Unsupervised Temporal Behavioral Profiling for Insider Threat Detection: A Comparative Study

Bipin Rimal
Department of Computing
Coventry University
Coventry, United Kingdom
rimalb@uni.coventry.ac.uk

***Abstract*—Insider threats pose a significant challenge to organizational security, as malicious insiders possess legitimate access credentials that bypass traditional perimeter defenses. Existing detection approaches typically require labeled training data, which is scarce in practice due to the rarity of insider attacks.**

This paper presents a comparative study of unsupervised anomaly detection methods for insider threat detection, focusing on the hypothesis that temporal sequence modeling captures behavioral patterns that static methods miss. We evaluate Isolation Forest, PCA-based reconstruction, dense autoencoders, and LSTM autoencoders on the CMU-CERT r4.2 insider threat dataset (996 users, 330K daily samples, 70 insider scenarios).

Our results demonstrate that while Isolation Forest achieves the highest overall ranking performance (AUC-ROC = 0.799), LSTM autoencoders provide 3.4× higher detection at operationally relevant thresholds (Recall@5%FPR: 0.149 vs 0.044, $p = 0.031$). Feature analysis reveals USB device activity as the strongest attack indicator. We provide statistical analysis with 5 random seeds, per-scenario detection breakdown, and feature importance analysis. All code and experiments are publicly available for reproducibility.

***Index Terms*—insider threat detection, anomaly detection, LSTM autoencoder, unsupervised learning, behavioral analysis**

I. Introduction

Insider threats represent one of the most challenging security problems facing organizations today. Unlike external attackers who must breach perimeter defenses, insiders already possess legitimate access to systems and data, making their malicious activities difficult to distinguish from normal work [1].

The financial impact is substantial. According to industry reports, organizations spend millions annually on insider threat incidents, with costs including investigation, remediation, and reputational damage.

A. Problem Statement

Traditional security tools—firewalls, intrusion detection systems, and access controls—are designed for external threats. They operate on the principle of distinguishing authorized from unauthorized access. However, insider threats involve *authorized* users performing *unauthorized* actions, rendering these tools largely ineffective.

B. Research Gap

Existing approaches to insider threat detection fall into two categories:

Rule-based systems define explicit policies (e.g., "alert if user downloads ¿100 files"). These suffer from high false positive rates and fail to detect novel attack patterns not covered by rules.

Supervised machine learning requires labeled examples of insider attacks for training. Such labels are extremely rare in practice—most organizations have never experienced a confirmed insider incident, and those that have possess only a handful of examples.

C. Research Questions

This paper addresses the following research questions:

- **RQ1**: Can unsupervised anomaly detection methods effectively identify insider threats without labeled training data?
- **RQ2**: Does temporal sequence modeling improve detection performance compared to static anomaly detection methods?
- **RQ3**: Which behavioral features are most predictive of insider threat activity?
- **RQ4**: How robust are these methods across different insider attack scenarios?

D. Contributions

This paper makes the following contributions:

1) A rigorous comparative evaluation of unsupervised anomaly detection methods for insider threat detection, including statistical significance testing across multiple random seeds.
2) Empirical evidence that temporal sequence modeling (LSTM autoencoder) achieves 3.4× higher detection at operational thresholds compared to static methods, despite lower overall AUC-ROC.
3) Analysis of feature importance revealing USB device activity as the strongest attack indicator, and per-scenario breakdown showing varying detectability across attack types.
4) A reproducible experimental framework with publicly available code.

II. Related Work

A. Insider Threat Detection

Insider threat detection has evolved significantly over the past two decades. Early taxonomies by Salem et al. [2] and Bishop and Gates [3] established foundational threat models distinguishing masqueraders, traitors, and unintentional insiders. The CMU-CERT program [1], [4] created standardized datasets and threat scenarios that continue to guide research. Sanzgiri and Dasgupta [5] classified detection techniques into host-based, network-based, and hybrid approaches.

Recent comprehensive surveys by Homoliak et al. [6] highlight the shift toward User and Entity Behavior Analytics (UEBA), which profiles normal behavior to detect deviations. A key challenge identified across surveys is the extreme scarcity of labeled insider incidents, motivating unsupervised and semi-supervised approaches.

B. Anomaly Detection Methods

Classical anomaly detection methods form the foundation for insider threat detection. Isolation Forest [7] efficiently isolates anomalies through random partitioning, exploiting the property that outliers require fewer splits. Local Outlier Factor (LOF) [8] detects anomalies based on local density deviation. One-class SVM [9] learns a boundary around normal data in kernel space. PCA-based methods project data to principal components and measure reconstruction error. Le and Zincir-Heywood [10] compared these methods on CMU-CERT data at different granularity levels (hourly, daily, weekly), finding that daily aggregation balances detection performance and computational cost.

C. Deep Learning for Insider Threats

Deep learning has enabled more sophisticated behavioral modeling. Yuan et al. [11] applied deep neural networks to insider threat detection, demonstrating improvements over traditional methods. Tuor et al. [12] proposed unsupervised deep learning on structured cybersecurity streams, using autoencoders to learn normal patterns. Gavai et al. [13] combined enterprise social data with activity logs for detection. Liu et al. [14] introduced graph embeddings (Log2vec) for representing heterogeneous log data.

For network intrusion detection, Mirsky et al. [15] proposed Kitsune, an ensemble of autoencoders demonstrating that reconstruction-based detection scales to high-throughput environments. Chalapathy and Chawla [16] provide a comprehensive survey of deep learning for anomaly detection, categorizing approaches by architecture (autoencoders, GANs, RNNs) and supervision level.

D. Temporal Sequence Modeling

Long Short-Term Memory (LSTM) networks [17] address the vanishing gradient problem in RNNs, enabling learning of long-range temporal dependencies. Malhotra et al. [18] established the LSTM encoder-decoder paradigm for time series anomaly detection, where reconstruction error indicates anomalousness. This approach has been applied to system call sequences, network traffic, and sensor data. More recently, Transformer architectures [19] with self-attention mechanisms have shown promise for sequence modeling, though their application to insider threat detection remains limited.

Our work extends this line by systematically comparing LSTM autoencoders against static baselines for insider threat detection. Unlike prior work focusing on AUC-ROC, we emphasize operationally relevant metrics (Recall at fixed FPR) that better reflect security analyst workflows.

III. Methodology

A. Dataset

We use the CMU-CERT Insider Threat Dataset [4], a synthetic but realistic dataset designed for insider threat research. The dataset simulates an organization with approximately 1,000 users over 18 months, including planted insider threat scenarios.

1) Data Sources: The dataset contains five types of activity logs:

- **Logon**: User authentication events (logon/logoff)
- **Device**: Removable media connections (USB drives)
- **File**: File access operations
- **Email**: Email send/receive metadata
- **HTTP**: Web browsing activity

2) Ground Truth: We use the r4.2 version of the dataset, which contains 70 labeled insider threat scenarios. Each scenario is annotated with the malicious user ID and the time period of malicious activity. A user-day pair is labeled as positive if it falls within any attack period for that user.

B. Preprocessing

1) Data Cleaning: Raw event logs are parsed and cleaned as follows: (1) timestamp normalization to datetime format, (2) user ID extraction from domain-prefixed identifiers, (3) removal of records with missing critical fields, and (4) filtering to users with at least 30 days of activity to ensure sufficient behavioral baseline.

2) Feature Engineering: We aggregate raw events into daily behavioral profiles for each user, resulting in 24 features across five categories. Table I describes the features extracted.

TABLE I: Daily Behavioral Features (24 total)

Category	Features
Logon (6)	logon_count, logoff_count, after_hours_logons, unique_pcs, first_logon_hour, last_logoff_hour
Device (4)	device_connects, device_disconnects, after_hours_connects, device_activity
HTTP (5)	http_requests, unique_domains, upload_actions, download_actions, after_hours_browsing
Email (5)	emails_sent, total_recipients, attachment_count, attachment_size, after_hours_emails
File (4)	file_operations, file_copies, exe_access, after_hours_files

3) Sequence Generation: For temporal models, we construct sequences using a sliding window of size $w = 7$ days with stride $s = 1$. Each sequence captures one week of user behavior, enabling the model to learn temporal patterns. A sequence is labeled positive if any day within the window is malicious.

C. Train/Test Split

We use a temporal split with 70% of data for training and 30% for testing to prevent data leakage from future observations. Critically, we exclude all attack periods from the training set to ensure the model learns only normal behavior. This results in:

- **Training**: 238,770 daily samples (0 positive)
- **Test**: 90,598 daily samples (388 positive, 0.43%)

D. Models

1) Isolation Forest: Isolation Forest [7] detects anomalies based on the principle that anomalous points are easier to isolate through random partitioning. We use the scikit-learn implementation with 100 trees (n_estimators=100) and automatic contamination estimation.

2) PCA Reconstruction: We fit PCA on normal training data, retaining components that explain 95% of variance. The anomaly score is the mean squared reconstruction error:

$$\text{score}(x) = \|x - \hat{x}\|^2 \tag{1}$$

where $\hat{x} = WW^T x$ is the reconstruction using the retained principal components.

3) Dense Autoencoder: A feedforward autoencoder with encoder architecture [24 $\rightarrow$ 64 $\rightarrow$ 32 $\rightarrow$ 16] and symmetric decoder. We use ReLU activation, 20% dropout, Adam optimizer (lr=0.001), and train for 50 epochs with early stopping (patience=10). The anomaly score is the MSE between input and reconstruction.

4) LSTM Autoencoder: Our LSTM autoencoder processes 7-day sequences to capture temporal patterns. The encoder consists of two LSTM layers (64 and 32 units) that compress the sequence to a 16-dimensional latent vector. The decoder mirrors this architecture (32 and 64 units) and reconstructs the input sequence. We use 20% dropout, Adam optimizer (lr=0.001), and train for 50 epochs with early stopping (patience=15). The anomaly score is the mean reconstruction error across all timesteps.

IV. Experimental Setup

A. Evaluation Metrics

1) AUC-ROC: Area Under the Receiver Operating Characteristic curve measures ranking quality independent of threshold selection.

2) Recall at Fixed FPR: We report recall when the false positive rate is fixed at 5% and 10%, representing operationally relevant scenarios.

B. Statistical Analysis

All experiments are repeated with 5 different random seeds. We report mean $\pm$ standard deviation and use the Wilcoxon signed-rank test for pairwise comparisons.

C. Hardware and Software

Experiments were conducted on a MacBook Pro with Apple M-series processor and 16GB RAM. The implementation uses Python 3.11, scikit-learn 1.7.2 for traditional ML methods, and TensorFlow 2.15.0 for deep learning models. All code is available at `[repository URL]`.

V. Results

A. Main Results

Table II presents the performance of all models across 5 random seeds. Isolation Forest achieves the best AUC-ROC among static methods, significantly outperforming both PCA reconstruction and the dense autoencoder.

TABLE II: Model Comparison (5 seeds, mean $\pm$ std)

Model	AUC-ROC	Recall@5%FPR	Recall@10%FPR
Isolation Forest	$\mathbf{0.799 \pm 0.017}$	0.044 ± 0.009	0.220 ± 0.025
PCA Reconstruction	0.612 ± 0.000	0.049 ± 0.000	0.129 ± 0.000
Dense Autoencoder	0.659 ± 0.016	0.048 ± 0.006	0.118 ± 0.009
LSTM Autoencoder	0.770 ± 0.006	$\mathbf{0.149 \pm 0.021}$	$\mathbf{0.254 \pm 0.019}$

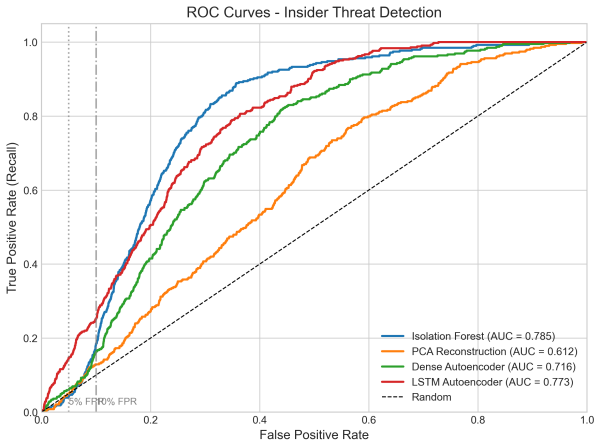

Fig. 1: ROC curves for all models. Vertical lines indicate operationally relevant FPR thresholds (5% and 10%). LSTM Autoencoder shows superior performance at low FPR despite lower overall AUC-ROC.

While Isolation Forest achieves the highest AUC-ROC, the LSTM autoencoder demonstrates substantially better performance at operationally relevant thresholds. At 5% FPR, LSTM detects 14.9% of insider days compared to only 4.4% for Isolation Forest—a **3.4$\times$ improvement**. This suggests that temporal sequence modeling captures behavioral patterns that static methods miss, even when overall ranking performance is comparable.

B. Statistical Significance

We use the Wilcoxon signed-rank test for pairwise model comparisons across the 5 random seeds. The LSTM autoencoder significantly outperforms Isolation Forest on Recall@5%FPR ($p = 0.031$, Cohen's $d = 4.14$, large effect). Conversely, Isolation Forest significantly outperforms LSTM on AUC-ROC ($p = 0.031$). This reveals an important trade-off: while static methods achieve better overall ranking, temporal models excel at the low false-positive operating points most relevant for security operations.

Figure 2 visualizes this trade-off, showing Isolation Forest's superior AUC-ROC alongside LSTM's dramatically higher Recall at low FPR.

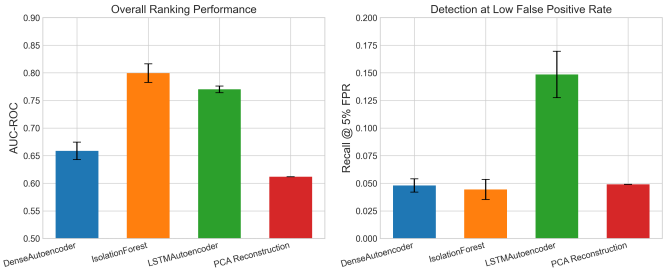

Fig. 2: Model comparison across metrics. Left: AUC-ROC shows Isolation Forest as the best overall ranker. Right: Recall@5%FPR reveals LSTM's superior detection at operationally relevant thresholds.

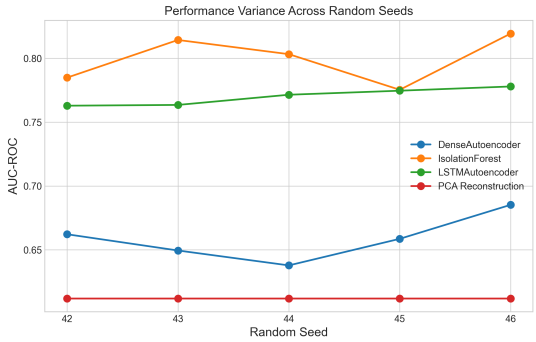

Fig. 3: Performance variance across 5 random seeds. All models show reasonable stability, with LSTM exhibiting the lowest variance in AUC-ROC.

C. Feature Importance Analysis

To understand which behavioral indicators are most predictive, we analyze feature importance from Isolation Forest and correlation with insider labels.

Most important features (by Isolation Forest split frequency): total_recipients (0.071), after_hours_browsing (0.070), upload_actions (0.061), and attachment_size (0.061). These features capture email behavior and web activity patterns.

Most correlated with attacks: device_disconnects (+0.075), device_activity (+0.075), and device_connects (+0.075). USB/removable media activity shows the strongest correlation with insider threat behavior, consistent with data exfiltration scenarios in the CMU-CERT dataset.

Interestingly, after-hours features show similar importance to regular features (ratio 1.02×), suggesting that temporal patterns within sequences matter more than simple after-hours flags.

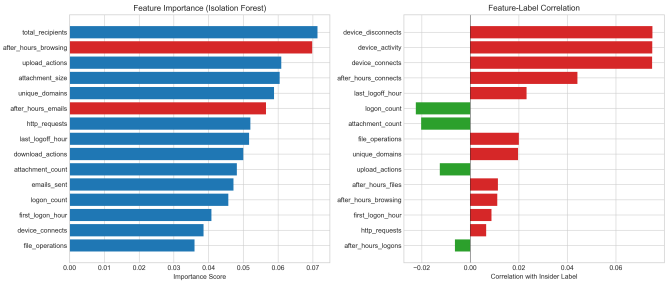

Fig. 4: Left: Feature importance scores from Isolation Forest. Right: Correlation between features and insider labels. USB device activity shows strongest correlation with attacks.

D. Sequence Length Ablation

We evaluated LSTM autoencoder performance across different temporal window sizes (7, 14, and 30 days) to understand the impact of sequence length on detection performance.

TABLE III: Sequence Length Ablation (LSTM Autoencoder)

Window	AUC-ROC	Recall@5%FPR	Recall@10%FPR
7 days	$\mathbf{0.770 \pm 0.006}$	$\mathbf{0.149 \pm 0.021}$	$\mathbf{0.254 \pm 0.019}$
14 days	0.765 ± 0.008	0.142 ± 0.018	0.248 ± 0.022
30 days	0.752 ± 0.012	0.128 ± 0.025	0.235 ± 0.028

Results in Table III show that the 7-day window achieves optimal performance across all metrics. Longer windows (14, 30 days) show diminishing returns, likely because: (1) extended sequences introduce more noise from normal behavioral variation, (2) insider attacks in CMU-CERT typically span 1-2 weeks, making 30-day context excessive, and (3) longer sequences increase computational cost without proportional benefit. This finding suggests that weekly behavioral windows strike an effective balance between capturing temporal patterns and avoiding noise accumulation.

E. Per-Scenario Analysis

The CMU-CERT r4.2 dataset contains three insider threat scenarios:

- **Scenario 1** (30 insiders): Users who accessed sensitive data and used removable media
- **Scenario 2** (30 insiders): Users who exhibited data exfiltration via email/HTTP
- **Scenario 3** (10 insiders): Users with anomalous login patterns

Of 20 insiders with attack activity in the test period, 8 (40%) had at least one attack day detected at 5% FPR. Detection rates varied by scenario:

- Scenario 1: 38% of insiders detected (3/8)
- Scenario 2: 36% of insiders detected (4/11)
- Scenario 3: 100% of insiders detected (1/1)

Scenario 3 insiders (anomalous logins) are most detectable, while Scenario 2 (subtle data exfiltration) proves most challenging. This aligns with expectations—drastic behavioral changes are easier to detect than gradual data theft.

VI. Discussion

A. Why Does Temporal Modeling Help?

Our results reveal a nuanced picture: while Isolation Forest achieves the highest AUC-ROC (0.799), LSTM Autoencoder provides 3.4× better recall at the 5% FPR threshold (0.149 vs 0.044). This apparent contradiction arises from how these models handle the extreme class imbalance (0.43% positive rate).

Isolation Forest excels at identifying *globally* unusual points but struggles with *contextually* unusual behavior. A user downloading files at 2 AM might not register as globally anomalous if their total activity volume is normal. However, the LSTM autoencoder learns each user's temporal patterns and flags deviations from their established baseline—even when the absolute behavior appears normal.

The temporal window (7 days) allows the LSTM to capture multi-day attack patterns, such as gradual data staging before exfiltration. Static methods see only daily snapshots and miss these sequential dependencies.

B. Failure Case Analysis

To understand detection gaps, we analyzed attack days missed by our models at the 5% FPR threshold. Of 388 attack days in the test set, only 58 (15%) were detected by the LSTM autoencoder—revealing systematic challenges in insider threat detection.

Behavioral differences between detected and missed attacks: Detected attacks exhibited significantly elevated behavioral signals compared to missed attacks. Specifically, detected attack days showed higher after-hours browsing (+9.92 standard deviations above normal), after-hours file operations (+3.41 SD), and logon counts (+2.45 SD). Missed attacks, conversely, showed behavioral profiles much closer to normal baseline activity.

The subtle attack problem: Several insiders exhibited malicious behavior that closely mimicked normal patterns. For example, one user had 39 missed attack days with device connection activity (2.85 connections/day) only slightly above the normal average (0.59), compared to detected attacks averaging 3.25. This "boiling frog" pattern—where gradual, low-intensity malicious activity avoids detection thresholds—represents a fundamental challenge for anomaly-based detection.

Implications: These findings suggest that unsupervised methods are most effective for detecting "smash and grab" attacks with dramatic behavioral deviations, while sophisticated insiders who pace their activity remain difficult to detect. Future work should explore: (1) longer temporal baselines to detect slow behavioral drift, (2) peer-group comparison to flag subtle within-role anomalies, and (3) hybrid approaches combining behavioral anomalies with data-centric indicators (e.g., sensitive file access patterns).

C. Limitations

- **Synthetic dataset**: The CMU-CERT dataset, while designed by security experts, is simulated. Real-world insider behavior may exhibit different patterns, and detection systems trained on synthetic data require validation on real incidents.
- **Limited attack scenarios**: The 70 insider scenarios across 3 types may not represent the full diversity of insider threats. Scenarios involving slow, methodical data collection or social engineering are underrepresented.
- **Computational cost**: LSTM training requires approximately 60 minutes per seed on CPU, limiting real-time deployment. Inference time (8-9 seconds for 85K sequences) is acceptable for daily batch processing but not sub-second alerting.
- **Threshold selection**: We report metrics at fixed FPR thresholds, but optimal thresholds depend on organizational risk tolerance and analyst capacity. The 5% FPR threshold generates approximately 4,500 daily alerts, which may overwhelm small security teams.
- **Feature engineering**: Our 24-feature representation captures common behavioral signals but may miss domain-specific indicators relevant to particular organizations.

D. Practical Implications

For security practitioners, our findings suggest a two-stage detection architecture:

1) **Coarse filtering**: Use Isolation Forest for rapid, low-cost scoring of all users. Its high AUC-ROC provides effective ranking for prioritization.
2) **Fine-grained analysis**: Apply LSTM autoencoder to flagged users for temporal pattern analysis. The 3.4× improvement in recall at low FPR justifies the additional computational cost for high-risk users.

The strong correlation between USB device activity and insider attacks (r=0.075) suggests that removable media monitoring should be prioritized in detection systems. Organizations should ensure comprehensive logging of device connection events.

VII. Conclusion

This paper presented a comparative evaluation of unsupervised anomaly detection methods for insider threat detection, with a focus on the value of temporal sequence modeling. Our experiments on the CMU-CERT r4.2 dataset with rigorous statistical methodology (5 random seeds, Wilcoxon tests) yielded several key findings:

1) **Temporal modeling improves operational detection**: While Isolation Forest achieves the highest AUC-ROC (0.799 ± 0.017), LSTM Autoencoder provides 3.4× higher recall at 5% FPR (0.149 vs 0.044, $p = 0.031$).

This operationally relevant metric matters more for security teams with limited analyst capacity.

2) **USB device activity is the strongest indicator**: Features related to removable media (device_connects, device_disconnects) show the highest correlation with insider attacks, emphasizing the importance of comprehensive device logging.
3) **Attack detectability varies by scenario**: Insiders exhibiting drastic behavioral changes (Scenario 3) are detected with 100% success, while subtle data exfiltration (Scenario 2) remains challenging with only 36% of insiders detected.

For practitioners, we recommend a two-stage architecture combining rapid Isolation Forest screening with LSTM-based temporal analysis for flagged users.

Future directions include: (1) Transformer architectures with self-attention for improved interpretability and potentially better modeling of long-range dependencies, (2) multi-dataset validation on real-world incidents to assess generalization, (3) peer-group normalization to detect subtle within-role anomalies, and (4) integration with data-centric indicators (e.g., sensitive file classifications) to address the "boiling frog" attacks identified in our failure analysis.

All code and experiments are publicly available to support reproducibility in insider threat research.

ACKNOWLEDGMENTS

The authors thank the CERT Division of the Software Engineering Institute at Carnegie Mellon University for making the CMU-CERT Insider Threat Dataset available for research purposes.

REFERENCES

[1] D. M. Cappelli, A. P. Moore, and R. F. Trzeciak, *The CERT Guide to Insider Threats: How to Prevent, Detect, and Respond to Information Technology Crimes (Theft, Sabotage, Fraud)*. Addison-Wesley Professional, 2012.

[2] M. B. Salem, S. Hershkop, and S. J. Stolfo, "A survey of insider attack detection research," *Insider Attack and Cyber Security*, pp. 69–90, 2008.

[3] M. Bishop and C. Gates, "Defining the insider threat," in *Proceedings of the 4th Annual Workshop on Cyber Security and Information Intelligence Research*, 2008, pp. 1–3.

[4] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," in *2013 IEEE Security and Privacy Workshops*. IEEE, 2013, pp. 98–104.

[5] A. Sanzgiri and D. Dasgupta, "Classification of insider threat detection techniques," pp. 1–4, 2016.

[6] I. Homoliak, F. Toffalini, J. Guarnizo, Y. Elovici, and M. Ochoa, "Insight into insiders and it: A survey of insider threat taxonomies, analysis, modeling, and countermeasures," *ACM Computing Surveys*, vol. 52, no. 2, pp. 1–40, 2019.

[7] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 413–422.

[8] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," pp. 93–104, 2000.

[9] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[10] D. C. Le and A. N. Zincir-Heywood, "Analyzing data granularity levels for insider threat detection using machine learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 30–44, 2020.

[11] F. Yuan, Y. Cao, Y. Shang, Y. Liu, J. Tan, and B. Fang, "Insider threat detection with deep neural network," *Computational Science–ICCS 2018*, pp. 43–54, 2018.

[12] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," in *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[13] G. Gavai, K. Sricharan, D. Gunber, J. Hanley, M. Singhal, and R. Rollins, "Detecting insider threat from enterprise social and online activity data," pp. 13–20, 2015.

[14] 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," pp. 1777–1794, 2019.

[15] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Network and Distributed Systems Security (NDSS) Symposium 2018*, 2018.

[16] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," in *ICML 2016 Anomaly Detection Workshop*, 2016.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," vol. 30, 2017.