modifications. In contrast, the last three (7–9) reflect deeper structural issues that result in overly complex architectures with poor scalability and limited support for real-time detection.

Exploring Simpler and Lightweight Architectures

To investigate simpler variants, we conducted extensive experiments totaling more than 453 days of computation on servers equipped with modern GPUs. We explored a range of new architectures by combining components from existing PIDSs and performing ablations to assess the impact of removing specific components. All variants were compared against existing PIDS baselines, with hyperparameters tuned and the best models selected for evaluation on nine DARPA TC and OpTC datasets simulating APT campaigns. Performance was assessed using attack detection precision (ADP), a pre-threshold metric that captures the ability to detect all attacks while maintaining high precision.

Surprisingly, our results show that a **much simpler architecture** not only **matches their performance** but does so with significantly lower runtime and memory overhead.

This variant, illustrated in Figure 4 and referred to as **Velox**, is a lightweight alternative to heavyweight GNNs and addresses all identified shortcomings by design. When processing an edge, it embeds the textual attributes of the source and destination nodes (file paths, process command lines, and socket IP addresses) using word2vec. These embeddings are then passed to a small neural network trained to predict the system call type associated with each edge. The prediction loss becomes the anomaly score, with nodes connected to high-scoring edges flagged as anomalous. Detection is performed through automatic thresholding calibrated on the validation set.

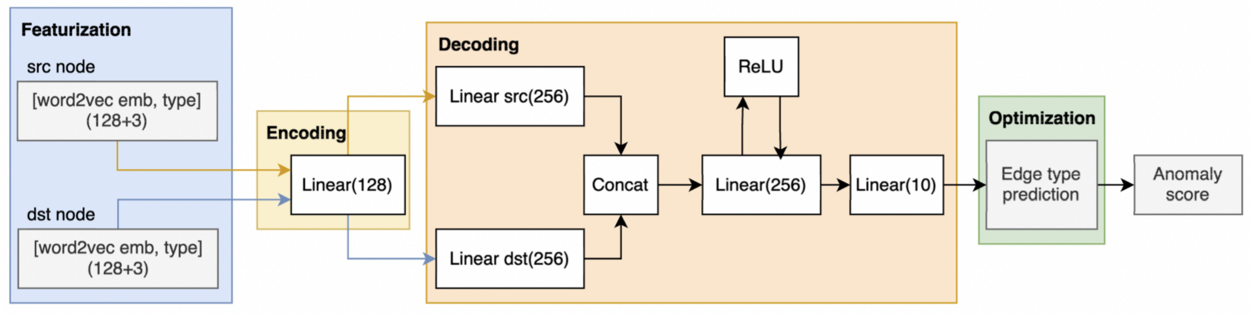


Figure 4. Neural network architecture of Velox.

Figure 5 shows ADP performance alongside runtime and memory usage on the E3-CADETS dataset, which contains three attacks (results are equivalent on other datasets).
In simple terms, ADP balances detection coverage with alert reliability. Formally, it is the area under the detected-attacks–precision curve, with ADP = 1 when all attacks are found with perfect precision, and ADP = 0 when detection occurs only at near-zero precision.

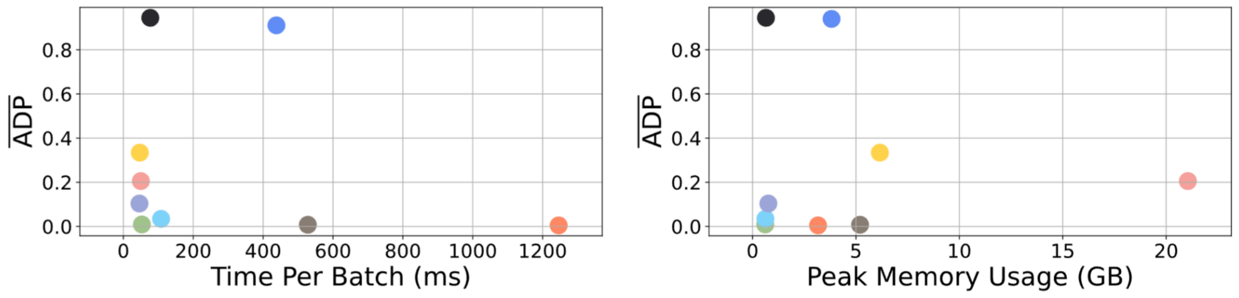


Figure 5. Attack detection precision versus time and memory. Baselines are Velox [3], Orthrus [4], NodLink [11], R-Caid [12], Flash [13], Kairos [14], Magic [15], ThreaTrace [16], SIGL [17].

At the node level, only Orthrus and Velox detect all attacks with high precision, reflected in their near-perfect ADP scores. The key difference is efficiency: Velox delivers the same detection capability as Orthrus but with far lower cost. This proves that, in this domain, simplicity can outperform complexity.

These findings are surprising and raise an important question about the growing complexity of PIDS architectures: ***is such complexity always necessary?***
We did not expect APT attacks to be detected so effectively using only the textual attributes of system entities. To avoid unnecessary complexity, we advocate for advanced ablation studies that empirically demonstrate that the only way to achieve such performance across multiple datasets is through integrating all components proposed in the paper.
Otherwise, simpler variants may exist but remain unexplored.

These results raise another important question: ***are the DARPA TC and OpTC datasets truly realistic benchmarks?*** Recent work [5] points out that these datasets include synthetic benign background activity, which artificially inflates true negative rates, while overly conspicuous malicious events inflate true positives. As a result, they reward systems for solving an easier problem than what analysts face in practice. Developing datasets that more accurately capture the messiness of real-world environments is, therefore, essential if we want PIDS evaluations in the literature to reflect genuine progress.

A Unified Framework to Support Future Research

To support the community, we are releasing **PIDSMaker** [18, 3], an open-source framework that unifies eight state-of-the-art systems into a single execution pipeline optimized for modularity and computational reuse. PIDSMaker was the foundation for the evaluations in both of our papers, and we hope it will serve as a common platform for future research, enabling consistent benchmarking and lowering the barrier to developing new systems.

PIDSMaker's pipeline (Figure 7) is organized into modular tasks, each with its own arguments. Every PIDS is specified in a YAML file that consolidates all relevant parameters, making configurations easy to manage and reproduce. Intermediate outputs are cached locally and indexed by a hash of the current and preceding task arguments, ensuring that previously computed results are not recomputed.

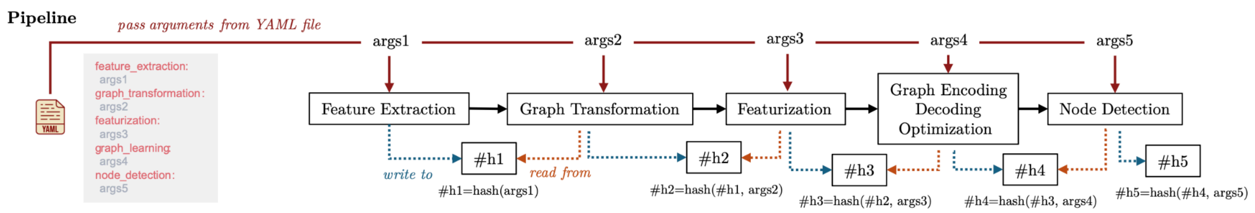


Figure 7. PIDSMaker framework's pipeline.

The framework supports a variety of features.

- **Combinatorial exploration of components:** The framework modularizes components from all baseline systems into ready-to-use building blocks, enabling flexible combinations. This design supports the creation of new PIDS variants by mixing and matching different encoders, decoders, training objectives, and thresholding strategies.
- **Consistent Evaluation:** All baselines share a common backbone and preprocessing pipeline, and are evaluated against the same ground truth, ensuring fair and consistent comparisons.
- **Hyperparameter Tuning:** Since PIDSs are highly sensitive to hyperparameter choices, the framework includes a unified tuning module to ensure consistent optimization across all baselines.
- **Instability Measurement:** PIDSs often produce variable results across runs. The framework supports repeated executions and quantifies instability using standard deviation metrics, providing a robust measure of variance.

We welcome researchers to contribute their systems to PIDSMaker and join us in extending and keeping the framework up to date.

References

[1] Hassan, Wajih Ul, et al. "Nodoze: Combatting threat alert fatigue with automated provenance triage." *Network and Distributed Systems Security Symposium*. 2019.

[2] Sundaramurthy, Sathya Chandran, et al. "A human capital model for mitigating security analyst burnout." S*ymposium on Usable Privacy and Security (SOUPS 2015)*. 2015.

[3] Bilot, Tristan, et al. "Sometimes Simpler is Better: A Comprehensive Analysis of State-of-the-Art Provenance-Based Intrusion Detection Systems." *USENIX Security Symposium*. 2025.

[4] Jiang, Baoxiang, et al. "ORTHRUS: Achieving High Quality of Attribution in Provenance-based Intrusion Detection Systems." USENIX *Security Symposium*. 2025.

[5] Liu, Jason, et al. "What We Talk About When We Talk About Logs: Understanding the Effects of Dataset Quality on Endpoint Threat Detection Research." IEEE *Symposium on Security and Privacy*. 2025.

[6] Dong, Feng, et al. "Are we there yet? an industrial viewpoint on provenance-based endpoint detection and response tools." *ACM Conference on Computer and Communications Security*. 2023.

[7] Cheng, Zijun, et al. "Kairos: Practical intrusion detection and investigation using whole-system provenance." *IEEE Symposium on Security and Privacy*. 2024.

[8] Rehman, Mati Ur, Hadi Ahmadi, and Wajih Ul Hassan. "Flash: A comprehensive approach to intrusion detection via provenance graph representation learning." *IEEE Symposium on Security and Privacy*. 2024.

[9] Arp, Daniel, et al. "Dos and don'ts of machine learning in computer security." *USENIX Security Symposium*. 2022.

[10] Fang, Pengcheng, et al. "Back-Propagating system dependency impact for attack investigation." *USENIX Security Symposium*. 2022.

[11] Shaofei Li, Feng Dong, et al. 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.

[12] 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

[13] 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.

[14] Zijun Cheng, Qiujian Lv, et al. Kairos: Practical Intrusion Detection and Investigation using Whole-system Provenance. In Symposium on Security and Privacy (S&P'24). IEEE, 2023.

[15] Zian Jia, Yun Xiong, et al. MAGIC: Detecting Advanced Persistent Threats via Masked Graph Representation Learning. In Security Symposium (USENIX Sec'24). USENIX, 2024.

[16] Su Wang, Zhiliang Wang, et al. 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.

[17] Xueyuan Han, Xiao Yu, et al. SIGL: Securing Software Installations Through Deep Graph Learning. In Security Symposium (USENIX Sec'21). USENIX, 2021.

[18] https://github.com/ubc-provenance/PIDSMaker