The Metric That Misleads: Why Temporal Behavioral Models Outperform Static Methods for Insider Threat Detection at Operational Thresholds

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

***Abstract*—When detecting insider threats with unsupervised anomaly detection, the most widely used evaluation metric—AUC-ROC—gives a misleading answer. On the CMU-CERT r4.2 dataset, Isolation Forest achieves the highest AUC-ROC (0.807), suggesting it is the best model. But at the 5% false positive rate where security teams actually operate, an LSTM autoencoder catches 3.8× more attacks (Recall@5%FPR: 0.164 vs. 0.043, $p = 0.031$, one-sided Wilcoxon).**

This divergence reveals something important: AUC-ROC averages performance across all operating thresholds, but insider threat detection is an extreme imbalance problem (0.43% positive rate) where only low false-positive thresholds matter. Static models rank anomalies well overall but concentrate their detection power at thresholds no practitioner would use. Temporal models, by learning sequential behavioral patterns, focus their discriminative power exactly where it is needed.

We present a comparative evaluation of four unsupervised methods across 5 random seeds with statistical significance testing. Feature analysis identifies USB device activity as the strongest attack indicator. Failure analysis reveals that 86% of attack sequences evade detection—the "boiling frog" attacks where insiders pace their behavior to stay within normal bounds.

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

I. Introduction

If someone breaks into your office, the alarm goes off. But if someone with a key walks in and starts stealing, the alarm stays silent—the system was designed to detect unauthorized access, not authorized people doing unauthorized things. That is the insider threat problem.

Insiders—employees, contractors, partners—already possess legitimate credentials. Firewalls, intrusion detection systems, and access controls all operate on the boundary between authorized and unauthorized. When the threat *is* authorized, these defenses have nothing to detect [1].

The natural response is behavioral monitoring: learn what normal work looks like, and flag deviations. But this approach requires answering two hard questions.

First, how do you learn "normal" without examples of "abnormal"? Labeled insider threat data is exceptionally scarce. Most organizations have never confirmed an insider incident. Those that have possess a handful of cases—not enough to train a supervised classifier. Unsupervised anomaly detection avoids this requirement entirely: it models normal behavior from clean data and treats anything that deviates as suspicious [2].

Second, does time matter? A static model sees each day independently. On any given day, an insider's behavior might look unremarkable—a few file downloads, some emails, a USB connection. But *across a week*, a pattern emerges: staging data on Monday, packaging it on Wednesday, exfiltrating on Friday. Temporal models see sequences, not snapshots.

This paper tests whether temporal sequence modeling improves insider threat detection, and in doing so uncovers a problem with how we measure success. The standard metric (AUC-ROC) says no. The operationally relevant metric (Recall at low false positive rates) says yes—emphatically.

A. Research Questions

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

B. Contributions

1) Evidence that temporal modeling (LSTM autoencoder) achieves 3.8× higher detection at operational thresholds despite lower AUC-ROC, with statistical significance ($p = 0.031$, one-sided Wilcoxon, Cohen's $d = 4.1$).
2) Analysis of why AUC-ROC misleads for extreme-imbalance security problems, and why threshold-specific metrics better reflect deployment reality.
3) Feature importance analysis identifying USB device activity as the strongest indicator, and failure analysis characterizing the "boiling frog" attacks that evade detection.

4) A reproducible framework with 5-seed statistical analysis and publicly available code.

II. Related Work

Insider threat detection has evolved from rule-based systems ("alert if downloads exceed threshold") through supervised classification to behavioral analytics [3], [4]. The trajectory follows a consistent pattern: each approach hits the same bottleneck—labeled data.

Classical anomaly detection. Isolation Forest [5] isolates anomalies through random partitioning, exploiting the geometric property that outliers sit in sparse regions and require fewer splits to separate. PCA-based methods project data onto principal components and measure reconstruction error—points that deviate from the dominant variance structure reconstruct poorly. These methods operate on fixed-length feature vectors: one observation, one score.

Deep unsupervised methods. Autoencoders learn a compressed representation of normal data and use reconstruction error as an anomaly score—the logic being that a model trained only on normal patterns will reconstruct normal inputs well and anomalous inputs poorly. Tuor et al. [6] applied this to structured cybersecurity streams. Mirsky et al. [7] demonstrated that ensembles of autoencoders scale to high-throughput network environments.

Temporal modeling. Malhotra et al. [8] established the LSTM encoder-decoder for time series anomaly detection. The key insight: LSTMs learn temporal dependencies—patterns that span multiple timesteps—that pointwise methods cannot capture. Yuan et al. [9] and Gavai et al. [10] applied deep learning to insider threats specifically, showing improvements over traditional methods.

The evaluation gap. Le and Zincir-Heywood [11] compared methods on CMU-CERT at different granularities, finding daily aggregation optimal. However, nearly all prior work reports AUC-ROC as the primary metric. For problems with extreme class imbalance and constrained operational capacity, this can be misleading—a point we make empirically in Section IV.

Our work extends this line by systematically comparing static and temporal methods *at operationally relevant thresholds*, revealing a performance reversal that AUC-ROC obscures.

III. Methodology

A. Dataset: CMU-CERT r4.2

The CMU-CERT Insider Threat Dataset [12] simulates an organization of approximately 1,000 employees over 18 months. It contains five activity streams—logon events, USB device connections, file operations, email metadata, and web browsing—plus 70 planted insider threat scenarios across three attack types.

The simulation is synthetic but designed by security experts to reflect realistic behavioral patterns. We use version r4.2, which provides the most diverse threat scenarios.

B. From Raw Logs to Behavioral Profiles

Raw event logs are noisy and heterogeneous—millions of individual logon timestamps, HTTP requests, and file operations. To make them amenable to anomaly detection, we aggregate them into **daily behavioral profiles**: for each user on each day, we extract 24 features that capture the *shape* of their workday.

The features are chosen to answer a practical question: *if you were watching an employee for signs of data theft, what would you observe?*

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

Each feature category tracks a different channel through which data could leave an organization. Device features capture USB activity—the most direct exfiltration path. Email features capture who receives what. HTTP features track uploads and unusual browsing. The "after_hours" variants flag the same activities occurring outside normal working patterns, when oversight is minimal.

Features are normalized using z-scores fitted on the training set.

C. Snapshots vs. Stories: Static and Temporal Representations

For static models, each user-day is a 24-dimensional vector—a single photograph of one day's behavior.

For temporal models, we construct **sequences** using a sliding window of 7 days with stride 1. Each sequence captures a week of behavior as a (7×24) matrix—not a photograph, but a short film. A sequence is labeled positive if any constituent day falls within an attack period.

The 7-day window is chosen to capture weekly behavioral cycles while remaining short enough to isolate attack patterns. We validate this choice empirically in Section IV-C.

D. Training Protocol

We split data temporally: the first 70% of days for training, the remaining 30% for testing. Critically, **all attack periods are excluded from training**—the models learn only normal behavior.

This yields 238,770 training samples (0 positive) and 90,598 test samples (388 positive, 0.43% prevalence). The extreme imbalance mirrors real-world conditions, where insider attacks are exceptionally rare events against a background of routine activity.

E. Models

We evaluate four models, spanning a spectrum from simple and fast to complex and expressive. Each represents a different assumption about what makes behavior "anomalous."

Isolation Forest [5] asks: *how easy is this point to isolate?* It builds 100 random trees, each splitting data on random features at random thresholds. Anomalies—points in sparse regions of feature space—require fewer splits to separate. The anomaly score is the average path length across trees. No reconstruction, no learned representation—purely geometric.

PCA Reconstruction asks: *does this point fit the dominant patterns in the data?* We retain principal components explaining 95% of training variance and measure reconstruction error: $\text{score}(x) = \|x - WW^T x\|^2$. Points that deviate from the learned variance structure reconstruct poorly. Deterministic and parameter-free once fitted.

Dense Autoencoder asks: *can this point be compressed and reconstructed?* An encoder ($24 \to 64 \to 32 \to 16$) compresses each day's profile to a 16-dimensional bottleneck; a symmetric decoder reconstructs it. Trained on normal data only, the model learns to reconstruct normal patterns efficiently. Anomalous inputs—patterns never seen during training—reconstruct poorly. The anomaly score is the mean squared error between input and reconstruction.

LSTM Autoencoder asks: *can this sequence be compressed and reconstructed?* This is the critical distinction. The encoder processes a 7-day sequence through two LSTM layers (64 and 32 units) to produce a 16-dimensional latent vector. The decoder mirrors this architecture and reconstructs the full sequence. The model learns not just *what* normal days look like, but *how they follow each other*. An insider who behaves normally on each individual day but whose week-long pattern deviates—staging, packaging, exfiltrating—will produce high reconstruction error in the temporal model even when static models see nothing unusual.

All neural models use Adam optimizer (lr=0.001), 20% dropout, and early stopping (patience 10 for dense, 15 for LSTM).

F. Evaluation: Why the Metric Matters

AUC-ROC measures overall ranking quality: if you pick a random attack and a random normal point, how often does the model score the attack higher? This is a useful general property, but it averages across *all* possible operating thresholds—including thresholds where the false positive rate is 50%, 80%, or 99%.

No security team operates there. A team with 10 analysts investigating 50 alerts per day cannot tolerate a 50% false positive rate across 1,000 users. They need a low false-positive operating point.

Recall@k%FPR measures detection rate at a fixed false positive rate. At 5% FPR, approximately 50 of 1,000 users are flagged daily—a plausible analyst workload. The question becomes: of the actual insider days in that flagged set, how many are caught?

We report both metrics to expose the tension between them. All experiments use 5 random seeds (42–46) with Wilcoxon signed-rank tests for pairwise significance.

IV. Results

A. The Reversal

Table II shows the central finding. Read the columns left to right: AUC-ROC tells one story; Recall@5%FPR tells the opposite.

TABLE II: Model Comparison (5 seeds, mean ± std)

Model	AUC-ROC	Recall@5%	Recall@10%
Isolation Forest	$\mathbf{0.807 \pm .012}$	$0.043 \pm .008$	$0.212 \pm .033$
PCA Recon.	$0.630 \pm .000$	$0.054 \pm .000$	$0.144 \pm .000$
Dense AE	$0.733 \pm .020$	$0.057 \pm .007$	$0.164 \pm .021$
LSTM AE	$0.774 \pm .004$	$\mathbf{0.164 \pm .022}$	$\mathbf{0.271 \pm .015}$

Isolation Forest is the best *ranker*—it orders anomalies most accurately across the full score distribution (AUC-ROC = 0.807). But at the 5% FPR threshold, it detects only 4.3% of insider days. The LSTM autoencoder detects 16.4%—a **3.8× improvement** ($p = 0.031$, one-sided Wilcoxon signed-rank, Cohen's $d = 4.1$).

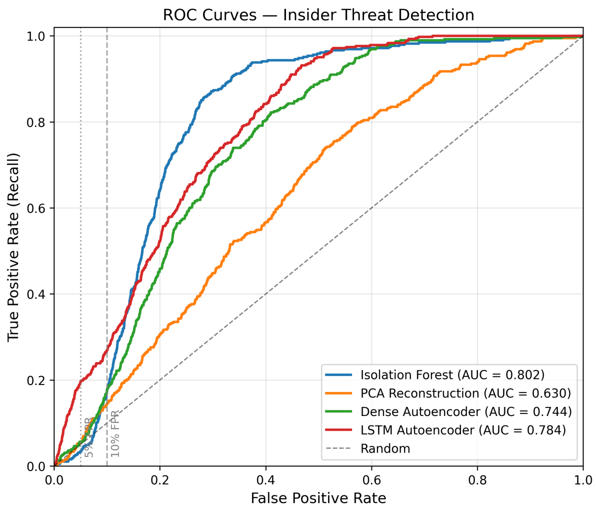

Fig. 1: ROC curves for all models. Vertical dashed lines mark 5% and 10% FPR. Isolation Forest dominates the mid-range, but the LSTM autoencoder pulls ahead at low FPR—exactly where security operations live.

Why does this happen? Isolation Forest identifies anomalies based on global isolation—how far a point sits from the bulk of the data. This produces good overall ranking but distributes its discriminative power across the entire score range. The LSTM autoencoder, by contrast, is specifically sensitive to *temporal anomalies*—unusual sequences that deviate from learned behavioral patterns. These temporal anomalies produce distinctly high reconstruction errors, concentrating the model's discriminative power at the extreme tail of the score distribution—precisely the region that maps to low FPR thresholds.

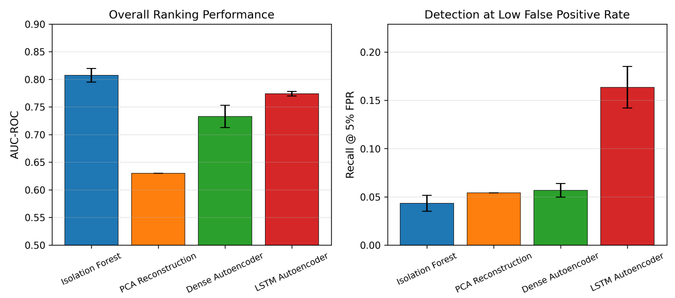

Fig. 2: The reversal. Left: AUC-ROC favors Isolation Forest. Right: Recall@5%FPR favors the LSTM autoencoder by 3.8×. The "best" model depends entirely on which metric you trust.

B. What Gives Insiders Away

Feature importance from Isolation Forest (split frequency) and correlation analysis with insider labels converge on the same finding: **USB device activity is the strongest signal**.

Device-related features (connects, disconnects, activity) show the highest correlation with attacks ($r = 0.075$). This is consistent with the CMU-CERT scenarios, where data exfiltration via removable media is a primary attack vector.

Interestingly, after-hours variants of features show nearly identical importance to their regular counterparts (ratio: 1.02×). This suggests that *what* an insider does matters more than *when* they do it—and that simple "after-hours = suspicious" rules would produce excessive false positives without proportional detection gains.

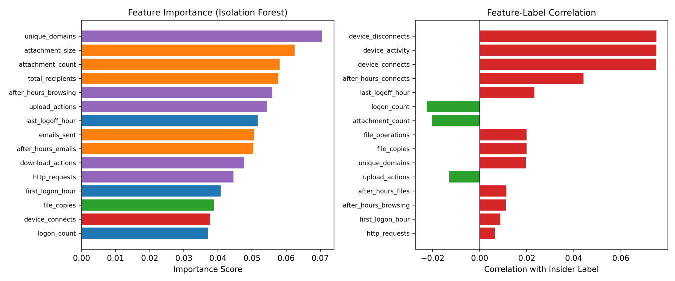

Fig. 3: Left: Feature importance (Isolation Forest). Right: Correlation with insider labels. USB device features dominate both analyses.

C. How Long a Window?

If temporal context helps, does more context help more?

TABLE III: Sequence Length Ablation (LSTM Autoencoder)

Window	AUC-ROC	Recall@5%	Recall@10%
7 days	$0.774 \pm .004$	$0.164 \pm .022$	$0.271 \pm .015$
14 days	$0.787 \pm .002$	$\mathbf{0.199 \pm .011}$	$\mathbf{0.279 \pm .007}$
30 days	$\mathbf{0.802 \pm .002}$	$0.174 \pm .004$	$0.246 \pm .000$

The answer depends on which metric you trust. AUC-ROC increases monotonically with window length ($0.774 \rightarrow 0.787 \rightarrow 0.802$), suggesting longer context captures richer behavioral patterns. But Recall@5%FPR—the operationally relevant metric—peaks at 14 days (0.199), then drops at 30 days (0.174). The 30-day window pushes the ROC curve higher overall but loses precision at tight operational thresholds.

Why the divergence? The 14-day window captures multi-day attack patterns (staging, escalation) while remaining short enough that attack behavior dominates the sequence. The 30-day window provides richer context for global ranking but dilutes attack signal within individual sequences. This is the AUC-ROC paradox applied to temporal resolution: the metric that averages across all thresholds disagrees with the metric that matters at the threshold you would actually use.

D. Which Attacks Are Catchable?

The dataset contains three scenario types:

- **Scenario 1** (30 insiders): Data staging and exfiltration via removable media
- **Scenario 2** (30 insiders): Subtle exfiltration via email and HTTP
- **Scenario 3** (10 insiders): Anomalous login patterns

Detection rates at 5% FPR vary sharply by scenario: Scenario 3 achieves 100% (2/2 attack-sequences), Scenario 1 achieves 69% (47/68), and Scenario 2 achieves only 3% (11/353). The pattern is stark: dramatic behavioral changes (anomalous logins, after-hours USB access) are detectable; quiet exfiltration via email and HTTP is nearly invisible because those activities are part of normal work.

E. The 86% That Got Away

At the 5% FPR threshold, the LSTM autoencoder detects only 60 of 423 attack sequences (14%). Understanding *why* 86% evade detection is as important as understanding what works.

Detected attacks are loud. Detected attack-sequences show elevated after-hours device connections (+1.62 SD above normal), logoff counts (+1.26 SD), after-hours logons (+1.22 SD), and after-hours file operations (+0.97 SD). These are multi-feature behavioral spikes—visible anomalies against the baseline.

Missed attacks are quiet. Missed attack-sequences show device activity at +1.04 SD—elevated, but along a single dimension while all other features remain near normal. The model detects multi-feature anomalies (Scenario 1: 69%) but misses single-dimension exfiltration (Scenario 2: 3%). This is the *boiling frog* pattern: quiet, steady data exfiltration that stays beneath the anomaly threshold because each individual day looks almost normal on most features.

This is a fundamental limitation of anomaly-based detection, not just a limitation of our models. Any system that defines "threat" as "deviation from normal" will miss threats that are designed to look normal. The most sophisticated insiders are precisely the ones who pace their activity to avoid triggering statistical thresholds.

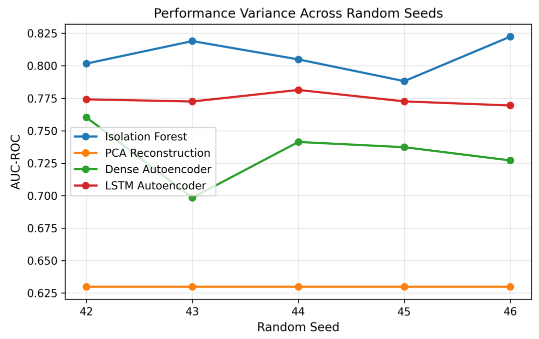

Fig. 4: Performance stability across 5 random seeds. All models show reasonable variance; the LSTM autoencoder exhibits the lowest AUC-ROC variance (±0.004).

V. Discussion

A. Why Temporal Models Win at Low FPR

The reversal between AUC-ROC and Recall@5%FPR is not a quirk of these particular models—it reflects a structural property of temporal versus static anomaly detection under extreme class imbalance.

Isolation Forest measures isolation in feature space. It identifies points that are globally unusual, assigning smoothly distributed anomaly scores across the population. At moderate FPR thresholds ($>$10%), this produces excellent detection. But at 5% FPR, the threshold sits deep in the tail of the score distribution, where global isolation provides little discrimination between the "slightly unusual" normal users and the "slightly unusual" insiders.

The LSTM autoencoder measures temporal reconstruction error. It does not ask "is this day unusual?" but rather "is this *sequence* of days consistent with learned patterns?" A user downloading files at 2 AM might not register as globally anomalous if their daily activity volume is unremarkable. But a week-long trajectory of escalating after-hours activity—normal Monday, late Tuesday, USB Wednesday, bulk downloads Thursday—produces high reconstruction error because the *sequence* deviates from the temporal patterns the model learned.

This concentrates the LSTM's discriminative power in the extreme tail of the score distribution, exactly where low-FPR thresholds operate.

B. A Two-Stage Architecture

These findings suggest a practical deployment: use Isolation Forest for rapid, low-cost screening of all users (its high AUC-ROC provides effective prioritization), then apply the LSTM autoencoder to the flagged subset for temporal analysis. The 3.8× detection improvement at low FPR justifies the additional computational cost for high-risk users.

C. Limitations

Synthetic data. CMU-CERT is designed by security experts but remains simulated. Real insider behavior may exhibit different patterns, and models trained on synthetic data require validation on real incidents.

Limited scenarios. The 70 scenarios across 3 types may not represent the full diversity of insider threats. Slow, methodical data collection and social engineering are underrepresented.

Computational cost. LSTM training requires ~60 minutes per seed on CPU. Inference (8–9 seconds for 85K sequences) is acceptable for daily batch processing but not real-time alerting.

Threshold sensitivity. The 5% FPR threshold generates approximately 50 daily alerts per 1,000 users. Whether this is tractable depends on organizational analyst capacity.

The boiling frog. Our failure analysis shows that 86% of attack sequences evade detection. Anomaly-based methods are most effective against "smash and grab" attacks, not sophisticated insiders who pace their behavior. This is not a limitation that better models will easily overcome—it is inherent to the anomaly detection paradigm.

VI. Conclusion

The standard evaluation metric for anomaly detection—AUC-ROC—gives a misleading picture of insider threat detection performance. On the CMU-CERT r4.2 dataset, Isolation Forest achieves the highest AUC-ROC (0.807 ± 0.012), but the LSTM autoencoder provides 3.8× higher recall at the 5% FPR threshold where security teams operate (0.164 vs. 0.043, $p = 0.031$).

The lesson is not merely that temporal models outperform static ones. It is that **how you measure matters as much as what you build**. AUC-ROC averages across all thresholds; operational deployment uses one. For extreme-imbalance problems with constrained analyst capacity, threshold-specific metrics like Recall@k%FPR are not supplementary—they are primary.

Three additional findings bear emphasis. First, USB device activity is the strongest attack indicator, suggesting that removable media monitoring should be prioritized in deployment. Second, 14-day windows optimize the operationally relevant metric (Recall@5%FPR = 0.199), even though 30-day windows achieve higher AUC-ROC—a further instance of the metric paradox. Third, and most soberly, 86% of attack sequences evade detection even under the best model. The "boiling frog" pattern—insiders who pace their activity to stay within normal bounds—represents a fundamental challenge for any system that defines threats as deviations from a learned baseline.

Future work should investigate: (1) peer-group normalization to detect subtle within-role anomalies, (2) hybrid approaches combining behavioral anomaly scores with data-centric indicators such as sensitive file classifications, and (3) Transformer architectures with self-attention for improved interpretability of temporal patterns.

All code and experiments are available at `[https://github.com/BipinRimal314/insider-detectio` to support reproducibility.

Acknowledgments

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

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.