Madeline: Continuous and Low-cost Monitoring with Graph-free Representations to Combat Cyber Threats

Wenjia Song, *Virginia Tech*
Hailun Ding, *Rutgers University*
Na Meng, *Virginia Tech*
Peng Gao, *Virginia Tech*
Danfeng Yao, *Virginia Tech*

2024 Annual Computer Security Applications Conference (ACSAC) | 979-8-3315-2088-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ACSAC63791.2024.00075

***Abstract*—Advanced persistent threats (APTs) have caused significant financial losses for enterprises, making the development of effective detection systems a critical priority. While existing provenance graph-based APT defenses demonstrate high accuracy, the high complexity and cost of graph operations (e.g., construction, iteration) require extensive processing time and computational resources, making them impractical for real-time detection. To address this challenge, we introduce Madeline, a graph-free, lightweight APT detection system that leverages historical system statistics. Through a multi-step state score calculation for a set of behavioral attributes, Madeline meticulously captures subtle, gradual system changes indicative of stealthy APT activities. Using an LSTM autoencoder, Madeline performs anomaly detection effectively without the need for prior knowledge or manual labeling of attacks. Additionally, Madeline supports continuous monitoring, enhancing the assessment of ongoing risks. This feature helps reduce the need for extensive investigative resources by prioritizing genuine high-risk periods. Our experiments show that Madeline achieves comparable detection accuracy with the state-of-the-art APT detection, with 0.996 recall and 0.011 false positive rate on average. Madeline also significantly reduces the computational overhead, with over 1000x reduction in processing time and 5x in memory utilization.**

1. Introduction

Advanced persistent threats (APTs) have shown a notable escalation, causing unprecedented damage. For example, the campaign conducted by Lazarus group in 2016 caused 81 million loss for the Central Bank of Bangladesh [1]. APTs have also been responsible for significant data breaches across various organizations, including Equifax [2] and the Personnel Management for the federal government [3]. To defend against such threats, detections using system audit logs have been widely developed to catch malicious behaviors going on in the system [4], [5].

System audit logs record various ongoing tasks and are great sources for threat hunting. The most popular APT detection approach is to derive a provenance graph from audit logs to analyze the causality of events. Provenance graphs depict system execution using system entities (e.g., processes, files) as nodes and actions (e.g., open, read) as edges. The dependency relationships between events can be inferred from the graph. The extracted information is then used to detect malicious behaviors at the edge or subgraph level, using signature matching [6]–[9] or learning-based methods [4], [5], [10]–[15]. For instance, HOLMES [6] summarizes known attack patterns and flags any matching behaviors. Unicorn [4] identifies abnormal system execution state through encoded provenance graph.

However, while achieving promising accuracy, graph-based analysis is also known to be complex and time-consuming [10]. The reason is that analyses need to be done at a fine-grained event level. Excessive investigative time and resources are required to trace the relevant and complex context of an event. As the system logs accumulate, the size of the provenance graphs increases substantially, taking a considerable amount of memory space (over 13GB in our experiments). Recent research has shown that graph generation accounts for more than 96% of the time required for attack analysis [16], posing a big challenge to prompt threat detection.

To enable lightweight real-time detection, we introduce Madeline, an efficient approach that utilizes system states derived from historical statistics. As a graph-free framework, Madeline demonstrates remarkably reduced processing time for audit logs and encapsulates the system state in a highly condensed manner. The logs are first digested state scores using historical behavioral distribution. Without complex graph construction, statistics-based computation executes very fast. Then, the model learns to reconstruct the benign states and detect the states that deviate from benign. This process does not require supervised learning and attack knowledge. Madeline further possesses a continuous monitoring feature, which helps comprehensively understand the risk over time and reduce false predictions.

One design requirement for using statistical methods for anomaly detection is to effectively capture the system states and reflect changes in behavior. System behaviors are inherently complex and noisy, with many tasks going on concurrently. With a vast amount of background activities, small changes may be buried under voluminous benign logs. It is critical to catch subtle yet abnormal patterns in system states.

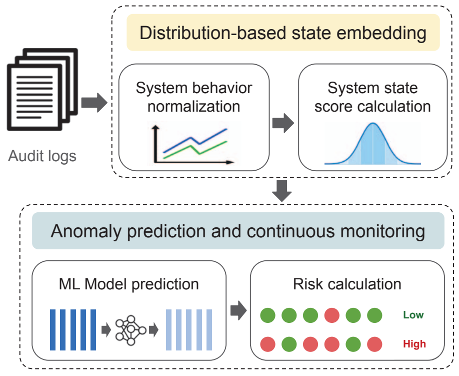
A further design goal is reducing false alarms, which is critical for all threat detection systems. Existing works filter false alarms by considering the anomaliness of related events in provenance graph [11], [17]. For distribution-based

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on July 11,2025 at 20:05:56 UTC from IEEE Xplore. Restrictions apply.

statistical models, when multiple tasks involve the same system behavior (e.g., file read), its frequency may vary and lead to an out-of-distribution score occasionally. Such unexpected fluctuations need to be properly handled.

We summarize our new features and contributions as follows:

- *Efficient detection framework.* We propose MADELINE, a lightweight detection designed to respond to advanced threats promptly. Research has shown that a fast response to such attacks can significantly reduce financial loss and help prevent future damage [10]. We extensively evaluate MADELINE on 3 APT attack scenarios and 10 attack-free scenarios from the DARPA OpTC dataset. MADELINE effectively detects all attack periods with an average recall of 0.988 and a FPR of 0.03. MADELINE-NCM further improves the recall to 0.996 and reduces the FPR to 0.011. We further compare MADELINE with state-of-the-art APT detections. MADELINE shows comparable accuracy and is significantly more lightweight than graph-based solution, achieving over 1000x reduction in processing time and up to 5.2x reduction in memory usage. This efficiency improvement is due to the elimination of entity- and event-level computations, which demand excessive computational time and resources.
- *Compact distribution-based state embedding.* To accurately represent the system state, we propose a multi-step method leveraging historical system behavior distribution. It converts complex system states to highly compact score vectors with minimal computational overhead. To prevent log accumulation during peak hours, we select sliding windows containing a fixed number of logs. In this way, busy hours with increased activities will be expanded into more windows, enhancing the likelihood of detecting subtle changes. We further normalize the occurrence frequencies by the window size to reflect the ratio of behaviors, depicting the relationship between different behaviors.
- *Multi-attribute system representation.* To mitigate the risk of false positives from fluctuations in a single behavior, we consider a collection of behavioral attributes and calculate a set of scores to represent the overall system state. The anomaly detection model receives the vector of scores as an ensemble, learns the pattern across behaviors, and thereby makes more informed predictions.
- *Neighbor-based continuous monitoring.* To enhance the assessment of ongoing risk, MADELINE incorporates continuous monitoring that evaluates the risk levels across successive time windows. Consecutive high-risk predictions are more indicative of a truly high-risk period, whereas isolated anomaly predictions may reflect unexpected fluctuations in behavior. We introduce neighbor-based continuous monitoring (NCM) which noticeably reduces false alarms by 63%, saving time and resources spent on investigating each alert. As a result, security analysts can concentrate on periods of genuine high risk.



Distribution-based state embedding
Audit logs
System behavior normalization
System state score calculation
Anomaly prediction and continuous monitoring
ML Model prediction
Risk calculation
Low
High

Figure 1: Overview of MADELINE.

2. Overview

In this section, we first describe the threat model. We then give a motivating example and present an overview of MADELINE.

2.1. Threat model

MADELINE aims to detect any abnormal behaviors that leave a record in the system audit log. Activities that cannot be captured by the audit logs are beyond the scope of our study. We assume the integrity of the audit logs that 1) they are not tempered or erased by any malicious actors to hide their traces, and 2) the benign period used for model training is attack-free.

2.2. Motivating example and security gap

We demonstrate the detection mechanism of MADELINE using a simplified attack example from the DARPA OpTC dataset [18].

Attack scenario. An enterprise has several endpoint machines running various routine tasks. One day, an attacker sends spearphishing emails containing malicious attachments named payroll.docx. A victim employee downloads and opens the file and the attacker checks in. The attacker first makes several attempts to modify the registry and sets persistence in the host to check back consistently. Then, they enumerate the files and search for "important, secret, classified" keywords to collect host information. Finally, they compress files in C:// documents for exfiltration.

Detection by MADELINE. MADELINE identifies system changes using behavioral statistics. At the beginning of the attack, as the attacker attempts to establish persistence, the proportion of registry-related activities tends to increase abnormally. Subsequently, during file exfiltration, compression will also exhibit distinct file operation patterns. For instance, file read increases during information gathering

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on July 11,2025 at 20:05:56 UTC from IEEE Xplore. Restrictions apply.

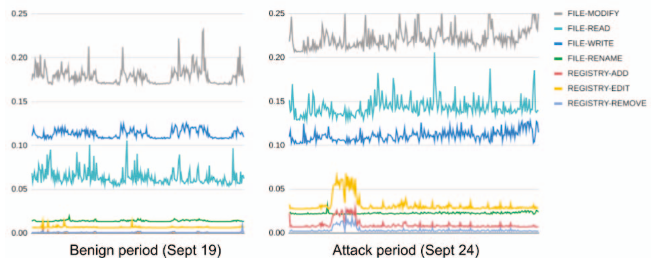


Figure 2: Behavior ratio of DARPA OpTC host 0501 during benign and attack periods. We only show 6 attributes in this figure to illustrate the idea. Calculation details are in Section 3.

and file compression (Figure 2). Variations in these operations will affect the ratio of other behaviors to varying degrees, collectively indicating a change in the system. When these abnormal states are compared to the historical system state distribution, they will receive an out-of-distribution score, which the anomaly detection model can flag later. Investigative resources can be conserved by prioritizing the analysis of these high-risk periods. Since MADELINE only relies on counting behavior frequencies and calculating state scores using statistical models, the computational overhead is minimal. This allows detection to occur within seconds even for large volumes of system logs, thus enabling real-time threat detection. Moreover, storing historical system behaviors as state scores consumes minimal space, requiring only a few megabytes for days of events. MADELINE can also complement existing fine-grained analysis approaches.

Limitations of existing solutions. Existing graph-based solutions [4]–[6], [19] require significant analysis time and computational resources, making prompt detection and response challenging. This is because event-level detection requires investigating every single event and its dependencies. To expedite the analysis, some approaches utilize multi-threading [5], [19], which may consume substantial computational power and potentially slow down other tasks. In this attack example, suspicious events include abnormal process creation from malicious file, unusual reads of files by unknown processes, and suspicious file creations by unknown processes. To pinpoint these events or the trace containing them, each event needs to be indexed and linked, while massive iterations are needed to trace event context. This results in substantial storage, memory, and computational overheads. Additionally, historical graphs often cannot be deleted because future anomaly scores may depend on them. We experimentally confirm the significant time and memory utilization reduction of MADELINE in Section 4.

2.3. Overview of MADELINE

We propose MADELINE, a threat detection leveraging system state computed based on historical statistics, requiring no supervised learning. MADELINE consists of two major steps: 1) distribution-based system state embedding, and 2) anomaly detection and risk evaluation (Figure 1).

To capture system state information in a condensed way, we first convert the audit logs to vectors with statistics of various behaviors. Each vector represents a short time window. When historical data accumulates, we can deduce a distribution and infer where a newly observed state falls in this distribution. Each behavior attribute has its own score, and together they represent the state of the current system. Then, an LSTM-autoencoder will learn to reconstruct the state score for a few consecutive time windows. The model is trained on only benign data. As a result, states close to observed normal data will have a lower reconstruction error, whereas states that deviate from normal will have a higher error.

The key idea behind our design is to let the model learn the relationship between each attribute in a time window and how this relationship changes over time across multiple time windows. Even though system behaviors could be complex and noisy, system states should reasonably repeat themselves during regular routines. Changes in one or more behavior attributes will make the calculated state deviate from the benign distribution. Log entries (i.e., activities) often do not follow a strict temporal order due to concurrent tasks and multithreading. Our design considers the activities accumulated during a time window, also helping reduce the impact of this variance.

3. Design of MADELINE

We detail the design of each phase below. *Distribution-based system state embedding* converts complex system states into score vectors. *Benign behavior learning and anomaly prediction* involves training the model using historical normal behaviors and identifying abnormal states based on the model's output. *Continuous monitoring* facilitates an enhanced assessment of ongoing risk.

3.1. Distribution-based system state embedding

The goal of the phase is to convert the system state to a compact vector representation (embedding).

Behavior attribution. First, MADELINE takes the audit log as input and categorizes the log entries by the action and related object. We consider one type of event, which is combination of object and action, as one attribute (e.g., PROCESS OPEN, FILE CREATE).

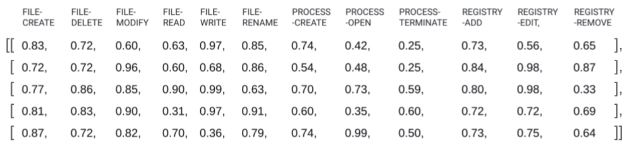
Behavior normalization. For a fixed-size window (e.g., 10,000 log entries), we count each attribute's frequency and normalize the frequencies by the total number of actions. That is, for each selected attribute i,

$$freq_i = \frac{\#\ occurrences\ of\ attribute_i}{length\ of\ window} \tag{1}$$

The calculated frequencies of all attributes form a vector, representing the activities happening in this time window. The fixed-sized window design helps us catch subtle changes during peak hours.

State score calculation. We then compute a score for each attribute utilizing historical data. After computing the

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on July 11,2025 at 20:05:56 UTC from IEEE Xplore. Restrictions apply.



	FILE-CREATE	FILE-DELETE	FILE-MODIFY	FILE-READ	FILE-WRITE	FILE-RENAME	PROCESS-CREATE	PROCESS-OPEN	PROCESS-TERMINATE	REGISTRY-ADD	REGISTRY-EDIT,	REGISTRY-REMOVE	
[[	0.83,	0.72,	0.60,	0.63,	0.97,	0.85,	0.74,	0.42,	0.25,	0.73,	0.56,	0.65	],
[	0.72,	0.72,	0.96,	0.60,	0.68,	0.86,	0.54,	0.48,	0.25,	0.84,	0.98,	0.87	],
[	0.77,	0.86,	0.85,	0.90,	0.99,	0.63,	0.70,	0.73,	0.59,	0.80,	0.98,	0.33	],
[	0.81,	0.83,	0.90,	0.31,	0.97,	0.91,	0.60,	0.35,	0.60,	0.72,	0.72,	0.69	],
[	0.87,	0.72,	0.82,	0.70,	0.36,	0.79,	0.74,	0.99,	0.50,	0.73,	0.75,	0.64	]]

Figure 3: A simplified example of our MADELINE embedding (system state scores) computed from the DARPA OpTC dataset. The example shows 5 windows, each with 12 attribute scores.

frequency vectors for a few time windows, we can form distributions of each attribute's frequency from past data. When given a new behavior frequency vector, the probability of each value happening can be induced from its historical distribution. Then, we use the cumulative distribution function (CDF) to calculate this probability for each attribute in the vector.

$$\begin{aligned} score &= 1 - P(\mu \ - \ \mathit{diff} < X < \mu \ + \ \mathit{diff}) \\ &= 1 - (F_X(\mu \ + \ \mathit{diff}) - F_X(\mu \ - \ \mathit{diff})) \end{aligned} \tag{2}$$

where X represents the distribution of the normalized behavioral frequency, μ is the distribution mean, *diff* is the difference between μ and the newly observed behaviroal frequency data point d:

$$\mathit{diff} = |\mu - d| \tag{3}$$

F_X is the CDF of X:

$$F_X(x) = P(X \leq x) = \int_{-\infty}^{x} f_X(t)\, dt \tag{4}$$

and f_X is the probability density function (PDF) of X:

$$f_X(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{5}$$

with σ being the standard deviation (std). The integral from negative infinity (Equation 4) computes the cumulative probability of a point falling below the upper bound. By subtracting the lower bound's cumulative probability from that of the upper bound (Equation 2), our attribute score quantifies the likelihood of the newly observed frequency deviating this far from the distribution mean.

Calculating these scores for the attributes generates a new vector representing the system state. Beyond the attribute relationship represented in the frequency vector, the state scores further encode information from the past, capturing the changing pattern of system behaviors. The score vectors are then used in the next step as the input for learning the anomaly detection model. We refer to the state score vectors as MADELINE embedding. Compared to existing embedding methods, such as word2vec [20] and log2vec [21], MADELINE relies solely on statistical models and does not require training, thus enabling rapid computation.

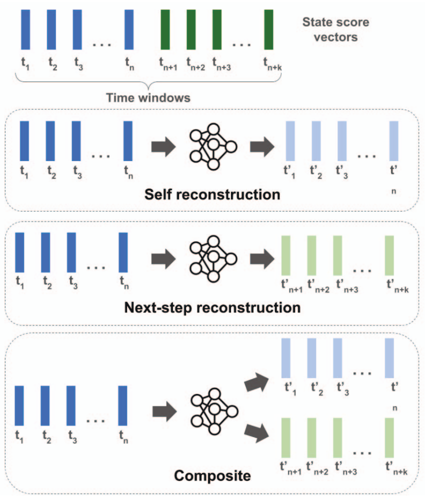


Figure 4: Illustration of the 3 LSTM reconstruction modes.

3.2. Benign behavior learning and anomaly prediction

This phase contains two steps. One is benign behavior learning, which aims to learn a model that accurately reconstructs the historical system state. The other is anomaly prediction, with a goal to identify abnormal system states from model predictions.

Benign behavior learning. We choose an LSTM autoencoder as our anomaly detection model. LSTM is known for its capability of handling sequential time series and the autoencoder enables learning without attack labels. This design, without supervised learning, helps address the challenge of the limited availability of attack data for training in a realistic setting. Compared to advanced Transformer-based models, LSTM has fewer parameters and requires less time for training. The model is implemented using an encoder-decoder LSTM architecture. We train the model with the previously computed state scores to learn the normal system state by a reconstruction task. The input is a few consecutive state score vectors and the learning goal is to minimize the difference between the recreated vectors and target vectors. The difference is measured by mean square error (MSE), which is used as the loss function. Specifically, we provide 3 reconstruction modes (Figure 4):

Self-reconstruction. In this mode, the model learns to reconstruct the input. The model reads the state score vectors, encodes them, decodes them, and tries to reconstruct them. That is, the model only handles the current system state.

Next-step reconstruction. In this mode, the decoder is modified to create the state vectors following the input. The model parses the current system state and is asked to predict the state for subsequent time windows.

Composite setting. In this mode, we combine self-reconstruction and next-step reconstruction with 2 decoders. One decoder is responsible for reconstructing the input and the other for recreating the next steps. Model's output

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on July 11,2025 at 20:05:56 UTC from IEEE Xplore. Restrictions apply.

includes both the reconstructed current state and the subsequent state.

Unlike existing LSTM-based anomaly detection methods, such as DeepLog [22], which can only handle a finite set of events, MADELINE can flexibly encode and learn system states from infinite combinations of behaviors.

Anomaly prediction. Next, we interpret the prediction to detect anomalies in system state. Each sliding window receives a separate security decision. We calculate the reconstruction error to quantify the deviation from normal. The reconstruction error is calculated as the mean absolute error (MAE) between the target vectors and the reconstructed vectors element-wise. The MAE for m time windows with n features are shown in Equation 6.

$$MAE_m = \sum_{i=1}^{m} \frac{1}{n} \sum_{j=1}^{n} |t_{ij} - r_{ij}| \tag{6}$$

Here, t and r represent the target and reconstruction vectors, respectively. t_{ij} and r_{ij} represent the jth element in the ith time window in t and r, respectively.

If the reconstruction error exceeds a pre-determined threshold, then the window is flagged as abnormal (positive); otherwise, it will be labeled as normal (negative).

$$\text{prediction} = \begin{cases} \text{positive (abnormal), if } MAE > th \\ \text{negative (normal), otherwise} \end{cases} \tag{7}$$

We select a decision threshold that is 2 standard deviations from the distribution mean (Equation 8) following the 68-95-99 rule, that is, theoretically, 95% of data from the same distribution should fall within this range.

$$th = \mu + 2\sigma \tag{8}$$

Here, μ is the distribution mean and σ is the standard deviation. The threshold is determined on the validation set.

3.3. Continuous monitoring and MADELINE-NCM

Continuous monitoring is critical for defending against APTs because of their low-and-slow characteristics. The goal is to support a more comprehensive understanding of ongoing risk by providing context to a specific decision. False alarms or miss detections are unavoidable due to the complex nature of system behaviors. For instance, when various tasks run simultaneously during heavy usage time, an unexpected ratio of behavioral attributes may appear in a sliding window and consequently lead to an alarm. This false prediction could be reduced by inspecting the risk level of adjacent periods. If the risk level is continuously low, investigative efforts can be prioritized to other higher-ranked alarms.

Adapting this idea, we further introduce a neighbor-based continuous monitoring (NCM) feature. This feature helps reduce false positives and missed detection by considering the predictions on consecutive windows and taking the majority vote. Given target window w_i and a neighbor size b, the decision for w_i becomes

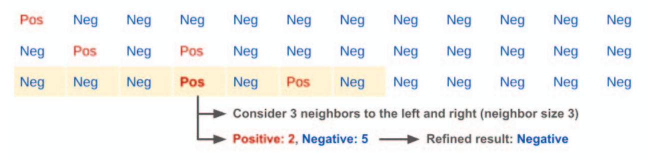

Pos Neg Neg Neg Neg Neg Neg Neg Neg Neg Neg Neg
Neg Pos Neg Pos Neg Neg Neg Neg Neg Neg Neg Neg
Neg Neg Neg Pos Neg Pos Neg Neg Neg Neg Neg Neg
Consider 3 neighbors to the left and right (neighbor size 3)
Positive: 2, Negative: 5
Refined result: Negative

Figure 5: An example of using our neighbor-based continuous monitoring (NCM) feature to refine the prediction result.

$$D_{NCM}(w_i) = \text{mode}\{D(w_{i+k}) \mid k \in [-b, b], 0 \leqslant i+k \leqslant e\} \tag{9}$$

where D is the decision of a window w and e is the ending index of currently available predictions.

We refer to the model incorporating this feature as MADELINE-NCM. Figure 5 shows a simple example, in which the prediction result of a particular time window (false positive) is replaced with the predominant vote among its neighbors.

4. Experimental Evaluation

We conduct extensive experiments in evaluating the efficacy and efficiency of MADELINE as a detection system. In particular, we investigate and aim to answer the following questions:

RQ1. How effective is MADELINE in detecting advanced threats in different attack scenarios? (Section 4.1)

RQ2. How do MADELINE compare with the state-of-the-art threat detection methods in terms of accuracy, training and prediction time, and computational resource utilization? (Section 4.2)

RQ3. How do different design choices impact the detection efficacy of MADELINE? (Section 4.3)

RQ4. How would continuous monitoring help reduce false alarms? (Sections 4.1 and 4.2)

We first evaluate the efficacy of MADELINE on 3 different APT attack scenarios and 10 attack-free scenarios from a large-scale DARPA dataset. We show that MADELINE effectively detects the attack periods with very few positives. We further compare MADELINE with 3 state-of-the-art APT detection approaches. The results show that MADELINE achieves comparable accuracy with the state-of-the-art solutions. We further compare the computation time and resources needed with graph-based solution KAIROS [5], demonstrating our advantage in lightweight online detection. We then conduct a comprehensive ablation study and discuss the impact of various design choices on the performance of MADELINE.

Experimental setup. All experiments are performed on a machine with Intel Core i7 11700K CPUs @ 3.6GHz, 64 GB memory, and an NVIDIA GeForce RTX 3090 GPU. We use Ubuntu 22.04 LTS as the operating system. Unless otherwise specified, we use an LSTM model with an encoder-decoder structure, optimized by Adam optimizer. The encoder contains 2 layers with 64 and 128 units, respectively. The de-

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on July 11,2025 at 20:05:56 UTC from IEEE Xplore. Restrictions apply.

coder contains 2 layers, with 128 and 64 units, respectively. We implement our models in Python using TensorFlow. The statistical models and scores are computed using Scipy. We use a sliding window size of 10,000 and a stride of 1000. For the LSTM self-reconstruction setting, we use an input and output size of 5 (i.e., 5 sequential score vectors). For the next-step reconstruction setting, we use an input size of 5 and an output size of 3. For the composite setting, we use an input size of 5, and output sizes of 5 (self-reconstruction) and 3 (next-step reconstruction) for the two decoders. We use recall and false positive rate (FPR) as evaluation metrics.
Datasets. We use 2 datasets for our evaluation. We release our pre-processed data and intermediate results[1].

DARPA OpTC. The DARPA Operationally Transparent Cyber (OpTC) [18] is a large-scale APT dataset containing log records for both benign and red-team simulated malicious activities. This dataset was collected in 2019 from around 1000 Windows 10 hosts. It describes both a benign period (September 17-23) and an attack period (September 23-25). During the attack period, the red team injected malicious behaviors with benign background activities running. A summary of the attack scenarios is shown in table 11. The data is publicly available in eCAR format as JSON files [23].

StreamSpot. The StreamSpot [24] dataset consists of data derived from 1 attack and 5 benign scenarios. The benign scenarios depict normal activities, such as browsing YouTube and playing video games. The attack involves a drive-by download triggered by visiting a malicious URL. For each scenario, 100 tasks were automatically executed on a Linux machine.

4.1. RQ1: Recall and FPR of MADELINE

We evaluate MADELINE on both attack scenarios and benign scenarios from the DARPA OpTC dataset. The results suggest that MADELINE can effectively detect malicious periods. We use 4 attacked hosts from all 3 attack scenarios and 10 randomly selected benign (i.e., attack-free hosts) to test the number of false alarms generated by MADELINE. For attacked hosts, we save the last benign period as validation and testing sets and use the rest as our training set. We label the data during the corresponding attack times in their evaluation directory as attack logs for evaluation. One thing worth noting is that we label the entire attack period as positive instead of labeling individual events (i.e., single log entries) because of the nature of our lightweight coarse-grained online detection. For attack-free hosts, we use a similar training and validation setting, with data from the day 1 attack period as testing. Because these hosts are not the target of the red team, there should be only benign background activities running even during the attack time. We use those data to extensively test the amount of false alarms generated by MADELINE.

For the DARPA OpTC dataset, we select 12 attributes from 3 different categories, namely file, process, and registry

1. https://github.com/wenjia7/madeline

TABLE 1: Attributes selected for DARPA OpTC dataset.

Category	Attributes
FILE	FILE-CREATE, FILE-DELETE, FILE-MODIFY, FILE-READ, FILE-WRITE, FILE-RENAME
PROCESS	PROCESS-CREATE, PROCESS-OPEN, PROCESS-TERMINATE
REGISTRY	REGISTRY-ADD, REGISTRY-EDIT, REGISTRY-REMOVE

(Table 1). We select those categories as they cover a wide range of adversary enterprise tactics, including Execution, Persistence, Privilege Escalation, Discovery, Lateral Movement, Collection, and Exfiltration [25]. Each attribute is a combination of an object and an action. We then calculate the frequency of each attribute and subsequently the state scores. A simplified input embedding example of input size 5 is shown in Figure 3.
Evaluation on APT attack scenarios. Tables 2 and 3 show a summary of the results on 4 attack hosts. The distributions of reconstruction errors are shown in Figure 6. We observe that, in all cases, benign and attack data show a good separation, achieving good recall and low FPR. Specifically, self-reconstruction and composite settings achieve an average recall of over 0.98 and an average FPR as low as 0.03. The composite setting shows slightly better separation between benign and attack compared to the reconstruction setting. Next-step reconstruction has a slightly lower recall at 0.93. One possible reason for the small number of missed detections is that, due to the complex nature of system behaviors, the scores of a few windows may fall in the normal range. Similarly for false alarms. We further show that missed detections and false alarms can be remarkably reduced with our continuous monitoring feature (Tables 2 and 3).

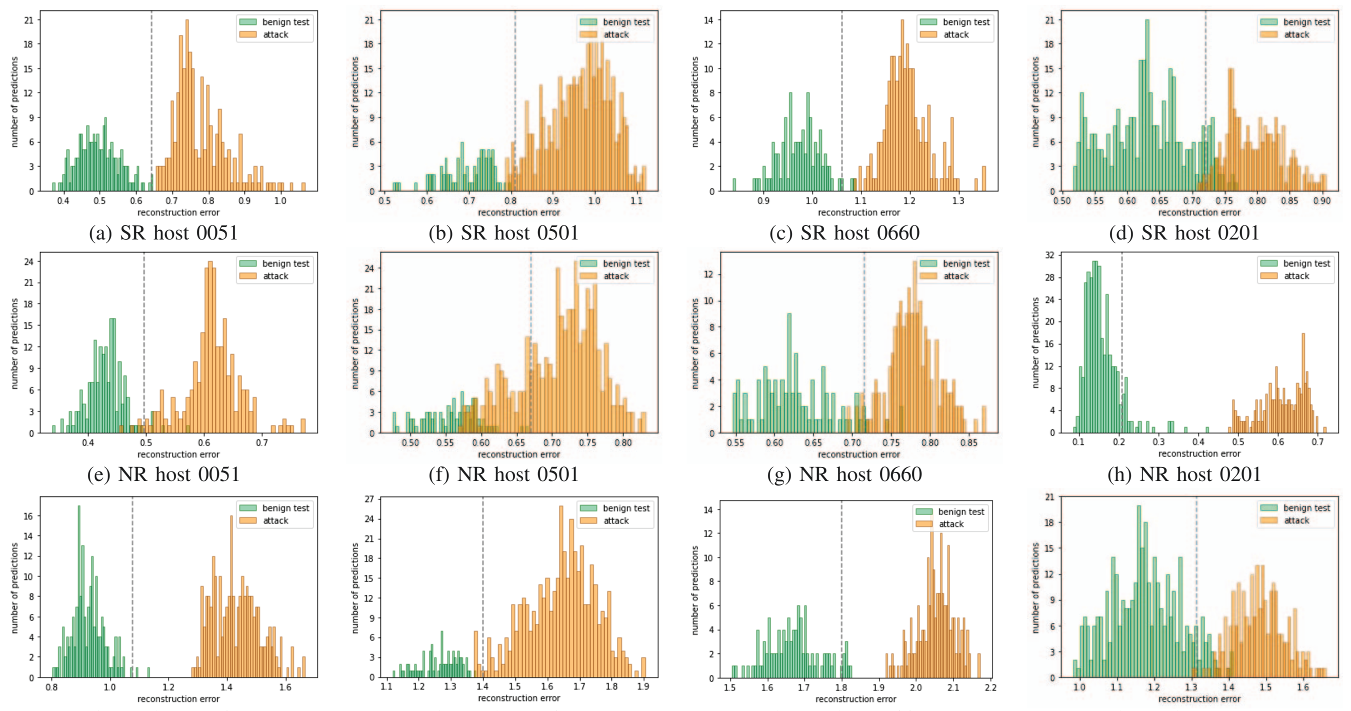
Self-construction setting approaches the performance of composite setting but requires less training time, making it a practical choice for most scenarios. However, when optimal accuracy is important, composite setting becomes a preferable option.

TABLE 2: Recall of MADELINE on attack scenarios from DARPA OpTC dataset with and without our neighbor-based continuous monitoring (NCM) feature (neighbor size 5). NCM improves recall in all cases.

Host	Self-reconstruction		Next-step reconstruction		Composite	
	-	NCM	-	NCM	-	NCM
0051	1.000	1.000	0.980	0.996	1.000	1.000
0501	0.971	0.985	0.759	0.768	0.980	1.000
0660	1.000	1.000	0.963	1.000	1.000	1.000
0201	0.979	1.000	1.000	1.000	0.991	1.000
avg	0.988	0.996	0.926	0.941	0.993	1.000

Evaluation on benign scenarios. We randomly select 10 attack-free hosts from DARPA OpTC dataset to further test the number of false alarms MADELINE generates. The findings indicate that, in addition to effectively detecting

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on July 11,2025 at 20:05:56 UTC from IEEE Xplore. Restrictions apply.



(a) SR host 0051 (b) SR host 0501 (c) SR host 0660 (d) SR host 0201

(e) NR host 0051 (f) NR host 0501 (g) NR host 0660 (h) NR host 0201

(i) CO host 0051 (j) CO host 0501 (k) CO host 0660 (l) CO host 0201

Figure 6: Distribution of data reconstruction errors on attack scenarios from DARPA OpTC dataset (SR for self-reconstruction, NR for next-step reconstruction, CO for composite).

TABLE 3: FPR of MADELINE on attack scenarios from DARPA OpTC dataset with and without our neighbor-based continuous monitoring (NCM) feature (neighbor size 5). NCM reduces FPR in all cases.

Host	Self-reconstruction		Next-step reconstruction		Composite	
	-	NCM	-	NCM	-	NCM
0051	0.005	0.000	0.029	0.000	0.011	0.000
0501	0.000	0.000	0.000	0.000	0.000	0.000
0660	0.019	0.000	0.039	0.000	0.049	0.000
0201	0.095	0.043	0.096	0.049	0.087	0.046
avg	0.030	0.011	0.041	0.012	0.037	0.012

abnormal behaviors within the system, MADELINE also successfully maintains a minimal rate of false alarms (Table 4 and Figure 11 in appendix).

TABLE 4: FPR of MADELINE on attack-free scenarios from DARPA OpTC dataset. SR for self-reconstruction, NR for next-step reconstruction, CO for composite.

Host	SR	NR	CO	Host	SR	NR	CO
0070	0.042	0.000	0.000	**0470**	0.011	0.004	0.019
0101	0.028	0.051	0.044	**0607**	0.031	0.021	0.016
0307	0.022	0.000	0.011	**0720**	0.000	0.000	0.000
0455	0.000	0.028	0.000	**0771**	0.000	0.008	0.000
0468	0.012	0.012	0.012	**0860**	0.040	0.005	0.082
				avg	0.019	0.013	0.018

4.2. RQ2: Comparison with state-of-the-art detections

We compare MADELINE with state-of-the-art advanced threat detections, evaluating detection accuracy, processing time, and computational resource utilization. We observe that MADELINE achieves comparable accuracy and is substantially more lightweight and faster than graph-based solutions.

Comparison of detection accuracy. We evaluate MADELINE on the StreamSpot dataset and compare state-of-the-art anomaly detection systems, including StreamSpot [24], Unicorn [4], and Kairos [5]. We choose StreamSpot dataset [24] for comparison because it provides an isolated attack scenario, ensuring compatibility of the attack label across various detection tools. While large APT datasets, such as DARPA OpTC, depict more advanced threats, the attack labeling process may vary from work to work depending on their attack analysis strategies and granularity. Therefore, although StreamSpot contains a relatively simple attack, it facilitates a more straightforward and fair comparison.

We train separate models using each benign scenario and a combined model using all 5 benign scenarios. In each setting, we use 80% data as the training set, 10% as the validation set for threshold selection, and 10% as the benign testing set to assess the FPR. Then, each trained model is evaluated against the attack scenario to determine recall.

Similar to our previous design, we choose attributes from file and process categories. The StreamSpot dataset does not

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on July 11,2025 at 20:05:56 UTC from IEEE Xplore. Restrictions apply.

have any registry-related objects as it was run on a Linux machine. The 26 selected attributes are shown in Table 14 in the appendix.

A summary of MADELINE's performance is shown in Table 5. MADELINE performs consistently well on all scenarios except for the GMail one. The GMail scenario has a notably small data size, which is only 36% of the Download scenario and 27% of the CNN scenario. This small data size may be inadequate for the deep learning model to effectively learn, leading to relatively high reconstruction errors for some benign time windows and subsequently a higher decision threshold. The prediction distribution is shown in Figure 12 in appendix.

TABLE 5: Performance of MADELINEon StreamSpot dataset.

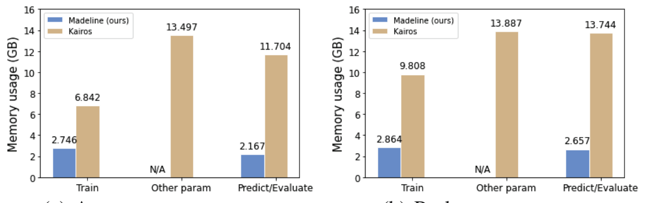
Scenario	**Self-reconstruction**		**Next-step reconstruction**		**Composite**	
	Recall	**FPR**	**Recall**	**FPR**	**Recall**	**FPR**
YouTube	1.000	0.003	0.984	0.025	1.000	0.000
GMail	0.537	0.081	0.615	0.061	0.947	0.037
VGame	1.000	0.002	0.964	0.010	1.000	0.000
Download	1.000	0.032	0.947	0.039	1.000	0.037
CNN	1.000	0.032	1.000	0.031	1.000	0.022
All	1.000	0.033	0.976	0.028	1.000	0.021
avg	0.923	0.031	0.914	0.032	0.991	0.020

We compare with the state-of-the-arts on the combined scenario (i.e., using combined data from the 5 benign scenarios for training). The results are presented in Table 6. MADELINE shows comparable detection accuracy. Next, we further show that MADELINE is notably more lightweight in terms of processing time and computational resource usage.

TABLE 6: Comparison of detection accuracy with state-of-the-art on StreamSpot dataset. The performance of StreamSpot [24] and Unicorn [4] are reported in the original Unicorn paper [4]. The performance of Kairos is reported in the original Kairos paper [5]. The performance of FLASH is reported in the original FLASH paper [14]. The Performance of THREATRACE is reported in the original THRATRACE paper [26]. Performance of MADELINE is based on the self-reconstruction setting. NCM refined performance uses a neighbor size of 5.

	Recall	**FPR**	**Accuracy**
StreamSpot	N/A	N/A	0.660
UNICORN	0.930	0.020	0.940
KAIROS	1.000	0.000	1.000
FLASH	0.960	0.000	0.960
THREATRACE	0.990	0.004	0.990
MADELINE (ours)	1.000	0.033	0.979
MADELINE-NCM (ours)	1.000	0.004	0.997

Comparison of processing time and resource utilization. We compare MADELINE's processing time and memory utilization with state-of-the-art detection Kairos [5] on the DARPA OpTC dataset. Because Kairos uses a multi-host training setting, we consider MADELINE's training and evaluation time for all 4 hosts together for a fair comparison. We follow the implementation of Kairos available on GitHub [27]. Besides training and prediction, Kairos has an extra step of calculating each node's inverse document frequency (node_IDF) for anomalous score evaluation. We report this part as other parameter calculation (labeled as other param in Table 7 and Figure 7). Both sets of experiments are conducted on the same machine as described previously.



(a) Average memory usage (b) Peak memory usage

Figure 7: Comparison of memory usage with Kairos [5] on DARPA OpTC dataset.

Due to the ability of MADELINE to efficiently condense the system state into a few score vectors, both the training and prediction phases are executed very fast. A breakdown of the processing time needed for each stage is shown in Table 7, and memory utilization is shown in Figure 7. MADELINE achieves over 1000x speed up in detection processing time in total and up to 5.2x reduction in memory utilization. This efficiency improvement is because of the fact that fine-grained detection at the node and edge level requires extensive computational effort. Specifically, in the case of Kairos, each node must be checked against every event in all preceding time windows to assess the node's rarity and, subsequently, to determine the benignness of future events involving this node. While providing fine-grained information for attack flow reconstruction, the substantial amount of processing time significantly increases the difficulty of deploying graph-based detection as a real-time solution. Although MADELINE presents coarse-level prediction over time intervals, its fast reaction allows early intervention in response to attack behaviors. It is important to note that MADELINE can also serve as a complement to existing fine-grained analyses, helping prioritize the investigation on high-risk time intervals, thereby conserving time and reducing manual efforts. A detailed use scenario is further discussed in Section 5.

TABLE 7: Comparison of processing time with Kairos [5] on DARPA OpTC dataset.

	Train	**Other param**	**Predict/ Evaluate**	**Total**
MADELINE (ours)	48s	N/A	4s	52s
Kairos	229m50.2s	797m18s	65m12s	1092m20s

4.3. RQ3: Ablation study

We then analyze the influence of various design choices using the self-construction setting on attack scenarios from the DARPA OpTC dataset. We independently adjust one design choice at a time to examine its impact on MADELINE's performance. We discuss the findings below.

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on July 11,2025 at 20:05:56 UTC from IEEE Xplore. Restrictions apply.

Decision threshold. We choose the default decision threshold as 2 stds from the mean (Equation 8) following the 68-95-99 rule of normal distribution. Next, we examine the impact of selecting different thresholds. Table 8 shows the experimental results. Shifting the threshold to the left enhances sensitivity to abnormal data, thereby increasing both the recall and the FPR. Using 1 standard deviation from the mean as the threshold, the FPR increases by 5 times. On the other hand, shifting the threshold to the right lowers the sensitivity, reducing the value of both metrics. Using 3 standard deviation from the mean as the threshold, the average recall drops to 0.85. Those numbers show that 2 stds from the mean achieve a good balance.

TABLE 8: Comparison of decision thresholds on reconstruction error.

Host	mean + 1 * std		mean + 2 * std (default)		mean + 3 * std	
	Recall	FPR	Recall	FPR	Recall	FPR
0051	1.000	0.068	1.000	0.005	0.904	0.000
0501	1.000	0.148	0.971	0.000	0.862	0.000
0660	1.000	0.154	1.000	0.019	0.985	0.000
0201	1.000	0.243	0.979	0.095	0.654	0.000
avg	1.000	0.153	0.988	0.030	0.851	0.000

Input size. Input size refers to the number of steps (i.e., continuous time windows) we feed into the LSTM autoencoder models. Figure 3 shows an example of input size 5. Increasing this size provides more context for the models to learn, at a cost of slightly elevated training time. A comparison of 3 different input sizes is shown in Table 9. An input size of 5 achieves good performance while further increasing the size trivially improves it.

TABLE 9: Comparison of reconstruction sizes on attack scenarios.

Host	Input size 3		Input size 5 (default)		Input size 8	
	Recall	FPR	Recall	FPR	Recall	FPR
0051	0.896	0.017	1.000	0.005	0.984	0.006
0501	0.954	0.056	0.971	0.000	0.973	0.000
0660	1.000	0.038	1.000	0.019	1.000	0.029
0201	0.887	0.107	0.979	0.095	1.000	0.081
avg	0.934	0.054	0.988	0.030	0.989	0.029

Sliding size. This value determines the number of logs included in one single sliding window. A small sliding size records a shorter period and provides us with more data points at the time of investigation. However, a size that is too small allows limited context in one window, possibly making the scores fluctuate a lot and thus hard for the model to learn. A balance needs to be established between these two aspects. We compare 5 sliding sizes on 2 attack scenarios (Table 10). We find that small sliding sizes (e.g., 5000 and 8000) result in an obvious overlap of benign and attack data and subsequently low recall. Larger sizes starting from 10,000 give a good separation. Increasing the sliding window size further from 10,000 trivially improves the recall and slightly lifts the FPR. Thus, we select 10,000 as the default size.

TABLE 10: Comparison of sliding window sizes on attack scenarios.

Sliding size	Host 0051		Host 0501	
	Recall	FPR	Recall	FPR
5000	0.102	0.006	0.670	0.000
8000	0.542	0.000	0.895	0.034
10000 (default)	1.000	0.005	0.971	0.000
15000	1.000	0.029	0.956	0.035
20000	1.000	0.012	0.998	0.024

State score calculation. We explore a few alternative methods for calculating the state scores. An alternative way of normalizing the frequency is to divide each attribute's frequency by the total number of entries in its category. We refer to this setting as categorical normalization. For state score calculation, besides using the single attributes, we also investigate alternatives using joint attribute distributions with 2 or 3 attributes. Formulas used in these alternative methods are discussed in the appendix.

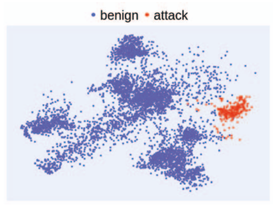
We visualize the distribution of benign and attack scores calculated by various methods using a PCA analysis (Figure 8). Scores calculated using the default normalization and single attribute distribution show the best separation between benign and attack data (Figure 8a). Scores calculated using the default normalization and joint distribution of multiple attributes also show good separation, with some overlap on the boundary (Figures 8b and 8c). Attack data points lay in the middle of benign points when using normalized frequencies without state score calculation (Figures 8d and 8h). Scores calculated based on categorical frequencies (Figures 8e, 8f, and 8g) exhibit large overlap between benign and attack data.

Moreover, calculating joint distribution requires significantly more time than calculating single attribute distribution (23% slower for 2 attributes and 58 times slower for 3 attributes on average). Therefore, we use scores based on single attribute distribution for our method.

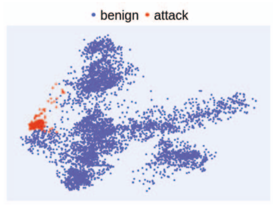
Anomaly detection model. We test the efficacy of one-class support vector machine (OC-SVM) and autoencoder as the anomaly detection model. The autoencoder evaluated in this section is a regular one without LSTM layers. We show the average performance on the 4 attacked hosts from the DARPA OpTC dataset.

For OC-SVM, we find that no single setup works for all scenarios. Detailed results are shown in Table 12 in the appendix. Using a linear kernel function and setting the hyperparameter nu at 0.95, the model performs well on Host 0051, but does not achieve this success across other hosts. A similar pattern is observed with other fixed configurations. We further conduct a grid search of 76 configurations to confirm this pattern. The 76 models are all possible combinations of 4 kernel functions (radial basis function (rbf), linear function, polynomial function, and sigmoid function) and a nu value from 0.05 to 0.95 with a 0.05 increment. Although we find that there is at least one configuration performs well on each attack scenario, the fatal issue is that attack knowledge is required to select a model with good

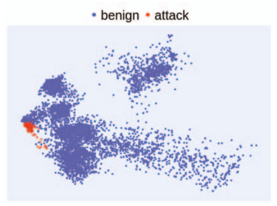
Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on July 11,2025 at 20:05:56 UTC from IEEE Xplore. Restrictions apply.



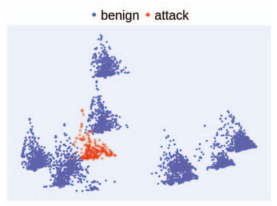
(a) state scores using single attribute distribution



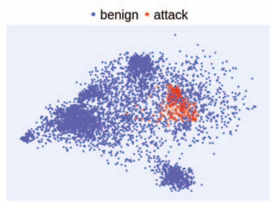
(b) state scores using joint distribution (2 attributes)



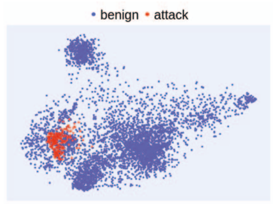
(c) state scores using joint distribution (3 attributes)



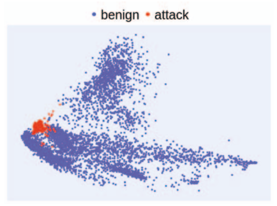
(d) percentage of attributes (no score calculation)



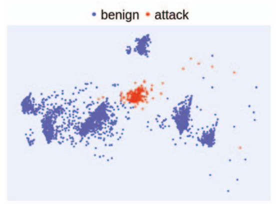
(e) state scores using single attribute distribution (categorical)



(f) state scores using joint distribution (2 attributes, categorical)



(g) state scores using joint distribution (3 attributes, categorical)



(h) percentage of attributes (no score calculation, categorical)

Figure 8: PCA analysis of benign and attack data of host 0051.

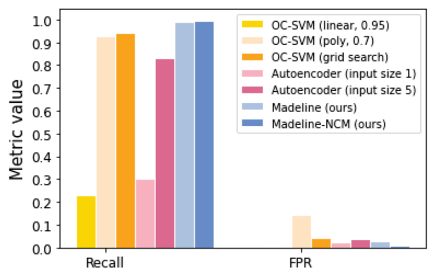


Figure 9: Performance comparison with OC-SVM and autoencoder as baselines. Linear stands for linear kernel, poly for polynomial kernel, 0.95 and 0.7 are values for the nu parameter. MADELINE's performance is based on the self-reconstruction setting.

recall. In a realistic setting, attack data is unlikely to be available at the model development stage.

For autoencoder, we test its performance using an input size of 1 (i.e., one time window) and an input size of 5 (i.e., 5 consecutive windows). Table 13 in the appendix shows the results. Input size 1 yields a recall of only 0.30 and input size 5 improves it to 0.83. The low performance of autoencoder could be because of its incapability of handling time sequence time, capturing irrelevant associations among random input elements yet failing to learn the crucial temporal relationships between scores.

We summarize our major experimental findings as follows:

- MADELINE effectively detects all 3 APT scenarios on 4 different hosts from the DARPA OpTC dataset, with a recall of 0.988 and a FPR as low as 0.03.
- MADELINE-NCM further enhances the detection recall and reduce the FPR in all scenarios. Specifically, it lowers the FPR by 63% for the detection on the DARPA OpTC dataset.
- As a lightweight real-time detection, MADELINE is remarkably more time- and resource-efficient than graph-based solutions, achieving over 1000x reduction on processing time and 5x reduction in memory usage while showing comparable detection effectiveness with state-of-the-art.
- MADELINE's anomaly detection model LSTM autoencoder outperforms other baseline models. It outperforms the best OC-SVM and autoencoder by 5% and 19%, respectively, in terms of recall. It also shows 32% and 23% lower FPR when compared with these two baselines.

5. Discussion

Prediction explainability. Beyond simply making a binary decision on each time window, MADELINE also offers insights into potential malicious behaviors. We infer the top abnormal attributes and category by inspecting the individual reconstruction error of each attribute (Figure 15 in the appendix). Figure 10 correlates the identified top abnormal categories with ground truth attack behaviors on host 0501 from the DARPA OpTC dataset as an example. The top abnormal category is registry when the red team attempted to modify the registry and establish persistence. When data exfiltration occurred, the file category was identified as the most abnormal. Security analysts can utilize this information and focus on top abnormal behaviors, thereby enhancing efficiency. Note that exact correspondence between the predictions and the actual attack is not always guaranteed. One possible reason is the slight misalignments between the attack records logged by the red team and the system logs collected by other teams, while another reason could be our statistical model needs the logs to accumulate to a detectable state change. Despite that, the insights remain valuable for understanding the attack dynamics. For instance, a sequential pattern of [process $\rightarrow$ registry $\rightarrow$ file] might indicate

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on July 11,2025 at 20:05:56 UTC from IEEE Xplore. Restrictions apply.

suspicious processes being created for persistence establishment and file manipulation, which can be compared against known attack patterns to identify potential threats.

Attack log (host 0501, Sept 24)		Madeline Predicted top anomaly category	
11:23:27	Attempted to bypass UAC by modifying a registry entry	11:23:16 - 11:46:30 11:25:42 - 11:49:05 11:27:13 - 11:50:45 ... 11:37:14 - 11:58:42 11:39:55 - 12:00:13	REGISTRY
11:34:56	Set persistence using WMI subscription		
11:37:35	Enumerated registry to determine trusted documents and trusted locations		
...			
13:31:29	Compressed documents in C:\documents for exfiltration	13:30:34 - 13:37:53 13:31:12 - 13:37:56 13:32:05 - 13:38:00 ... 13:38:16 - 13:39:36 13:38:21 - 13:39:40	FILE
13:44:34	Exfiltrated export.zip	13:43:35 - 13:52:52 13:44:40 - 13:53:15 13:45:27 - 13:54:01	FILE
13:45:12	Cleaned up export.zip		

Figure 10: Attack log for red team activities on host 0501 from DARPA OpTC dataset and top abnormal categories for corresponding prediction windows predicted by MADELINE.

Use scenario. In addition to being effective as a standalone detection system, MADELINE can serve as an initial layer of defense within a comprehensive APT investigation and response framework, complementing more detailed attack analyses. Figure 14 in the appendix presents an overview of the entire investigation workflow. Once a risk is verified, high-risk periods can be forwarded to further inspection through entity- and event-level log analyses (such as graph-based [5] and embedding-based [16] analyses) to precisely identify attack activities and reconstruct the attack story. This helps better allocate investigative efforts for genuine high-risk time.

Model choices. We select LSTM autoencoder as the anomaly detection model, despite the existence of more sophisticated sequential models, such as Transformer-based models. The reason is that one major advantage of those large models is their ability to learn token embedding based on the context. However, our statistical model already encapsulates the relationship between current and historical system states within the score vector. Additional context learning might not offer substantial benefits and could potentially disrupt the scores. Additionally, our goal is to develop a lightweight real-time detection. Large models, with significantly more parameters, may require considerably longer training and prediction, affecting the efficiency of monitoring.

Assumption for Gaussian distribution. By using the CDF to calculate probability score for behavioral attributes, we make an implicit assumption about system behavioral frequencies' distribution. While frequencies of behavioral attributes do not strictly follow a Gaussian distribution, we find their frequencies have a normal-distribution-like shape. The scores also give useful information for the downstream prediction. We show an example of their distributions from DARPA OpTC host 0051 during a time period between Sept 20-23 in Figure 13 in the appendix.

Concept drift and model updating. System behaviors may change over time when new tasks are introduced, leading to an increase in false alarms. MADELINE can be retrained and updated quickly within a few seconds, enabling a regularly based (e.g., daily) update of the model. Once it is confirmed that the risk associated with a specific time period is low, the data from this period can be utilized to update the anomaly detection model. We assume the integrity of benign logs used for training. Although the detection of data poisoning is an important field to research, it is beyond the scope of this study.

Limitations. Regarding dataset choices, our evaluation was constrained due to the mixture of attack and benign data in some datasets, such as the popular DARPA Transparent Computing (TC) dataset. As an unsupervised learning framework, MADELINE needs attack-free data for training. Because of the substantial overlap of attack and benign traces, removing attack entries from the log inevitably affects the statistics of benign behaviors and thus fails to reflect a real-world benign setting. While event-level analyses can precisely remove attack edges and exclude abnormal interactions, our coarse-grained detection may be influenced implicitly by the distorted statistic. We take the evaluation of MADELINE on a more diverse set of APT attacks as our future work when more suitable datasets become available.

Similar to other detection systems, MADELINE's efficacy may be affected by finely crafted, sophisticated mimicry attacks that disguise themselves as benign activities [28]–[30]. However, such evasive attacks require significant effort to acquire historical system knowledge, calculate a seemingly benign behavior ratio, and make adjustments with consideration of the ongoing background system activities. Additionally, MADELINE may struggle to generalize to unseen or unknown benign behaviors. That is, models trained on the historical data of one host may not perform well on another host with different behaviors.

6. Related Work

We discuss related works in areas of APT attack detection and investigation, anomaly detection, and LSTM autoencoder.

APT detection and investigation. Existing attack detections mainly utilize provenance graphs, including signature-based [6]–[9], [31]–[33] and learning-based solutions [4], [5], [10]–[15], [17], [19], [26]. Specifically, some works use graph or event statistics. NoDoze [11] computes an anomaly score for every node along a dependency path based on their occurrence frequencies. Unicorn [4] converts counts of system provenance subgraphs to a system state representation to detect any outliers. P-gaussian [32] uses Gaussian distribution to describe attack sequences and identify similar suspicious behaviors. MADELINE adopts a different graph-free statistical approach, enabling lightweight coarse-grained detection with minimal computational resources. Other APT defense works include network-level approaches [34], [35].

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on July 11,2025 at 20:05:56 UTC from IEEE Xplore. Restrictions apply.

Attack investigation happens after the detection of attacks to gain intelligence of the attack, clear threats in the system, and help strengthen the protection. MADELINE can be used along with these systems as the first layer of attack detection. Most investigation works focus on analyzing the causality of events using provenance graphs [21], [36]–[39]. Recent research also uses natural language processing (NLP) methods to track and recover the attack story [16], [40]–[42]. Studies have also been done on provenance graph reduction to save investigative efforts [43]–[48] and on effective logging systems to reduce overhead [49]–[52].

Log anomaly detection. Besides attack analysis, anomaly detection has also been developed for other types of logs to identify task failures or system errors. Sequential and NLP-based solutions are widely used in this field [22], [53]–[58]. Researchers also use autoencoders to reconstruct to extract features for anomaly detection [59]. Different from these works, MADELINE reconstructs preprocessed system state scores instead of single events.

LSTM autoencoder. Previous applications of LSTM autoencoder fall in the field of video reconstruction [60]. Recent studies apply it to detect anomalies in network traffic [61]–[63]. MADELINE applies LSTM autoencoder to reconstruct the system state in a more complex APT detection scenario.

7. Conclusion

We propose MADELINE, a novel advanced threat detection facilitating prompt online identification of malicious system state. MADELINE utilizes statistical models to condense the voluminous system audit logs and effectively capture high-risk periods that deviate from normal. Our evaluation shows that MADELINE is effective against various APT scenarios, achieving comparable accuracy with state-of-the-art tools while being significantly efficient in terms of computational resource utilization. The fast detection enables organizations to respond to attacks quickly, preventing potential financial losses.

Acknowledgement

This work has been supported by the Office of Naval Research under Grant N00014-22-1-2057 and the Virginia Commonwealth Cyber Initiative (CCI).

References

[1] "The great bangladesh cyber heist shows truth is stranger than fiction." [Online]. Available: https://www.dhakatribune.com/opinion/op-ed/122939/the-great-bangladesh-cyber-heist-shows-truth-is

[2] "Home — equifax data breach settlement." [Online]. Available: https://www.equifaxbreachsettlement.com/

[3] "Inside the opm hack, the cyberattack that shocked the us government — wired." [Online]. Available: https://www.wired.com/2016/10/inside-cyberattack-shocked-us-government/

[4] X. Han, T. F. J. Pasquier, A. Bates, J. Mickens, and M. I. Seltzer, "UNICORN: runtime provenance-based detector for advanced persistent threats," in *Proceedings of Symposium on Network and Distributed System Security (NDSS)*, February 2020.

[5] Z. Cheng, Q. Lv, J. Liang, Y. Wang, D. Sun, T. Pasquier, and X. Han, "KAIROS: Practical intrusion detection and investigation using whole-system provenance," in *2024 IEEE Symposium on Security and Privacy (SP)*, 2024.

[6] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan, "Holmes: Real-time APT detection through correlation of suspicious information flows," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 1137–1152.

[7] W. U. Hassan, A. Bates, and D. Marino, "Tactical provenance analysis for endpoint detection and response systems," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 1172–1189.

[8] M. N. Hossain, S. Sheikhi, and R. Sekar, "Combating dependence explosion in forensic analysis using alternative tag propagation semantics," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 1139–1155.

[9] S. M. Milajerdi, B. Eshete, R. Gjomemo, and V. Venkatakrishnan, "Poirot: Aligning attack behavior with kernel audit records for cyber threat hunting," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 1795–1812.

[10] Y. Liu, M. Zhang, D. Li, K. Jee, Z. Li, Z. Wu, J. Rhee, and P. Mittal, "Towards a timely causality analysis for enterprise security." in *Proceedings of Symposium on Network and Distributed System Security (NDSS)*, 2018.

[11] W. U. Hassan, S. Guo, D. Li, Z. Chen, K. Jee, Z. Li, and A. Bates, "Nodoze: Combatting threat alert fatigue with automated provenance triage," in *Proceedings of Symposium on Network and Distributed System Security (NDSS)*, 2019.

[12] W. U. Hassan, D. Li, K. Jee, X. Yu, K. Zou, D. Wang, Z. Chen, Z. Li, J. Rhee, J. Gui *et al.*, "This is why we can't cache nice things: Lightning-fast threat hunting using suspicion-based hierarchical storage," in *Proceedings of the 36th Annual Computer Security Applications Conference*, 2020, pp. 165–178.

[13] Q. Wang, W. U. Hassan, D. Li, K. Jee, X. Yu, K. Zou, J. Rhee, Z. Chen, W. Cheng, C. A. Gunter *et al.*, "You are what you do: Hunting stealthy malware via data provenance analysis." in *Proceedings of Symposium on Network and Distributed System Security (NDSS)*, 2020.

[14] M. U. Rehman, H. Ahmadi, and W. U. Hassan, "Flash: A comprehensive approach to intrusion detection via provenance graph representation learning," in *IEEE Symposium on Security and Privacy (SP)*, 2024, pp. 139–139.

[15] F. Yang, J. Xu, C. Xiong, Z. Li, and K. Zhang, "PROGRAPHER: An anomaly detection system based on provenance graph embedding," in *32nd USENIX Security Symposium (USENIX Security 23)*, Anaheim, CA, Aug. 2023, pp. 4355–4372.

[16] H. Ding, J. Zhai, Y. Nan, and S. Ma, "Airtag: Towards automated attack investigation by unsupervised learning with log texts," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 373–390.

[17] F. Dong, L. Wang, X. Nie, F. Shao, H. Wang, D. Li, X. Luo, and X. Xiao, "DISTDET: A cost-effective distributed cyber threat detection system," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 6575–6592.

[18] "Fivedirections/optc-data." [Online]. Available: https://github.com/FiveDirections/OpTC-data

[19] J. Zeng, X. Wang, J. Liu, Y. Chen, Z. Liang, T.-S. Chua, and Z. L. Chua, "Shadewatcher: Recommendation-guided cyber threat analysis using system audit records," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 489–506.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on July 11,2025 at 20:05:56 UTC from IEEE Xplore. Restrictions apply.

[21] F. Liu, Y. Wen, D. Zhang, X. Jiang, X. Xing, and D. 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*, 2019, pp. 1777–1794.

[22] M. Du, F. Li, G. Zheng, and V. 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*, 2017, pp. 1285–1298.

[23] "Darpa optc ecar - google drive." [Online]. Available: https://drive.google.com/drive/u/0/folders/1NwaCWRyr_coyPbF2SvScbani5O9MXp7_

[24] E. Manzoor, S. M. Milajerdi, and L. 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*, ser. KDD '16, 2016, p. 1035–1044.

[25] "Tactics - enterprise — mitre att&ck." [Online]. Available: https://attack.mitre.org/tactics/enterprise/

[26] S. Wang, Z. Wang, T. Zhou, H. Sun, X. Yin, D. Han, H. Zhang, X. Shi, and J. Yang, "Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3972–3987, 2022.

[27] "Provenanceanalytics/kairos - github." [Online]. Available: https://github.com/ProvenanceAnalytics/kairos/blob/main/DARPA/OpTC/optc_graph_learning.ipynb

[28] A. Goyal, X. Han, G. Wang, and A. 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.

[29] D. Wagner and P. Soto, "Mimicry attacks on host-based intrusion detection systems," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, 2002, pp. 255–264.

[30] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," in *International conference on machine learning*. PMLR, 2018, pp. 1115–1124.

[31] C. Xiong, T. Zhu, W. Dong, L. Ruan, R. Yang, Y. Cheng, Y. Chen, S. Cheng, and X. Chen, "Conan: A practical real-time apt detection system with high accuracy and efficiency," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 1, pp. 551–565, 2020.

[32] Y. Xie, Y. Wu, D. Feng, and D. Long, "P-gaussian: provenance-based gaussian distribution for detecting intrusion behavior variants using high efficient and real time memory databases," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 6, pp. 2658–2674, 2019.

[33] Y. Xie, D. Feng, Y. Hu, Y. Li, S. Sample, and D. Long, "Pagoda: A hybrid approach to enable efficient real-time provenance based intrusion detection in big data environments," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 6, pp. 1283–1296, 2018.

[34] O. Bajaber, B. Ji, and P. Gao, "P4control: Line-rate cross-host attack prevention via in-network information flow control enabled by programmable switches and ebpf," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2024, pp. 147–147.

[35] Y. Ji, S. Lee, M. Fazzini, J. Allen, E. Downing, T. Kim, A. Orso, and W. Lee, "Enabling refinable {Cross-Host} attack investigation with efficient data flow tagging and tracking," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1705–1722.

[36] M. N. Hossain, S. M. Milajerdi, J. Wang, B. Eshete, R. Gjomemo, R. Sekar, S. Stoller, and V. Venkatakrishnan, "Sleuth: Real-time attack scenario reconstruction from cots audit data," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 487–504.

[37] P. Gao, X. Xiao, Z. Li, F. Xu, S. R. Kulkarni, and P. Mittal, "Aiql: Enabling efficient attack investigation from system monitoring data," in *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, 2018, pp. 113–126.

[38] A. Alsaheel, Y. Nan, S. Ma, L. Yu, G. Walkup, Z. B. Celik, X. Zhang, and D. Xu, "Atlas: A sequence-based learning approach for attack investigation," in *30th USENIX security symposium (USENIX security 21)*, 2021, pp. 3005–3022.

[39] K. Pei, Z. Gu, B. Saltaformaggio, S. Ma, F. Wang, Z. Zhang, L. Si, X. Zhang, and D. Xu, "Hercule: Attack story reconstruction via community discovery on correlated log graph," in *Proceedings of the 32Nd Annual Conference on Computer Security Applications*, 2016, pp. 583–595.

[40] P. Gao, F. Shao, X. Liu, X. Xiao, Z. Qin, F. Xu, P. Mittal, S. R. Kulkarni, and D. Song, "Enabling efficient cyber threat hunting with cyber threat intelligence," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 193–204.

[41] S. Nedelkoski, J. Bogatinovski, A. Acker, J. Cardoso, and O. Kao, "Self-attentive classification-based anomaly detection in unstructured logs," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 1196–1201.

[42] Y. Shen and G. Stringhini, "{ATTACK2VEC}: Leveraging temporal word embeddings to understand the evolution of cyberattacks," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 905–921.

[43] P. Fang, P. Gao, C. Liu, E. Ayday, K. Jee, T. Wang, Y. F. Ye, Z. Liu, and X. Xiao, "{Back-Propagating} system dependency impact for attack investigation," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 2461–2478.

[44] Z. Xu, P. Fang, C. Liu, X. Xiao, Y. Wen, and D. Meng, "Depcomm: Graph summarization on system audit logs for attack investigation," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 540–557.

[45] J. Zeng, Z. L. Chua, Y. Chen, K. Ji, Z. Liang, and J. Mao, "Watson: Abstracting behaviors from audit logs via aggregation of contextual semantics." 2021.

[46] Y. Tang, D. Li, Z. Li, M. Zhang, K. Jee, X. Xiao, Z. Wu, J. Rhee, F. Xu, and Q. Li, "Nodemerge: Template based efficient data reduction for big-data causality analysis," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1324–1337.

[47] Z. Xu, Z. Wu, Z. Li, K. Jee, J. Rhee, X. Xiao, F. Xu, H. Wang, and G. Jiang, "High fidelity data reduction for big data security dependency analyses," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 504–516.

[48] T. Van Ede, H. Aghakhani, N. Spahn, R. Bortolameotti, M. Cova, A. Continella, M. van Steen, A. Peter, C. Kruegel, and G. Vigna, "Deepcase: Semi-supervised contextual analysis of security events," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 522–539.

[49] S. Ma, X. Zhang, and D. Xu, "Protracer: Towards practical provenance tracing by alternating between logging and tainting," in *23rd Annual Network And Distributed System Security Symposium (NDSS 2016)*, 2016.

[50] H. Ding, S. Yan, J. Zhai, and S. Ma, "{ELISE}: A storage efficient logging system powered by redundancy reduction and representation learning," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 3023–3040.

[51] H. Ding, J. Zhai, D. Deng, and S. Ma, "The case for learned provenance graph storage systems," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 3277–3294.

[52] W. U. Hassan, L. Aguse, N. Aguse, A. Bates, and T. Moyer, "Towards scalable cluster auditing through grammatical inference over provenance graphs," in *Network and Distributed Systems Security Symposium*, 2018.

[53] H. Guo, S. Yuan, and X. Wu, "Logbert: Log anomaly detection via bert," in *2021 international joint conference on neural networks (IJCNN)*. IEEE, 2021, pp. 1–8.

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on July 11,2025 at 20:05:56 UTC from IEEE Xplore. Restrictions apply.

[54] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun *et al.*, "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs." in *IJCAI*, vol. 19, no. 7, 2019, pp. 4739–4745.

[55] Y. Lee, J. Kim, and P. Kang, "Lanobert: System log anomaly detection based on bert masked language model," *Applied Soft Computing*, vol. 146, p. 110689, 2023.

[56] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li *et al.*, "Robust log-based anomaly detection on unstable log data," in *Proceedings of the 2019 27th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*, 2019, pp. 807–817.

[57] Q. Cheng, A. Saha, W. Yang, C. Liu, D. Sahoo, and S. Hoi, "Logai: A library for log analytics and intelligence," *arXiv preprint arXiv:2301.13415*, 2023.

[58] Z. Chen, J. Liu, W. Gu, Y. Su, and M. R. Lyu, "Experience report: Deep learning-based system log analysis for anomaly detection," *arXiv preprint arXiv:2107.05908*, 2021.

[59] A. Farzad and T. A. Gulliver, "Unsupervised log message anomaly detection," *ICT Express*, vol. 6, no. 3, pp. 229–237, 2020.

[60] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *International conference on machine learning*. PMLR, 2015, pp. 843–852.

[61] M. Said Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Network anomaly detection using lstm based autoencoder," in *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, ser. Q2SWinet '20, 2020, p. 37–45.

[62] J. Ashraf, A. D. Bakhshi, N. Moustafa, H. Khurshid, A. Javed, and A. Beheshti, "Novel deep learning-enabled lstm autoencoder architecture for discovering anomalous events from intelligent transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4507–4518, 2021.

[63] H. Homayouni, S. Ghosh, I. Ray, S. Gondalia, J. Duggan, and M. G. Kahn, "An autocorrelation-based lstm-autoencoder for anomaly detection on time-series data," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 5068–5077.

Appendix

TABLE 11: Attack scenarios of DARPA OpTC dataset.

Attack senario	Date	Attacked hosts
Plain PowerShell Empire	Sept 23 (day 1)	0201, 0660
Custom Powershell Empire	Sept 24 (day 2)	0501
Malicious Upgrade	Sept 25 (day 3)	0051

TABLE 12: Performance of OC-SVM on attack scenarios as a baseline.

	Fixed (linear, 0.95)		Fixed (poly, 0.7)		Grid search	
Host	**Recall**	**FPR**	**Recall**	**FPR**	**Recall**	**FPR**
0051	0.929	0.006	1.000	0.455	0.929	0.006
0501	0.000	0.000	0.904	0.033	0.904	0.033
0660	0.000	0.000	0.940	0.075	0.950	0.094
0201	0.000	0.000	0.860	0.000	0.983	0.043
avg	0.232	0.002	0.926	0.141	0.942	0.044

TABLE 13: Performance of Autoencoder on attack scenarios as a baseline.

Host	Input size 1		Input size 5	
	Recall	**FPR**	**Recall**	**FPR**
0051	0.352	0.006	0.787	0.000
0501	0.566	0.022	0.862	0.000
0660	0.283	0.019	0.871	0.087
0201	0.004	0.052	0.797	0.069
avg	0.301	0.025	0.829	0.039

TABLE 14: Attributes selected for StreamSpot dataset.

Category	Attributes
file	file-execve, file-access, file-open, file-fstat, file-mmap2, file-close, file-read, file-stat, file-write, file-unlink, file-listen, file-chmod, file-connect, file-writev, file-recv, file-ftruncate, file-sendmsg, file-send, file-recvmsg, file-accept, file-sendto, file-recvfrom, file-truncate, file-bind
process	process-clone, process-waitpid

Alternative state score calculation. An alternative way of normalizing the frequency is to divide each attribute's frequency by the total number of entries in its category:

$$freq_i = \frac{\#\ occurrences\ of\ attribute_i}{\#\ entries\ in\ this\ category} \tag{10}$$

where $length\ of\ window = \sum_j \#\ entries\ in\ category_j$. We refer to this setting as categorical normalization.

For state score calculation, besides using the single attributes, we also investigate alternatives using joint attribute distributions. A score using a joint distribution for 2 attributes is calculated as

$$\begin{aligned} score = 1 - P(\mu_1 \ - \ diff_1 < X_1 < \mu_1 \ + \ diff_1, \\ \mu_2 \ - \ diff_2 < X_2 < \mu_2 \ + \ diff_2) \end{aligned} \tag{11}$$

where $diff_1$ is the difference between the distribution mean μ_1 for the first attribute and the newly observed data point d_1 for the first attribute, similarly for $diff_2$:

$$diff_1 = |\mu_1 - d_1|\,,\ diff_2 = |\mu_2 - d_2| \tag{12}$$

and $F_{X_1,\ X_2}$ is the CDF of $X_1,\ X_2$:

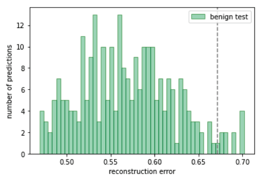
$$\begin{aligned} F_{X_1,X_2}(x_1,x_2) &= P(X_1 \le x_1, X_2 \le x_2) \\ &= \int_{-\infty}^{x_1}\int_{-\infty}^{x_2} f_{X_1,X_2}(u,v)\,dv\,du \end{aligned} \tag{13}$$

and f_X is the PDF of $X_1,\ X_2$:

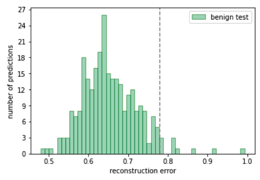
$$\begin{aligned} &f(\mathbf{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \\ &\frac{1}{\sqrt{(2\pi)^2\det(\boldsymbol{\Sigma})}}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \end{aligned} \tag{14}$$

with $\mathbf{x}$ being the vector for newly observed values ($[d_1,\ d_2]$ in this case), μ being the vector for distribution mean, $\boldsymbol{\Sigma}$ being the covariance matrix, $\det(\boldsymbol{\Sigma})$ being the determinant of the covariance matrix. Using the joint distribution of multiple variables results in an increase in the dimensions of the score vector relative to using single variable distribution.

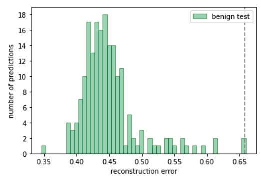
Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on July 11,2025 at 20:05:56 UTC from IEEE Xplore. Restrictions apply.



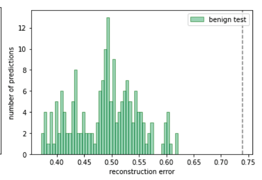
(a) host 0070



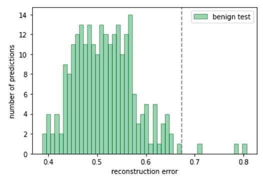
(b) host 0101



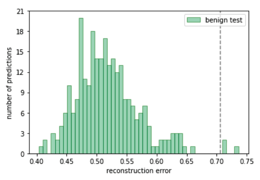
(c) host 0307



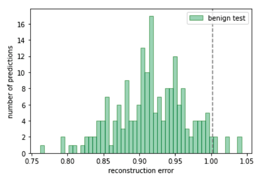
(d) host 0455



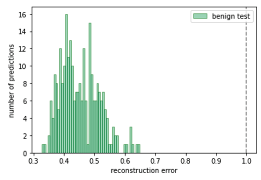
(e) host 0468



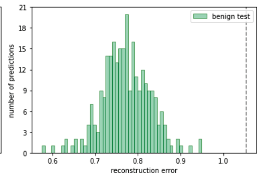
(f) host 0470



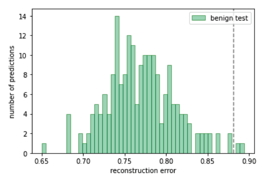
(g) host 0607



(h) host 0720

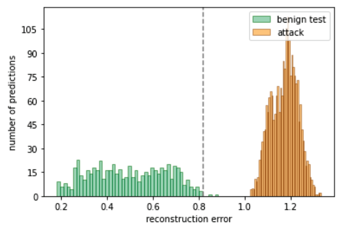


(i) host 0771

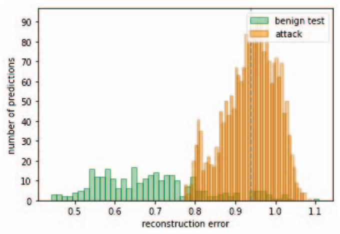


(j) host 0860

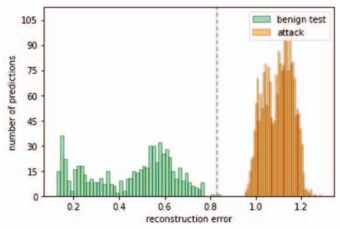
Figure 11: Distribution of data reconstruction errors on attack-free hosts (self-reconstruction).



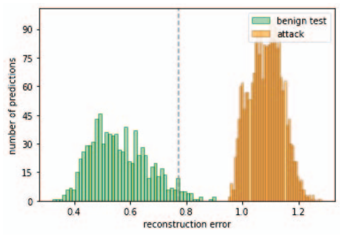
(a) YouTube



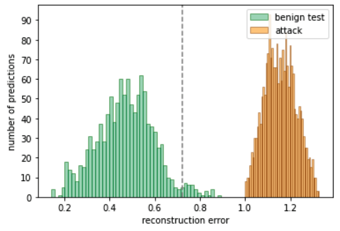
(b) GMail



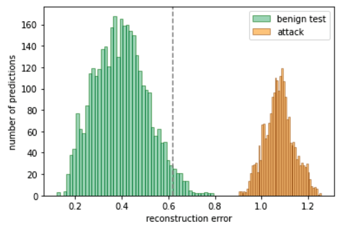
(c) VGame



(d) Download



(e) CNN



(f) All

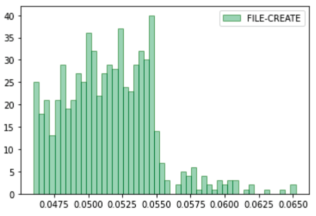
Figure 12: Distribution of data reconstruction errors on StreamSpot dataset (self-reconstruction).

That is, when considering all possible joint distributions of k variables chosen from the n behavior attributes from the previous step, the dimension of the score vector computes as

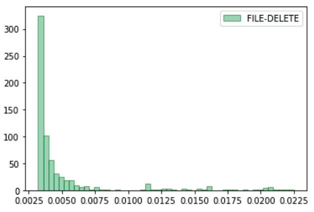
$$dimension = C(n,k) = \binom{n}{k} = \frac{n!}{k!(n-k)!} \qquad (15)$$

where n is the number of selected attributes and k. For instance, in the case that we have 12 behavior attributes and employ joint distributions of 2 variables for score calculation, the score vectors will have a dimension of 66. We omit the details for the joint distribution of 3 attributes here, which follows the same procedure.

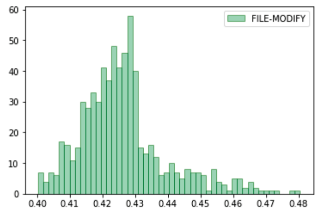
Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on July 11,2025 at 20:05:56 UTC from IEEE Xplore. Restrictions apply.



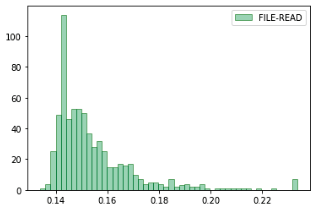
(a) FILE CREATE



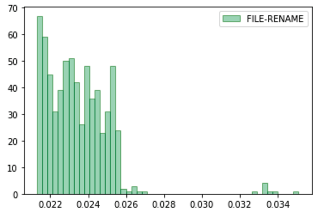
(b) FILE DELETE



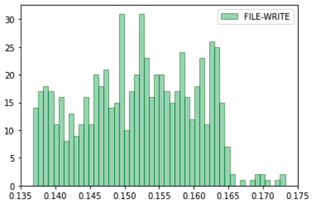
(c) FILE MODIFY



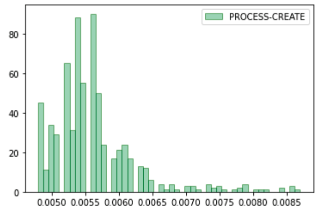
(d) FILE READ



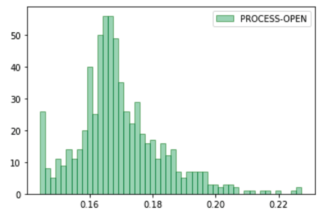
(e) FILE RENAME



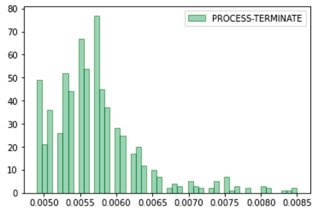
(f) FILE WRITE



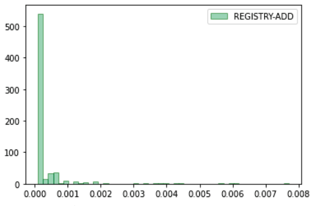
(g) PROCESS CREATE



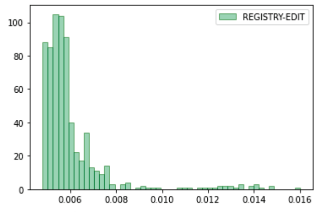
(h) PROCESS OPEN



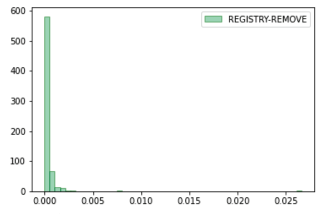
(i) PROCESS TERMINATE



(j) REGISTRY ADD



(k) REGISTRY EDIT



(l) REGISTRY REMOVE

Figure 13: Distribution of system behavioral attributes. The example is based on a time period on DARPA OpTC host 0051 between Sept 20-23 (AIA-51-75.ecar-2019-12-08T00-56-58.175 in the original dataset).

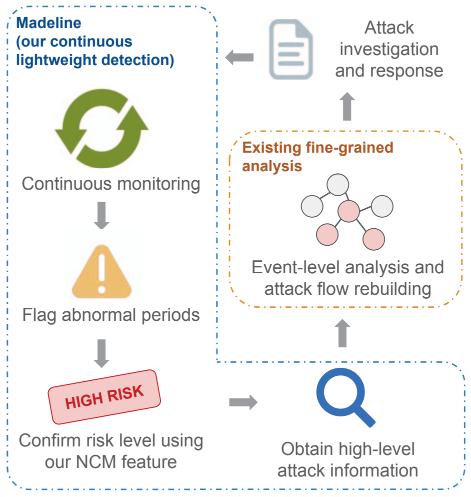
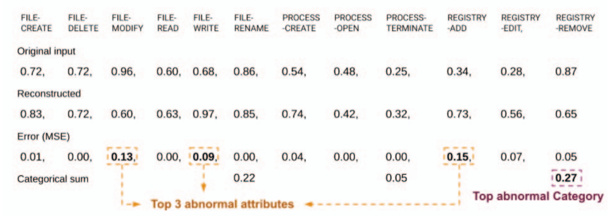


Figure 14: Continuous monitoring and attack investigation life cycle.



	FILE-CREATE	FILE-DELETE	FILE-MODIFY	FILE-READ	FILE-WRITE	FILE-RENAME	PROCESS-CREATE	PROCESS-OPEN	PROCESS-TERMINATE	REGISTRY-ADD	REGISTRY-EDIT,	REGISTRY-REMOVE
Original input	0.72,	0.72,	0.96,	0.60,	0.68,	0.86,	0.54,	0.48,	0.25,	0.34,	0.28,	0.87
Reconstructed	0.83,	0.72,	0.60,	0.63,	0.97,	0.85,	0.74,	0.42,	0.32,	0.73,	0.56,	0.65
Error (MSE)	0.01,	0.00,	**0.13,**	0.00,	**0.09,**	0.00,	0.04,	0.00,	0.00,	**0.15,**	0.07,	0.05
Categorical sum						0.22			0.05			**0.27**

Top 3 abnormal attributes
Top abnormal Category

Figure 15: Example of calculating top abnormal attributes and category using reconstruction errors.

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on July 11,2025 at 20:05:56 UTC from IEEE Xplore. Restrictions apply.