Tidal: Tackling Concept Drift in Provenance-Based Advanced Persistent Threats Detection

Yajie Zhou
University of Maryland, College Park, MD, USA

Nengneng Yu
University of Maryland, College Park, MD, USA

Tuo Zhao
Georgia Institute of Technology, Atlanta, GA, USA

Zaoxing Liu
University of Maryland, College Park, MD, USA

Abstract

Advanced Persistent Threats (APTs) pose significant challenges to cybersecurity due to their evolving nature and ability to evade detection. This paper introduces Tidal, a novel provenance-based intrusion detection system (PIDS) that is specifically designed to address concept drift in APT detection. Tidal designs a modified Transformer architecture tailored for transfer learning, including a Multi-head Transformer (MHT) with shared layers for learning common knowledge and task-specific head layers for learning unique patterns. The system uses a pre-training and fine-tuning workflow to achieve high post-drift adaptation and pre-drift retention accuracy. Additionally, Tidal customizes its data embedding for detection on flexible audit log lengths and computes entity relevance scores alongside classified attacks to aid in attack investigation. We evaluate Tidal by simulating concept drift scenarios with real-world datasets. Results demonstrate that compared to state-of-the-art detection systems, Tidal achieves an average of 27% higher recall and 31% higher precision with only half of new training data for post-drift adaptation accuracy.

2012 ACM Subject Classification Security and privacy

Keywords and phrases Advanced Persistent Threat (APT), Provenance-based Intrusion Detection (PIDS), Concept Drift, Transfer Learning, Machine Learning for Security

Digital Object Identifier 10.4230/OASIcs.NINeS.2026.1

Acknowledgements We thank all the anonymous reviewers for their constructive feedback and valuable suggestions, which greatly improved the quality of this paper. This work was supported in part by the U.S. NSF grants SaTC-2415754, CNS-2415758, and CNS-2431093. We extend our special thanks to our shepherd for the thoughtful guidance throughout the revision process. We are grateful to Chao Zhang, Simiao Zuo, Yue Yu for their early feedback, which helped shape the first idea of this work.

1 Introduction

As cybersecurity challenges escalate, *advanced persistent threats* (APTs) raise particular concern due to their stealthy, prolonged, and adaptive nature. Often orchestrated by well-resourced groups, APTs involve meticulously planned, multi-stage campaigns designed to infiltrate and persist within target systems over extended periods. These threats continuously evolve their tactics, techniques, and procedures (TTPs) to evade detection, adapting dynamically based on target defenses.

To address the adaptive behavior of APTs, *provenance-based intrusion detection systems* (PIDSes) leverage kernel-level audit logs, which capture detailed system events, to model a history of system execution as provenance graphs to guide detection efforts. Recent research into PIDSes employ machine learning-based approaches [23, 5, 64, 63, 17, 16, 1, 33, 53, 50, 9], demonstrating improved accuracy in APT detection across a variety of environments.

As new APTs emerge over time, however, ML-based PIDSes are challenged by *concept drift*, or the evolution of data that invalidates previously learned patterns. Concept drift can cause a drop in model performance as the relationships learned previously decrease in relevancy. This is of particular concern with APTs, which exhibit adversarial concept drift. Unlike natural distribution shifts in traditional ML domains, attackers deliberately evolve tactics to evade detection. Although recent research on PIDSes acknowledges concept drift as in issue in APT detection [50, 9], there has been limited in-depth exploration of how to address it.

In this paper, we identify three key challenges that arise when adapting ML-based PIDSes for APT detection to address concept drift: 1) retraining a ML model often requires a large amount of labeled data, which is scarce in APT contexts due to resource limitations and privacy concerns, 2) continuously training an existing model may require fewer new labeled samples but introduces the risk of *catastrophic forgetting*, where performance on previous data declines, and 3) while concept drift is a commonly addressed issue in machine learning, general solutions may fall short when facing APT-specific challenges, such as ensuring timely detection and in-depth attack investigation.

We then introduce Tidal, a PIDS that specifically addresses the above challenges to tackle concept drift in APT detection. Tidal is based on a simple insight: pre-drift and post-drift APT data may exhibit similar patterns while differing in specific attack characteristics. This drives Tidal's two-phase design: first, it pre-trains on pre-drift data to capture general system activity patterns as shared knowledge, then fine-tunes on limited post-drift data to adapt to emerging attack (or benign) patterns in each dataset. This approach effectively demonstrates transfer learning to tackle concept drift, and incorporates the following key ideas:

A modified Transformer architecture tailored for transfer learning among APT datasets. Inspired by the success of domain adaptation with large language and vision models [58, 60, 21], we explore a standard Transformer architecture [52], but find it limited in its ability to share knowledge among APT datasets. Based on domain-specific data insights, Tidal instead incorporates a Multi-head Transformer (MHT) that contains shared layers to capture common knowledge and multiple APT task-specific head layers that specialize in each dataset's unique patterns. Notably, MHT supports both signature-based and anomaly-based detection through interchangeable head-layer implementations. In our experiments, MHT improves upon adaptation accuracy in concept drift scenarios for APT detection compared to baseline models that use Long Short-Term Memory (LSTM) and Graph Neural Networks (GNN).

A pre-training and fine-tuning workflow to optimize accuracy under concept drift. The learning process begins with pre-training a base model on existing APT datasets. When concept drift occurs, Tidal initiates a fine-tuning phase to adapt to new attacks. Fine-tuning involves creating a new head atop shared layers to isolate post-drift attack patterns, freezing the first half of shared encoder layers to preserve pre-drift knowledge, and applying a lower learning rate for gradual adaptation. This strategy effectively balances the influence of pre-drift and post-drift data, stabilizing transfer learning and preventing catastrophic forgetting.

A customized data embedding that supports flexible detection lengths. Tidal aims to maximize information density within model inputs by focusing on semantic context from audit logs, which tends to change more dramatically than the provenance graph structure

Table 1 Overview of recent PIDSes for APT detection under concept drift.

	Signature -based PIDS	**Anomaly -based PIDS**	**Post-drift adaptation accu.**	**Pre-drift retention accu.**
Atlas [5]	✓	–	Low	Low
Unicorn [23]	–	✓	Low	Low
Flash [50]	–	✓	Low	Low
Kairos [9]	–	✓	Low	Low
Ours	✓	✓	High	High

over short time windows. We complement the semantics with a careful selection of structural provenance data, enabling detection on flexible time frames. This flexibility eliminates assumptions about data size during TIDAL's inference, allowing human investigators to quickly respond with classified anomalies.

Entity relevance scores for attack investigation. TIDAL generates relevance scores for each system entity within the input provenance graph, indicating their relative importance to detected anomalies. While not completely demystifying the machine learning model's explainablility, this feature significantly enhances transparency for the decision-making process. By setting an appropriate relevance score threshold, TIDAL highlights the most relevant malicious events, effectively guiding human investigators to focus on the most pertinent aspects of the provenance graph rather than being overwhelmed by extraneous details. This targeted approach prevents information overload, potentially saving human time and effort at investigation.

TIDAL's learning pipeline is implemented in PyTorch, comprising 3.5K lines of Python and 400 lines of C++ code. To evaluate TIDAL's performance in concept drift scenarios, we simulated two drift conditions using the DARPA E3 and E5 datasets, as publicly available datasets exhibiting concept drift in APT scenarios are scarce. The evaluation focuses on two key aspects: accuracy adaptation on post-drift data (i.e., an adapted model's accuracy on newly available data) and accuracy retention on pre-drift data (i.e., an adapted model's accuracy on previously collected data). Since current state-of-the-art PIDSes do not directly address concept drift, we compare by continuously training alternative PIDSes on the same data as TIDAL to ensure fairness. TIDAL demonstrates significantly better post-drift adaptation accuracy, achieving average 27% higher recall and 31% higher precision when only half of new data is available. Furthermore, TIDAL's training workflow also demonstrates accuracy retention on pre-drift data (mitigating the forgetting of prior knowledge), achieving average 43% higher recall and 38% higher precision than baseline PIDSes.

In summary, this paper makes the following main contributions:

- We present TIDAL, an end-to-end learning framework designed to improve APT detection accuracy in concept drift scenarios. To the best of our knowledge, this is the first PIDS specifically tailored to address concept drift in APT detection.
- We introduce novel model structure and a pre-training/fine-tuning workflow that demonstrates effective transfer learning for APT detection under concept drift.
- TIDAL considers APT detection-specific challenges, supporting timely detection inference without assumptions on input lengths and providing in-depth explanations of results to assist human-led APT investigations.
- Compared to continuous training of existing PIDSes, TIDAL improves both adaptation accuracy on post-drift data and retention accuracy on pre-drift data. TIDAL will be available online at `https://github.com/Froot-NetSys/TIDAL` to facilitate future research.

2 Background and Motivation

2.1 Provenance-based Intrusion Detection Systems (PIDSes) for APT

System audit logs serve as a primary basis for APT detection and investigation. Collected through monitoring tools like Linux Audit [39] and Windows ETW [38], these logs record kernel-level system calls. Each log entry contains a timestamp, an action between system entities (e.g., process p_1 connecting to process p_2), containing file paths and names that represent contextual semantics. To enhance analysis and provide a more structured representation, audit logs can also be represented as provenance graphs (Figure 4), where nodes denote system entities and edges represent relationships such as read or write operations. Provenance graphs offer a powerful abstraction of the log data, summarizing system activities over specific time windows and capturing historical causal structures among system entities.

Provenance-based intrusion detection systems (PIDSes) leverage provenance data from audit logs to drive machine learning approaches for APT detection and investigation. It can be broadly categorized into two main types:

Anomaly-based PIDSes [23, 64, 63, 16, 53, 9, 50] model benign system activities by training on audit system logs known to be free of attacks. They then raise alarms when behavior deviates from their training data. These approaches are *attack agnostic*, in that they do not depend on *a priori* knowledge of specific attacks. As such, they have the potential to capture zero-day attacks. On the other hand, since any logs, whether benign or not, that differ from the training data will be flagged as abnormal, they can be prone to high rates of false positives.

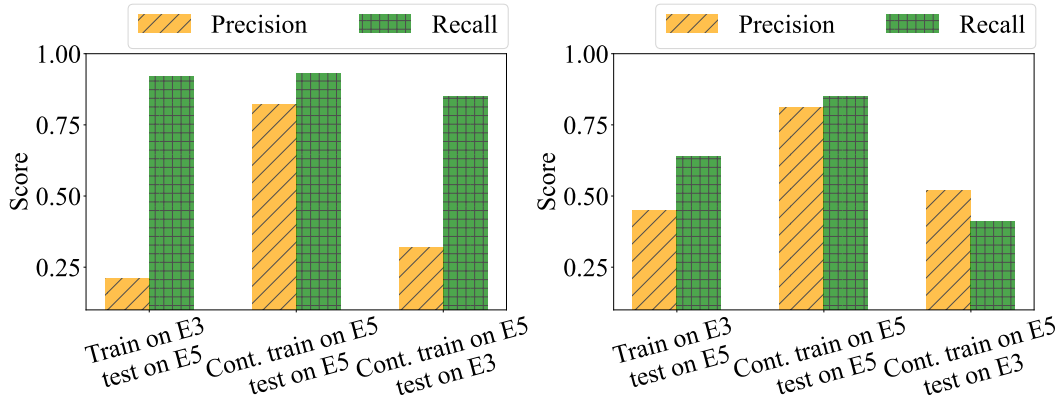
Signature-based PIDSes [5, 17, 1, 33] model specific attack signatures to detect known attack types by training a classification model on both attack and benign data. While these systems show high accuracy in identifying previously encountered attacks and limit false positives, they are constrained to known signatures and may fail to capture zero-day attacks when confronted with new, unseen attack types.

2.2 Concept Drift in APT

In machine learning, *concept drift* describes the evolution of data that invalidates what was learned previously. This leads to a drop in model performance as the relationships learned during training become less relevant [57, 18]. In APT detection, concept drift can manifest in the dynamic evolution of attack patterns, tactics, and malware signatures over time. This phenomenon occurs as sophisticated attackers continuously modify their techniques (including disguised benign patterns) to evade detection systems, making previously learned detection patterns less effective. Recent work by Goyal et al. [19] and Mukherjee et al. [44] demonstrates adversarial mimicry attacks against ML-based PIDSes. These attacks manipulate nodes in the attack graph to mimic benign activities at the embedding level, making it harder for models to distinguish. This highlights the importance of regularly updating detection models to adapt to evolving threat techniques, if PIDSes are to remain resilient against APT.

2.2.1 Motivating Scenario Example

To demonstrate the challenge of model adaptation in APT detection, we simulate a concept drift scenario using two public datasets from the DARPA Transparent Computing (TC) program [15]. The DARPA TC datasets capture red team activities that mimic real-world APT attacks on critical enterprise services such as web, email, and SSH servers, alongside benign activities.



(a) Anomaly-based (Unicorn). **(b)** Signature-based (Atlas).

Figure 1 Detection performance during data drift from E3 Cadets to E5.

We focus on the DARPA E3 [12] and E5 [13] datasets, collected in 2018 and 2019 respectively. These datasets contain similar sub-datasets and attack types, including Cadets, Theia, ClearScope, and FiveDirection (Table 3). To examine shifts in the provenance system over time, we analyze the Cadets sub-dataset from both E3 and E5. While both E3 and E5 Cadets involve a Nginx exploit, the attack signatures evolve: in E3, attackers gain shell access and execute a malicious payload, whereas in E5, they employ HTTP POST requests for remote code execution. Additionally, benign system log patterns shift significantly over time as the red team adapts its behavior to simulate more subtle system activities.

To evaluate model performance under concept drift, we tested one signature-based PIDS and one anomaly-based PIDS by training on E3 Cadets and testing on E5 Cadets without continuous training. As shown in the left bars of Figure 1, the signature-based PIDS experienced substantial drops in precision and recall when applying the E3-trained model to E5, due to its inability to recognize new attack signatures accurately. In contrast, the anomaly-based PIDS achieved high recall on E5 by identifying non-benign behaviors but generated high false positives with low precision, misclassifying new benign patterns in E5 as attacks due to their absence in the E3 training data. We then implemented continuous training on E5 Cadets data to adapt the models, as recommended by previous ML-based PIDS approaches [23, 64, 50, 9]. We retrained the E3 model using E5 Cadets and evaluated it on both E5 and E3 Cadets. As shown in Figure 1 (middle and right bars), performance improved on E5 but declined on E3, highlighting the difficulty of retaining prior accuracy under concept drift and the need for better adaptation strategies in APT scenarios.

2.2.2 Model Adaptation Challenges Under Concept Drift

Building upon our motivating example, we highlight three key challenges for adapting a PIDS under concept drift conditions.

- *Limited new labeled data post-drift:* Retraining detection models requires accurately labeled new APT data, often constrained by resource availability [28]. For example, anomaly-based PIDSes assume that benign training data is captured in a controlled environment, ensuring the absence of attackers [9]. Otherwise, falsely labeled benign data could poison the model, resulting in low precision and recall. Similarly, signature-based PIDSes rely on newly labeled attack data to be accurate and complete so that it learns

from new attack signatures. The key challenge is improving data usage efficiency: with limited new labeled data, how quickly and accurately can detection models adapt to emerging APT scenarios?

- *Accuracy retention on pre-drift data:* Many recent ML-based IDSes for APT detection use deep neural networks (DNNs), including GNNs [64, 50, 9], Transformers [17], and RNNs [5]. When concept drift occurs, existing PIDSes can retrain using both pre-drift data (from prior training) and post-drift data (collected since the last training). However, as data accumulates, this approach becomes impractical, particularly for smaller DNNs due to data capacity limitations. Continuous online training with only new APT data alleviates data capacity issues but risks "catastrophic forgetting", where the model loses previously learned APT knowledge [31, 29]. Thus, the challenge is adapting to post-drift APT data while retaining knowledge from pre-drift data.
- *Insufficiency of traditional concept drift adaptation methods:* While general model drift adaptation strategies (e.g., periodic retraining, online training, and ensemble methods) exist [18, 36], they may not fully address the unique challenges of APT detection. A crafted APT can deploy an initial malicious step and remain dormant for long periods before executing the next phase, requiring models to understand long-term dependencies between system entities. Additionally, timely detection and response are crucial for APT detection and forensic analysis; waiting for a certain amount of data may allow the system to be evaded. Therefore, the challenge lies in designing a model architecture and retraining process that effectively addresses APT-specific issues while adapting to new concept drifts.

Since real-world APT scenarios will likely include concept drift conditions, our goal in this paper is to design a PIDS for APT detection that addresses the above challenges.

3 Threat Model and Assumptions

We consider an APT attacker who performs sparse malicious actions blended with benign behavior, persisting over long periods. The OS, audit logging system, and other security-critical components are part of the trusted computing base (TCB). As in prior provenance-based systems [25, 50, 9, 23, 64, 35, 40], audit logs are assumed to comprehensively capture host activities, including provenance graphs and semantics such as actions, subjects, and paths. Logs are also assumed to be free of misconfigurations, corruption, or compromise, supported by secure logging and tamper-resistant storage [47, 6, 48, 2, 46].

Unlike prior work [9, 50, 5, 55] that assumes complete visibility and stable behavior during training, we account for concept drift. We define concept drift as gradual changes in system behavior that retain structural similarity to earlier patterns, such as new forms of APTs. This motivates Tidal to prioritize both detection accuracy and adaptation efficiency. We also assume that labeled data may be limited or partially inaccurate after drift, due to delayed labeling or adversarial interference. Tidal is therefore designed to learn under weak supervision and maintain robustness when labels are noisy or incorrect.

4 Design of Tidal

Tidal consists of four key modules (Figure 2): 1) a data embedding module that processes streaming system audit logs into contextual and structural embeddings as model input (§4.1); 2) a pre-training module that uses existing APT data at T_0 (pre-drift) to train a Multi-head Transformer (MHT) to learn general knowledge across multiple APT datasets (§4.2); 3) a

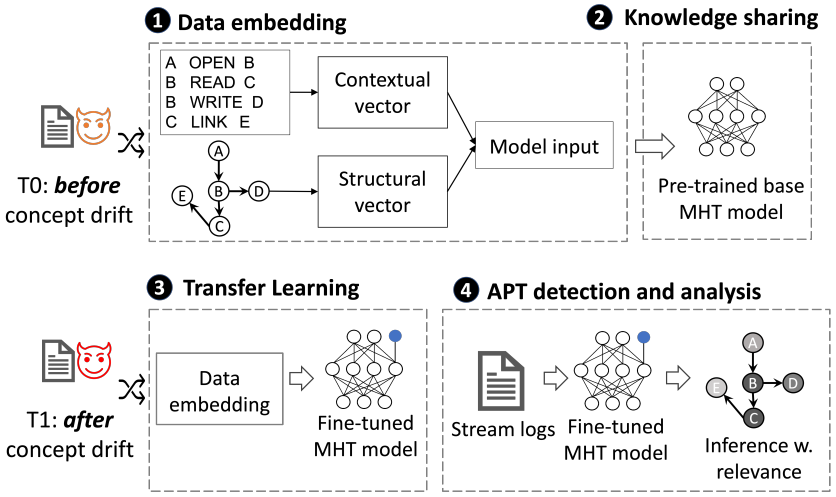


Figure 2 TIDAL Workflow Overview.

Table 2 System log semantics covered by embedding.

System Entity	Attributes	Relationships
Process	image name,	Start, Close, Open
File	path name,	Read, Write, Open, Execute, Resolve, Delete, Fork
Network	ip, src port, dst port	Connect, Send, Receive, Bind

fine-tuning module that transfers previously learned knowledge when new, often limited data becomes available at T_1 (post-drift) (§4.3); and 4) a detection module that deploys the latest model to infer APT attacks in real time, classify instances as benign or anomalous, and output a relevance score to assist human investigation (§4.4).

4.1 Contextual and Structural Data Embedding

At the data embedding stage, TIDAL builds upon domain-specific insights about streaming system audit log data to maximize the signal for APT detection when data is limited. Specifically, we observe that over shorter time frames, the natural language semantics of audit log context may still change dramatically in comparison to the structure of the provenance graph data. Therefore, in order to maximize the signal when system activity may be sparse (such as under drift conditions), TIDAL considers contextual semantics from audit logs as primary, which is then complemented by a careful selection of structural provenance graph data. As a result, TIDAL makes fewer assumptions about audit log length, which allows for greater flexibility with regard to detection granularity.

Contextual embedding for semantics. TIDAL captures audit log semantics for system entities, including types, attributes, and relationships summarized in Table 2. For any relationships that represents a dingle action (e.g., read, connect), it can be easily embedded to a unique vector space. For system entities that has hierarchical structure, TIDAL aims to make its embedded vectors close to each other, so that the model learns their hierarchical relevance in the embedding. To achieve this, TIDAL employs a frequency-based

hierarchical tokenizer for semantic embedding that dynamically adapts as new file paths emerge. Paths, like `/usr/firefox/file_1` and `/usr/firefox/file_2`, are parsed into segments `[usr, firefox, file_1, file_2]` and organized in a hierarchical tree with each segment having a frequency count. When the frequency of a segment surpasses a threshold (default set as 100), it becomes a new token, ensuring commonly used paths remain stable while rare paths are generalized to avoid overfitting. To maintain consistency when token thresholds change, embedding updates are handled through smoothing. For instance, if `/usr/firefox/file_3` surpasses the threshold and becomes a new token, its embedding can be computed as a weighted average of its previous embeddings, ensuring smooth adaptation. This allows the model to balance between continuity and adaptability, even as frequency distributions shift. While we considered existing tokenizers such as Word2Vec [37] and WordPiece [51], these methods do not capture hierarchical similarity. As a result, they cannot ensure that related file paths are embedded closely based on their hierarchical relationships.

Structural embedding for provenance graphs. TIDAL embeds provenance graphs from audit logs to add structural relationships that complement its semantic embedding. Rather than embedding the entire historical provenance graph, TIDAL captures relationships within recently updated subgraphs. It first identifies recent graph updates within the same temporal window as the log input, then constructs a subgraph from all unique nodes and edges, also including one-hop neighbors outside of the current temporal window for each node to capture potential long-term dependencies (sufficient for capturing local structural relationship, see details in §5.4). Afterward, we use a fast graph embedding method [32] to embed the graph into fixed-size vectors to prepare for model input. While TIDAL could integrate other graph embedding methods, such as Graph2Vec [45], these approaches are more computationally demanding than TIDAL's embedding methods, potentially slowing APT detection during inference.

4.2 Knowledge Sharing with Pre-Drift Data

To address the concept drift issue, one common workflow is to pre-train a large model on comprehensive historical data (i.e. pre-drift data), then fine-tune it on new, small and specific datasets (i.e. post-drift data) [54]. However, unlike general intrusion detection tasks, where substantial data is available for pre-training [30], APT detection data is limited. The size of APT datasets is often small, and collecting and labeling new data is challenging [3]. To tackle these limitations, we design a novel model structure tailored for pre-training with limited APT data, based on our analysis of public datasets.

Our analysis of public APT datasets shows that some APTs share certain level of similar strategies. We examine the DARPA TC E3 and E5 [12, 13] datasets, comparing attack types and graph embedding similarity. As shown in Table 3, these datasets feature similar attack types, with the average cosine similarity of graph embeddings (ranging from -1 to 1) also indicating partial similarity across E3 and E5 sub-datasets. This trend extends beyond the DARPA datasets; other APTs, such as APT-42, are known to have historical ties to APT-35, as highlighted in Mandiant reports [41]. Unlike prior approaches that train models for each APT attack separately in implementation [4, 8, 49], we propose pre-training a model across multiple APT attacks to capture shared patterns, enhancing model performance on limited datasets. Joint pre-training enables the model to learn comprehensive system pattern and attack strategies, improving its adaptability and robustness. For this purpose, we design a multi-head Transformer model optimized for cross-APT learning.

Table 3 DARPA TC APT dataset sshare similar attack types from E3 (2018) to E5 (2019) deployments.

	Cadets	Theia	Trace	FiveD
E3 dataset	Nginx backdoor	Firefox backdoor	Firefox backdoor; Browser Extension	Firefox backdoor; Browser Extension
E5 dataset	Nginx backdoor	Firefox backdoor	Firefox injection	Firefox backdoor
Avg. graph embed similarity	0.65	0.48	0.34	0.51

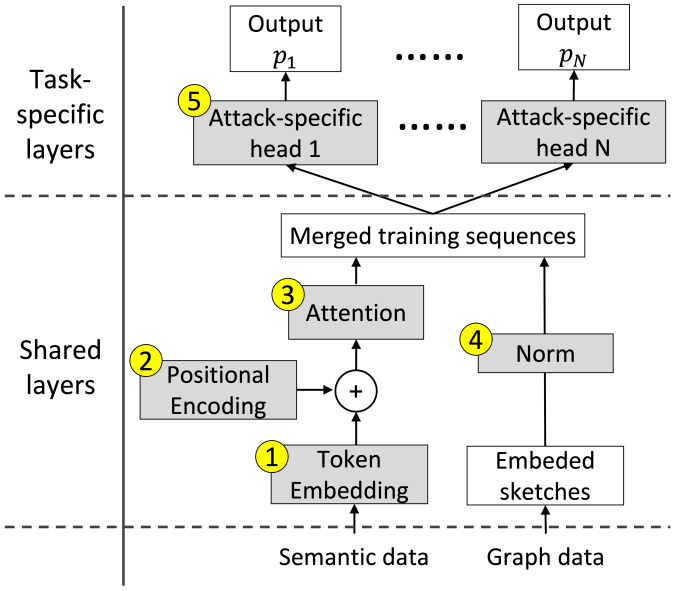


Figure 3 TIDAL's Multi-head Transformer (MHT) architecture. Layers 1-4 are shared among all attacks, while layer 5 is unique to each attack case.

Multi-head Transformer (MHT) model. The MHT model is pre-trained on a combined dataset of multiple APT datasets to learn general representations while maintaining attack-specific knowledge. Although Transformers are commonly used for cross-domain learning [58], we find that a standard Transformer architecture lacks the capacity for effective knowledge sharing in APT detection due to unique patterns within each attack (Vanilla Transformer baseline in §5.4). To achieve knowledge sharing without sacrificing specificity, TIDAL introduces a Multi-Head Transformer (MHT) model optimized for learning across APTs. As shown in Figure 3, the architecture consists of: ① a token embedding layer for embedding semantic data from system audit logs, ② a positional encoding layer to represent token positions within a sequence, ③ a self-attention layer processing the outputs from ① and ②, ④ a normalization layer capturing causal relationships on the provenance graph, merging the outputs of ③ and ④ via vector concatenation, and ⑤ attack-specific heads (signature-based or anomaly-based) specializing in each APT dataset.

TIDAL propose a novel architecture to flexibly supports both signature-based and anomaly-based detection by adapting training data and loss functions at the last layer.

- **Signature-based classification head:** To use TIDAL as a signature-based classification system, denoted as TIDAL (signature), the training data will contain both attack and benign data samples. At the last layer, we use cross-entropy loss for classification, where

N represents number of training data samples, y_i represents the label of each sample (0 for benign, 1 for attack), and $\hat{y}_i$ represents predicted probability of the sample being an attack.

$$L_{signature} = -\frac{1}{N}\sum_{i=1}^{N}(y_i log\,(\hat{y}_i) + (1-y_i) log(1-\hat{y}_i)) \tag{1}$$

- **Anomaly-based reconstruction head:** to use TIDAL as an anomaly-based detection system, denoted as TIDAL (anomaly), the training data will contain only benign data samples. At the last layer, we use Mean Squared Error (MSE) for reconstruction loss, where N represents number of benign data samples, x_i represents the original benign data samples, and $\hat{x}_i$ represents reconstructed samples from the model.

$$L_{anomaly} = \frac{1}{N}\sum_{i=1}^{N}\|x_i - \hat{x}_i\|^2 \tag{2}$$

After selecting the detection system type, TIDAL pre-trains the model with existing data and creates a unique head for each specific attack. This structure allows shared layers to capture general knowledge across APT attacks, while attack-specific heads enable specialization and adaptation to individual attacks, as detailed in Algorithm 1.

Algorithm 1 Pre-training on Pre-drift Data.

Input: Data embeddings at pre-drift T_0
Output: Pre-trained APT-detection model
Initialize Ω_{shared} ▷ Initialize shared transformer layers
for each attack dataset t **do**
    Initialize H_t ▷ Initialize attack-specific head
end for
for each epoch **do**
    for each batch in combined dataset **do**
        Compute shared representation:
        $\theta^{T_0} \leftarrow$ FORWARDPASS($batch.input$)
        for each task t **do**
            Compute head-specific prediction:
            $pred[t] \leftarrow$ FORWARDPASSHEAD(t, θ^{T_0})
            Compute loss for task t:
            $L_{T_0}(\theta) \leftarrow$ LOSSFUNCTION($pred[t], batch.labels[t]$)
        end for
        Backpropagation: BACKWARDPASS($L_{T_0}(\theta)$)
        Update model parameters θ^{T_0}
    end for
end for

4.3 Transfer Learning to Post-Drift Data

After pre-training on existing data, the model must be regularly updated to handle concept drift as new attacks appear. Updates are critical for both signature- and anomaly-based PIDS to recognize new attack signatures and evolving benign patterns. This reduces false negatives in signature-based systems and false positives in anomaly-based systems. Since labeled data for new attack is scarce [22], the model must adapt efficiently to emerging threats with limited samples [3].

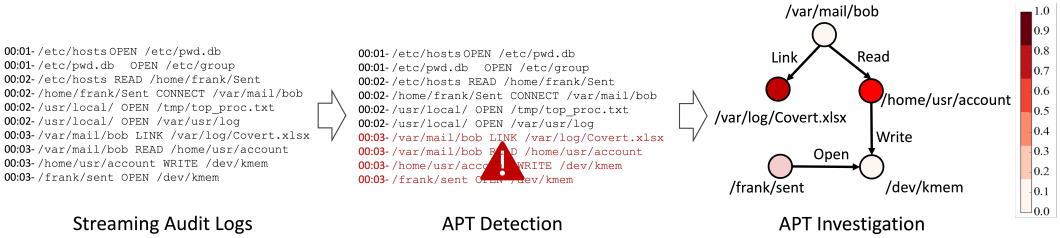


Figure 4 TIDAL flags the activity at timestamp 00:03 as malicious and raises an alarm. With a relevance score threshold as 0.5, it reveals that `/var/log/Covert.xlsx` is malicious. Although reading `/home/usr/account` is not directly related to attack, it is part of the attack activity because `/var/mail/bob` has been exploited. Other nodes are likely to be benign.

TIDAL leverages previously learned system patterns to adapt to concept drift. For example, while specific backdoor exploitation attacks may differ, they often share common behaviors in system audit logs, such as stealthy operations that appear benign but later contribute to an attack. By initially training a base model to capture general representations across multiple APTs, we can efficiently fine-tune it on smaller datasets to adapt to emerging attacks. This approach, introduced as a form of transfer learning, allows TIDAL to address new threats as new system provenance data becomes available.

Transfer learning methodology. Transferring knowledge to new APT attacks requires adapting the model to capture unique features of emerging threats. However, a key challenge is the risk of the model over-adjusting to new data, potentially "forgetting" important patterns from its original pre-training. Specifically, if the model is pre-trained on data T_0 with parameters optimized the loss on T_0: $\theta^{T_0} = \underset{\theta}{argmin} L_{T_0}(\theta)$. When fine-tuning on new APT data T_1, the model parameters are updated to minimize the loss on T_1: $\theta^{T_1} = \underset{\theta}{argmin} L_{T_1}(\theta)$. The forgetting issue occurs when the optimization for minimizing $L_{T_1}(\theta)$ conflicts with minimizing $L_{T_0}(\theta)$. As a result, the updated parameters θ^{T_1} may no longer be optimal for the original T_0, leading to an accuracy drop on prior data.

To mitigate forgetting, TIDAL employs three fine-tuning strategies. First, rather than updating all shared layers, TIDAL freezes the first half of the encoder layers, only fine-tuning the remaining half. The fine-tuned parameters $\theta^* = \underset{\theta^*}{argmin} L_{T_1}(\theta_{frozen}, \theta^*)$, where θ_{frozen} prevent fine-tuning on T_1 from disrupting knowledge preserved from the pre-training data T_0. This allows general representations captured by the frozen layers on T_0 to remain intact, while enabling specific adjustments for T_1. Second, TIDAL applies a lower learning rate during updates to the unfrozen layers, allowing gradual adaptation to T_1, which balances influences from both tasks. This also helps stabilize the fine-tuning by limiting parameter shifts at each step. Finally, to effectively capture new attack-specific features, TIDAL initializes a new head layer for the new APT dataset, isolating unique patterns while retaining broad knowledge. The complete fine-tuning approach is detailed in Algorithm 2.

4.4 Attack Inference and Investigation Helper

After the model is fine-tuned for a drifted scenario, TIDAL performs attack inference with aggregated results from multi-heads. Using TIDAL (anomaly) as an example, each anomaly detection head $n \in \{1, \cdots N\}$ provides an output p_n, representing the probability that the n_{th} head classifies the sample as benign. If p_n is smaller than the detection threshold (set as

Algorithm 2 Transfer Learning on Post-drift Data.

Input: Pre-trained model; Data embeddings at post-drift T_1
Output: Fine-tuned model for new APT
procedure KnowledgeTransfer($task, data$)
    Load Ω_{shared} ▷ Load pre-trained shared layers
    Initialize H_i ▷ Initialize a new head
    Freeze θ_{frozen} of the first half of Ω_{shared} ▷ Prevent updates to initial shared layers for knowledge retention
end procedure
for each epoch **do**
    for each batch in *data* **do**
        Compute shared representation:
        $\theta^{T_1} \leftarrow$ ForwardPassSharedLayers($batch.input$)
        Compute task-specific prediction:
        $pred \leftarrow$ ForwardPassHead(t, θ^{T_1})
        Compute loss:
        $L_{T_1}(\theta) \leftarrow$ LossFunction($pred, batch.labels$)
        Backpropagation: BackwardPass($L_{T_1}(\theta)$)
        Update model parameters θ^{T_1}
    end for
end for

0.5), it means the decision results from current head n is anomaly. The final decision for anomaly detection is based on the aggregation of the decisions from all N heads. If any one of the N heads detects an anomaly (i.e., classifies the sample as an anomaly), then the final decision R is anomaly, as shown below.

$$R = \begin{cases} Anomaly,\ if\ \exists n \in N,\ p_n = anomaly \\ Benign,\ if\ \forall n \in N,\ p_n = benign \end{cases} \tag{3}$$

Tidal (signature) uses the same aggregation logic, except its p_n represents the probability of the input being an attack. Therefore, with Tidal (signature), if p_n is greater than the detection threshold (set as 0.5), then the decision from head n is classified as an attack.

Post inference, attack explanation is essential for analysts to effectively investigate and mitigate future threats [27]. APT often persist in systems for long periods, disguising themselves among benign operations. Existing systems [26, 5] that reconstruct attack story graphs typically include thousands of entities labeled as either attack-related or benign, which can overwhelm experts during review. Therefore, assessing the relevance of each system event is helpful for pinpointing malicious activities.

Relevance Score. Tidal leverages the Transformer's self-attention mechanism to provide insight into specific model decisions [11]. Self-attention weights is trained through back-propagation [7], represent dependencies across entities by linking input to various parts of the neural network. The model's pairwise attention mechanism allows distant elements to interact, enabling the capture of long-range dependencies over extended time frames.

To enhance explainability, Tidal aggregates attention weights across all self-attention heads, generating an average relevance score for each system entity. Let $A_i^{(h)}$ denote the attention weight of the i_{th} entity in self-attention head. The relevance score S_i is computed by averaging the attention weights across all heads H (heads number set as 8), and normalizing across entities in the current input sequence:

$$S_i = \frac{1}{H} \sum_{h=1}^{H} \frac{A_i^{(h)}}{\sum_j A_j^{(h)}} \tag{4}$$

This approach provides a quantitative measure of each entity's importance in the model's decision-making process. By highlighting the most critical events and interactions, the relevance scores offer context to the APT detection results. Visualizing these scores enables security analysts to swiftly focus on pivotal attack operations, facilitating faster and more accurate responses to potential threats.

A Running Example. To demonstrate TIDAL's deployment functionality, we use a phishing email APT log snapshot from DARPA-TC3. In this scenario, an attacker sends a spreadsheet attachment to Bob containing an encoded Powershell command, which downloads and executes a script, resulting in a remote connection. The attacker then surveys the target using shell access, including reading user account files. TIDAL processes streaming audit logs, aggregating them at each timestamp for input to the detection model. While it classifies inputs from the first two timestamps as benign, it flags activity at timestamp 00:03 as malicious, triggering an alarm. Figure 4 shows the reconstructed graph used for investigation, with relevance scores calculated for each internal event.

5 Evaluation

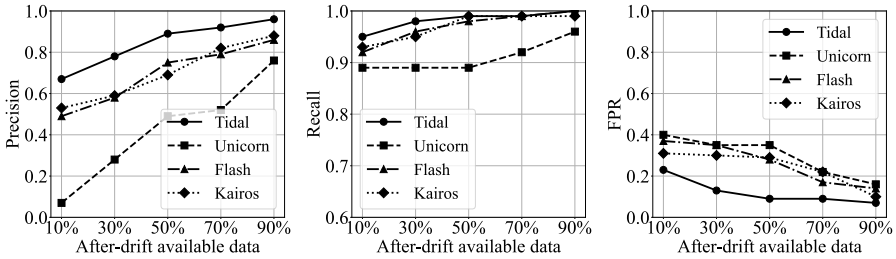
TIDAL is implemented in approximately 3,500 lines of Python and 400 lines of C++ code. All experiments are deployed on an Ubuntu 18.04 system equipped with an NVIDIA RTX A100 GPU. To thoroughly evaluate TIDAL when concept drift occurs in APT, we address the following research questions:

- RQ1: How effective is TIDAL compared to continuously training the baselines when concept drift happens?
- RQ2: Does TIDAL support accurate detection with varied lengths of input data?
- RQ3: How concise is TIDAL's relevance score for attack investigation?
- RQ4: Is TIDAL robust against adversarial scenarios?
- RQ5: How do TIDAL's design choices hold up under an ablation study?

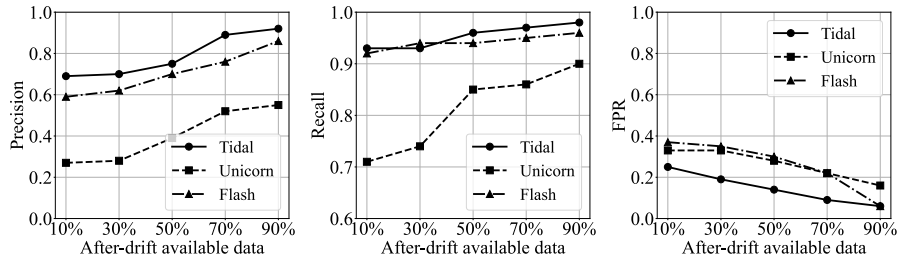
Concept Drift Evaluation Methodology. Concept drift evaluation in APT detection is challenging due to data paucity and ambiguous drift definitions [9, 50]. Existing APT datasets like StreamSpot [42] and OpTC [14] capture attack patterns at fixed points in time but lack temporal evolution patterns necessary for systematic drift evaluation.

We use DARPA's Transparent Computing (TC) datasets E3 and E5 [15] due to their temporal separation (from April 2018 to May 2019) and comparable attack scenarios. We define E5 as concept-drifted from E3 based on this temporal separation. This enables systematic drift evaluation across 129GB of data spanning Linux (Cadets, Theia, Trace) and Windows (FiveDirection) scenarios.

Data Labeling. We label data using attack timestamps from DARPA's ground truth [12, 13]: events within attack periods are marked as attacks, others as benign. Attack-related keywords in the ground truth help identify malicious system entities, which we use to manually compare TIDAL 's investigation graphs with ground-truth graphs. Details on training, validation, and detection splits are in the Appendix.



(a) Linux system drift.



(b) Windows system drift.

Figure 5 Post-drift effectiveness comparison with anomaly-based PIDSes.

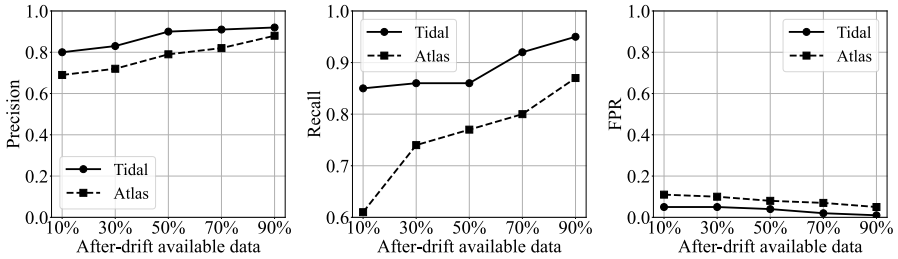
Comparison Study. TIDAL is the first APT detection system designed for concept drift, making direct comparison with existing PIDS challenging. To ensure fairness, we implement continuous training for other ML-based PIDSes to adapt to drift. We compare TIDAL with signature-based ATLAS [5] and anomaly-based Unicorn [23], Flash [50], and Kairos [9]. Selection prioritizes open-source availability rather than perceived importance. Broader systems are discussed qualitatively in §7.

Metrics. We evaluate performance under concept drift using Precision (P), Recall (R), and False Positive Rate (FPR). Precision measures the accuracy of positive predictions, Recall measures the ability to identify true attacks, and FPR quantifies benign misclassification. Together they offer a balanced view of detection performance, supporting meaningful comparisons and addressing real-world concerns such as accurate detection and reduced alert fatigue. Additional metrics are discussed in later subsections.

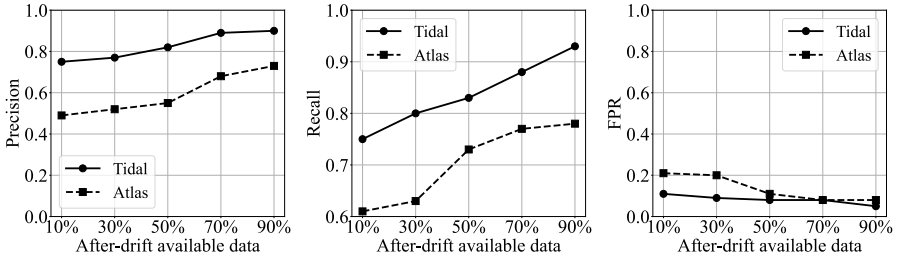
5.1 Performance Under Concept Drift

We evaluate TIDAL under concept drift along two dimensions: *adaptation accuracy on post-drift data* and *retention of pre-drift accuracy*. This assesses how well models adapt to new patterns with limited labeled data and whether adaptation preserves prior detection capabilities.

Adaptation Evaluation. We first compare TIDAL with other PIDSes under stable conditions without drift to reproduce their reported performance. This establishes a baseline for both signature- and anomaly-based systems. We exclude Kairos results on FiveDirection because its implementation did not support this dataset and we could not tune it successfully. Table 4 shows all anomaly-based systems, including TIDAL, achieve similarly high detection accuracy and low false positives under stable conditions. For ATLAS, which did not originally use



(a) Linux system drift.



(b) Windows system drift.

Figure 6 Post-drift effectiveness comparison with signature-based PIDSes.

Table 4 *Pre-drift* results without addressing concept drift.

	Cadet (E3)			Theia (E3)			Trace (E3)			FiveD (E3)		
	P↑	R↑	FPR↓	P↑	R↑	FPR↓	P↑	R↑	FPR↓	P↑	R↑	FPR↓
ATLAS	0.85	0.88	0.05	0.90	0.91	0.03	0.87	0.95	0.02	0.89	0.89	0.05
TIDAL (signature)	0.99	1.00	0.01	0.95	0.99	0.01	0.99	1.00	0.02	0.95	1.00	0.01
Unicorn	0.98	1.00	0.01	1.00	1.00	0.00	0.92	1.00	0.02	0.65	0.91	0.03
Flash	0.94	0.99	0.01	0.92	0.99	0.01	0.95	0.99	0.01	0.72	0.93	0.01
Kairos	0.80	1.00	0.01	0.91	1.00	0.01	0.93	1.00	0.01	N/A	N/A	N/A
TIDAL (anomaly)	0.95	1.00	0.01	0.91	1.00	0.01	0.95	0.99	0.01	0.88	0.97	0.01

DARPA TC data, we adapted its embedding module for compatibility while keeping its model structure unchanged. We attribute any performance gap between TIDAL (signature) and ATLAS to unoptimized hyperparameters on this dataset.

We then simulate drift from E3 to E5, training all models with the same amount of data during the transition. Because Linux (Cadets, Theia, Trace) and Windows (FiveDirection) logs differ significantly, we treat them as separate drift scenarios. Figure 5 shows that with equal amounts of new labeled data, all anomaly-based systems achieve high recall even at 10% of new data. However, TIDAL (anomaly) improves precision by more than 34% with less than 50% new data, reducing false alarms more effectively than others. As labeled data approaches 90%, differences narrow. A similar pattern appears for signature-based systems in Figure 6. Because they rely on attack-labeled data, which is scarcer than benign labels under drift, both TIDAL (signature) and ATLAS show reduced recall. Nevertheless, TIDAL (signature) adapts more effectively than ATLAS. We attribute TIDAL 's overall advantage to its ability to learn broad behavioral patterns during pre-training and transfer them to drifted data during fine-tuning, even with limited new labels.

Table 5 *Post-drift* model test on *pre-drift data* with knowledge retention.

	Cadet (E3)			Theia (E3)			Trace (E3)			FiveD (E3)		
	P↑	R↑	FPR↓	P↑	R↑	FPR↓	P↑	R↑	FPR↓	P↑	R↑	FPR↓
ATLAS	0.23	0.51	0.56	0.42	0.41	0.33	0.47	0.35	0.42	0.29	0.38	0.54
TIDAL (signature)	0.84	0.92	0.03	0.85	0.91	0.05	0.83	0.88	0.03	0.80	0.89	0.04
Unicorn	0.58	0.69	0.08	0.65	0.87	0.05	0.50	0.61	0.10	0.52	0.53	0.12
Flash	0.57	0.90	0.04	0.62	0.85	0.03	0.70	0.76	0.08	0.68	0.81	0.05
Kairos	0.64	0.85	0.04	0.61	0.79	0.04	0.72	0.79	0.05	N/A	N/A	N/A
TIDAL (anomaly)	0.85	0.94	0.03	0.78	0.96	0.02	0.85	0.97	0.02	0.83	0.93	0.04

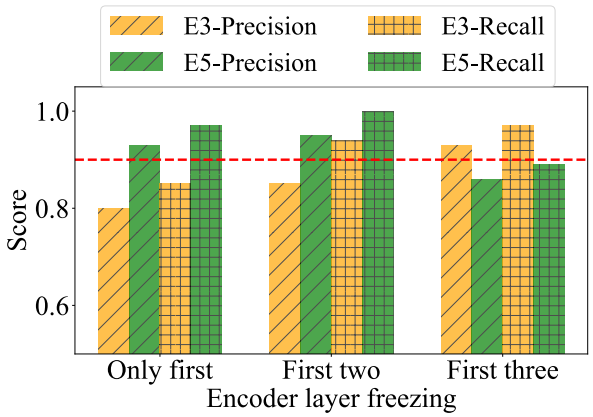


Figure 7 Layer-freezing effect on knowledge retention.

Knowledge Retention Evaluation. A key challenge in fine-tuning under drift is catastrophic forgetting of prior knowledge. We evaluate models updated with 90% of E5 labels on the pre-drift E3 dataset. Table 5 shows that all PIDSes decline relative to Table 4, but other systems experience 15–40% precision loss and 20–50% recall loss, whereas TIDAL drops only 7–15% in precision and 2–11% in recall.

We attribute TIDAL 's reduced forgetting to two design choices. First, the Transformer's self-attention mechanism emphasizes relevant input features, helping retain knowledge from older data. Second, freezing early encoder layers preserves pre-drift representations while allowing later layers to adapt. We test TIDAL (anomaly) with different numbers of frozen encoder layers (1, 2, and 3 of 4 total) and evaluate on E5 Cadets (post-drift) and E3 Cadets (pre-drift). Figure 7 shows the trade-off: freezing three layers maximizes retention but hurts adaptation on E5, while freezing one layer maximizes E5 accuracy but increases forgetting on E3. TIDAL freezes the first two encoder layers to balance performance between pre- and post-drift data.

5.2 Attack Investigation Assistance

While some previous PIDSes [5] focus on reconstructing an entire attack graph after processing a big chunk of data (which may require hours of data accumulation), TIDAL is designed to provide explanations for each input shortly after detection (e.g., within a second). Specifically, TIDAL calculates a relevance score for each system event in relation to the model's decision for the current input. This score is then used to generate an investigation-assistance graph,

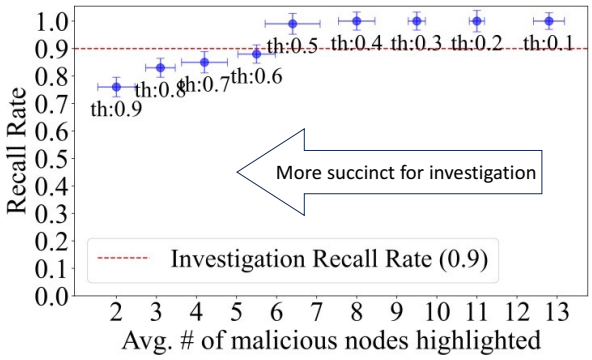


Figure 8 Investigation tradeoff with different threshold of relevance score.

where the relevance of each event is visualized according to its impact on the model's decision, as shown in Figure 4. A tricky challenge is setting an appropriate threshold for the relevance score to filter highlighted, attack-relevant events. Ideally, these filtered events should be both accurate (covering all malicious activities) and concise (minimizing false alarms that might confuse investigators).

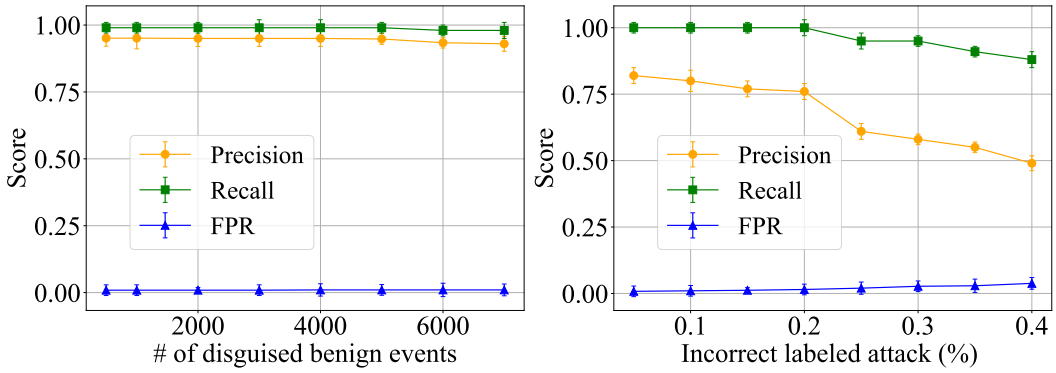
We note that the DARPA APT dataset lacks explicit system event-level ground truth for each attack, making it challenging to precisely evaluate the precision of reconstructed malicious events, though approximate recall can still be assessed. To evaluate the accuracy of relevance scores, we use keywords mentioned in the ground truth report for each attack as a proxy for ground truth, which has also been employed by recent APT detection PIDSes [9].

For each input detected as an attack in the E3 dataset, we vary the threshold from 0.1 to 0.9 to filter the attack relevant system events. In Figure 8, the x-axis shows the average number of system events highlighted as malicious at each threshold, while the y-axis represents recall compared to ground truth keywords. At a low threshold (e.g., 0.1), recall remains high but includes numerous nodes as malicious, potentially increasing investigation overhead for human analysts. When the threshold is too high (e.g., 0.9), the highlighted malicious nodes are concise but may omit relevant attack events. Therefore, we recommend that security analysts select a balanced threshold that aligns with their accuracy and resource requirements. In the E3 dataset, relevance threshold as 0.5 offers a reasonable balance.

5.3 Robustness against Adversarial Scenario

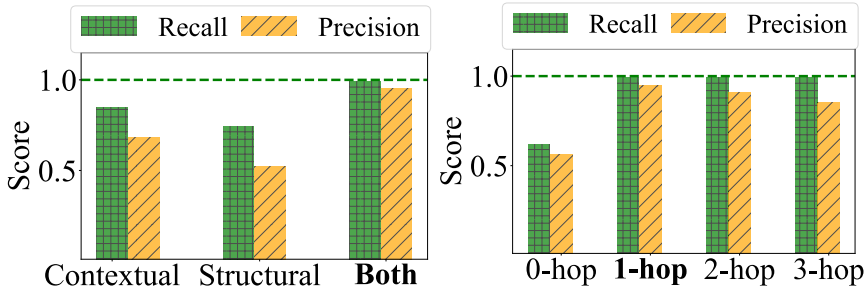
One key concern in ML-based PIDS is robustness under imperfect conditions. Prior work [19, 44] shows that graph-based IDS can be vulnerable to adversarial mimicry, where attackers manipulate node embeddings to resemble benign behavior. Concept drift can also lead to mislabeled data, either from user error or adversarial contamination. To assess TIDAL 's robustness, we simulate two test cases.

Adversarial Mimicry Robustness. We simulate mimicry attacks by injecting benign events into E3 Trace attacks. These events are randomly sampled from the benign pool to resemble real benign behavior. Figure 9a shows that TIDAL maintains strong detection performance even with thousands of added benign events. This resilience results from TIDAL 's data embeddings, which combine contextual and structural information. Even when the provenance graph resembles benign patterns, semantic indicators such as operations on sshd files flag high suspicion. Additionally, the graph embedding includes one-hop neighbors, so benign-looking noise linked to prior attacks still receives a high anomaly score.



(a) Detection w/ mimicry attacks. **(b)** Training w/ data contamination.

Figure 9 Robustness analysis against adversarial scenarios.



(a) Ablation study. **(b)** Embedded neighbors.

Figure 10 Design choice for data embedding.

Robustness to False Labels. High-quality labels are difficult to obtain in APT settings, especially during concept drift. Users may report erroneous alarms, and attackers may poison data with mislabeled inputs. We evaluate TIDAL by progressively injecting false positive (attack) labels into the E3 Trace training set, from 5% up to 40%. For each setting, we train with five random seeds and report mean and standard deviation.

Figure 9b shows that as false positive noise increases, precision drops by about 25% but recall stays above 0.85 and the false positive rate remains near 0.01. When less than 20% of data is mislabeled, TIDAL shows good resilience. We attribute this to the multi-head design in TIDAL 's MHT model, where gradients from noisier heads are averaged with those from cleaner heads, reducing their impact.

5.4 Design Choice Analysis

We evaluate TIDAL's design across data embedding, model architecture, and training hyper-parameters using the E3 Trace dataset.

Data Embedding. We ablate the use of semantic and graph embeddings. Figure 10a shows both are essential for detection accuracy, with semantic embedding contributing more. Semantics help distinguish short logs more effectively than graph structure alone. For graph embedding, we include 1-hop neighbors in each log sequence to capture connectivity with historical context. Figure 10b compares 0-hop to 4-hop settings. Accuracy drops sharply at 0-hop due to missing context. Using more than 1-hop slightly reduces precision, likely due to compressed representations in fixed-size embeddings. We recommend using 1-hop for the best balance.

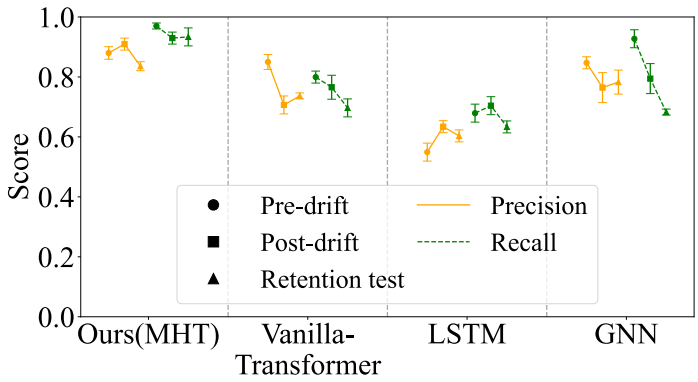
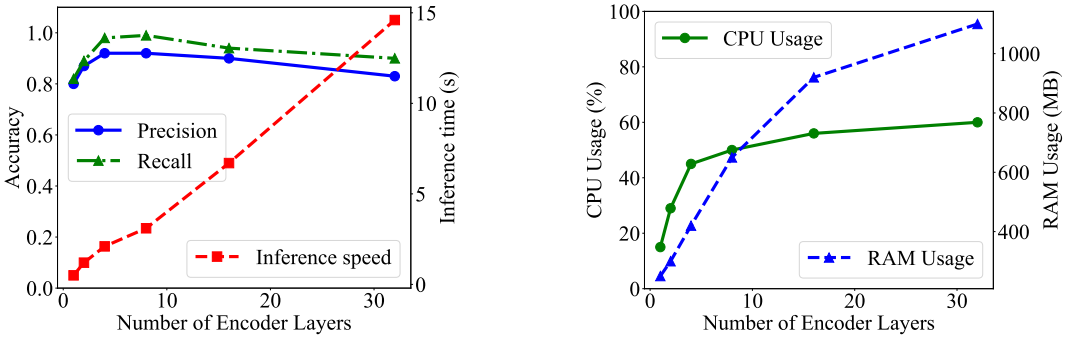


Figure 11 Model choice comparison.



(a) Detection performance.

(b) Resource efficiency.

Figure 12 Design choice for data embedding.

Multi-head Transformer Architecture. TIDAL uses a multi-head Transformer to improve adaptability to concept drift. We compare it with three common ML-based PIDS models, each pre-trained on E3 and fine-tuned on E5. Evaluation follows §5.1: performance on E3 (pre-drift), E5 (post-drift), and knowledge retention from E3. Figure 11 shows the vanilla Transformer performs well on E3 but fails to adapt post-drift. Adding a new head in TIDAL preserves pre-drift knowledge while adapting to new data. LSTM performs worse due to its limited ability to model long dependencies and separate post-drift patterns. GNNs handle pre-drift well but degrade post-drift, likely due to structural graph changes.

Encoder Layer Depth. We test encoder depths from 1 to 32 layers on E3 Cadets to analyze the trade-off between performance and resource usage. Figure 12a shows accuracy improves with more layers but declines after a point due to overfitting. Inference time grows linearly. Figure 12b shows CPU efficiency improves but plateaus around eight layers, while RAM usage increases significantly with deeper models. Based on this, TIDAL uses 4 encoder layers for optimal accuracy, reasonable inference time (around 2.1s), and balanced resource usage.

6 Discussion and Limitations

Concept drift evaluation as a research challenge. One limitation TIDAL faces is the lack of APT detection datasets that exhibit realistic temporal evolution. While the DARPA TC datasets support our drift simulation for evaluation, they capture only a limited subset of how

behavior changes over time. This dataset gap also highlights several important directions for future work. Mirroring ongoing research in networking and systems on applying machine learning when the boundary between offline training and online deployment continually shifts [62, 61, 59], future APT detection research could focus on: (1) developing long-term, continuously adapting datasets with documented drift patterns; (2) establishing standardized evaluation platforms; (3) defining metrics that separate meaningful drift from benign noise; and (4) building frameworks for sharing sensitive attack data while preserving privacy.

Adapting to attacks with zero similarity. TIDAL's workflow operate under a threat model where the data represents concept drift, i.e., while the attack and benign activities evolve over time, some underlying similarity remains between pre-drift and post-drift data. If there is an abrupt and complete change in the data distribution, meaning no common features or relationships exist between the old and new data, it goes beyond the concept drift scope and is typically referred to as concept shift or even domain shift. In such cases, TIDAL is no longer optimal for adaptation and may require full model retraining to address the shift. While TIDAL takes a step forward by addressing concept shift scenarios, adapting models for future, unpredictable attacks remains an open challenge that needs further discussion.

7 Related Work

There exists many anomaly-based PIDSes for APT detection. UNICORN [23], DISTDET [16], Log2vec [34] and TBDetector [53] transforms streaming graphs into fixed-size sketches, which are then clustered to identify anomalies. FRAPpuccino [24] uses a windowing technique for graph analysis. ShadeWatcher [64] introduces a recommendation system, allowing the model to be further refined after encountering false positives. PrioTracker [35] computes priority scores based on node statistics to improve accuracy. NoDoze [25] reduces false positives by propagating and aggregating anomaly scores in dependent sub-graphs. ProGrapher [63] embeds the entire graph and learns normal system behaviors using RCNN. NODLINK [33] focuses on 10-second-level detection granularity and uses an online optimization algorithm to model the graph.

Several studies have used NLP techniques to detect anomalies and attacks in log data. DeepLog [17], LogShield [1], LogBERT [20] employ NLP models to capture patterns in benign log events and identify deviations that may indicate malicious activity. LogAnomaly [43] takes a different approach, training a LSTM model to detect both sequential and quantitative anomalies in log data. It uses Template2Vec [43] to embed semantic information from synonyms and antonyms, which helps the model understand the relationships between log entries. However, simply using discrete symbols from log files can make it challenging to capture the temporal relationships between events. To address this limitation, Wang et al. [56] proposed OC4Seq, which learns sequential information from both global and local perspectives. While effective, the forget gate in Gated Recurrent Units (GRUs) [10] used by OC4Seq may cause it to lose information over time, which could be problematic for detecting long-term APT.

Compared to existing systems, TIDAL focuses specifically on enhancing model adaptation accuracy and efficiency in the presence of concept drift. Unlike traditional concept drift scenarios, APT environments present unique challenges where adversarial evolution demands specialized approaches. We view TIDAL's insights and workflow complements existing systems, offering a foundation to explore new advancements in handling evolving attack scenarios.

8 Conclusions

We present TIDAL, a provenance-based intrusion detection system designed to address concept drift in APT datasets. TIDAL employs a multi-head Transformer model with a pre-training and fine-tuning workflow to enhance adaptation accuracy and efficiency during concept drift. It achieves strong knowledge retention, minimizing the forgetting of previous data. TIDAL tackles APT-specific challenges by supporting flexible input log lengths and providing explainability for model decisions. Our experiments demonstrate that TIDAL maintains high accuracy in concept drift scenarios, even with limited new data, and exhibits robustness against adversarial attacks and mislabeled data.

References

1 Sihat Afnan, Mushtari Sadia, Shahrear Iqbal, and Anindya Iqbal. Logshield: A transformer-based apt detection system leveraging self-attention. *arXiv preprint arXiv:2311.05733*, 2023. doi:10.48550/arXiv.2311.05733.

2 Adil Ahmad, Sangho Lee, and Marcus Peinado. Hardlog: Practical tamper-proof system auditing using a novel audit device. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1791–1807. IEEE, 2022. doi:10.1109/SP46214.2022.9833745.

3 Khandakar Ashrafi Akbar, Yigong Wang, Md Shihabul Islam, Anoop Singhal, Latifur Khan, and Bhavani Thuraisingham. Identifying tactics of advanced persistent threats with limited attack traces. In *Information Systems Security: 17th International Conference, ICISS 2021, Patna, India, December 16–20, 2021, Proceedings 17*, pages 3–25. Springer, 2021. doi:10.1007/978-3-030-92571-0_1.

4 Abdulellah Alsaheel, Yuhong Nan, Shiqing Ma, Le Yu, Gregory Walkup, Z Berkay Celik, Xiangyu Zhang, and Dongyan Xu. Atlas open-sourced code. https://github.com/purseclab/ATLAS.

5 Abdulellah Alsaheel, Yuhong Nan, Shiqing Ma, Le Yu, Gregory Walkup, Z Berkay Celik, Xiangyu Zhang, and Dongyan Xu. {ATLAS}: A sequence-based learning approach for attack investigation. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3005–3022, 2021. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/alsaheel.

6 Adam Bates, Dave Jing Tian, Kevin RB Butler, and Thomas Moyer. Trustworthy {Whole-System} provenance for the linux kernel. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 319–334, 2015.

7 Adrien Bibal, Rémi Cardon, David Alfter, Rodrigo Wilkens, Xiaoou Wang, Thomas François, and Patrick Watrin. Is attention explanation? an introduction to the debate. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3889–3900, 2022. doi:10.18653/V1/2022.ACL-LONG.269.

8 Zijun Cheng, Qiujian Lv, Jinyuan Liang, Yan Wang, Degang Sun, Thomas Pasquier, and Xueyuan Han. Kairos open-sourced code. https://github.com/ProvenanceAnalytics/kairos/tree/main.

9 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 *2024 IEEE Symposium on Security and Privacy (SP)*, pages 3533–3551. IEEE, 2024. doi:10.1109/SP54263.2024.00005.

10 Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

11 Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*, 2019. arXiv:1906.04341.

12 DARPA transparent computing e3. https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md.

13 DARPA transparent computing e5. https://github.com/darpa-i2o/Transparent-Computing.

14 Operationally transparent cyber (OpTC) data release. https://github.com/FiveDirections/OpTC-data.

15 DARPA transparent computing program. https://www.darpa.mil/program/transparent-computing.

16 Feng Dong, Liu Wang, Xu Nie, Fei Shao, Haoyu Wang, Ding Li, Xiapu Luo, and Xusheng Xiao. Distdet: A cost-effective distributed cyber threat detection system. *USENIX Security Symposium*, 2023.

17 Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 1285–1298, 2017. doi:10.1145/3133956.3134015.

18 João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014. doi:10.1145/2523813.

19 Akul Goyal, Xueyuan Han, Gang Wang, and Adam Bates. Sometimes, you aren't what you do: Mimicry attacks against provenance graph host intrusion detection systems. In *30th Network and Distributed System Security Symposium*, 2023.

20 Haixuan Guo, Shuhan Yuan, and Xintao Wu. Logbert: Log anomaly detection via bert. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021. doi:10.1109/IJCNN52387.2021.9534113.

21 Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):87–110, 2022. doi:10.1109/TPAMI.2022.3152247.

22 Xueyuan Han, James Mickens, Ashish Gehani, Margo Seltzer, and Thomas Pasquier. Xanthus: Push-button orchestration of host provenance data collection. In *Proceedings of the 3rd International Workshop on Practical Reproducible Evaluation of Computer Systems*, pages 27–32, 2020. doi:10.1145/3391800.3398175.

23 Xueyuan Han, Thomas Pasquier, Adam Bates, James Mickens, and Margo Seltzer. Unicorn: Runtime provenance-based detector for advanced persistent threats. *arXiv preprint arXiv:2001.01525*, 2020. arXiv:2001.01525.

24 Xueyuan Han, Thomas Pasquier, Tanvi Ranjan, Mark Goldstein, and Margo Seltzer. {FRAPpuccino}: Fault-detection through runtime analysis of provenance. In *9th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 17)*, 2017.

25 Wajih Ul Hassan, Shengjian Guo, Ding Li, Zhengzhang Chen, Kangkook Jee, Zhichun Li, and Adam Bates. Nodoze: Combatting threat alert fatigue with automated provenance triage. In *network and distributed systems security symposium*, 2019.

26 Md Nahid Hossain, Sanaz Sheikhi, and R Sekar. Combating dependence explosion in forensic analysis using alternative tag propagation semantics. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1139–1155. IEEE, 2020. doi:10.1109/SP40000.2020.00064.

27 Muhammad Adil Inam, Yinfang Chen, Akul Goyal, Jason Liu, Jaron Mink, Noor Michael, Sneha Gaur, Adam Bates, and Wajih Ul Hassan. Sok: History is a vast early warning system: Auditing the provenance of system intrusions. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 2620–2638. IEEE, 2023. doi:10.1109/SP46215.2023.10179405.

28 Roberto Jordaney, Kumar Sharad, Santanu K Dash, Zhi Wang, Davide Papini, Ilia Nouretdinov, and Lorenzo Cavallaro. Transcend: Detecting concept drift in malware classification models. In *26th USENIX security symposium (USENIX security 17)*, pages 625–642, 2017.

29 Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

30 Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):1–22, 2019.

31 Diego Klabjan and Xiaofeng Zhu. Neural network retraining for model serving. *arXiv preprint arXiv:2004.14203*, 2020. `arXiv:2004.14203`.

32 Aapo Kyrola, Guy Blelloch, and Carlos Guestrin. Graphchi: Large-scale graph computation on just a {PC}. In *Presented as part of the 10th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 12)*, pages 31–46, 2012. URL: `https://www.usenix.org/conference/osdi12/technical-sessions/presentation/kyrola`.

33 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. *arXiv preprint arXiv:2311.02331*, 2023. `doi:10.48550/arXiv.2311.02331`.

34 Fucheng Liu, Yu Wen, Dongxue Zhang, Xihe Jiang, Xinyu Xing, and Dan Meng. Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1777–1794, 2019. `doi:10.1145/3319535.3363224`.

35 Yushan Liu, Mu Zhang, Ding Li, Kangkook Jee, Zhichun Li, Zhenyu Wu, Junghwan Rhee, and Prateek Mittal. Towards a timely causality analysis for enterprise security. In *NDSS*, 2018.

36 Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31(12):2346–2363, 2018. `doi:10.1109/TKDE.2018.2876857`.

37 Long Ma and Yanqing Zhang. Using word2vec to process big text data. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2895–2897. IEEE, 2015. `doi:10.1109/BIGDATA.2015.7364114`.

38 Shiqing Ma, Kyu Hyung Lee, Chung Hwan Kim, Junghwan Rhee, Xiangyu Zhang, and Dongyan Xu. Accurate, low cost and instrumentation-free security audit logging for windows. In *Proceedings of the 31st Annual Computer Security Applications Conference*, pages 401–410, 2015. `doi:10.1145/2818000.2818039`.

39 Shiqing Ma, Juan Zhai, Yonghwi Kwon, Kyu Hyung Lee, Xiangyu Zhang, Gabriela Ciocarlie, Ashish Gehani, Vinod Yegneswaran, Dongyan Xu, and Somesh Jha. {Kernel-Supported}{Cost-Effective} audit logging for causality tracking. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 241–254, 2018. URL: `https://www.usenix.org/conference/atc18/presentation/ma-shiqing`.

40 Shiqing Ma, Xiangyu Zhang, and Dongyan Xu. Protracer: Towards practical provenance tracing by alternating between logging and tainting. In *23rd Annual Network And Distributed System Security Symposium (NDSS 2016)*. Internet Soc, 2016.

41 Mandiant. How many alerts is too many to handle? https://services.google.com/fh/files/misc/apt42-crooked-charms-cons-and-compromises.pdf, 2022.

42 Emaad Manzoor, Sadegh M Milajerdi, and Leman Akoglu. Fast memory-efficient anomaly detection in streaming heterogeneous graphs. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1035–1044, 2016. `doi:10.1145/2939672.2939783`.

43 Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun, et al. Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In *IJCAI*, volume 19, pages 4739–4745, 2019.

44 Kunal Mukherjee, Joshua Wiedemeier, Tianhao Wang, James Wei, Feng Chen, Muhyun Kim, Murat Kantarcioglu, and Kangkook Jee. Evading {Provenance-Based}{ML} detectors with adversarial system actions. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 1199–1216, 2023. URL: `https://www.usenix.org/conference/usenixsecurity23/presentation/mukherjee`.

45 Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017. arXiv:1707.05005.

46 Riccardo Paccagnella, Pubali Datta, Wajih Ul Hassan, Adam Bates, Christopher Fletcher, Andrew Miller, and Dave Tian. Custos: Practical tamper-evident auditing of operating systems using trusted execution. In *Network and distributed system security symposium*, 2020.

47 Riccardo Paccagnella, Kevin Liao, Dave Tian, and Adam Bates. Logging to the danger zone: Race condition attacks and defenses on system audit frameworks. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1551–1574, 2020. doi:10.1145/3372297.3417862.

48 Thomas Pasquier, Xueyuan Han, Mark Goldstein, Thomas Moyer, David Eyers, Margo Seltzer, and Jean Bacon. Practical whole-system provenance capture. In *Proceedings of the 2017 Symposium on Cloud Computing*, pages 405–418, 2017. doi:10.1145/3127479.3129249.

49 Mati Ur Rehman, Hadi Ahmadi, and Wajih Ul Hassan. Flash open-sourced code. https://github.com/DART-Laboratory/Flash-IDS.

50 Mati Ur Rehman, Hadi Ahmadi, and Wajih Ul Hassan. Flash: A comprehensive approach to intrusion detection via provenance graph representation learning. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 139–139. IEEE Computer Society, 2024.

51 Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. Fast wordpiece tokenization. *arXiv preprint arXiv:2012.15524*, 2020.

52 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

53 Nan Wang, Xuezhi Wen, Dalin Zhang, Xibin Zhao, Jiahui Ma, Mengxia Luo, Sen Nie, Shi Wu, and Jiqiang Liu. Tbdetector: Transformer-based detector for advanced persistent threats with provenance graph. *arXiv preprint arXiv:2304.02838*, 2023. doi:10.48550/arXiv.2304.02838.

54 Pingfan Wang, Nanlin Jin, Duncan Davies, and Wai Lok Woo. Model-centric transfer learning framework for concept drift detection. *Knowledge-Based Systems*, 275:110705, 2023. doi:10.1016/J.KNOSYS.2023.110705.

55 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, 2022. doi:10.1109/TIFS.2022.3208815.

56 Zhiwei Wang, Zhengzhang Chen, Jingchao Ni, Hui Liu, Haifeng Chen, and Jiliang Tang. Multi-scale one-class recurrent neural networks for discrete event sequence anomaly detection. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3726–3734, 2021. doi:10.1145/3447548.3467125.

57 Geoffrey I Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4):964–994, 2016. doi:10.1007/S10618-015-0448-4.

58 Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*, 2022. arXiv:2202.07125.

59 Zhengxu Xia, Yajie Zhou, Francis Y Yan, and Junchen Jiang. Genet: Automatic curriculum generation for learning adaptation in networking. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 397–413, 2022. doi:10.1145/3544216.3544243.

60 Peng Xu, Xiatian Zhu, and David A Clifton. Multimodal learning with transformers: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):12113–12132, 2023. doi:10.1109/TPAMI.2023.3275156.

61 Francis Y Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. Learning in situ: a randomized experiment in video streaming. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 495–511, 2020.

62 Francis Y Yan, Jestin Ma, Greg D Hill, Deepti Raghavan, Riad S Wahby, Philip Levis, and Keith Winstein. Pantheon: the training ground for internet congestion-control research. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 731–743, 2018.

63 Fan Yang, Jiacen Xu, Chunlin Xiong, Zhou Li, and Kehuan Zhang. {PROGRAPHER}: An anomaly detection system based on provenance graph embedding. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 4355–4372, 2023. URL: `https://www.usenix.org/conference/usenixsecurity23/presentation/yang-fan`.

64 Jun Zengy, Xiang Wang, Jiahao Liu, Yinfang Chen, Zhenkai Liang, Tat-Seng Chua, and Zheng Leong Chua. Shadewatcher: Recommendation-guided cyber threat analysis using system audit records. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 489–506. IEEE, 2022. `doi:10.1109/SP46214.2022.9833669`.

A Details of DARPA dataset splitting

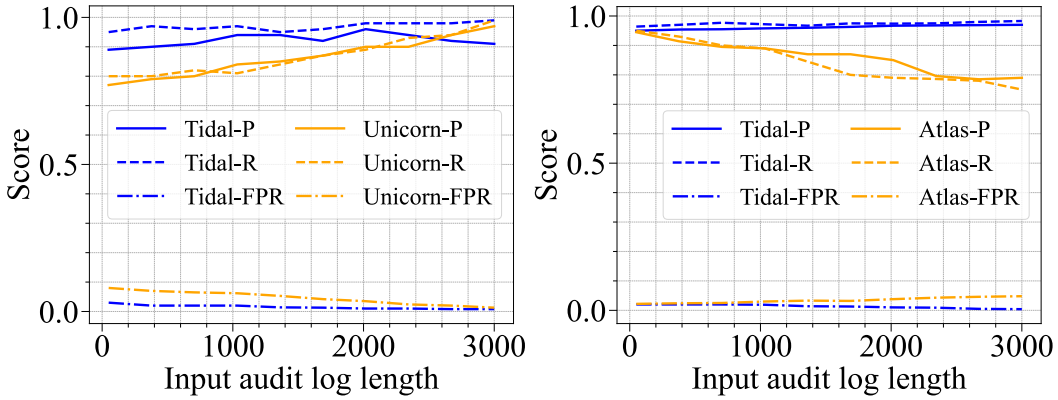
Table 6 Data splitting for evaluation.

	Training set	Validation set	Testing set
E3-Cadets	2018-04-06, 2018-04-11	2018-04-12	2018-04-13
E3-Theia	2018-04-10	2018-04-10	2018-04-12
E3-Trace	2018-04-10	2018-04-12	2018-04-13
E3-FiveD	2018-04-11	2018-04-11	2018-04-12
E5-Cadets	2019-05-16	2019-05-16	2019-05-17
E5-Theia	2019-05-14	2019-05-14	2019-05-14
E5-Trace	2019-05-14	2019-05-14	2019-05-17
E5-FiveD	2019-05-09	2019-05-17	2019-05-19

Table 6 show the details of data splitting approach used for evaluation on the DARPA dataset. Due to the limited availability of attack data, some training, validation, and testing sets are deployed on the same day. In such cases, we use timestamps to separate the data for that day, allocating the first 80% for training, another 10% for validation, and the rest 10% for testing.

A.1 Detection on Varying Input Lengths

Earlier graph-based PIDSes [23, 5] often make specific assumptions about input graph size to ensure that the graph structure provides sufficient information for accurate model predictions. In contrast, Tidal requires no assumptions about input length at streaming time, allowing detection length to be tailored to a security engineer's operational requirements. Recent graph-based approaches also address detection granularity. For example, Flash [50] provides node-level detection granularity, while ShadeWatcher [64] focuses on edge-level granularity. Since the primary focus of Tidal is addressing concept drift, we do not expect it to surpass recent end-to-end PIDSes across every metric. Therefore, we compare Tidal with two earlier graph-based PIDSes to highlight its support of flexible detection granularity as a streaming detection system.



(a) Anomaly-based System. **(b)** Signature-based System.

Figure 13 Detection performance under flexible length of input data.

We evaluate the impact of increasing input log length, from short to long, on TIDAL 's detection performance using the E3 dataset. With TIDAL (anomaly), Figure 13a shows that while detection accuracy improves as input length increases, the recall rate remains consistently above 0.95. In contrast, the baseline system's recall and precision decline significantly with shorter input lengths. TIDAL's superior performance on shorter input lengths is attributed to its effective integration of contextual and structural information. When graph information is insufficient in shorter inputs, TIDAL (anomaly) focuses on semantic information to better differentiate among short logs.

For TIDAL (signature), Figure 13b demonstrates that TIDAL outperforms the baseline system on longer input lengths. We attribute this performance difference to TIDAL 's enhanced ability to capture long-term dependencies. The baseline system, which employs an LSTM model, may suffer from gradient explosion. During back propagation, repeated multiplications across many timesteps can cause gradients to grow or diminish exponentially. Additionally, LSTMs tend to prioritize recent states, potentially overlooking distant dependencies that can be critical for detecting stealthy attack patterns. In contrast, TIDAL 's Transformer-based model avoids gradient explosion over long input sequences by employing self-attention mechanisms rather than recurrent structures. This design enables attack-related information to flow directly across all system events without repeated multiplications, keeping gradients stable across long sequences and enhancing detection reliability.

B Model hyper parameters of each layer

Using Torch 1.13.1, our model is implemented with one token embedding layer, one positional encoding layer, and four transformer encoder layers (Figure 14). We use one normalization layer on top of the input for the embedded graph vectors. After concatenating both data types, we implement a sequential classifier for each APT task as a head to calculate its prediction probability. For learning rate, we use a layer-wise decayed learning rate starting with 1e-3, which we find performs well during training.

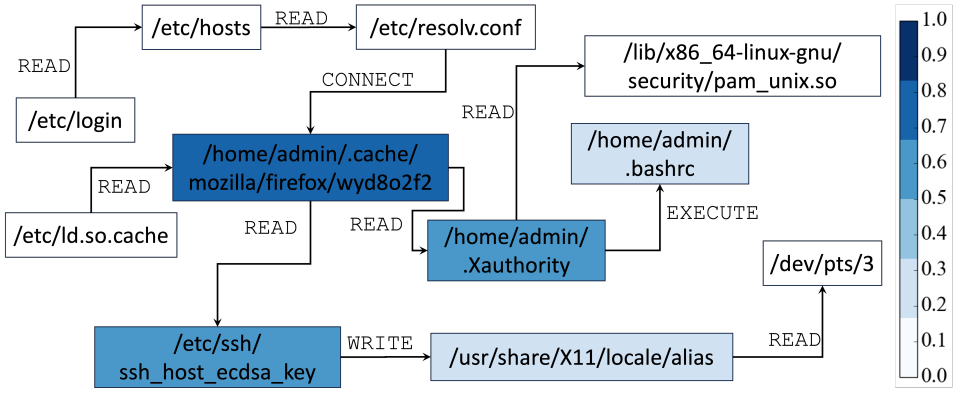
C Case study of attack investigation

To demonstrate the investigation details, we present three case studies using Firefox backdoor attacks in the E3 datasets. Figure 15a shows that TIDAL identifies the later SSH key reading operation and bash execution as highly related to the attack. In Figure 15b, TIDAL identifies

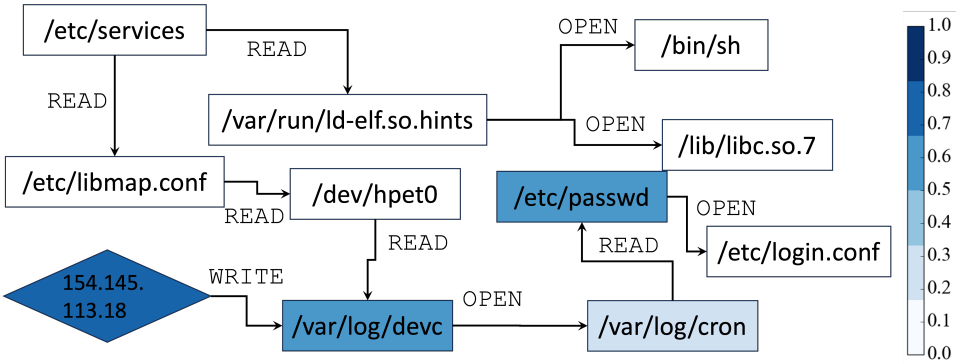
Module	Layer	Num of layers	Dims
Token Embedding	embedding	1	(902, 512)
Positional Encoding	Dropout	1	p=0.2
self_attention	Linear	6	(512, 512)
	Dropout	3	P=0.2
	Norm	2	(512,)
Classifier	Linear	2	(, 64)
	Dropout	1	p=0.2

Figure 14 Model architecture in TIDAL.

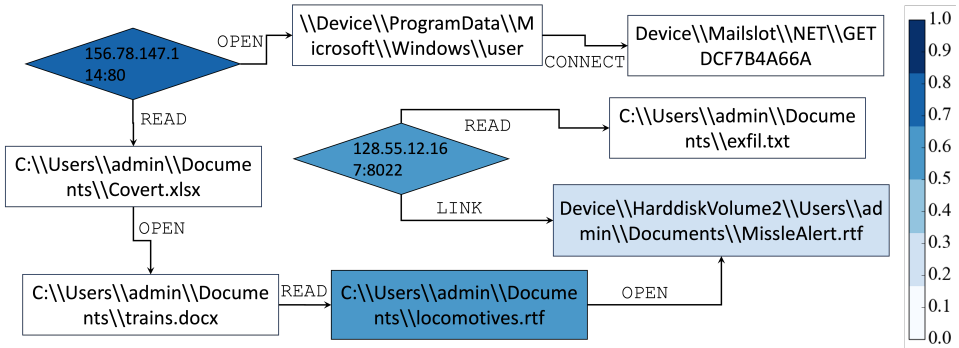
the malicious IP address and also the passward reading malicious activities by /var/log/devc. In Figure 15c, TIDAL detects an unconnected IP address, 156.78.147.114, as malicious, even though it is four hops away from the malicious IP address detected by ATLAS. Identifying the malicious IP address early enables security engineers to blacklist it, preventing future exploitation.



(a) Tidal on Cadet.



(b) Tidal on Trace.



(c) Tidal on FiveD.

Figure 15 Relevance score in Tidal helps analyst to understand more complete attack story on malicious entities.