

Sequence-Based Insider Threat Detection: An Evaluation of LSTM Autoencoders on the CERT Dataset

Shouq Alrowili¹, Asma A. Alhashmi², Abdulbasit Abdullah Darem^{3*}

Department of Computer Science, College of Science, Northern Border University, Arar 73213, Saudi Arabia^{1,2}
Center for Scientific Research and Entrepreneurship, Northern Border University, Arar 73213, Saudi Arabia³

Abstract—Insider activity is hard to evaluate through sporadic events since the same actions may occur normally. In this study, we explore the hypothesis that a small history of user activities might be used as a baseline. We work with CERT r4.2 and merge logon, device, HTTP, file, and e-mail logs into daily profiles. Daily profiles are then mapped into five days' overlapping windows. The first 70% of each user's history is taken for training, and the last 30% is left for testing to ensure that the test is chronologically posterior. The LSTM Autoencoder is trained on benign sequences without any labeling regarding their malignancy; malicious labels are not a part of the task objective. MAE between the input and the reconstruction by the Autoencoder is used as the anomaly score. At a 0.5709 cut-off value, the model shows 0.94 Precision, 0.97 recall, and 0.96 F1-Score. Both RF and SVM achieve better benchmark metrics; however, both algorithms learn directly from labeled examples and use SMOTE in the supervised training branch. That is why we consider our experiment as a comparison of detection settings rather than a comparison of two models. In addition, the choices that have a crucial impact on the result are recorded. The experiment shows that a benign-only sequence model can recover a strong anomaly signal on CERT r4.2 while preserving a realistic direction of time.

Keywords—Insider threat detection; User Behavior Analytics (UBA); Long Short-Term Memory (LSTM); autoencoder; anomaly detection; CERT dataset; sequence modeling; cybersecurity

I. INTRODUCTION

However, the practical problem is that access remains legitimate until it turns out to be illegitimate; that is, until the user logs on using proper credentials, operates on an authorized machine, opens files, emails, and browses the Web, or attaches some removable device without breaking any access rule. It becomes impossible to judge the legitimacy of any single event. The difference is made by context: it concerns whether the event is legitimate behavior for this user, happens at an unusual time, changes its volume, and is accompanied by some other events. Thus, the behavioral analysis of activity plays an important role in insider threat detection research since static signatures are not suitable for misuse composed of legitimate events [1], [2].

Time is one of the elements of context. For example, after-hours login is an uninteresting event in itself – it just may mean that there is some deadline. However, when this login event takes place among several days of increased file activity, frequent USB connections, and/or rapid changes in communication, it becomes more suspicious. Time-based

features allow describing the current counts. This is precisely why sequence models appeal. Recurrent neural networks, attention systems, transformers, and continuous-time models have all been employed to encode some idea of ordering or behavioral context [9], [12], [19], [21], [26]. We use a simpler model than most of these extensions and see whether a five-day history has anything informative to tell us.

However, there is another problem that one can easily ignore: the evaluation process can undermine the temporal nature of the task, even if the model is sequential in itself. The random shuffling of security logs means that past behavior will influence the training process, and overlapping windows will appear on opposite sides of the train-test split. A good performance metric achieved this way could mean nothing more than data leakage rather than future classification performance. Experimental bias in security machine learning has been pointed out time and again by several studies [3], [8]. In our experiment, chronology will be established before applying any model-specific operations. Future behavior will be withheld, and all balancing will occur at the training stage.

We utilize CERT r4.2 because it provides several audit sources in one controlled environment and offers ground truth for injected insider scenarios [4], [40]. These characteristics make CERT useful to test a pipeline of feature extraction from raw activity data up to final alert score generation. They, however, put constraints on what conclusions can be drawn. Users and attacks are artificial here, and labels are a lot cleaner than they would be in a realistic setting. We, therefore, employ CERT as a benchmark for the choices made in our design, not as a proxy for alert volume or attack prevalence in real production. It also becomes important when we compare RF and SVM with the Autoencoder since only the former have malicious labels available for training.

The research is structured around three questions. First, can we reduce five different CERT audit sources to daily user profiles without losing behavioral information necessary for a sequence-based classifier? Second, what can be achieved with a benign-only LSTM Autoencoder with respect to time-aware evaluation and comparison with supervised RF and SVM models based on the same engineered behavioral features? Third, how should the scores be interpreted in light of class imbalance, thresholding, lack of equal label access, and the fact that CERT is artificially generated? These issues ensure that our

*Corresponding author.

study concentrates on the experiment that was done, rather than on claims of universal insider threat detection.

The primary contribution of the study is in the design of the experiment and the clarity of the pipeline itself. First, the time window is defined, and after that, SMOTE is applied to the process of supervised learning. Autoencoders are fit using benign sequences, not attacks. Five-day windows overlap, mean absolute error is the reconstruction error, and the 0.5709 cutoff is the one maximizing the F1-score. None of these decisions is obscured by one headline statistic. This is intentional. In the given case, a score cannot be reasonably interpreted without an understanding of the order of the samples, information available at each branch, and transformation of the continuous anomaly score to the binary decision.

II. RELATED WORK

A. Behavior-Focused Detection on CERT

Much of the previous insider threat literature starts with activity-engineered variables and a classifier under supervision. This approach makes sense provided the malicious class is labeled well enough to train on. RF and SVM serve as great references in this case since both are established and strong models for tabular data rather than weak baselines [7], [30], [31]. Others focus on representations of individuals. User and role profiles are introduced by Legg et al., and Kim et al. introduce a user-behavior model and anomaly detection [6], [11]. The common point is clear: an action is not dangerous by itself but depends on who did it, when, and how typical it is.

The work of CERT simplified control comparisons in a way that several enterprise-like logs were packaged with the corresponding insider threat scenarios [4], [40]. Such an approach has its price: there is no unified 'CERT task'. Different publications use different release dates, aggregation level, feature engineering approach, positive sample definition, sampling approach and even evaluation split. D. C. Le et al. demonstrated that changes in granularity can change the results drastically [15]. Reviews come to the same conclusions about the whole field: high scores are common, but the experiments answer different questions [13], [14], [20]. For Therefore, previous works will be used mostly for justification purposes and not for showing cross-paper F1 scores as though they come from a single standardized leaderboard.

B. Learning Without Attack Labels

Anomaly Detection takes a different starting point. Rather than asking the classifier to detect the boundary between benign and malicious samples, Anomaly Detection is about modeling normal behavior and giving large scores to outliers. There are multiple ways of doing so through Isolation Forests, one-class learning, or reconstruction [32], [33], [38]. There are multiple insider threat works that use some of these techniques. For instance, Le and Zincir-Heywood study unsupervised ensembles, whereas Pantelidis et al. investigate deep and variational Autoencoders [17], [18]. The latter is useful in case of rare attack labels.

The appeal of an Autoencoder is pragmatic: good samples can be used for input and target for training. Here we are asking the model to reproduce familiar patterns instead of assigning a

class of attack. The whole realm of representation learning offers background knowledge of how to learn structure from unlabeled data [36], [37]. For anomaly detection, the difference between input and reconstruction becomes an indicator of deviation. Malhotra et al. generalize the idea to an LSTM Encoder-decoder architecture to reconstruct ordered sequences, instead of rows [5]. This corresponds well to our question since the subject we want to assess is a sequence of five days of a user's activity.

Reconstruction is not a judgment call. A high value of the reconstruction error indicates that the sequence cannot be easily represented using benign samples; it doesn't mean anything else. A career change, travel, maintenance activities, an unexpected deadline, or even a legitimate bulk transfer of information could produce the same mathematical phenomenon. So, we use reconstruction errors as a sign of triage. This cautious interpretation is consistent with general anomaly-detection work and with insider-threat research that treats anomalous behavior as something to investigate rather than an automatic accusation [13], [33].

C. Time-Aware Models and the Position of this Study

Various sequential insider-threat models have been proposed before. For example, Tuor et al. make use of deep and recurrent models to score structured security streams in an unsupervised fashion, Yuan et al. look into deep neural detection, and Sharma et al. use an LSTM Autoencoder for UBA [9], [10], [12]. More recently, Villarreal-Vasquez et al. investigated LSTM-based anomaly detection for threat hunting where malicious events occur mixed with legitimate events [21]. These models are quite different in their implementation but share one common good assumption that behavior can take place over many observations, and this progression gets lost in case the records get classified independently from each other.

However, recent studies introduce new models not only with respect to architecture, but also to the representation of time itself. For example, Pal et al. place attention on temporally aggregated behavior, and Song et al. model rhythmic behavioral patterns along with user adaptation [19], [22]. AISlaiman et al. look into a temporal LSTM-based model that is intended to decrease the number of false positives and false negatives in the context of CERT r4.2 [23]. The self-attention mechanism allows for sequence modeling without recurrences [39], and Elbasheer and Akinfaderin use this mechanism for user-based CERT sequence modeling [24] [25]-[27]. Such diversity of approaches makes it crucial to distinguish architectural innovation from innovations in data creation.

We try to keep the core architecture as simple as possible. Incorporating attention, graph information, or a transformer would make the model more sophisticated, but would also add more potential factors to explain the change in performance. In the current experiment, we need a simpler baseline: five days of aggregated behavior, an LSTM reconstruction goal, chronological division, and absence of any malicious instances in Autoencoder training. If such a baseline turns out to be valuable, then further experiments will gradually replace individual components and study their effects. It is less architecturally ambitious but helps us defend the current interpretation of the experiment better.

In Table I, we provide a comparison of several temporal models depending on how they represent history. We decided to use this approach as there is no direct comparison of the underlying tasks. Different CERT releases, features, session definitions, and verification procedures could shift the metrics significantly. Studying the idea of temporal modeling - recurrent history, attention, sessions, continuous time, or graph context - is much more valuable for our architecture than comparing artificially constructed F1 scores.

TABLE I. COMPARISON OF SEVERAL TEMPORAL MODELS

Study	Main Model	Temporal Representation	Relevance Here
Tuor <i>et al.</i> [9]	Deep/RNN models	Structured user-activity streams	Shows that recurrent structure can support unsupervised scoring of user activity.
Sharma <i>et al.</i> [12]	LSTM Autoencoder	UBA sequences from CERT activity	Provides direct precedent for reconstructing normal CERT behavior with an LSTM Autoencoder.
Villarreal-Vasquez <i>et al.</i> [21]	LSTM anomaly detector	Ordered event sequences	Uses ordered sequences to detect malicious steps embedded in legitimate activity.
Pal <i>et al.</i> [22]	Attention ensemble	Temporally aggregated activity	Shows how attention can weigh different parts of a temporally aggregated history.
Song <i>et al.</i> [19]	BiLSTM/FFN	Time-aware behavioral rhythm	Adds explicit behavioral rhythm and user adaptation to sequential modeling.
Elbasheer and Akinfaderin [24]	Transformer encoder	User-based sequences	Provides a self-attention alternative to recurrent user-sequence modeling on CERT.
Mateo-Muñoz <i>et al.</i> [25]	Multiple ML models	Activity sessions	Replaces fixed daily windows with activity sessions and includes CERT/SPEDIA evaluation.
Jurišić <i>et al.</i> [26]	LSTM vs. NCDE	Discrete vs. continuous time	Compares discrete LSTM steps with continuous-time modeling of user anomalies.
Paduvilan <i>et al.</i> [27]	Sequence-graph fusion	Temporal and relational context	Combines temporal evidence with relational context through sequence-graph fusion.

Moreover, the table above outlines what is not included in the current research design. There is no attention mechanism, no organizational chart, no content-based embeddings, and no second temporal architecture used in the model. Daily unit of analysis, 5-day window, feature vector, and anomaly reconstruction objective remain the same. Such constrained experimental design allows for future studies to keep a stable point of comparison when using the Transformer instead of LSTM, and vice versa, using the session as a daily window while maintaining the same objective.

III. DATA AND METHOD

In this research, we use the five types of data sources provided by the CERT r4.2 package: logon activity, device activity, HTTP requests, file access, and email activity, with ground truth of insider scenarios. The data sources come as raw event files, however, the machine learning algorithm in our case does not operate on the level of event but rather on aggregated user-calendar day level. Thus, the input of our temporal model becomes a daily behavioral profile rather than HTTP requests, file accesses, or emails.

Aggregation at the level of one day is a compromise. An event-level model would keep order and precise time information but create sequences too long and dependent on high-volume sources like HTTP. Aggregation per week would decrease volume but may aggregate development that spans several days into one. The daily level allows for having a common clock across all sources and short histories to be examined. As previous work has shown, granularity has an effect on insider threat performance [15]; therefore, the choice of daily as the aggregation level is included in the experimental design.

Table II summarizes the set of features chosen for each daily profile. All of them are counts except those that track the timing or specific categories of actions: after-hours use, cloud storage browsing, or searching for a job; executable files, or compression files. The decision is deliberately made not to go further than audit data. Thus, there is no analysis of emails' contents or semantic representations, which would introduce another research question into the experimental design.

TABLE II. SUMMARIZES THE SET OF FEATURES

Log Source	Extracted Features	Operational Meaning
Logon	total_logons, after_hours_activity	Overall access activity and access occurring outside the normal working period.
Device	distinct_pcs_used, total_usb_connects, usb_after_hours	Workstation use and removable-media activity, including USB activity outside normal hours.
HTTP	total_http_visits, http_after_hours, suspicious_web_visits	Browsing volume, timing, and selected categories such as cloud-storage or job-seeking domains.
File	total_file_actions, file_after_hours, suspicious_file_types	File-operation volume, timing, and activity involving selected executable or compressed file types.
Email	total_emails, email_after_hours, emails_with_attachments	Communication volume, after-hours messaging, and messages containing attachments.

The variables cannot be seen as handwritten attack rules. A programmer is allowed to copy many files; an administrator might be on duty at nighttime; using USB may be a common practice for a certain position. The important thing here is whether there are changes in the number of these actions and whether there are several such changes simultaneously. This is why the set of features may be quite useful for a reconstruction model: when a sequence scores badly, the variables themselves will retain their meanings known to the analyst through audit logs. The drawback is obvious, though.

The labels of CERT are used in two different ways for learning. RF and SVM models use benign and malicious labels

for training data; the autoencoder-based approach does not. The set used for fitting the Autoencoder is limited to benign sequences only, and the labels are introduced only when scoring the test data. Maintaining this separation is important for the research question raised. If malicious cases were used during Autoencoder fitting, this experiment would not be a simulation of the benign-only modeling of normal activity.

Records are sorted per user before separating the learning pipeline into training and testing phases. The first 70% of the history sequence is used in training, while the remaining 30% serves as the test set. Shuffling of records between these two sets is not done. The result is quite simple yet crucial: all test sequences are chronologically located after the behavior corresponding to fitting the user's part of the model. The process is closer to the direction the detector would work once deployed than a random holdout method.

Overlapping windows show the reason why that boundary needs to be there. In two adjacent five-day windows, four days overlap. Even without shuffling in random cross-validation, it could easily happen that nearly the same history would be in training and testing datasets except for rows being the same. That way, the model would be incentivized to generalize to some sequence that is partially known to it. Chronological splitting prevents that particular type of leakage and adheres to recommendations of more general time-dependent security evaluation practices [3], [8]. It does not eliminate all dependencies in the data, but it prevents mixing future local histories into the training phase.

SMOTE is used only after the temporal split has been made and only in the RF/SVM training sub-branch. The latter testing phase is unchanged. This order avoids the possibility of artificially created minority examples being contaminated by records from the future evaluation phase. We do this consistently for all transformations based on learning from data: train using training data, use on held-out data. In the case of this study, the placement of SMOTE is a matter of leakage control, not just class balance.

Let the daily feature vector of user u at time t be represented by $\mathbf{x}_{u,t}$. An input to the model up to day t consists of that day plus the four previous days. This creates a 5-day history, where d is the number of daily features:

$$\mathbf{X}_{u,t} = [\mathbf{x}_{u,t-4}, \mathbf{x}_{u,t-3}, \mathbf{x}_{u,t-2}, \mathbf{x}_{u,t-1}, \mathbf{x}_{u,t}] \in \mathbb{R}^{5 \times d}$$

Overlapping windows are used. Once the first four days have been established to form a history, each subsequent eligible day is able to be the end of the next sampling period. More temporal information is kept by using overlapping windows rather than non-overlapping windows, but this also means that adjacent windows are dependent because they are built on a common four-day basis. This dependency is fine within a partition because that is exactly why random train-test folds would be unacceptable. It is the time border, not sample independence, that prevents a near-duplicate local history from being used in testing.

Five days is a matter of design rather than a belief about how long insider behavior lasts in the real world. We do not optimize over one-, three-, seven-, or fourteen-day windows and pick the

best-performing option. The window size is chosen for the experiment for the purpose of providing the model with an easily interpretable short local history. Different representations of time are possible in session-based and continuous-time models [25], [26]. A horizon study is required at a later stage, and the current result does not imply optimality of five days.

Both RF and SVM serve as supervised benchmarks. They both take labeled samples of benign and malicious classes from the prior training stage and work with point-in-time engineered behavior features and not the five-day Autoencoder input. RF is valuable since it can model non-linear relations between features with minimal pre-processing, while SVM offers a powerful margin-based benchmark [30], [31]. It is not necessary to replicate the anomaly detection framework. The two serve different and important purposes: How separable are the engineered CERT patterns given samples of the malicious class in training?

Malicious examples being rare, the supervised training set is balanced using SMOTE [28]. For any minority sample $\mathbf{x}_i$, and for a randomly selected minority sample $\mathbf{x}_j$, a synthetic sample is created through interpolation:

$$\tilde{\mathbf{x}} = \mathbf{x}_i + \lambda(\mathbf{x}_j - \mathbf{x}_i), \quad 0 \leq \lambda \leq 1$$

None of the samples in the testing phase are synthetic. The out-of-sample samples maintain their original distribution such that Precision, Recall, and F1 will be calculated in the rare-event context. Oversampling has been considered in similar work on insider-data leakage [16]; however, the key point here is when oversampling takes place. If oversampling were applied before the temporal split, this would blur the line between training and testing. Using oversampling only in supervised learning leaves the testing phase independent of this balancing process.

The Autoencoder receives the five-day sequence both as input and output. The Autoencoder's encoder encodes the ordered daily vectors into a latent space representation, and its decoder decodes the latent space representation back into the original sequence. LSTMs are employed due to their gated memory units, which can remember information through time from previous timesteps in ordered observations [29]. This is not the case for predicting attack labels using LSTM. The model learns to reproduce benign activity, and failing to do so later becomes the indicator.

The contribution to the Autoencoder loss function comes only from benign sequences within the chronological training period. If f_θ denotes the encoder-decoder mapping, fitting minimizes reconstruction loss across the N benign training sequences:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^N \mathcal{L}(\mathbf{X}_i, f_{\theta}(\mathbf{X}_i))$$

After fitting, each test sequence is run through the model and scored against its reconstruction. Mean Absolute Error (MAE) is used as the sequence-level score according to the common formulation of LSTM encoder-decoder anomalies [5]:

$$s(\mathbf{X}) = \frac{1}{Td} \sum_{t=1}^T \sum_{k=1}^d |x_{t,k} - \hat{x}_{t,k}|$$

A high MAE score indicates that the test sequence is difficult for the benign model to reconstruct. We do not take the score as a probability of malicious activity. This score just shows the level of divergence from the patterns observed by the benign model during training. It is important since legitimate changes may be highly unusual. The Autoencoder is not trained to predict the CERT scenario that generated the alert, providing just one anomaly score per sequence.

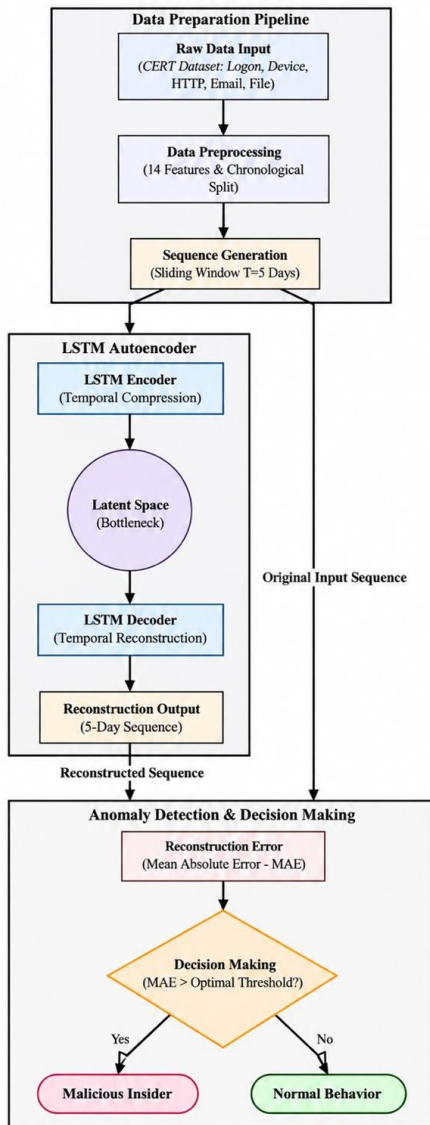


Fig. 1. LSTM Autoencoder used to reconstruct short sequences of daily user behavior.

Since the Autoencoder output is continuous, a threshold needs to be established in order for it to count as a detection. Cutoffs are assessed using the precision-recall tradeoff curve, and we state the one with maximum F1 for this particular benchmark as 0.5709. Fig. 1 is specific to this experiment, and it is NOT a universal CERT threshold; hence it is not to be used

by anyone else in any other place. Another user distribution or alert budget may warrant a different operating point.

Precision, Recall, and F1-Score are the classification metrics that we report. If TP, FP, and FN represent the number of true positives, false positives, and false negatives, then the formulas are:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}$$
$$F1 = 2 \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

The ROC curves are also included to illustrate how the ranking evolves as a function of the threshold. However, since there is an inherent class imbalance problem, the precision-recall curves become especially important here [34], [35]. Just a few percent of false positives can generate many benign alerts because the majority of cases in the stream are regular. For this reason, no balancing procedure is performed on the test set. The statistics are computed on the natural held-out distribution, not the artificial mixture of the two classes.

Table III is included since the two learning branches do not receive the same data. While RF and SVM receive the labels for malicious instances and use SMOTE in the training process, the Autoencoder receives the labels for the benign instances only and generates the reconstruction score. Placing these options next to each other helps to avoid a misunderstanding of the performance table. The statistics can be compared as results obtained over the same period of the benchmark, but the supervision is not identical here. The table also includes the aggregation unit, the sequence length, the splitting rule, the anomaly score, the threshold, and the main metrics.

TABLE III. EXPERIMENTAL PROTOCOL AT A GLANCE

Choice	Reported Setting
Dataset	CERT r4.2
Aggregation	User-day profiles from five log sources
Sequence length	5 consecutive days
Split	Earliest 70% train; latest 30% test
RF/SVM labels	Benign and malicious labels in training
RF/SVM balancing	SMOTE on training partition only
LSTM AE labels	Benign sequences only during fitting
Anomaly score	Mean Absolute Error
Selected cutoff	0.5709
Primary metrics	Precision, Recall, F1-Score

IV. RESULTS

As for the later 30% holdout, the LSTM Autoencoder attains 0.94 Precision, 0.97 Recall, and 0.96 F1-Score (see Table IV). The most important component of the anomaly detection result is Recall: most of the malicious sequences have passed the selected threshold despite the absence of any malicious sequence among the malicious targets for training the Autoencoder. However, the Price of the high Recall is 0.94 precision, which means that some sequences considered to be benign are sufficiently different from the trained ones. Operationally speaking, the algorithm catches most of the

periods that are known to be malicious but does not reduce the amount of the analyst's work.

The supervised models perform better in this regard: SVM attains 1.00 Precision and 0.99 Recall, with F1 equal to 1.00, while RF obtains 1.00 Precision, Recall, and F1. The values above clearly indicate that the engineered features are extremely separable with the given dataset split after the learner gets access to malicious sequences. They do not prove that other insider strategies can be easily classified, either. Therefore, the results can be taken as evidence of a good, supervised CERT solution rather than the evidence of a complete insider-threat solution.

TABLE IV. PERFORMANCE EVALUATION OF PROPOSED VS. BASELINE MODELS

Model	Precision	Recall	F1	Evaluation
SVM	1.00	0.99	1.00	Point-in-Time (Chronological)
RF	1.00	1.00	1.00	Point-in-Time (Chronological)
LSTM AE	0.94	0.97	0.96	Sequence-Based (Chronological)

The numbers bring out facts that the round table does not reveal. As per Fig. 3, the Autoencoder gives some rise to the number of false alarms but guarantees very good coverage of malicious sequences. Fig. 2 depicts how the ranking changes with the change in the threshold. It is helpful to consider both of them since the real-world cost of missing one malicious sequence is distinct from the cost of presenting the extra benign sequence to an analyst. One aggregate figure will never be able to capture that difference.

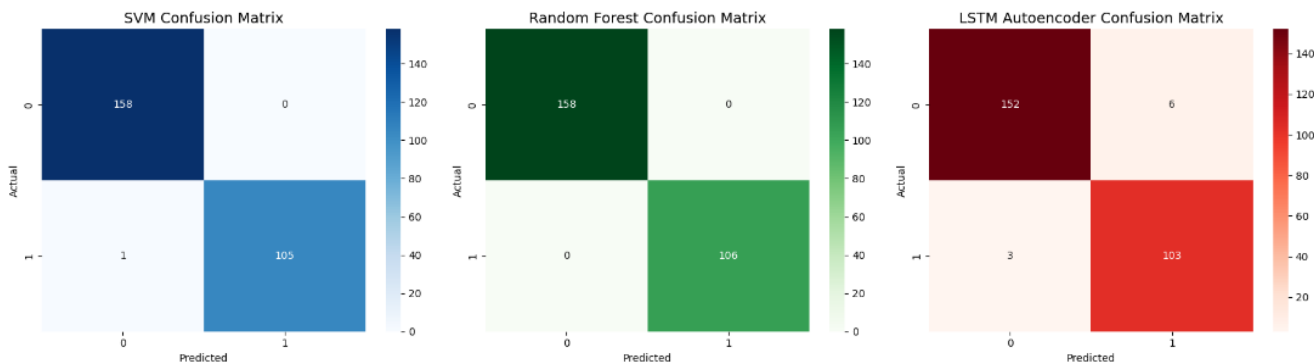


Fig. 3. Confusion matrices for the supervised baselines and the LSTM autoencoder.

This does not mean that the supervised approach is unjust or flawed. With reliable labels for attacks available, a production detector has to use them. The query posed by RF and SVM in this context is simply a narrower one: is it possible to recognize malicious patterns based on CERT-like criteria after they were labeled? And the answer is affirmative with this feature set. Issues of limited ground truth, changing behavior, and transfer across environments remain problems in the review of insider threat detection methods [13], [20]. A future strategy of insiders will probably go beyond the supervised decision region.

An Autoencoder, however, solves the complementary problem. It queries whether the behavior is different enough from those benign ones that were used in the process of fitting. The 0.96 F1-Score here makes sense since the model maintains

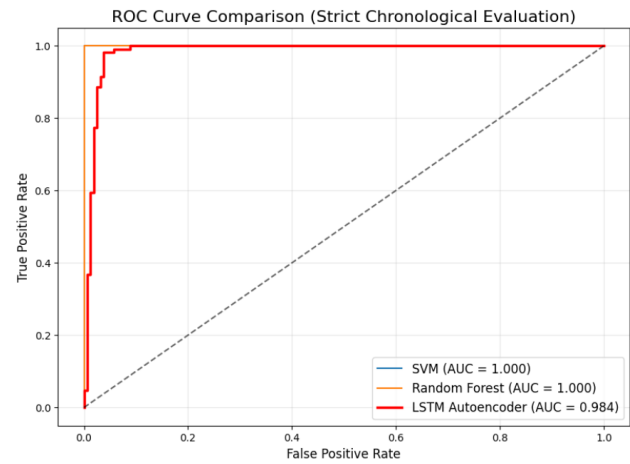


Fig. 2. Receiver Operating Characteristic (ROC) curve comparison.

It will be easy to see the distinction between near-perfect RF/SVM scores and the Autoencoder when we consider the training set. RF and SVM are provided with labeled instances of the malicious samples belonging to the previous period. The CERT scenarios may have particular combinations of features related to timing, files, devices, and communication. This allows the supervised model to learn a direct decision boundary around the labeled instances. With SMOTE, the weight of the small class of malicious sequences becomes even more significant.

anomaly patterns in its predictions without ever being trained on a malicious class. In the actual monitoring stack, both approaches can co-exist: the first one can be applied to known patterns, and the latter to anomalous behaviors.

The difference lies in the evidence that the model has at its disposal. The single user-day vector is able to demonstrate that file activity or USB usage is high on the present day. On the other hand, a sequence of five days is capable of demonstrating whether the increase happens after some quiet days, follows an increase which has started before or is going back towards the recent normal level. It is the temporal pattern that is the motivation behind the use of the LSTM network as opposed to transforming each day into a sample independently from the rest.

It is impossible to state how much gain the use of five-day sequences brings because the same experiment was not carried out with a one-day Autoencoder with the same feature set and train/test split. Therefore, any statement in the form of 'LSTM model improves F1 by X percentage in comparison with static Autoencoder' is not appropriate for the experiment. However, it is appropriate to say that the five-day reconstruction model performs well within the described chronological pipeline. Recurrent networks and attention models encourage this pipeline [12], [19], [21], [22], but they cannot be a replacement for the ablation study.

The same caveat holds true for the length of history as well. Five days have not been chosen after considering many alternatives. Shorter history could have been more sensitive to sudden changes; a longer history could have been more sensitive to slow drifting. Formulations of session-based and continuous-time models get rid of having any fixed daily window size [25], [26]. It would be interesting to conduct another set of experiments while keeping all other parameters constant and varying the number of days in history.

The threshold value of 0.5709 is the value that maximizes F1 score for the current setup. This is an operating decision rather than intrinsic to the LSTM Autoencoders. When misclassification of any suspicious sequence has a high cost, then the threshold could be reduced, and more benign sequences could be investigated. On the other hand, when the availability of analysts is limited, the threshold could be increased by lowering recall. This is why reporting only the model architecture is not enough.

Imbalance contributes to making this decision operationally relevant. The occurrence of normal activity greatly outweighs that of malicious activity. Consequently, a low false positive rate will lead to many false alarms, which contribute to the computational cost. The precision metric is related to that aspect of the problem, whereas overall accuracy may be outstanding even in case of poor performance on the positive class. In this case, we do not alter the distribution of the held-out data set and consider precision-recall curves along with ROC curves for the very same reason [34], [35]. The test set is supposed to remain hard, just like the rare event problem is hard.

There is one reservation regarding the term 'benign-only.' While the Autoencoder is trained only on benign samples, the choice of threshold value is done based on the labeled dataset due to the fact that F1 needs ground truth. This means that the representation learning process is unsupervised as regards attack labels. However, the whole procedure is supervised in terms of the benchmarking procedure since it requires labeled datasets.

The occurrence of a false positive detection from the anomaly detector does not automatically mean that there is randomness. It is possible that the user is benign and simply changes shifts, joins a new project, conducts an authorized bulk operation, or uses another suite of machines. It is possible that the algorithm correctly identifies an anomaly but incorrectly identifies that this behavior is indicative of a threat. False positives are significant because they are costly in terms of analyst effort. They also point to scenarios where the behavioral baseline is incomplete from a contextual perspective.

A false negative, on the other hand, represents a different problem. The anomalous pattern was close enough to the benign region to remain under the threshold. This may be because the behavior is not well-expressed by feature counts, or the variance of the user's normal activity is very high. The appropriate fix will depend on which reason holds true. If the problem is poor feature encoding, additional context is needed. If the user's activity variance is high, greater personalization is needed. The size of the neural network will not help.

Since the results from our experiments do not have errors partitioned per user or scenario, we do not fabricate such an analysis based on confusion matrices. More useful analysis would consist of identifying all false positives and false negatives, grouping them according to scenarios and users, and examining the log sources that were changed near them. This type of analysis should be done for RF and SVM too. With a positive class being very rare, an F1 value of 1.00 could mask even a systematic but relatively infrequent failure mode. It is necessary to aggregate metrics, but they are not the only errors we need to look at.

V. DISCUSSION

Just using the LSTM itself is not sufficient for making an experiment truly temporal in nature. The data splitting must respect the time dimension as well. Due to overlapping windows, the random split can put almost identical past data in training and in testing, while all samples will be formally distinct. The chronological holdout approach makes sure that this particular way to simplify things will not happen. The model learns from past behavior and predicts future behavior. Directionality is an essential part of what is being asserted through this experiment.

This is also true for the sequence of SMOTE application. Oversampling of the minority class is relevant for the supervised classifiers but occurs only after the training and testing periods have been separated. In another case, malicious examples from the future would be able to affect synthetic samples that were generated to be used for fitting in the past. Timing is set first, and balancing remains within supervised learning. This is consistent with the guidelines of leakage avoidance in security machine learning [3], [8].

Furthermore, we differentiate between the process of learning about normal behavior and the decision when the anomaly score is high enough to raise an alarm. The neural network was trained using sequences of normal traffic only. Only after this point, when candidate thresholds are evaluated, labels of malicious traffic become available again. This allows us to say definitely that the network learned to recognize normal behavior independently of any malicious examples. But it does not allow us to claim that there is no labeled data at all.

The daily feature vector is deliberately kept small. Counts for logons, USBs, web accesses, file activity, email volume, and attachment volume can all be derived from known audit data. In evaluating an alert, an analyst can question whether that particular day had an unusually large number of removable media events or file activity during odd hours. This is helpful. It increases the likelihood that a high anomaly score will not be

incomprehensible to the person deciding which alert to examine next.

There is a trade-off, however. Two days may both have the same counts for emails and attachments, but one day has an email sent to an unusual external address. The normal amount of file activity does not say anything about the sensitivity of those files. This information, and other semantic features, are missing from the current input set. Context of the recipient, the sensitivity of the file, or organizational relationships could provide more signal, but would also make the model more invasive and less directly comparable to the current baseline.

That is the reason why neither the design of features nor the architecture of the model can be analyzed separately. No matter how big the size of the LSTM or Transformer is, it will not be able to retrieve the lost data before the training process. Similarly, the superiority of the graph-sequence model might be due to the fact that the graph-sequence model gets relational information that the baseline does not receive. Both Transformer and graph models are natural progressions [24], [27], yet a proper test should indicate whether input information is consistent.

There are several ideas proposed in recent works that can be tested on this particular baseline model. The BRITD introduces behavioral rhythms of time awareness and adaptation [19]. Pal et al. implement attention on the aggregation of activity [22]. The Transformer approach uses self-attention in place of recurrence [24], [39]. Session-based and continuous-time models challenge the concept of daily steps, whereas sequence-graph combination incorporates relations between users, resources, and actions [25]-[27].

That is why we refrain from referring to the 0.96 F1-Score as state-of-the-art. The published papers lack commonality in terms of the release of the CERT dataset, sampling, balancing, and splitting techniques, and even the granularity of the aggregation step affects the problem difficulty [15]. The defensible claim is precise: the benign-only LSTM Autoencoder achieves the stated result using the protocol laid out chronologically in the present study. Any comparison in terms of score ranking implies a certain degree of comparability that does not exist.

A more appropriate comparison would be achieved by manipulating one variable at a time. Using the same features, split, threshold selection process, and hold-out sequences, the LSTM could be substituted for a Transformer Autoencoder and a continuous-time model. Another experiment would include keeping the LSTM fixed but changing the temporal unit from days to sessions. Such an approach would help attribute any improvements to a particular modeling choice.

A simple MAE value will not suffice for the analyst. MAE = 0.63 doesn't explain anything. The usefulness of the metric increases when there is also information about which specific days and features were most responsible for the reconstruction error – such as a rise in after-hours file access combined with repeated USB usage towards the end of the five days. The current study considers only the scoring system and not the full analyst interface, so feature-level attribution is a necessary part of implementation rather than a finding that we claim to achieve.

The next practical question is the problem of cold start. A new hire or a person who has switched positions will have no stable personal history. Baseline based purely on individual usage will be relatively ineffective in exactly this situation. One approach that can be considered in this case is role-aware profiling [6], where the initial expectations can be set based on other people in the same position and then shifted according to the actual individual behavior. The current system of representation does not use this extra step.

Long-term usage results in the opposite dilemma. While the static baseline becomes outdated with changing normal work patterns, the actively refreshed baseline risks being tainted by suspicious events as part of 'normal' activity. Both time awareness and adaptiveness to users are driven, in part, by this contradiction [19]. Thus, an online application of our pipeline would require an update strategy equally well designed as the inference algorithm. The update policy would define which recent behavior was allowed to modify the baseline and which could be used only for assessment purposes.

The monitoring of employees also involves governance considerations which are not covered by the benchmarking framework. Login times, browsing categories, metadata for communications, file activities, and device usage are privacy-sensitive data. For a production system, there would have to be a policy in place concerning access rights, storage policies, and using the anomaly score as evidence of a violation without additional investigation. We do not examine any governance mechanisms here. Thus, the model must be seen as an experiment in detection rather than a complete policy.

VI. VALIDITY AND REPRODUCIBILITY

One of the most significant internal validity threats in the current study is time leakage. The 70%/30% timeline split and SMOTE usage solely in training attempt to ensure that future observations do not affect past fitting procedures. The windows overlap remain dependent both inside the training window and inside the test window; this dependency is preserved. What the timeline boundary ensures is that there is no random mixing of nearly the same local history between train and test sets. Any replication that uses a different split must be aware that it changes the difficulty of the benchmark as well.

Threshold selection is yet another potential source of variability in the results. The precision/recall at 0.5709 is a property of a particular model-threshold pair, not only of the model. Changing the threshold would shift these values, while keeping all the reconstruction scores constant. That is why we choose to report the way the threshold was selected instead of hiding it as a tuning parameter. In the anomaly detection case, it is particularly relevant, as the ranking/score generation takes place long before any binary decision is made.

Uncertainty per run is not quantified in the given experiment. Neural network initialization, the ordering of mini-batches, and the nature of stochastic optimization might shift the boundary of reconstruction. One trained Autoencoder is thus not enough to establish the robustness of small metric differences. Running the same experiment chronologically on different random seeds and then presenting the average value together with the standard deviation would reinforce the empirical result without altering

the research problem. We identify this as missing validation rather than suggesting that the provided single run proves the variance.

The CERT tool provides something that most real-world organizations do not have complete ground truth for the inserted insider scenarios. This makes the evaluation possible; however, this also means that the benchmark is more idealized than actual operational data. Synthetic users simply cannot emulate all the peculiarities of a particular enterprise, such as local practices, exceptions to company policy, seasonal workload, changes in an organization, or any other undocumented activity. Therefore, the results should be interpreted as the ones obtained in the controlled environment.

Testing on a second data set would be more indicative of generalization than testing on an additional architecture on the same CERT split. Recent session-based methods covering both CERT and SPEDIA demonstrate one such path forward [25]. The portability of this current pipeline would not be simple because of the differences between the two schema and label sets, but this difficulty is precisely what an external validation test is meant to reveal. Any degree of transfer will confirm whether the behavioral signal persists when it moves beyond the simulated generation of CERT r4.2.

Daily aggregation constitutes another limit on external validity. Two users might have the same daily activity counts even while having completely different sequences and times of those events. When these details are condensed down to a daily summary, the LSTM loses all information about them. Both session-based approaches and continuous-time methods keep more temporal information [25], [26]. In light of this, the current model must be characterized as detecting changes in multi-day behavior aggregates. It does not analyze event sequences.

However, a replication should do more than just say "LSTM Autoencoder." The benchmark employed in this case is based on CERT r4.2; logon, device, HTTP, file, and email sources; user/day aggregations; the features outlined in Table II; five-day windows with overlap; a 70%/30% split where the earliest is for training and the latest for testing; no random shuffling between windows in terms of boundary crossing; SMOTE applied to supervised training only; benign training for Autoencoder; MAE scoring; and cutoff chosen through F1 at 0.5709. Any modification of any of these settings may alter the learning task itself. They should all be fixed when comparing results before accepting credit for a better or worse score.

Several analyses which would strengthen the validity of the experiment are not included in the output available: multi-seed variance; errors clustered by user and scenario; window size ablation; source-wise feature ablation; and dataset transfer. We highlight these absences explicitly because they tend to be confused with experiments done in a long discussion section. They should be performed as separate, controlled experiments in future work. For now, however, they remain proposed methods of validation.

VII. LIMITATIONS

First, there is a problem that the supervised and anomaly branches do not get equal training. Both RF and SVM get their malicious label data plus SMOTE, while the Autoencoder learns

only from the benign sequences. The scores may be shown side by side, since they are tested using the same hold-out time period, yet the table is not an architecture comparison. It is two approaches to detecting threats, with unequal information.

Second, there is a question of the features used. There are counts and selected timing/category information, but no semantics of the messages, resource sensitivity details, or organization network structure. It makes the analysis easier and puts the content monitoring out of our test scope, but also leads to certain weaknesses. There might be some misuse cases that will go unnoticed due to being semantic, not quantitative. Any sequence architecture will miss data that was never included in the input.

Finally, the temporal configuration is static, not optimal. Five-day overlapping sequences without regard to the event order within a day are used. No experiment conducted in this study proves that five days is better than one, three, seven, or fourteen. The cutoff value is also tuned using labeled evaluation data even though the Autoencoder training process does not make use of any malicious labels. Another dataset and another organization will need a different validation phase or explicit criteria concerning how many and how costly the alerts they can afford.

Lastly, the proof comes from a single benchmark dataset, a single temporal holdout, and a single neural network run. Replicating the model across different seeds, iterating over the temporal holdouts over several time periods, analyzing the errors by scenarios, and testing on another dataset will all contribute to increased credibility over simply adding layers to the neural network. These limitations do not undermine the stated result; rather, they set the scope of the interpretation.

VIII. CONCLUSION

A benign-only LSTM Autoencoder was trained on 5 days of data generated by CERT r4.2, using logon, device, HTTP, file, and email events. This experiment maintains the temporal ordering of data; the earlier data is used for training, while the later 30% is set aside for testing. The Autoencoder learns to reconstruct benign sequences and uses Mean Absolute Error as its anomaly detection score. At the stated threshold of 0.5709, the results are 0.94 Precision, 0.97 Recall, and 0.96 F1-Score. This demonstrates that a small temporal window contains an anomaly signal even when not explicitly instructing the Autoencoder about any malicious class in the fitting phase.

Both RF and SVM achieve better benchmark scores, but they tackle a more supervised task. Both models are given the labels of the malicious class from the training phase, in addition to SMOTE oversampling. The Autoencoder, on the other hand, is given benign sequences only. Therefore, this comparison provides evidence for two different types of anomaly detection: recognizing known labeled sequences and detecting deviation from a benign sequence. It must be noted that there is no reason these two cannot coexist.

The key point here is methodology in nature. The temporal distance, the use of balancing, the overlap of sequence windows, the level of features' granularity, and the selection of thresholds modify the interpretation of the final score. The subsequent experiments will include running over several seeds, different

lengths of history, the analysis of the errors for each user and scenario, and finally testing on a new data set. At that point, the Transformer, continuous-time, and graph-sequence approaches can be tested against the LSTM model under comparable conditions [24], [26], [27].

ACKNOWLEDGMENT

The authors extend their appreciation to the Deanship of Scientific Research at Northern Border University, Arar, KSA, for funding this research work through the project number “NBU-FFRGS-2026”.

CONFLICT OF INTERESTS

The authors declare that they have no conflicts of interest regarding the present study.

REFERENCES

- [1] L. Liu, O. De Vel, Q.-L. Han, J. Zhang, and Y. Xiang, “Detecting and preventing cyber insider threats: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1397–1417, 2018.
- [2] I. Homoliak, F. Toffalini, J. Guarnizo, J. Elovici, and A. 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.
- [3] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, “Dos and don’ts of machine learning in computer security,” in *Proc. 31st USENIX Security Symposium*, 2022, pp. 3971–3988.
- [4] J. Glasser and B. Lindauer, “Bridging the gap: A pragmatic approach to generating insider threat data,” in *Proc. IEEE Security and Privacy Workshops*, 2013, pp. 98–104.
- [5] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, “LSTM-based encoder-decoder for multi-sensor anomaly detection,” *arXiv preprint arXiv:1607.00148*, 2016.
- [6] P. A. Legg, O. Buckley, M. Goldsmith, and S. Creese, “Automated insider threat detection system using user and role-based profile assessment,” *IEEE Systems Journal*, vol. 11, no. 2, pp. 503–512, 2017.
- [7] A. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.
- [8] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, “TESSERACT: Eliminating experimental bias in malware classification across space and time,” in *Proc. 28th USENIX Security Symposium*, 2019, pp. 729–746.
- [9] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, “Deep learning for unsupervised insider threat detection in structured cybersecurity data streams,” in *AAAI Workshop on Artificial Intelligence for Cyber Security*, 2017, pp. 224–231.
- [10] F. Yuan, Y. Cao, Y. Shang, Y. Liu, J. Tan, and B. Fang, “Insider threat detection with deep neural network,” in *Computational Science–ICCS 2018*, Springer, 2018, pp. 43–54, doi: 10.1007/978-3-319-93698-7_4.
- [11] J. Kim, M. Park, H. Kim, S. Cho, and P. Kang, “Insider threat detection based on user behavior modeling and anomaly detection algorithms,” *Applied Sciences*, vol. 9, no. 19, p. 4018, 2019, doi: 10.3390/app9194018.
- [12] B. Sharma, P. Pokharel, and B. Joshi, “User behavior analytics for anomaly detection using LSTM autoencoder—insider threat detection,” in *Proc. 11th International Conference on Advances in Information Technology (IAIT)*, 2020, pp. 1–9, doi: 10.1145/3406601.3406610.
- [13] S. Yuan and X. Wu, “Deep learning for insider threat detection: Review, challenges and opportunities,” *Computers & Security*, vol. 104, p. 102221, 2021, doi: 10.1016/j.cose.2021.102221.
- [14] A. Kim, J. Oh, J. Ryu, J. Lee, and K. Kwon, “SoK: A systematic review of insider threat detection,” *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 10, no. 4, pp. 46–67, 2019, doi: 10.22667/IOWUA.2019.12.31.046.
- [15] D. C. Le, A. N. Zincir-Heywood, and M. I. 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, doi: 10.1109/TNSM.2020.2967721.
- [16] T. Al-Shehari and R. A. Alsowail, “An insider data leakage detection using one-hot encoding, synthetic minority oversampling and machine learning techniques,” *Entropy*, vol. 23, no. 10, p. 1258, 2021, doi: 10.3390/e23101258.
- [17] D. C. Le and A. N. Zincir-Heywood, “Anomaly detection for insider threats using unsupervised ensembles,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1152–1164, 2021, doi: 10.1109/TNSM.2021.3071928.
- [18] E. Pantelidis, G. Bendiab, S. Shialeles, and N. Kolokotronis, “Insider threat detection using deep autoencoder and variational autoencoder neural networks,” in *Proc. IEEE International Conference on Cyber Security and Resilience (CSR)*, 2021, pp. 129–134, doi: 10.1109/CSR51186.2021.9527925.
- [19] S. Song, N. Gao, Y. Zhang, and C. Ma, “BRITD: Behavior rhythm insider threat detection with time awareness and user adaptation,” *Cybersecurity*, vol. 7, art. 2, 2024, doi: 10.1186/s42400-023-00190-9.
- [20] F. R. Alzaabi and A. Mehmood, “A review of recent advances, challenges, and opportunities in malicious insider threat detection using machine learning methods,” *IEEE Access*, vol. 12, pp. 30907–30927, 2024, doi: 10.1109/ACCESS.2024.3369906.
- [21] M. Villarreal-Vasquez, G. Modelo-Howard, S. Dube, and B. Bhargava, “Hunting for insider threats using LSTM-based anomaly detection,” *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 1, pp. 451–462, 2023, doi: 10.1109/TDSC.2021.3135639.
- [22] P. Pal, P. Chattopadhyay, and M. Swarnkar, “Temporal feature aggregation with attention for insider threat detection from activity logs,” *Expert Systems with Applications*, vol. 224, p. 119925, 2023, doi: 10.1016/j.eswa.2023.119925.
- [23] M. Alslaiman, M. I. Salman, M. M. Saleh, and B. Wang, “Enhancing false negative and positive rates for efficient insider threat detection,” *Computers & Security*, vol. 126, p. 103066, 2023, doi: 10.1016/j.cose.2022.103066.
- [24] M. Elbasheer and A. Akinfaderin, “User-based sequential modeling with Transformer encoders for insider threat detection,” *arXiv preprint arXiv:2506.23446*, 2025.
- [25] A. Mateo-Muñoz, X. Larriva-Novo, C. Sánchez-Zas, M. Álvarez-Campana, and V. A. Villagra, “A session-based insider threat detection approach leveraging the CERT and SPEDIA datasets,” *Machine Learning with Applications*, vol. 25, p. 100946, 2026, doi: 10.1016/j.mlwa.2026.100946.
- [26] M. Jurišić, I. Tomićić, and A. Bernik, “Time is of the essence: A comparative study of continuous (NCDE) and discrete (LSTM) time models for user anomaly detection,” *Signals*, vol. 7, no. 4, p. 59, 2026, doi: 10.3390/signals7040059.
- [27] A. K. Paduvilan, S. A. Babu, J. Sekar, A. K. Elengovan, N. Seshagiri, K. Jangiti, and A. K. Selvam, “Attention-driven fusion of sequential and graph neural models for insider threat detection in UEBA,” *Scientific Reports*, vol. 16, art. 21117, 2026, doi: 10.1038/s41598-026-41242-7.
- [28] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.
- [29] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [30] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, pp. 5–32, 2001.
- [31] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, pp. 273–297, 1995.
- [32] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in *Proc. 8th IEEE International Conference on Data Mining*, 2008, pp. 413–422.
- [33] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Computing Surveys*, vol. 41, no. 3, art. 15, 2009.

- [34] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PLoS ONE*, vol. 10, no. 3, p. e0118432, 2015.
- [35] J. Davis and M. Goadrich, "The relationship between Precision-Recall and ROC curves," in *Proc. 23rd International Conference on Machine Learning*, 2006, pp. 233–240.
- [36] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.
- [37] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
- [38] L. Ruff, R. Vandermeulen, N. Görmitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *Proc. 35th International Conference on Machine Learning*, 2018, pp. 4393–4402.
- [39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [40] B. Lindauer, J. Glasser, M. Rosen, and K. C. Wallnau, "Generating test data for insider threat detectors," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 5, no. 2, pp. 80–94, 2014, doi: 10.22667/JOWUA.2014.06.31.080.