Unsupervised Anomaly Detection in OpenStack Logs via Fine-Tuned RoBERTa Embeddings

Janit Rajkarnikar, Nishan Poudel and Nick Rahimi*

School of Computer Sciences, University of Southern Mississippi, Hattiesburg, MS, USA

Abstract: As cloud computing infrastructures increasingly depend on reliable operation, proactively detecting anomalies in system logs becomes indispensable. Conventional log analysis techniques often produce high false-alarm rates and exhibit limited semantic understanding. To address these limitations, we developed an unsupervised anomaly detection framework that leverages fine-tuned RoBERTa-base model embeddings to capture contextual patterns within OpenStack log event sequences. We apply a crucial filtering step to remove high-frequency, non-discriminatory events, ensuring our models learn from nuanced contextual signals rather than simple indicators. From these refined sequences, we construct a custom vocabulary and fine-tune RoBERTa with Parameter-Efficient Fine-Tuning (PEFT) using LoRA. These contextualized embeddings inform unsupervised classifiers, including Isolation Forest and One-Class SVM, trained solely on normal data. Our approach demonstrates excellent and robust performance on a holdout test set (Anomaly F1-Score up to 0.97), significantly outperforming traditional LSTM-based baselines on the same task. These results demonstrate that contextualized transformer embeddings provide a powerful and resilient foundation for log-based anomaly detection, reducing false alarms and improving detection accuracy in complex cloud environments.

Keywords: Anomaly Detection, Transformer Embeddings, RoBERTa, Isolation Forest, One-Class SVM, OpenStack Logs, Unsupervised Learning, PEFT, LoRA.

1. INTRODUCTION

As cloud computing infrastructures continue to scale and support critical applications, real-time monitoring of system health through log analysis has become essential. System logs offer rich operational data, yet traditional log-mining solutions, often relying on template-based parsing or statistical methods like PCA, frequently struggle with high false-alarm rates and a lack of semantic depth needed to detect nuanced deviations in event sequences [1-4].

Recent advances in deep learning, particularly transformer-based language models, present a significant opportunity to overcome these challenges. For instance, LogBERT demonstrated that fine-tuned BERT embeddings can capture contextual dependencies within log messages, outperforming classical baselines on public datasets [5]. Furthermore, Parameter Efficient Fine-Tuning (PEFT) techniques enable the application of large language models to log mining with reduced computational overhead while achieving competitive performance [6].

Despite these advancements, many studies focus on benchmark datasets like HDFS and BGL, leaving cloud-native platforms such as OpenStack, a widely adopted, complex, and mission-critical infrastructure, relatively underexplored by these sophisticated semantic approaches. While some work on OpenStack logs has employed methods like Robust PCA [4] to achieve respectable detection, such statistical approaches often do not fully leverage the sequential and semantic richness inherent in log instance streams. This gap is compounded by the known variability in log template extraction quality across different sources [1-3], underscoring the critical need for methods that integrate robust log parsing with deep contextual modeling, especially for multifaceted systems like OpenStack.

At the same time, there remains a broader gap in how these methods contribute to the security and forensic value of logs. In operational environments, especially within Security Operations Centers (SOCs), excessive false alarms increase analyst fatigue and extend mean time to respond. Moreover, limited contextual understanding of log sequences weakens the ability to reconstruct incident timelines or trace abnormal behaviors in forensic analysis. Bridging this methodological gap is therefore crucial not only for improving detection accuracy but also for strengthening cyber defense and post incident response capabilities in cloud infrastructures.

This paper addresses these challenges by presenting an unsupervised anomaly detection framework specifically tailored for OpenStack system logs (illustrated in Figure **1**). We begin by applying the Drain algorithm to parse raw logs into instance level event sequences. From a dataset of 5,452 such sequences, we then construct a domain-specific

*Address correspondence to this author at School of Computer Sciences, University of Southern Mississippi, Hattiesburg, MS, USA;
E-mail: nick.rahimi@usm.edu

Figure 1: Overview of the proposed unsupervised anomaly detection framework for OpenStack logs using fine-tuned RoBERTa embeddings. This diagram illustrates the end-to-end workflow: raw OpenStack logs are first parsed with the Drain algorithm to extract structured event templates. After filtering high-frequency events, sequences of EventIDs are generated per VM instance. These sequences are used to build a custom vocabulary and fine-tune a RoBERTa-base model with Parameter-Efficient Fine-Tuning (LoRA). The resulting contextual embeddings are then used to train Isolation Forest and One-Class SVM detectors, which classify unseen log sequences as normal or anomalous.

vocabulary and fine-tune a RoBERTa-base model using PEFT (LoRA) with Optuna-driven hyperparameter search. This process yields contextualized 768-dimensional embeddings. Finally, we feed these embeddings into Isolation Forest and One-Class SVM classifiers, trained exclusively on normal data. Our approach significantly reduces false alarms and improves detection accuracy across diverse OpenStack failure scenarios, outperforming traditional template- and PCA-based baselines and demonstrating the power of contextualized embeddings for this domain.

2. TRAINING METHODS

This section details the comprehensive methodology we employed in our unsupervised anomaly detection framework. Our process encompasses dataset preparation, log preprocessing, RoBERTa-base model fine-tuning, contextual embedding extraction, and the training of anomaly detection classifiers.

A. Dataset Preparation

The dataset consists of instance-level event sequences parsed from raw OpenStack logs. We begin by loading the structured CSV outputs produced by the Drain parser (see Section II-B). This dataset includes four distinct log families:

- **Normal VM creation** (openstack nova normal vm create): 4,944 instances • DHCP off (openstack nova dhcp off): 199 instances
- **Immediate VM destroy** (openstack vm destroy immediately after create): 196 instances
- **Undefine VM after create** (openstack nova un define vm after create): 113 instances

This yields a total of 5,452 parsed log instances, each represented as an ordered sequence of event templates (or EventIDs). The distribution of these instances is shown in Figure **2**.

Before partitioning the data, we perform a crucial filtering step to enhance our evaluation. We first analyzed the event frequency across all log types and identified two high-frequency event IDs that acted as overly strong, simplistic discriminators between normal and abnormal sequences. To ensure our models learn from broader contextual patterns rather than relying on these “telltale” events, we remove them from all sequences. This step creates a more challenging and realistic detection scenario, testing the model’s ability to understand nuanced event relationships.

We label each log instance as “normal” or “anomaly” based on its source family. Subsequently, we perform an 85/15 stratified split of the entire dataset, carefully preserving the original class proportions. This split creates a development set of 4,634 instances (4,202 normal, 432 anomaly) and a holdout test set of 818 instances (742 normal, 76 anomaly). We use stratified sampling on the instance labels to ensure a balanced representation of each failure scenario in both subsets.

Finally, we serialize the development and test splits into NumPy archives (.npz files). These archives store the event sequence arrays and their corresponding label vectors, facilitating efficient data loading for subsequent model fine-tuning and evaluation.

B. Preprocessing

Our preprocessing pipeline (Figure **3**) transforms raw OpenStack log files into structured event sequences suitable for transformer modeling. First, we ingest the raw logs. We apply a regular expression that matches the start of each log entry (timestamp + level). Any subsequent lines not matching this pattern are concatenated to their predecessor with a literal “\n” separator; this step preserves multi-line traces and wrapped messages as single logical events.

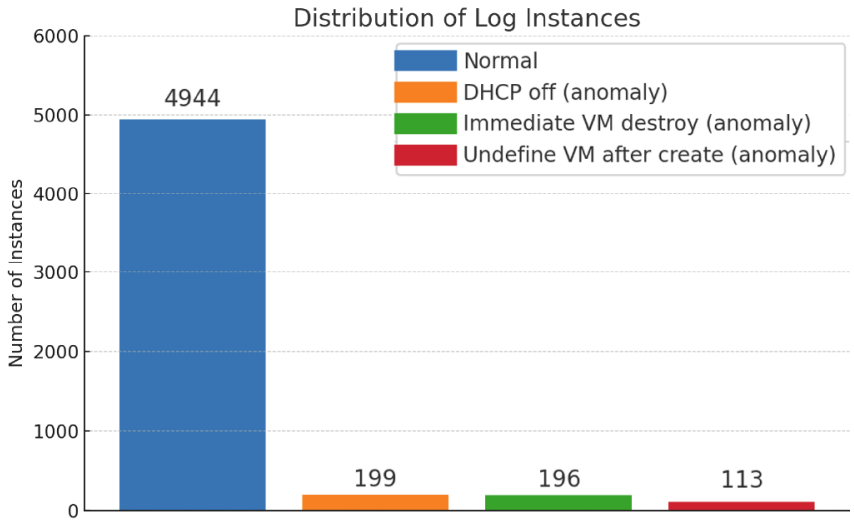

Figure 2: Distribution of log instances across normal and anomaly categories in the full OpenStack dataset. The bar chart shows the number of parsed VM lifecycle instances belonging to each event family: 4,944 normal VM creations and three anomaly types: 199 DHCP-off, 196 immediate-destroy, and 113 undefine-after-create cases, yielding a total of 5,452 instances used for model development and testing.

Next, we parse these consolidated logs using the Drain algorithm. Drain clusters raw messages into templates (e.g., “VM instance <ID> started”) and emits structured CSV files. For each event, these files contain fields such as EventId (a unique string identifier assigned to each template, e.g., “E12345”) and ParameterList (containing extracted variables like instance IDs).

To reduce noise and improve template generalization, we replace variable content, such as numeric values, UUIDs, IP addresses, and other free-form parameters, with generic placeholders like <NUM> and <ID> via regex before parsing with Drain.

Following parsing, we construct per-instance sequences. We group together all events sharing the same instance value (extracted from the ParameterList column) and order them by timestamp. This process results in ordered lists of string EventIds for each VM lifecycle. At the end of this stage, each log instance is represented as a sequence of string-based EventId tokens (representing log templates), which serve as input for the subsequent vocabulary construction and model fine-tuning.

C. Vocabulary Construction

We construct a custom vocabulary exclusively from the development set (4,634 instances) to prevent data leakage from the holdout test set. First, we aggregate all event sequences within the development set. Then, we identify all unique string EventId tokens (e.g., “E5”, “E23”) present across these sequences. The vocabulary comprises these unique EventIds plus five standard special tokens: [PAD], [UNK], [CLS], [SEP], and [MASK]. We assign a unique integer index to each unique EventId string and each special token. This process yields a final vocabulary of 369 tokens. We serialize the resulting vocabulary dictionary, which maps string tokens to integer indices, to a JSON file (event_vocab_dev.json) for use during model fine-tuning and embedding extraction.

D. Masked Language Model Fine-Tuning

To adapt the transformer to OpenStack event-sequence semantics, we fine-tune a pretrained RoBERTa-base model with a Masked Language Modeling (MLM) objective, as illustrated conceptually in Figure **4**. First, we resize the model’s vocabulary and

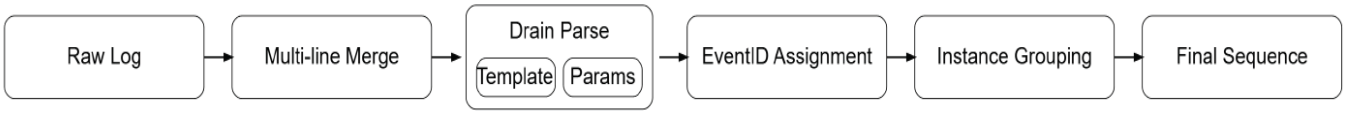

Figure 3: Log preprocessing pipeline transforming raw OpenStack logs into structured EventID sequences. The pipeline consolidates multi-line messages, applies the Drain log parser to produce structured templates, replaces variable content (e.g., UUIDs, IP addresses) with placeholders, and groups events by VM instance. The ordered sequences of EventIDs produced at the end of this process form the tokenized input for vocabulary construction and RoBERTa fine-tuning.

embedding layer to accommodate our custom 369 token event-ID vocabulary.

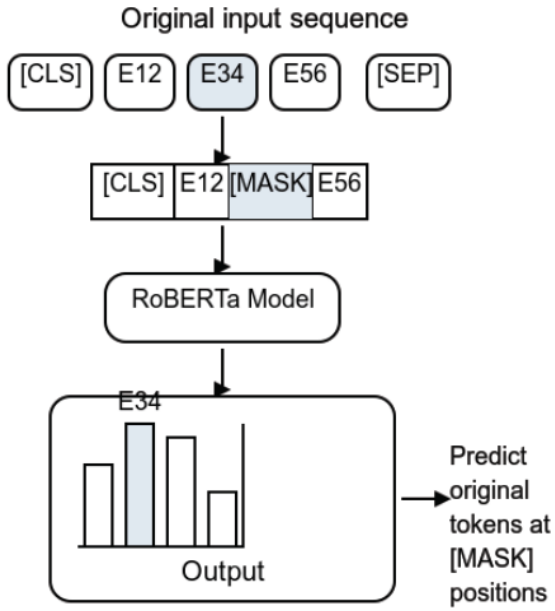

Figure 4: Masked Language Model (MLM) fine-tuning objective applied to the RoBERTa-base model for EventID sequences. During training, approximately 15 % of EventIDs in each sequence are randomly masked, and the model learns to predict them based on surrounding context. This self-supervised task enables RoBERTa to capture sequential and semantic relationships between log events, producing domain-adapted contextual embeddings for anomaly detection.

The development set sequences which were tokenized into integer ID sequences by a custom tokenizer using the vocabulary built in the previous step, are padded or truncated to a fixed length of 64. These sequences are stratified into a 90% training subset (4,170 instances) and a 10% validation subset (464 instances). We employ PEFT via Low-Rank Adaptation (LoRA) [6] with parameters r=8, lora_alpha=32, and lora_dropout=0.1 applied to the query, key, and value projection layers of the transformer.

During training, we mask 15% of tokens and optimize the standard MLM cross-entropy over masked positions.

We tuned LoRA/optimizer settings with a brief Optuna search (5 trials, 5 epochs per trial), selecting the checkpoint with lowest validation loss. The best run used a learning rate of 6.17×10^{-4}, batch size 32, weight decay ≈ 0.0237, and warmup ratio ≈ 0.1739. We then trained up to 10 epochs with early stopping (patience 3) and kept the best checkpoint (validation loss 0.3375). A concise summary is in Table **1**.

Table 1: MLM Fine-Tuning Summary

Item	Value
Search budget	5 trials, 5 epochs/trial
Mask rate	15%
Best LR	6.17×10^{-4}
Batch size	32
Weight decay	≈ 0.0237
Warmup ratio	≈ 0.1739
Early stopping	Patience 3 (max 10 epochs)
Best val. loss	0.3375

E. Embedding Extraction

After fine-tuning the Masked Language Model, we extract fixed-dimensional vector representations (embeddings) for each event sequence in both the development and holdout test sets. This process involves using the fine-tuned RoBERTa-base model obtained from the previous stage. The saved LoRA adapters are loaded and merged into the base model for efficient inference.

For each sequence (represented as a list of string EventIds), we first apply a tokenization process specific to this stage using the custom vocabulary constructed previously. This involves prepending a [CLS] token and appending a [SEP] token to the sequence of integer IDs corresponding to the EventIds. We then pad or truncate the resulting sequence to a fixed length of 64 tokens and generate an attention mask. We feed these processed sequences into the encoder part (specifically, the roberta attribute) of the fine-tuned RobertaForMaskedLM model, thereby bypassing the MLM prediction head. We use CLS pooling on the final encoder layer to obtain a 768-dimensional sequence embedding per instance.

These 768-dimensional embeddings are generated for all instances in both the development and holdout test sets. We then save the resulting embedding arrays, along with their corresponding instance identifiers and labels, to separate NumPy archives (.npz files) for use in subsequent training and evaluation of anomaly detection classifiers.

F. Anomaly Detector Training

With the 768-dimensional embeddings generated for each instance, we train two unsupervised anomaly detection models using the development set: Isolation

Table 2: Detector Tuning Summary (Concise)

Model	Search	Selected Config
IF	Optuna (50 trials)	n estimators = 400, max samples ≈ 0.799, contam. ≈ 0.0107, max features ≈ 0.681, bootstrap = False
OCSVM	Optuna (75 trials)	Linear kernel, $\nu \approx 0.0575$

Forest (IF) and One-Class Support Vector Machine (OCSVM). Crucially, both detectors are trained exclusively on the embeddings corresponding to normal instances (4,202 samples) from the development set, making the approach unsupervised. We tuned Isolation Forest (IF) and One-Class SVM (OCSVM) with Optuna, training on development-set normals only and selecting configurations that maximized anomaly F1 on the development split. The selected OCSVM used a linear kernel with $\nu \approx 0.0575$; IF used n_estimators = 400, max_samples ≈ 0.799, contamination ≈ 0.0107, max_features ≈ 0.681, and no bootstrap. The final models were refit on all normal embeddings from the development set. Table **2** summarizes the search succinctly.

A key difference between the two detectors lies in how they set the decision boundary for anomalies:

- **Isolation Forest (IF)**: After training, we compute decision_function scores on the development set (normal + anomaly) and choose the threshold that maximizes anomaly F1; this fixed threshold is then applied to the test set.
- One-Class SVM (OCSVM): Uses its intrinsic boundary (controlled by ν) and classifies via predict (+1 normal,-1 anomaly).

Finally, the trained Isolation Forest model and its determined optimal threshold, along with the trained One-Class SVM model, are saved (.joblib files) for subsequent evaluation of the embeddings of the unseen holdout test set.

3. EVALUATION METHODOLOGY

We evaluate the final performance of our trained anomaly detection models on the unseen holdout test set. This set, comprising 818 instances (742 normal, 76 anomaly), represents approximately 15% of the total dataset. We preserved its original class distribution via stratified sampling and ensured it was not used during any phase of model training or hyperparameter tuning.

For evaluation, we use the precomputed 768-dimensional embeddings from the holdout test set instances as input. We load the saved, optimally tuned Isolation Forest (IF) and One Class SVM (OCSVM) models.

- For Isolation Forest, anomaly scores are obtained using the model's decision_function. An instance is classified as anomalous if its score is below the optimal threshold determined previously during the post-hoc tuning step on the development set (threshold value ≈0.0000).
- For One-Class SVM, classification is performed directly using the model's prediction method. The output (-1 for anomaly, +1 for normal) is mapped to binary labels (1 for anomaly, 0 for normal).

We assess model performance using standard metrics suitable for potentially imbalanced anomaly detection tasks. Our primary focus is on Precision, Recall, and F1-Score for the anomaly class (label=1), defined as:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 - Score = \frac{2 \times precision \times recall}{precision + recall}$$

Where TP, FP, and FN are True Positives, False Positives, and False Negatives for the anomaly class.

Additionally, we compute the Area Under the Receiver Operating Characteristic Curve (ROC-AUC), using the negated anomaly scores from each model's decision_function as input for ranking instances. A full Classification Report (including accuracy and metrics for both classes) and a Confusion Matrix (True Negatives, False Positives, False Negatives, True Positives) are also generated. The evaluation metrics are calculated using the scikit-learn library.

To further validate the robustness of our models' performance on the holdout set, we conducted a 10-fold stratified cross-validation. For this procedure, the holdout set was partitioned into 10 folds, preserving the

class distribution in each. We then evaluated our single, optimally tuned IF and OCSVM models on each fold sequentially. This process allows us to assess the stability and consistency of the models' performance across different subsets of the unseen data, ensuring that our reported metrics are not the result of a favorable, single partition.

4. RESULTS AND DISCUSSION

This section presents the performance of our unsupervised anomaly detection framework on the holdout test set. We first detail the quantitative results achieved by our RoBERTa embedding-based approach with Isolation Forest and One Class SVM. We then discuss the effectiveness of contextual embeddings, compare our work with related studies, analyze the performance of the two anomaly detectors, and acknowledge the limitations of our study.

A. Quantitative Results

We assessed the performance of our fine-tuned RoBERTa embedding approach, combined with optimized Isolation Forest (IF) and One-Class SVM (OCSVM) anomaly detection classifiers, on the unseen holdout test set (818 instances: 742 normal, 76 anomaly). Tables **3** and **4** present detailed classification reports, including Precision, Recall, F1-Score, and Support for both normal (0) and anomaly (1) classes for IF and OCSVM, respectively. For a compact, side-by-side summary that includes baselines, see Table **5**.

Both models achieve excellent overall accuracy (> 0.99). For the crucial anomaly class, the Isolation Forest model yields an F1-score of 0.9744, driven by a perfect recall of 1.0000. The One-Class SVM also performs very well, achieving an F1-score of 0.9467. The ROC-AUC scores further confirm strong discrimination ability, with Isolation Forest achieving 0.9952 and One-Class SVM achieving 0.9569. The separability of instances is illustrated in Figure **5**. These results demonstrate that our contextual RoBERTa embeddings enable effective discrimination even when obvious indicator events are removed.

To confirm the stability of these results, we performed a 10-fold cross-validation on the holdout set. The Isolation Forest model demonstrated exceptional consistency, achieving an average Anomaly F1-Score of 0.9749 ± 0.0308 and an average ROC-AUC of 0.9950 ± 0.0062. The One-Class SVM was similarly stable, with an average Anomaly F1-Score of 0.9442 ± 0.0757 and an average ROC-AUC of 0.9571 ± 0.0665. The low standard deviation across folds for both models validates that their high performance is robust and not an artifact of a specific data partition.

B. Discussion: Effectiveness of Contextual Embeddings

The quantitative results presented in Section IV-A strongly validate the use of fine-tuned RoBERTa embeddings for representing OpenStack log event sequences in anomaly detection. This performance represents a substantial improvement over traditional log analysis techniques like PCA, which often struggle

Table 3: Classification Report for RoBERTa + Isolation Forest on Hold-Out Test Set

Class	Precision	Recall	F1-Score	Support
Normal (0)	0.9986	0.9946	0.9973	742
Anomaly (1)	0.9500	1.0000	0.9744	76
Accuracy			0.9951	818
Macro Avg	0.9750	0.9973	0.9861	818
Weighted Avg	0.9953	0.9951	0.9952	818

Table 4: Classification Report for RoBERTa + One-Class SVM on Hold-Out Test Set

Class	Precision	Recall	F1-Score	Support
Normal (0)	0.9933	0.9959	0.9946	742
Anomaly (1)	0.9595	0.9342	0.9467	76
Accuracy			0.9902	818
Macro Avg	0.9764	0.9651	0.9705	818
Weighted Avg	0.9901	0.9902	0.9901	818

with system logs and yield lower F1-scores, whereas our approach delivered excellent F1-scores of 0.9744 (Isolation Forest) and a robust 0.9467 (One-Class SVM) on the unseen holdout data.

The key advantage of our approach stems from the ability of the fine-tuned transformer embeddings to capture rich semantic and sequential context. Unlike methods that treat logs as isolated events or rely solely on statistical distributions, the RoBERTa model, adapted via Masked Language Modeling on sequences of EventIDs, learns the typical patterns and relationships between events within an instance's lifecycle. These 768-dimensional embeddings encode this contextual understanding, allowing the downstream anomaly detectors (IF and OCSVM) to effectively model the manifold of normal behavior. Anomalies, which often manifest as unexpected events, incorrect event ordering, or missing events within a sequence, result in embeddings that lie further from this normal manifold, enabling their successful identification.

Furthermore, the domain adaptation achieved by fine-tuning RoBERTa specifically on the OpenStack EventID vocabulary likely contributes to this success, tailoring the representations to the specific operational language of this cloud platform. The high precision and recall observed for both detectors (Tables **3** and **4**) suggest that these contextual embeddings provide a robust feature space that minimizes the false alarms and limited semantic understanding often associated with conventional methods. Therefore, the results strongly support the use of fine-tuned transformer embeddings as a powerful foundation for high-fidelity, unsupervised anomaly detection in complex system logs, such as those from OpenStack.

C. Comparison with Baseline Models

To rigorously evaluate the effectiveness and robustness of our contextual RoBERTa embeddings, we established a strong baseline using a traditional LSTM Autoencoder, a common method for sequential anomaly detection. We trained an LSTM Autoencoder on the normal log sequences from the development set to learn to reconstruct them. The encoder part of this trained model was then used to generate 128-dimensional embeddings for all instances in both the development and holdout sets. The same Isolation Forest and One-Class SVM classifiers were then tuned and tested on these LSTM-based embeddings.

The performance of these baselines on the filtered holdout test set reveals the significant advantage of our transformer-based approach (Table **5**).

The LSTM-based models struggle on this dataset, with the Isolation Forest baseline achieving an F1-score of only 0.7956 and the OCSVM baseline collapsing to 0.5139. This indicates that while LSTMs can learn basic sequential patterns, they are not sufficient to capture the deep contextual information required when simplistic "telltale" event indicators are absent. In contrast, our RoBERTa-based models maintain state-of-the art performance, demonstrating their ability to learn from the entire sequence context. This directly validates that the contextual understanding provided by the fine-tuned transformer is superior and more resilient than the sequential pattern recognition of LSTMs for this task.

D. Comparison with Related Work

Our framework's strong performance on the OpenStack holdout set (F1 > 0.94, Tables **3**, **4**) positions it favorably compared to related approaches in log-based anomaly detection, as summarized in Figure **6**.

LogBERT [5], which pioneered using BERT for this task via Masked Log Key Prediction and other self-supervised objectives, reported F1-scores of 0.823 on HDFS, 0.908 on BGL, and 0.966 on the Thunderbird dataset using their combined training tasks. While direct comparison is limited due to dataset differences, our achieved F1-scores (0.9744 for IF, 0.9467 for OCSVM) are highly competitive and even exceed LogBERT's performance on some standard

Table 5: Performance Comparison with LSTM Autoencoder Baselines on the Filtered Holdout Set.

Model	Anomaly F1-Score	ROC-AUC
RoBERTa + IF	0.9744	0.9952
RoBERTa + OCSVM	0.9467	0.9569
LSTM + IF (Baseline)	0.7956	0.9696
LSTM + OCSVM (Baseline)	0.5139	0.7514

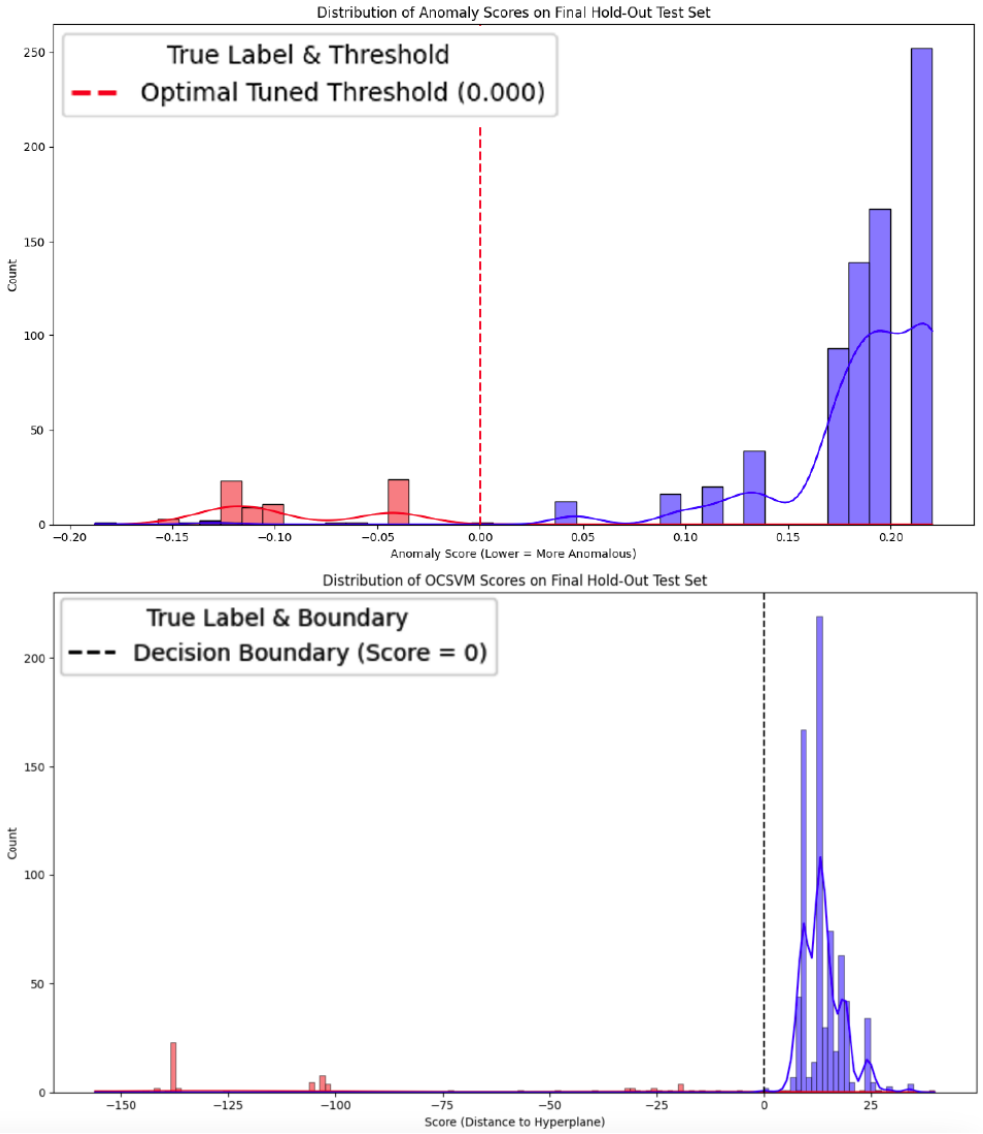

Figure 5: Anomaly-score distributions for normal (blue) and anomalous (orange) instances on the hold-out test set. Subfigure (**a**) shows scores from the Isolation Forest model, and (**b**) from the One-Class SVM. Lower scores correspond to higher anomaly likelihood, illustrating clear separability between normal and abnormal log sequences.

benchmarks. This suggests that our approach of fine-tuning RoBERTa with PEFT (LoRA) specifically on sequences of abstract EventIds is highly effective for capturing anomalous patterns within the structured sequences derived from OpenStack logs.

When comparing our method to statistical techniques applied specifically to OpenStack logs, Kalaki *et al*. [4] utilized an improved Robust PCA (RPCA) approach on a dataset they generated. They reported an F1-score of 0.93 for their PRPCACS method (see Figure **4** in [4]). Although this represents a strong result for an RPCA-based technique, our framework achieved even higher F1-scores on our dataset. This difference further suggests that modeling sequential and contextual information via transformer embeddings offers a more powerful approach for discriminating anomalies in OpenStack event sequences than statistical matrix decomposition methods like RPCA. Despite their improvements, such statistical methods may not fully capture the nuanced, context-dependent patterns that our approach detects.

Our adoption of PEFT (LoRA) aligns with recent research focusing on efficient log anomaly detection, such as the work by Lim *et al*. [6], who explored various PEFT techniques. Our results confirm that applying LoRA to a RoBERTa-base model, specifically adapted via MLM fine-tuning on EventID sequences, effectively achieves state-of-the-art performance for this OpenStack task.

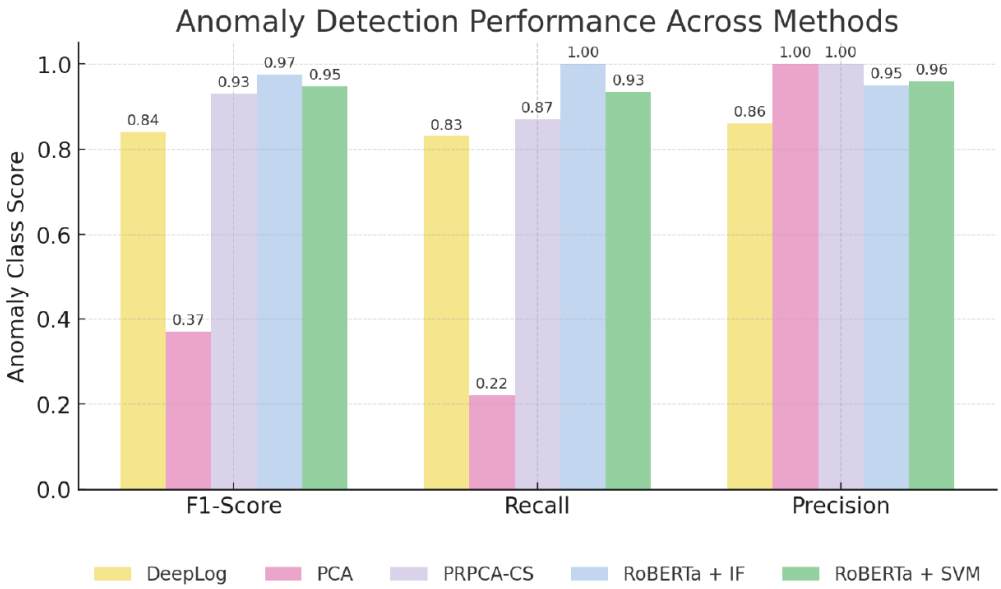

Figure 6: Comparison of anomaly F1-scores between the proposed RoBERTa-based models and related methods. The bar chart compares RoBERTa + Isolation Forest and RoBERTa + One-Class SVM against LSTM Autoencoder baselines and prior studies such as LogBERT [5]. The results highlight the superior performance of the fine-tuned transformer embeddings on OpenStack logs.

The success of our method inherently depends on the quality of the upstream log parsing provided by the Drain algorithm. As well-documented [1, 2], accurate and effective parsing is a critical prerequisite, and its challenges underscore this dependency. By focusing on OpenStack data, our work contributes to the evaluation of anomaly detection techniques on logs from modern, complex cloud systems, thereby addressing potential limitations of relying solely on older benchmark datasets [3].

In conclusion, our approach leverages domain-adapted transformer embeddings via PEFT on parsed event sequences, demonstrating performance that is highly competitive with or exceeds state-of-the-art methods on various benchmarks, and appears significantly more effective than statistical techniques like RPCA when applied to OpenStack log sequences.

E. Analysis of Anomaly Detectors (IF vs. OCSVM)

Both Isolation Forest (IF) and One-Class SVM (OCSVM), when applied to the fine-tuned RoBERTa embeddings, proved highly effective, though they exhibited slightly different performance profiles (Tables **3**, **4**).

Isolation Forest achieved a marginally superior F1-score (0.9744 vs. 0.9467) and ROC-AUC (0.9952 vs. 0.9569), driven by a perfect recall of 1.0000. This suggests its ensemble-based isolation mechanism was slightly better at capturing the overall distribution variance between normal and anomalous embeddings, resulting in zero false negatives.

Conversely, One-Class SVM achieved a slightly higher precision (0.9595 vs. 0.9500), resulting in fewer false positives (3 vs. 4). The linear kernel identified during hyperparameter optimization learned a tight decision boundary around the dense cluster of normal embeddings, excelling at minimizing false alarms but consequently misclassifying a few more anomalies.

The difference in thresholding strategy (post-hoc optimal F1 for IF vs. intrinsic OCSVM boundary) also contributes to this performance trade-off. The ROC-AUC, being threshold independent, confirms that IF has a stronger overall ranking ability on this dataset.

Ultimately, the choice between them could depend on operational tolerance for false positives versus false negatives. However, the strong performance of both underscores the quality of the RoBERTa embeddings in creating a highly separable feature space for unsupervised anomaly detection.

F. Implications for Cybersecurity Operations and Forensics

The high-performance metrics achieved by our framework have direct and practical applications in addressing critical challenges within real-world cybersecurity operations. The modern Security

Operations Center (SOC) is widely reported to be in a state of crisis, suffering from overwhelming alert volumes and high false-positive rates that lead to analyst burnout and missed threats [7], [8]. Our approach, particularly the Isolation Forest model achieving an Anomaly F1-score of 0.9744 with high precision, can serve as a powerful automated triage engine. By autonomously filtering many benign log sequences, it would dramatically reduce the low fidelity alert queue, directly combating the alert fatigue epidemic. This automation frees human analysts from repetitive, low-level tasks, enabling a strategic shift from a reactive "firefighting" posture to proactive threat hunting, where expert time is dedicated to investigating the high-confidence anomalies surfaced by the model [9].

Furthermore, in the context of post-breach Digital Forensics and Incident Response (DFIR), our method functions as a forensic accelerator. Traditional forensic analysis requires investigators to manually sift through massive volumes of historical log data to reconstruct an attack timeline, a process that is slow and prone to missing subtle indicators [10]. The superior contextual understanding of our RoBERTa-based model, which proved far more resilient than an LSTM, is particularly well-suited for uncovering the "low and slow" attack patterns characteristic of advanced persistent threats. These sophisticated attacks often consist of a sequence of seemingly benign events spread over long periods, which evade signature-based tools and simple sequential models [11]. By presenting investigators with a pre-computed and correlated set of all anomalous sequences, our framework provides a data-driven starting point, enabling faster root cause analysis and a more complete and accurate reconstruction of the entire attack chain.

G. Ethical and Legal Considerations

As we note in our conclusion, a key interdisciplinary potential for this framework lies in its application to digital forensics and "legally defensible forensic reporting." However, a significant ethical and legal barrier for any "black box" model, including our fine-tuned RoBERTa, is its inherent opacity. For an automated alert to be truly "legally defensible," the logic behind the judgment must be transparent, auditable, and explainable to a human investigator or a court.

This creates a critical need for Explainable AI (XAI) to bridge this gap. A crucial area for future work is integrating methods to interpret these complex models. Techniques such as post-hoc rule extraction, which can generate human-readable rules for unsupervised models [12], or feature attribution methods that evaluate the contribution of each log event to an alert [13], are essential. Applying these XAI techniques would transform a statistical anomaly score into an actionable, defensible finding, thereby satisfying the legal and ethical demands for transparency.

H. Limitations

While our proposed framework demonstrates promising results, we acknowledge several limitations:

- Dataset Specificity: We evaluated our approach on OpenStack Nova logs featuring specific VM lifecycle events and three known failure types. Further testing is required to ascertain generalizability to other OpenStack services, different cloud platforms, or novel anomaly types.
- Parser Dependency: The framework's effectiveness hinges on the quality and consistency of the upstream Drain parser [1], [2]. Inaccuracies or inconsistencies in log parsing could negatively impact downstream performance.
- Static Evaluation: We assessed performance on a static holdout set. Investigating real-time, streaming performance and the framework's adaptability to evolving log patterns remains future work.
- Hyperparameter Sensitivity: The framework's performance relies on careful tuning of both the RoBERTa fine-tuning process and the downstream anomaly detectors (IF and OCSVM). Optimal parameters might vary across different log sources or distributions of anomalies.
- Model Transparency and Robustness: This study does not address the "black box" nature of the model, which is a barrier to operational trust and forensic defensibility. Future work must integrate Explainable AI (XAI) methods, such as feature attribution [13], to provide interpretability. Furthermore, the model's resilience against a malicious adversary was not evaluated. The system must also be tested against adversarial attacks, such as data poisoning, which are a unique threat to ML-based security systems [14].

5. CONCLUSION

In this paper, we presented and validated an effective unsupervised anomaly detection framework for OpenStack logs, leveraging fine-tuned RoBERTa-base model embeddings. Our methodology, which involves parsing logs into event sequences, adapting RoBERTa via Masked Language Modeling (MLM) and Parameter-Efficient Fine-Tuning (PEFT) with LoRA, and then applying these contextual embeddings to unsupervised classifiers like Isolation Forest and One-Class SVM, achieves high detection accuracy.

On a holdout test set, our framework demonstrated excellent performance: Isolation Forest yielded an Anomaly F1-score of 0.9744 and a ROC-AUC of 0.9952, while One-Class SVM achieved an F1-score of 0.9467 and a ROC-AUC of 0.9569. These results confirm our approach's ability to effectively distinguish normal operations from failures within OpenStack logs. Notably, these F1-scores substantially surpass those of traditional methods like PCA (which can score as low as 0.37 on some OpenStack datasets [4]) and are highly competitive with state-of-the-art transformer-based approaches such as LogBERT [5] on standard benchmarks.

The central contribution of this work is the demonstration that fine-tuned, domain-adapted contextual embeddings provide a highly effective and, crucially, robust representation for anomaly detection. This was validated by our framework's consistent high performance on a filtered dataset where simpler sequential models, such as an LSTM Autoencoder, failed significantly. This finding confirms that our approach learns true contextual patterns rather than relying on simplistic event indicators, offering a promising direction for developing more resilient and semantically aware monitoring solutions for complex infrastructures.

Future research will focus on evaluating our framework across a broader range of OpenStack services and anomaly types, exploring alternative transformer architectures and PEFT methods, and investigating the impact of different parsing strategies. Beyond technical improvements, our framework offers significant interdisciplinary potential, particularly at the intersection of cybersecurity, digital forensics, and legal compliance. The deterministic and data-driven nature of transformer-based anomaly scores could serve as machine-generated auditable evidence in legal proceedings, strengthening incident response documentation, and regulatory compliance efforts. Furthermore, integration with forensic data frameworks would enable seamless incorporation of our anomaly detection outputs into comprehensive incident investigation workflows, bridging the gap between automated detection and legally defensible forensic reporting.

ACKNOWLEDGEMENT

The authors acknowledge the support from the University of Southern Mississippi. The code for this research is publicly available at https://github.com/aniJani/roBERTaAnomaly. The dataset is available at https://github.com/ParisaKalaki/openstack-logs [4].

REFERENCES

[1] Zhu J, He S, Liu J, He P, Xie Q, Zheng Z, *et al*. Tools and benchmarks for automated log parsing. In: Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice. Montreal, QC, Canada. IEEE 2019; pp. 121-30. https://doi.org/10.1109/ICSE-SEIP.2019.00021

[2] He P, Zhu J, He S, Li J, Lyu MR. An evaluation study on log parsing and its use in log mining. In: Proceedings of the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. IEEE 2016; pp. 654-61. https://doi.org/10.1109/DSN.2016.66

[3] Landauer M, Skopik F, Wurzenberger M. A critical review of common log data sets used for evaluation of sequence-based anomaly detection techniques. Proceedings of the ACM on Software Engineering 2024; 1(FSE): 1354-75. https://doi.org/10.1145/3660768

[4] Kalaki PS, Shameli-Sendi A, Abbasi B. Anomaly detection on OpenStack logs based on an improved robust principal component analysis model and its projection onto column space. Software: Practice and Experience 2022; 53: 665-81. https://doi.org/10.1002/spe.3164

[5] Guo H, Yuan S, Wu X. LogBERT: log anomaly detection via BERT. arXiv. 2021 [cited 2025 Nov 2]. Available from: https://arxiv.org/abs/2103.04475

[6] Lim YF, Zhu J, Pang G. Adapting large language models for parameter-efficient log anomaly detection. arXiv. 2025 [cited 2025 Nov 2]. Available from: https://arxiv.org/abs/2503.08045

[7] Alahmadi BA, Axon L, Martinovic I. 99% false positives: a qualitative study of SOC analysts' perspectives on security alarms. In: Proceedings of the 31st USENIX Security Symposium; 2022 Aug 10-12; Boston, MA. Berkeley (CA): USENIX Association; 2022. p. 2783-800. Available from: https://www.usenix.org/conference/usenixsecurity22/presentation/alahmadi

[8] Nepal S, Hernandez J, Lewis R, Chaudhry A, Houck B, Knudsen E, *et al*. Burnout in cybersecurity incident responders: exploring the factors that light the fire. Proceedings of the ACM on Human-Computer Interaction. 2024; 8(CSCW1): 27: 1-35. https://doi.org/10.1145/3637304

[9] Chen W, Zhang J. Elevating security operations: the role of AI-driven automation in enhancing SOC efficiency and efficacy. Journal of Artificial Intelligence and Machine Learning in Management 2024; 8(2): 1-13. Available from: https://journals.sagescience.org/index.php/jamm/article/view/128

[10] Oliner AJ, Ganapathi A, Xu W. Advances and challenges in log analysis. Communications of the ACM 2012; 55(2): 55-61. https://doi.org/10.1145/2076450.2076466

[11] Buchta R, Gkoktsis G, Heine F, Kleiner C. Advanced persistent threat attack detection systems: a review of approaches, challenges, and trends. Digital Threats: Research and Practice 2024; 5(4): Art. 39:1-37. https://doi.org/10.1145/3696014

[12] Li R, Li Q, Zhang Y, Zhao D, Jiang Y, Yang Y. Interpreting unsupervised anomaly detection in security via rule extraction. In: Oh A, Naumann T, Globerson A, Saenko K, Hardt M, Levine S, editors. Advances in Neural Information Processing Systems 36. Curran Associates 2023; pp. 62224-43. Available from: https://proceedings.neurips.cc/paper_files/paper/2023/file/c43b987f23fd5ea840df2b2be426315c-Paper-Conference.pdf

[13] Dahal A, Bajgai P, Rahimi N. Analysis of zero day attack detection using MLP and XAI. In: Daimi K, Arabnia HR, Deligiannidis L, editors. Security and Management and Wireless Networks. CSCE 2024. Communications in Computer and Information Science, vol. 2254. Cham: Springer 2025; pp. 1-10. https://doi.org/10.1007/978-3-031-86637-1_5

[14] Rahimi N, Maynor J, Gupta B. Adversarial machine learning: difficulties in applying machine learning in existing cybersecurity systems. In: Lee G, Jin Y, editors. Proceedings of the 35th International Conference on Computers and Their Applications, vol. 69. EasyChair 2020; pp. 40-7. Available from: https://easychair.org/publications/paper/XwRv

Received on 13-09-2025 Accepted on 28-10-2025 Published on 04-11-2025

https://doi.org/10.65879/3070-5789.2025.01.02