Detecting Anomalous Cybersecurity Events Using LSTM Autoencoders

A Case Study on the BETH Dataset

AUTHOR
Lionel & Killian (Advisor: Dr. Cohen)

PUBLISHED
August 5, 2025

Slides: Slides.html (Go to `Slides.qmd` to edit)

I- Introduction

As modern digital infrastructures become increasingly interconnected and complex, attackers are relying more on process-level intrusions that carefully blend into legitimate activity. Such sophisticated threats often unfolding slowly over time or interleaving malicious commands with normal operations can evade static, signature-based defenses. Traditional systems, which are designed around pre-defined rules or pattern matching, fail to account for subtle changes in temporal context: an action that appears normal in isolation may signal an attack when it occurs in an unusual sequence. Detecting these threats requires models capable of understanding how events unfold over time. This challenge has spurred research toward dynamic, data-driven sequence modeling methods. Among these, Long Short-Term Memory (LSTM) networks have emerged as particularly effective. Introduced by Hochreiter and Schmidhuber (Hochreiter and Schmidhuber 1997), LSTM networks use gated memory cells that retain information across long sequences, effectively overcoming the vanishing gradient problem in traditional recurrent neural networks. This architecture allows LSTMs to capture dependencies spanning dozens or even hundreds of events, making them well-suited for modeling the temporal structure of cybersecurity logs, where the sequence of system calls and process operations is often critical.

The versatility of LSTM networks has been demonstrated across diverse domains. In speech recognition, Graves, Mohamed, and Hinton (Graves, Mohamed, and Hinton 2013) showed that deep LSTM architectures can learn the temporal structure of audio signals, significantly reducing error rates on benchmark tasks. In natural language processing, Sundermeyer et al. (Sundermeyer, Schlüter, and Ney 2012) used LSTMs for language modeling, demonstrating that the model's ability to retain long-range context leads to more accurate text generation compared to traditional n-gram methods. In the clinical domain, Lipton et al. (Lipton et al. 2016) applied LSTM models to patient vitals time series, revealing how memory-enabled models can detect subtle health anomalies as they develop. Across these applications, a core insight emerges: when the data is fundamentally sequential, LSTMs excel at modeling it. This modeling power has translated into cybersecurity research as well. Kim et al. (Kim et al. 2016) applied LSTM-based classifiers to intrusion detection, demonstrating that LSTMs can recognize abnormal sequences of system and network behavior with lower false-positive rates than static feature approaches. Malhotra et al. (Malhotra et al. 2016) advanced this work by introducing an LSTM encoder decoder for anomaly detection. Instead of directly classifying events, the model reconstructs input sequences, identifying anomalies by elevated reconstruction errors an approach especially well suited to unlabeled or novel attacks, as it learns solely from normal behavior. Yin et al. (Yin et al. 2017) extended this concept using recurrent neural networks over raw flow data, achieving improved generalization in network intrusion detection. Building further, Cinque et

al. (Cinque, Della Corte, and Pecchia 2022) introduced Micro2vec, which embeds log message patterns as numeric vectors and leverages LSTM networks to detect anomalies in microservice environments, showing how sequence models can scale to complex, real-world systems.

Yet, progress in these models has often been constrained by outdated benchmarks. Seminal datasets such as KDD Cup 1999 (Hettich and Bay 1999), NSL-KDD (Tavallaee et al. 2009), and ISCX 2012 (Shiravi et al. 2012) provided early test beds for intrusion detection research, but are now hampered by synthetic traffic, unrealistic features, and obsolete system contexts limiting their generalizability. A model trained on these datasets may detect known patterns, but often fails in real-world scenarios where attackers use new commands, timing patterns, or process chains.

In response, Highnam et al. (Highnam et al. 2021) introduced the BETH dataset, marking a significant advance in realism and operational relevance. Instrumented with eBPF across live honeypots, BETH captures over eight million process-level events including system calls, arguments, return values, and timestamps. Importantly, each data instance cleanly separates benign system behavior from adversarial activity, providing labeled, temporally coherent sequences. This makes BETH an ideal platform for training sequence-based anomaly detection models in a semi-supervised setting. Our approach builds directly on this foundation. We deploy an LSTM autoencoder trained exclusively on benign process sequences from BETH. The encoder compresses temporal sequences into fixed-length latent representations, while the decoder attempts to reconstruct them. When the model encounters novel malicious behavior, its reconstruction fails, producing high reconstruction error, this serves as our anomaly signal. This method aligns with recent advances in unsupervised anomaly detection (Gökstorp et al. 2024; Greff et al. 2017), removing the need for prior knowledge of attack types and allowing rapid adaptation to emerging threats.

This work makes three key contributions. First, it demonstrates the effectiveness of LSTM-based sequence modeling for detecting anomalies in real-world, process-level logs. Second, it presents a rigorous evaluation on the BETH dataset, using classification metrics and reconstruction thresholds to quantify detection capabilities. Third, it bridges the gap between advanced deep learning techniques and practical cybersecurity needs, delivering an interpretable, flexible, and deployable detection framework grounded in realistic system behavior.

II- Methods

This section presents the methodological framework for detecting anomalies in cybersecurity event logs using Long Short-Term Memory (LSTM) autoencoders. The approach is organized into six interconnected components and together, these steps form a comprehensive and scalable pipeline for identifying irregularities in large-scale log data

1. Long Short-Term Memory (LSTM) Networks

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) developed by Hochreiter and Schmidhuber (1997) (Hochreiter and Schmidhuber 1997) specifically to overcome the vanishing gradient problem found in traditional RNNs. In cybersecurity applications, the ability to model long-range temporal dependencies is crucial, as malicious activity often unfolds over extended sequences of system calls, network packets, or access attempts.

An LSTM cell maintains two key internal states at each time step: the cell state C_t and the hidden state h_t. The cell state serves as a memory that runs across the sequence, allowing information to persist over time. Three gating mechanisms regulate this process:

- **Forget Gate**: Determines which parts of the previous memory to discard.

  $$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

  where h_{t-1} is the output in state $t-1$, W_f and b_f is the weight matrices and the bias of the forget gate.

- **Input Gate and Candidate Update**: Controls how much new information is added to the memory.

  $$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i), \quad \tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$$

- **Cell State Update**:

  $$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

  where (W_i, b_i) and (W_c, b_c) are the weight matrices and the biases of input gate and memory cell state, respectively

- **Output Gate and Hidden State**: Controls what the cell outputs.

  $$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o), \quad h_t = o_t * \tanh(C_t)$$

  where W_o and b_o are the weight matrix and the bias of output gate, determines a part of the cell state being outputed.

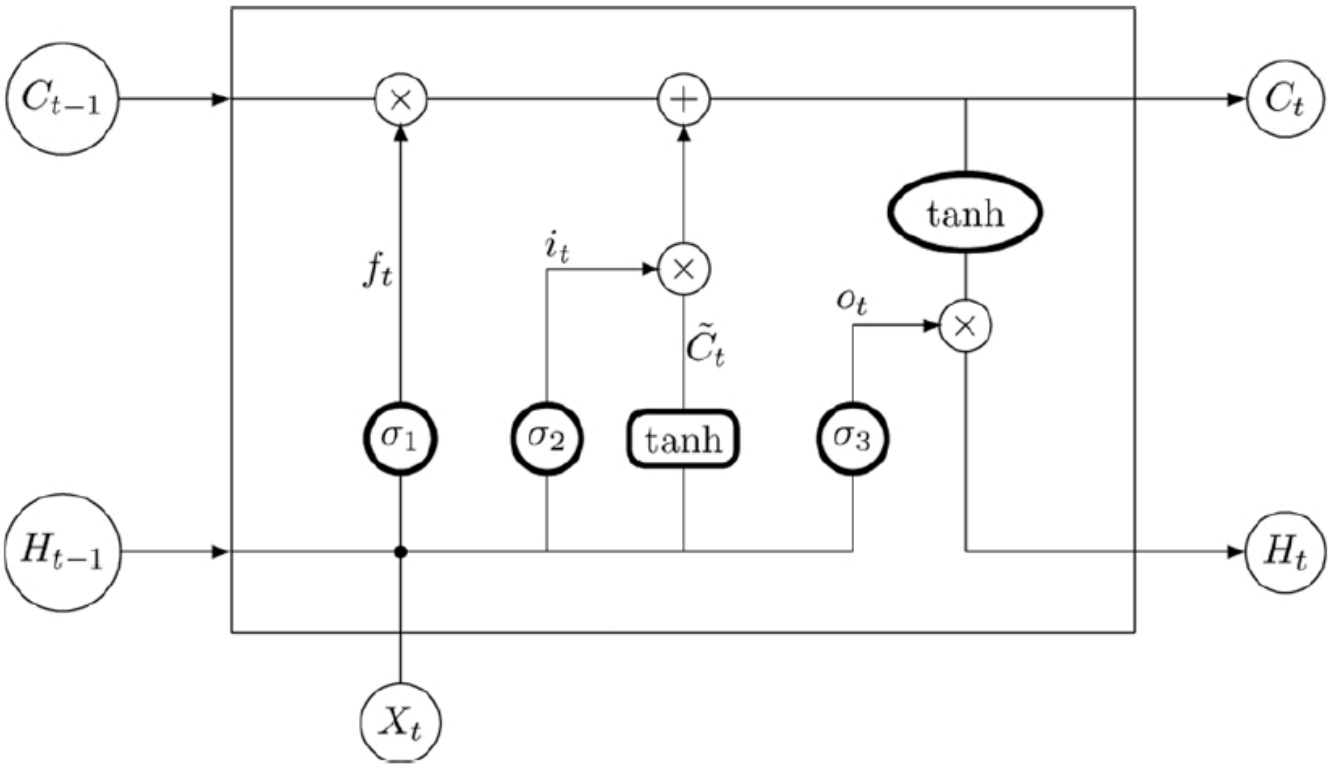


Figure 1: A module of LSTM network (Trinh et al. 2021)

LSTM's memory mechanisms make it especially effective for modeling system behavior over time, where a sequence of seemingly benign events may, when considered together, signal an attack. This makes LSTM a preferred choice for modeling logs in intrusion detection systems (Lipton et al. 2016; Greff et al. 2017; Graves, Mohamed, and Hinton 2013).

2. LSTM Autoencoder for Anomaly Detection

An LSTM autoencoder combines an encoder–decoder architecture with LSTM layers to learn latent representations of temporal data in an unsupervised manner. The encoder compresses input sequences into a fixed-length latent vector, while the decoder attempts to reconstruct the sequence from this vector. If the model is trained exclusively on normal data, it learns to reconstruct familiar sequences accurately. Anomalous sequences, which deviate from learned patterns, yield high reconstruction error a signal of abnormal behavior.

Formally, for an input sequence $x = (x_1, \ldots, x_T)$, the autoencoder is trained to minimize the reconstruction loss:

$$L = \frac{1}{T} \sum_{t=1}^{T} \|x_t - \hat{x}_t\|^2$$

where $\hat{x}_t$ is the output generated by the decoder at each time step.

During inference, this reconstruction error serves as an anomaly score. A sequence with error above a threshold θ is flagged as anomalous. This technique, called reconstruction-based anomaly detection, is

effective when anomalous samples are rare or unknown, which is often the case in cybersecurity contexts.

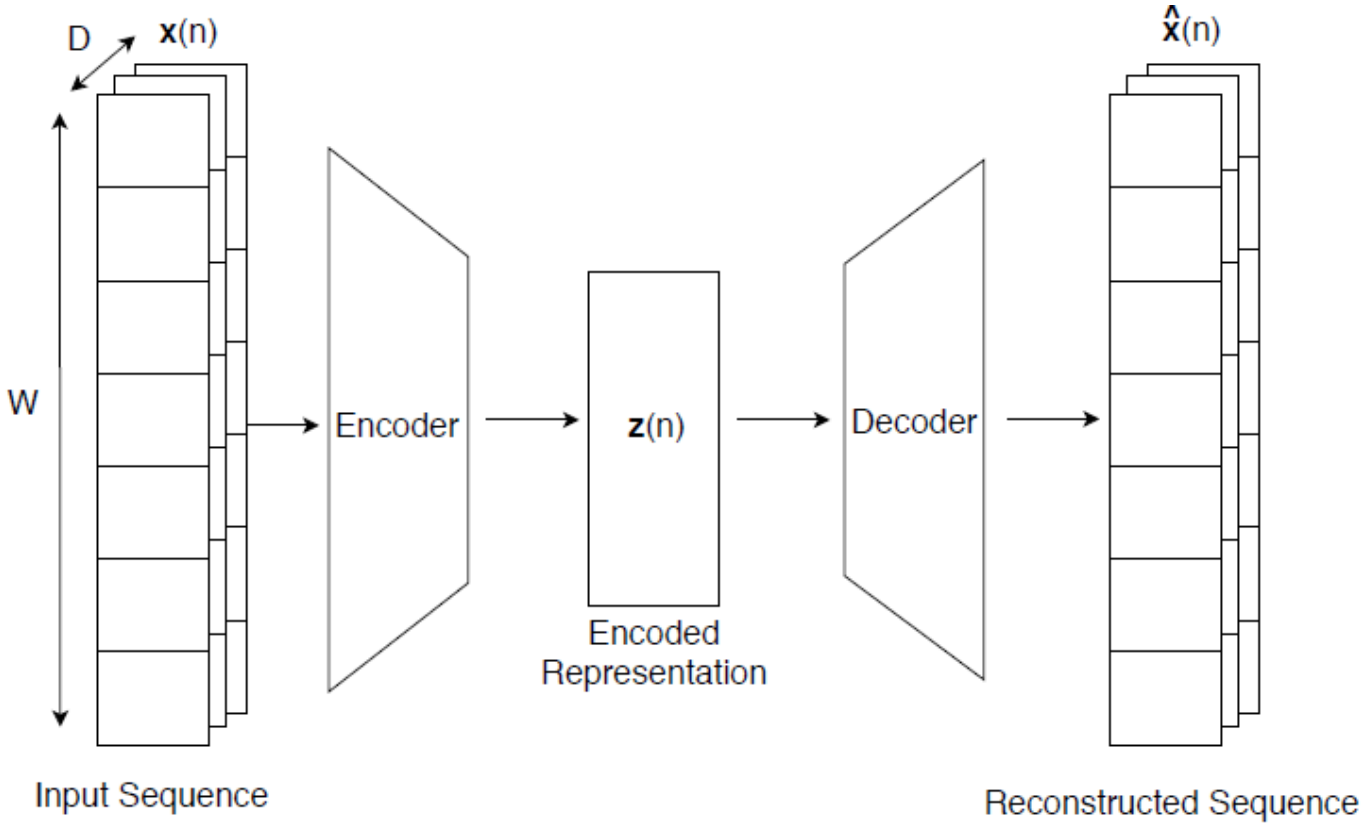


Figure 2: LSTM Autoencoder for Anomaly Detection (Trinh et al. 2021)

Past research has demonstrated the effectiveness of this approach in domains such as multi-sensor fault detection (Malhotra et al. 2016), system log anomaly detection (Gökstorp et al. 2024), and network intrusion detection (Kim et al. 2016; Cinque, Della Corte, and Pecchia 2022). Its unsupervised nature and ability to model complex time-dependent features make it a natural fit for the challenges of modern intrusion detection systems.

3. Anomaly Detection Using LSTM Autoencoders

Once trained on normal data, an LSTM autoencoder becomes a powerful tool for identifying anomalous behavior in sequential inputs such as system logs or network traces. The core idea is that the model becomes highly proficient at reconstructing sequences that follow the familiar temporal patterns it learned during training. However, when presented with sequences that deviate from these learned patterns such as those containing rare, unexpected, or malicious events, the model fails to reconstruct them accurately, producing a noticeably higher reconstruction error.

This reconstruction error serves as a quantifiable measure of anomaly. Specifically, the anomaly score for a given input sequence is computed as the **mean squared error (MSE)** between each input vector x_t and its reconstructed counterpart x_t:

$$\text{Reconstruction Error} = \frac{1}{T}\sum_{t=1}^{T} \|x_t - \hat{x}_t\|^2$$

Here, T is the total number of time steps in the sequence. A higher score indicates greater divergence from normal behavior, and thus, a higher likelihood that the sequence is anomalous.

To operationalize this detection process, a **threshold** is established. This threshold is usually selected by analyzing the distribution of reconstruction errors in the training or validation dataset, often by selecting a statistical percentile or optimizing detection metrics such as the F1-score. Any sequence with a reconstruction error exceeding the threshold is classified as **anomalous**, while sequences with lower errors are assumed to represent **normal activity**.

This approach offers several key benefits in the context of cybersecurity anomaly detection:

- **Unsupervised learning**: There is no need for labeled attack data during training, which is ideal since real-world attack examples are rare, imbalanced, or unknown at the time of deployment.
- **Temporal awareness**: LSTM cells allow the model to consider context across long event sequences—making it effective in identifying subtle but meaningful deviations in system behavior over time.
- **Detection of unknown attacks**: Since the model is not trained on known attack signatures, it generalizes better to novel or zero-day threats, simply by recognizing that "this sequence doesn't look like normal behavior."

The method has been successfully used in a variety of security-critical environments. For instance, Malhotra et al. (Malhotra et al. 2016) used it to detect failures in multi-sensor environments, while Gökstorp et al. (Gökstorp et al. 2024) applied it to analyze security logs. These studies show that LSTM autoencoders can reliably distinguish benign from anomalous sequences, even in high-noise or high-volume contexts.

In this project, we leverage the LSTM autoencoder's ability to detect such behavioral anomalies in the BETH dataset. By training solely on benign log sequences and evaluating reconstruction performance on new data, our model provides a flexible, adaptive, and scalable mechanism for real-time anomaly detection.

4. Justification for Approach and Dataset

Cybersecurity anomaly detection requires models that are flexible, context-aware, and capable of adapting to new or unseen attack patterns. Traditional rule-based or signature-based detection systems are limited by their reliance on predefined patterns, making them ineffective against zero-day attacks or novel behavior.

Unsupervised learning, particularly autoencoder-based methods, allows us to model what "normal" looks like and detect deviations from it, without requiring labeled attacks. This approach is grounded in the notion that abnormal behavior is, by definition, rare and structurally distinct from routine system activity (Yin et al. 2017).

The BETH dataset (Highnam et al. 2021) was selected to evaluate our approach. It includes over 8 million system log events collected from 23 honeypots designed to mimic real-world server environments. Each honeypot was exposed to the internet and captured one targeted attack (at most), alongside a large volume of benign activity. Events are labeled using two binary fields: Sus for suspicious activity and Evil for confirmed malicious events. the table below give a description of our all dataset.

Table1: The description and type of each feature within the DNS logs

Feature	Description
`timestamp`	Date and time When the event occurred (float)
`processId`	ID of the process generating the event
`threadId`	ID of the thread performing the operation
`parentProcessId`	ID of the parent process
`userId`	User running the process/event
`mountNamespace`	Kernel namespace for filesystem isolation
`processName`	Name of the executable or program
`hostName`	Name or IP of the machine
`eventId`	Numeric identifier for the event
`eventName`	Name/type of system call/event
`stackAddresses`	List of memory addresses (call stack)
`argsNum`	Number of arguments for the event
`returnValue`	Return value of the system call/event
`args`	List of arguments (name, type, value)
`sus`	1 if flagged suspicious, 0 otherwise
`evil`	1 if event is malicious, 0 otherwise

The BETH dataset offers several key advantages:
- **Realism:** Data is collected in production-like environments with realistic attack vectors.
- **Anomaly-focused labeling:** The Sus and Evil labels allow fine-grained evaluation of anomaly detection systems.
- **Accessibility:** The dataset is publicly available, enabling reproducibility and peer comparison.

Compared to older datasets like KDD'99 (Tavallaee et al. 2009; Hettich and Bay 1999) or NSL-KDD, BETH reflects modern system-level behavior and attack techniques, providing a more accurate benchmark for current anomaly detection research.

5. Data Preprocessing and Sequence Modeling

▶ Code

```
train_dataset.csv already exists - loading from disk
validation_dataset.csv already exists - loading from disk
```

```
test_dataset.csv already exists - loading from disk
```

To prepare the BETH dataset for LSTM modeling, we applied the following preprocessing steps:

- **Feature extraction:** Key fields such as Timestamp, EventID, SyscallType, ProcessId, ReturnValue, and ArgsNum were selected based on relevance to behavioral analysis.
- **Encoding:** Categorical fields were one-hot encoded to retain discrete information. Numerical fields were normalized to the [0, 1] range using min-max scaling.
- **Sequence generation:** A sliding window of fixed size (e.g., 25–50 events) was used to convert raw logs into overlapping sequences. Each sequence was reshaped into a 3D tensor of shape (Batch × Time × Features), preserving temporal order and structural coherence.

Labeling for sequences was handled as follows:
- Sequences with only benign events were labeled as normal.
- Sequences containing at least one Sus or Evil event were labeled as anomalous.

This approach mirrors real-world monitoring scenarios, where logs are processed in fixed-length windows to detect behavioral anomalies over time.

6. Model Training, Thresholding, and Evaluation

6.1 Training and Architecture

The LSTM autoencoder is trained solely on normal sequences. The architecture includes two stacked LSTM layers in the encoder and decoder, with a fully connected layer at the bottleneck. The decoder mirrors the encoder in structure, and the model is trained to minimize reconstruction loss using the Adam optimizer.

To improve generalization: - Early stopping is used based on validation loss. - A dropout layer is added to mitigate overfitting. - 5-fold cross-validation is used to assess the stability of the model across different subsets of the benign data.

Hyperparameters (e.g., learning rate, batch size, window size, latent vector size) were selected based on preliminary tuning and aligned with best practices from related work (Nguyen et al. 2021; Malhotra et al. 2016).

6.2 Threshold Selection

After training, we determine an anomaly threshold θ by analyzing reconstruction errors on a labeled validation set. Multiple strategies are considered:

- Using a fixed percentile of training error distribution (e.g., 95th percentile)
- Optimizing the F1-score on validation anomalies
- Minimizing false positive and false negative rates jointly

The selected threshold is applied to test sequences during evaluation.

6.3 Evaluation Metrics

The model's performance is evaluated using the following metrics:

- **Precision:** The proportion of predicted anomalies that are true anomalies.
- **Recall:** The proportion of actual anomalies that are correctly identified.
- **F1-score:** The harmonic mean of precision and recall.
- **ROC-AUC:** Measures the trade-off between true positive and false positive rates.
- **Confusion Matrix:** Provides a granular view of model classification results.
- **Error distribution plots:** Visualize separation between normal and anomalous samples.

These metrics provide a robust, multidimensional assessment of the model's effectiveness in identifying anomalous behavior.

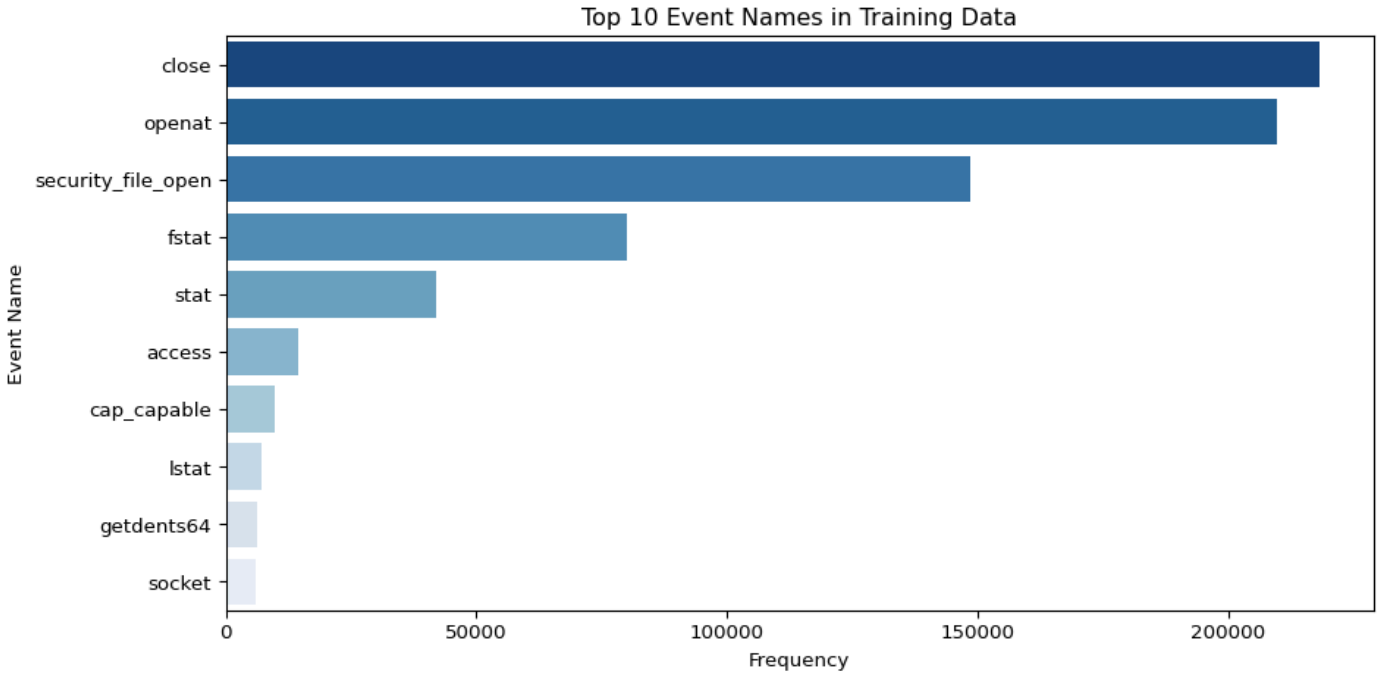
III- Analysis and Results

1- Data Exploration and Visualization

To better understand the structure and composition of the BETH dataset, we begin by analyzing the process logs that serve as input to our LSTM-based anomaly detection model. The dataset comprises a total of 8,004,918 system events, among which a benchmark subset of approximately 763,145 records is reserved for training. The dataset is rich in temporal and categorical information, including process and thread identifiers, process names, system call arguments, return values, and two binary labels: sus (suspicious) and evil (malicious). The logs are chronologically ordered and contain a temporal feature (timestamp) suitable for sequential modeling. Each record represents an individual system event collected from one of 23 cloud-based honeypots. The data was split into training, validation, and test sets using a 60/20/20 ratio, where only the test set contains attack sequences. A descriptive analysis of the features reveals the heterogeneous nature of the dataset. The eventName field, for example, exhibits a long-tailed distribution, with certain events (e.g., execve, read, openat) appearing frequently, while others are rare and potentially indicative of anomalous behavior. The processName and hostName attributes also show high cardinality, which suggests the need for careful encoding strategies when preparing the data for modeling. An initial examination of the BETH training and test subsets reveals several insights into the nature of system-level activity across cloud-hosted honeypots. These observations have guided the formulation of our LSTM-based anomaly detection model.

1.1 Event Frequency Distribution.

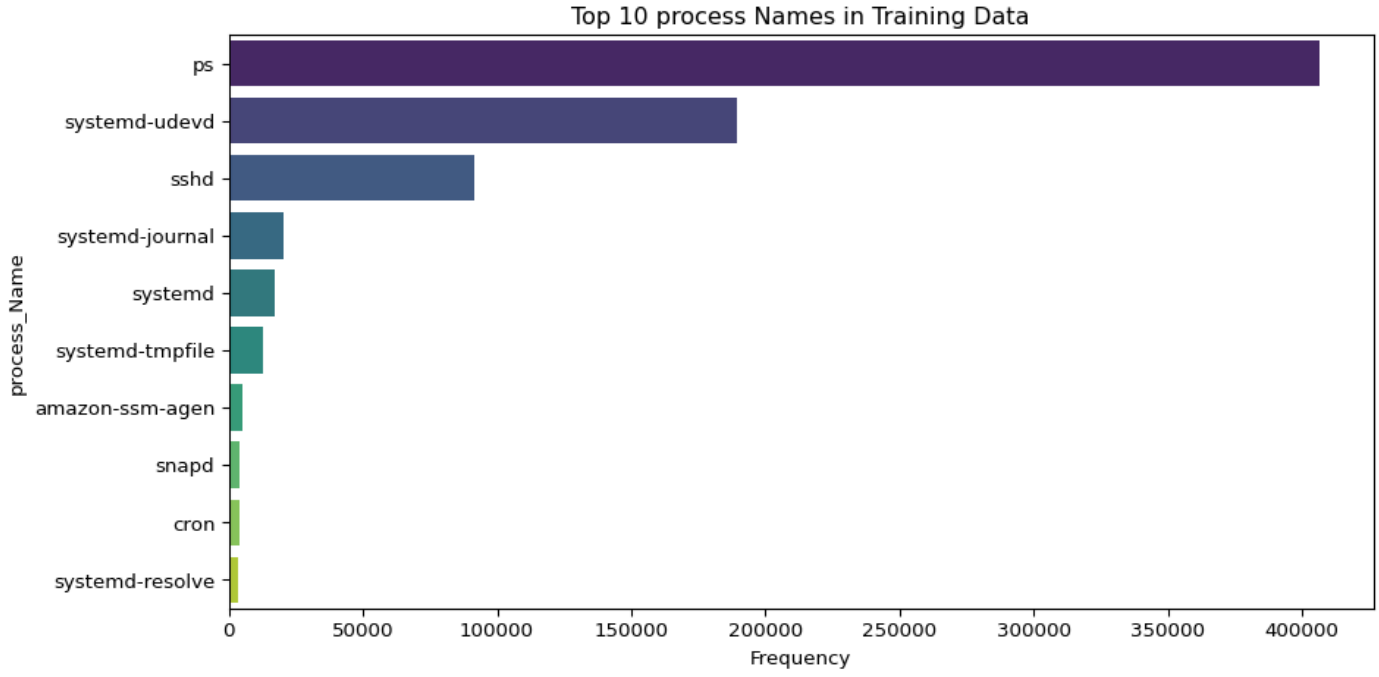
▶ Code



As shown in Figure 3, the distribution of system events in the training data is highly skewed. The top three event types close, openat, and security_file_open collectively dominate the logs, each appearing over 150,000 times. In contrast, lower-frequency events such as getdents64 and socket are comparatively rare. This long-tailed distribution suggests that certain system calls represent routine behavior, while others though rare may hold critical information for detecting anomalies. Therefore, proper handling of categorical sparsity and rare event embedding is essential during preprocessing.

1.2 Process Behavior

▶ Code



A similar distribution is observed in processName. A small set of background processes account for a significant fraction of system activity. These are expected in benign operational scenarios. However, malicious processes are often less frequent and unpredictable. This motivates the need for a temporal modeling approach that generalizes from dominant patterns while remaining sensitive to rare deviations.

1.3 Temporal Skew in Data Collection

▶ Code

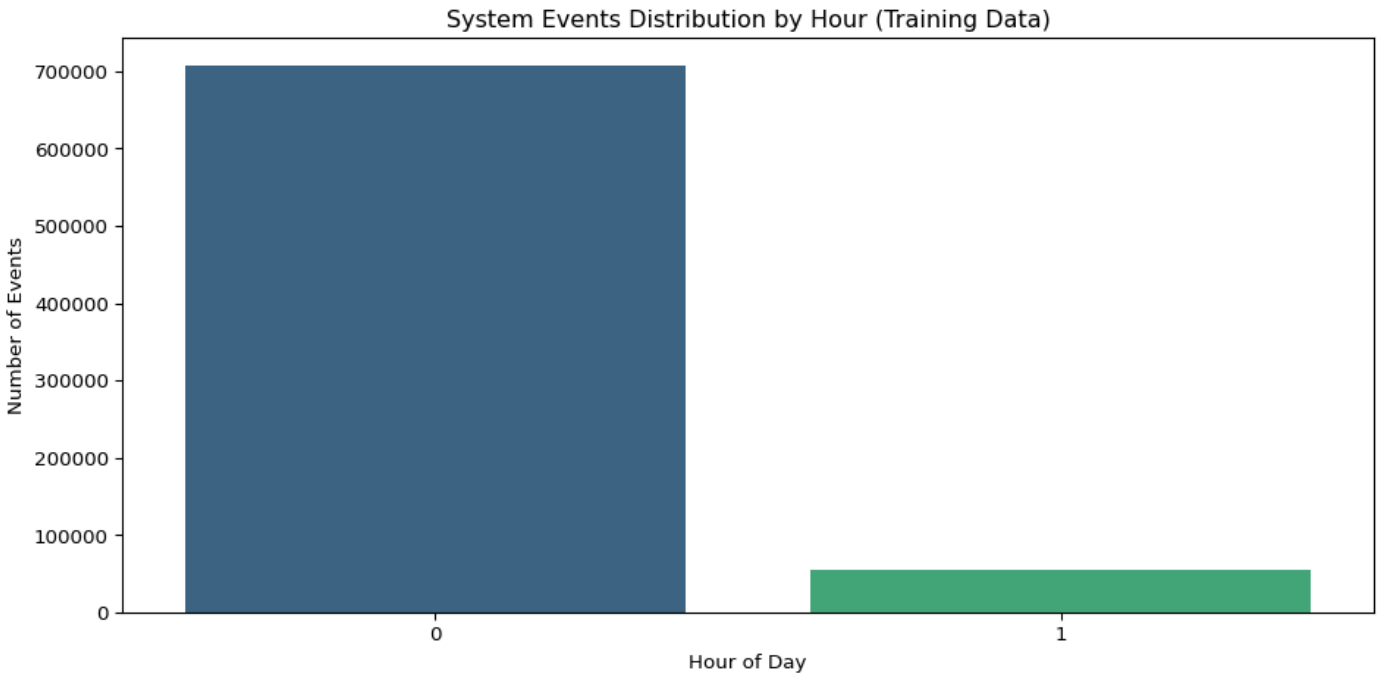
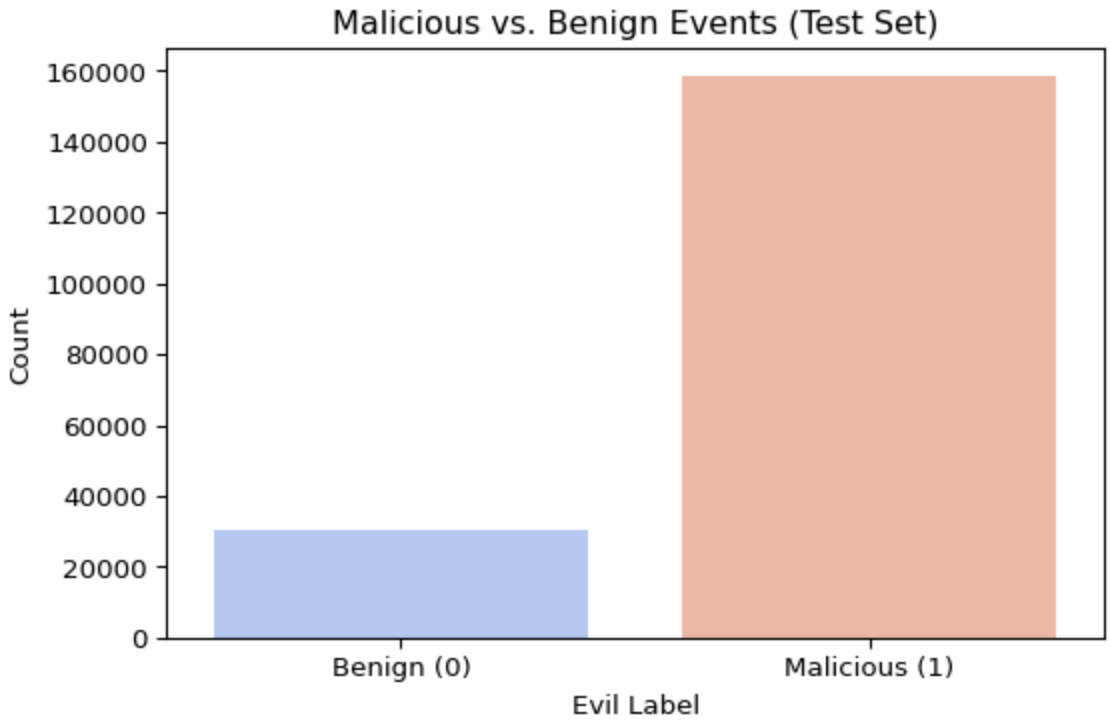


Figure 4 reveals a pronounced imbalance in the distribution of events across time. The vast majority of events in the training set are concentrated within a single hour of activity, with very limited representation beyond that point. This skew may reflect batch-mode logging or the constrained operational window of honeypot collection. As a result, the model must be designed to generalize temporal patterns from narrow windows of observed behavior, further motivating the use of memory-aware architectures like LSTM.

1.4 Label Imbalance and Anomaly Prevalence

▶ Code



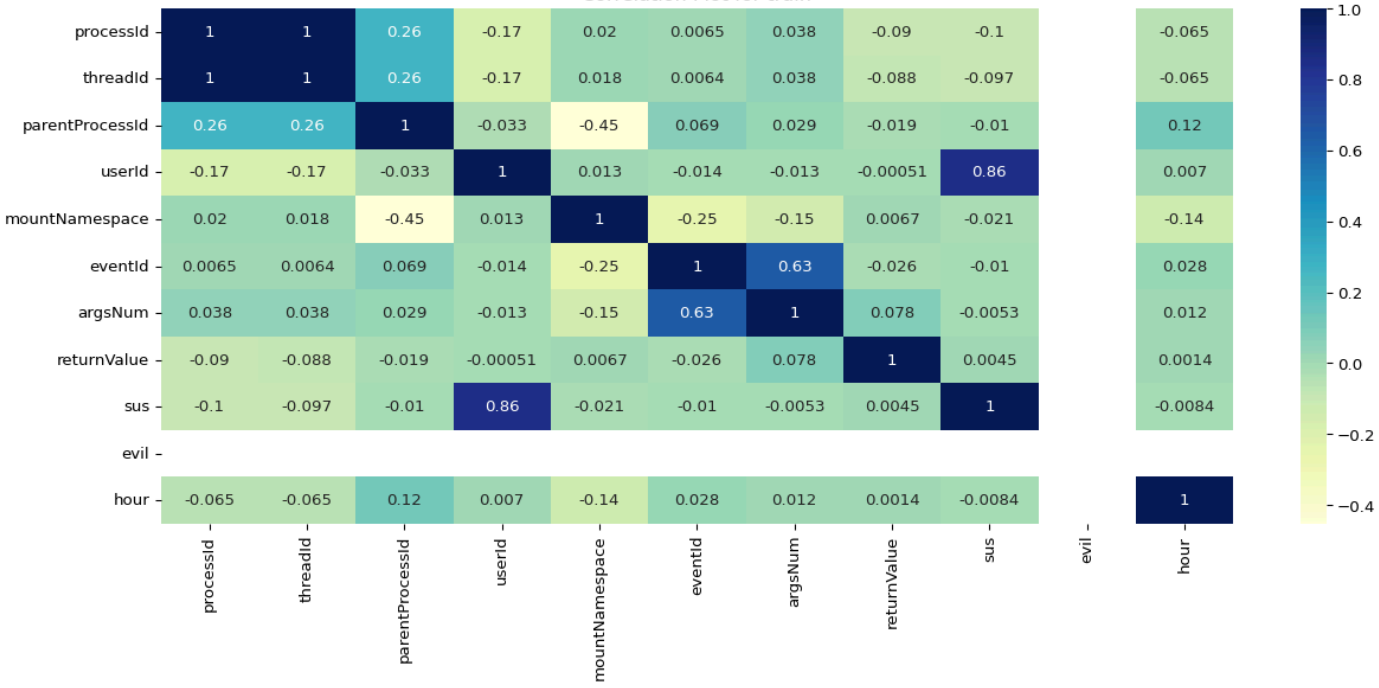
Contrary to initial expectations, Figure 4 indicates that a substantial portion of the test data is labeled as malicious (evil = 1), with malicious events significantly outnumbering benign ones. This suggests that the dataset may contain attack-heavy test sequences designed for stress-testing anomaly detection models. This observation presents both a challenge and an opportunity: while class imbalance is often a barrier in anomaly detection, the abundance of labeled attacks in the test set allows for richer evaluation of model performance using standard classification metrics such as precision, recall, and F1-score.

1.4 Correlation

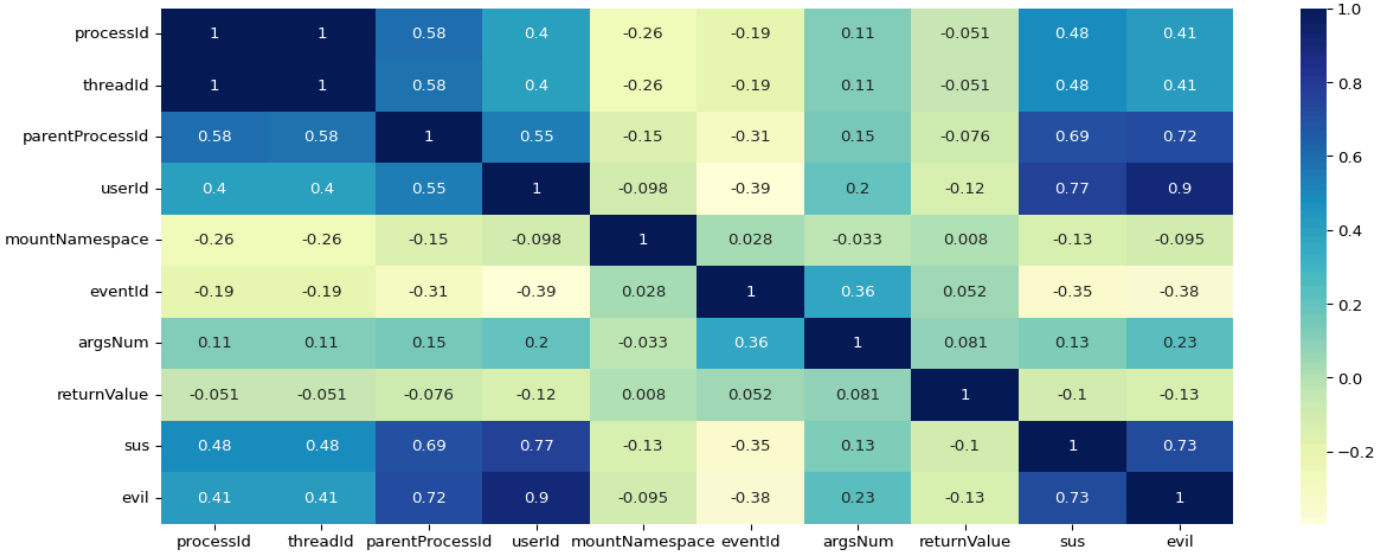
▶ Code

▶ Code

Correlation Plot for train



Correlation Plot for test

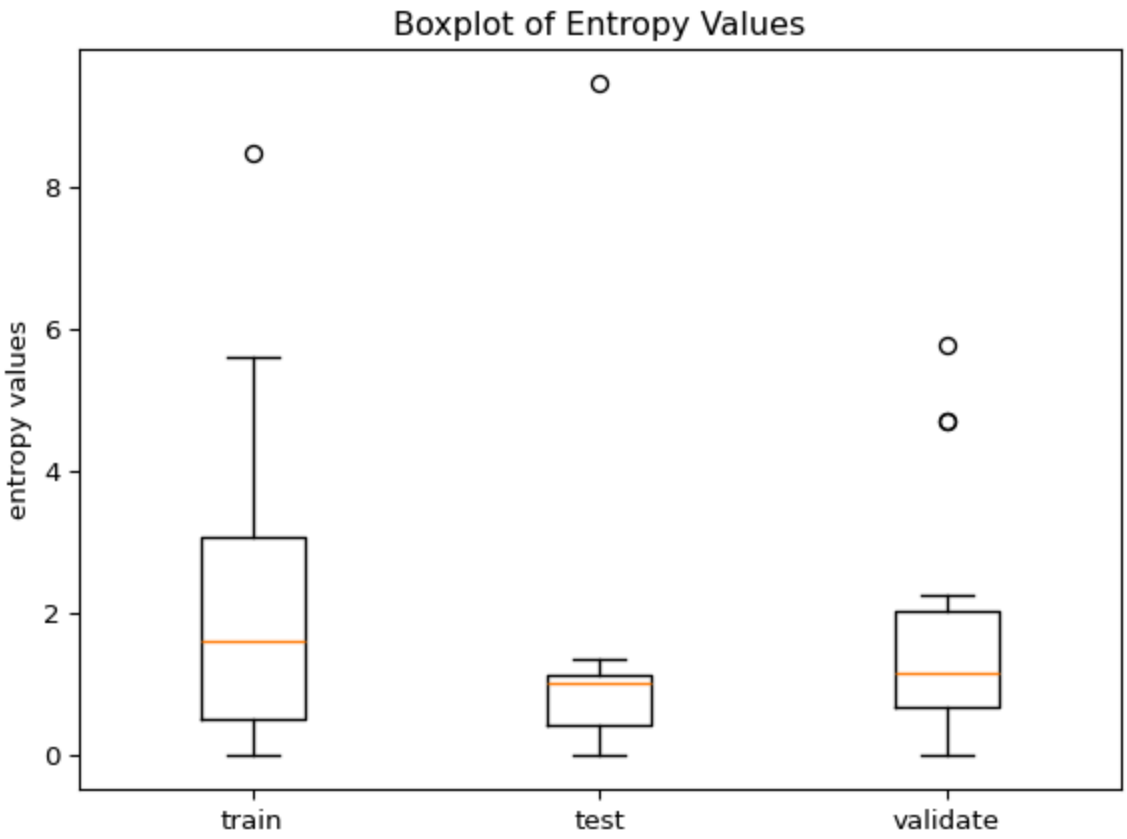


Correlation Plot for validation

All three of the datasets have a heavy correlation between userId and the associated labels which feature in the dataset (sus and/or evil). processid and threadid are highly correlated and seem to have similar correlation vaules across all three datasets. This means that they are representing pretty much the same thing and one of them could be dropped. The correlation plots all look significantly different.

1.4 Boxplot

▶ Code



Train set:

- Median entropy is around ~1.5, higher than both test and validation sets.
- Wider interquartile range (IQR), indicating more variability in entropy within the training data.
- Several high outliers (entropy > 6), suggesting some sequences in the training data are significantly more “disordered” or unpredictable than the rest.

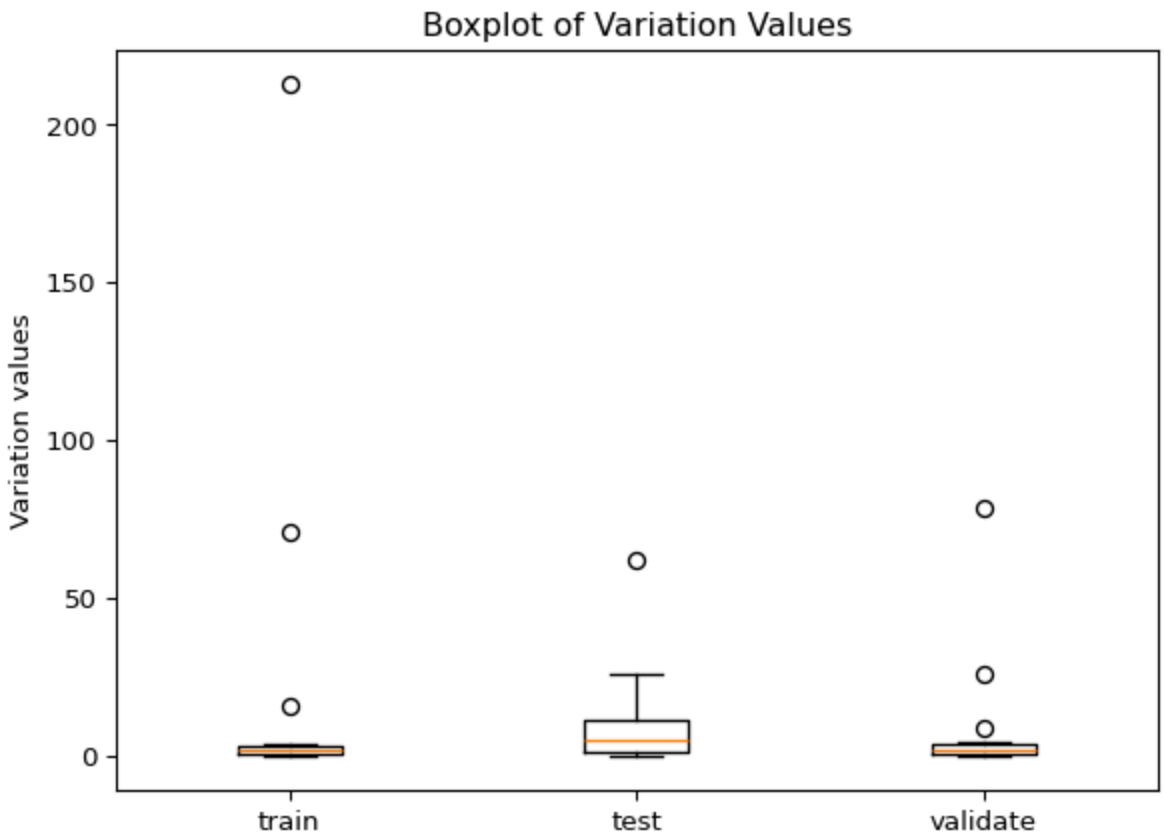
Test set: - Median entropy is low (~1), with a narrow IQR. - Distribution is more compact, meaning less variability in entropy. - Only a few outlines, but one very high outlier (> 8).

Validation set: - Median is slightly higher than the test set (~1.2–1.3), but still lower than the training set. - Slightly wider spread than the test set. - Some high outliers (~5–6).

Interpretation:

The training set has more diverse patterns of entropy, which may indicate it includes a wider range of sequence complexities compared to test and validation sets. This could reflect richer patterns in training data or possible inconsistencies in data collection.

▶ Code



Train set: - Median variation is near zero, suggesting most sequences have low variability. - Very small IQR — most values are tightly clustered. - Multiple extreme outliers, including very high values (>200).

Test set: - Higher median variation compared to the train set. - Slightly larger IQR than training set. - Several extreme outliers, though not as large as the biggest training outlier.

Validation set: - Similar to training set in median (near zero) and small spread. - Fewer extreme outliers than train and test sets.

Interpretation: Variation is generally low across all datasets, meaning that within most sequences, values are relatively stable. The high outliers represent occasional bursts of high variability, which might correspond to anomalous or attack-related sequences.

Overall in the boxplots, the training set appears more heterogeneous in both entropy and variation, suggesting it captures a broader range of system behaviors. Plus Test and validation sets are more compact but still contain significant outliers, which may correspond to rare but important events for anomaly detection. The extreme outliers in both metrics could be strong indicators of anomalies or unusual system states, potentially useful for your LSTM model to learn separation between normal and abnormal patterns.

1.5 Implications for Modeling

Taken together, these findings justify our modeling decisions in several ways: The long-tailed event distribution calls for embedding-based representations of categorical variables (e.g., eventName, processName).

The temporal compression of training data highlights the need for architectures that generalize across short but informative sequences.

The label distribution in the test set supports a semi-supervised evaluation protocol, where a model trained on benign data is evaluated on its ability to flag deviations (i.e., attacks) as anomalous.

2- Modeling and Results

2.1 Model Architecture

The architecture of our anomaly detection system is based on an LSTM autoencoder trained exclusively on benign process sequences. The input to the model is a sliding window of look_back=8 consecutive events, each described by 9 features. The encoder consists of two LSTM layers with 128 and 64 hidden units, respectively, followed by dropout layers to prevent overfitting. The latent representation is repeated and passed to a symmetric decoder consisting of two LSTM layers that reconstruct the input sequence. The model is trained to minimize the Mean Absolute Error (MAE) between the original and reconstructed sequences.

▶ Code

▶ Code

```
X_train shape: (763137, 8, 9)
X_val shape: (188960, 8, 9)
X_test shape: (188960, 8, 9)
y_test distribution: [ 30528 158432]
```

▶ Code

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 8, 9)	0
lstm (LSTM)	(None, 8, 128)	70,656
dropout (Dropout)	(None, 8, 128)	0
lstm_1 (LSTM)	(None, 64)	49,408
dropout_1 (Dropout)	(None, 64)	0
repeat_vector (RepeatVector)	(None, 8, 64)	0
lstm_2 (LSTM)	(None, 8, 64)	33,024
dropout_2 (Dropout)	(None, 8, 64)	0
lstm_3 (LSTM)	(None, 8, 128)	98,816

dropout_3 (Dropout)	(None, 8, 128)	0
time_distributed (TimeDistributed)	(None, 8, 9)	1,161

Total params: 253,065 (988.54 KB)

Trainable params: 253,065 (988.54 KB)

Non-trainable params: 0 (0.00 B)

The final input tensor shape is (batch_size, 8, 9), corresponding to 8 time steps and 9 features. The model has approximately 1.5 million trainable parameters.

2.2 Training and Validation Performance

We first evaluate the generalization capability of the model using a 5-fold time-series cross-validation performed on benign data only. For each fold, the model is trained on a different temporal split and evaluated on future sequences to simulate realistic detection scenarios.

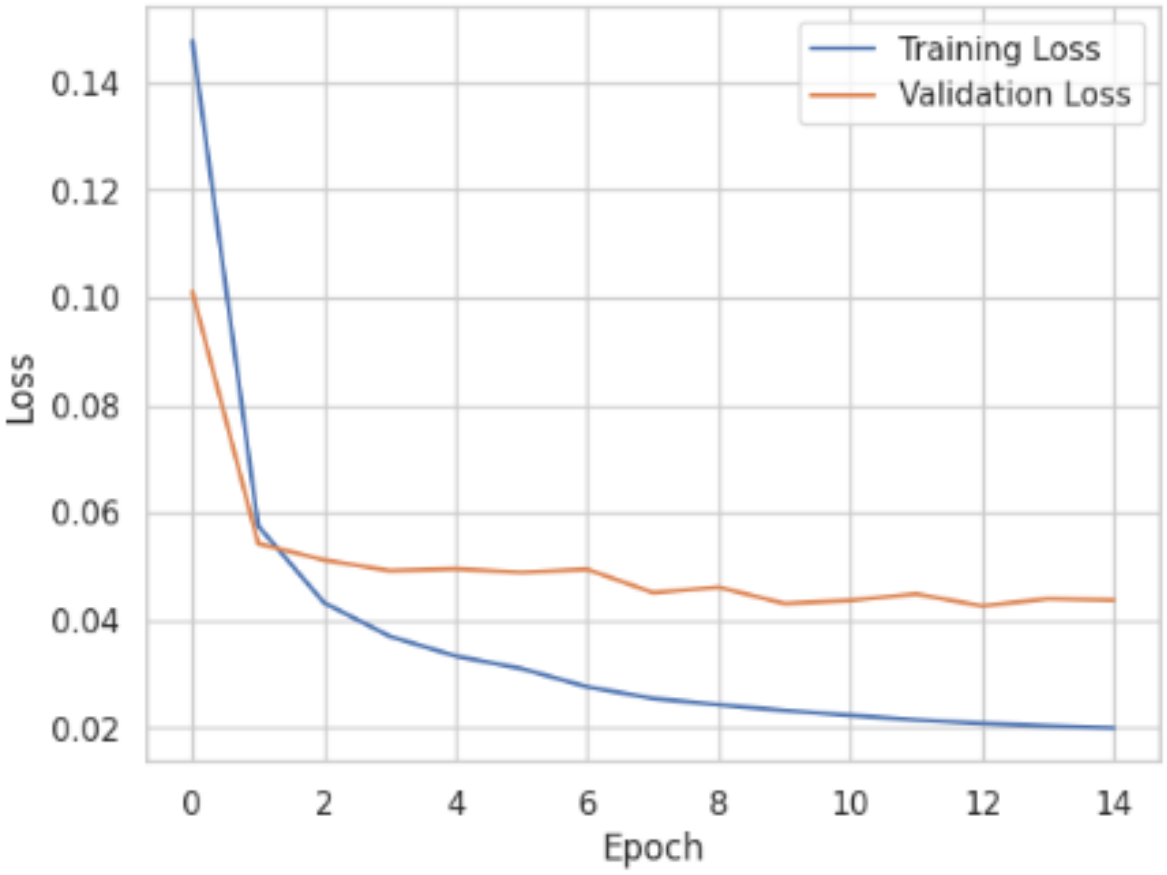


Figure 7: Training / validation Loss result

The average validation loss across folds was: 0.142291 (+/-0.128335). After cross-validation, the model was retrained on the full benign training set. The final training and validation losses indicate stable convergence without overfitting as we can see on the figure above

2.3 Anomaly Detection Threshold

The trained model was then used to compute reconstruction errors over the test set. The distribution of reconstruction errors clearly shows a distinguishable separation between normal and anomalous sequences

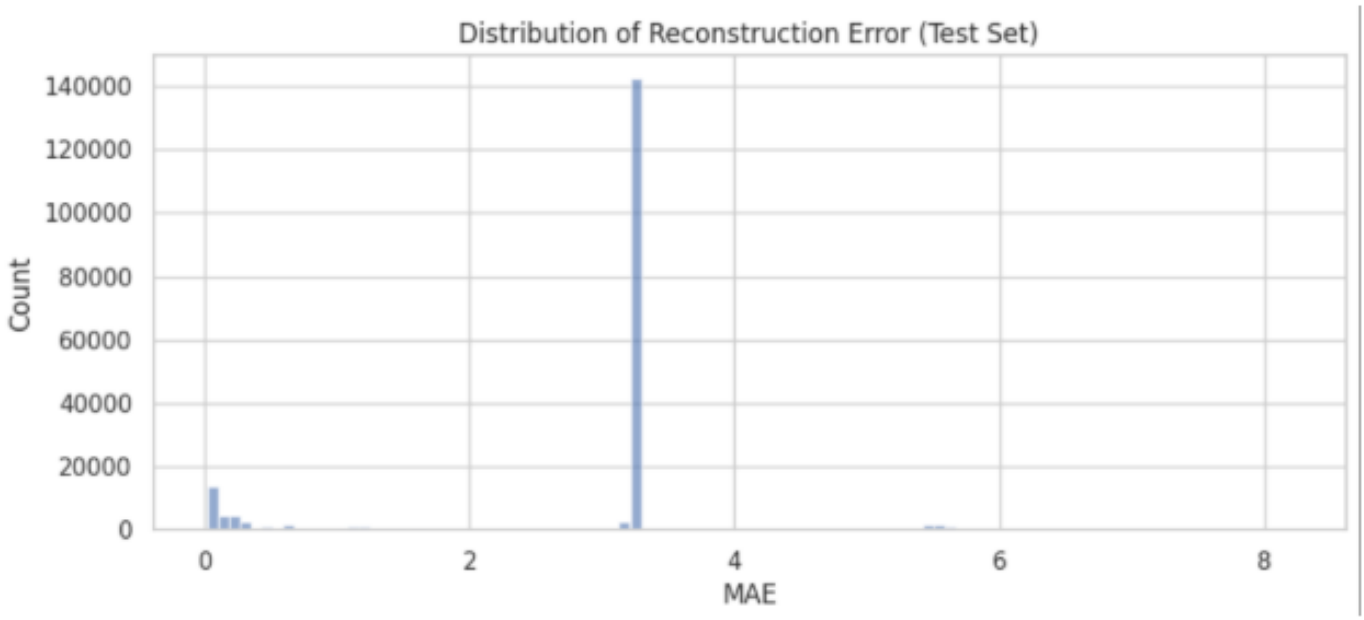


Figure 8: Histogram of test reconstruction errors (MAE)

To convert reconstruction error into binary predictions, we used the precision-recall curve to find the optimal threshold that maximizes the F1-score: 0.9549.

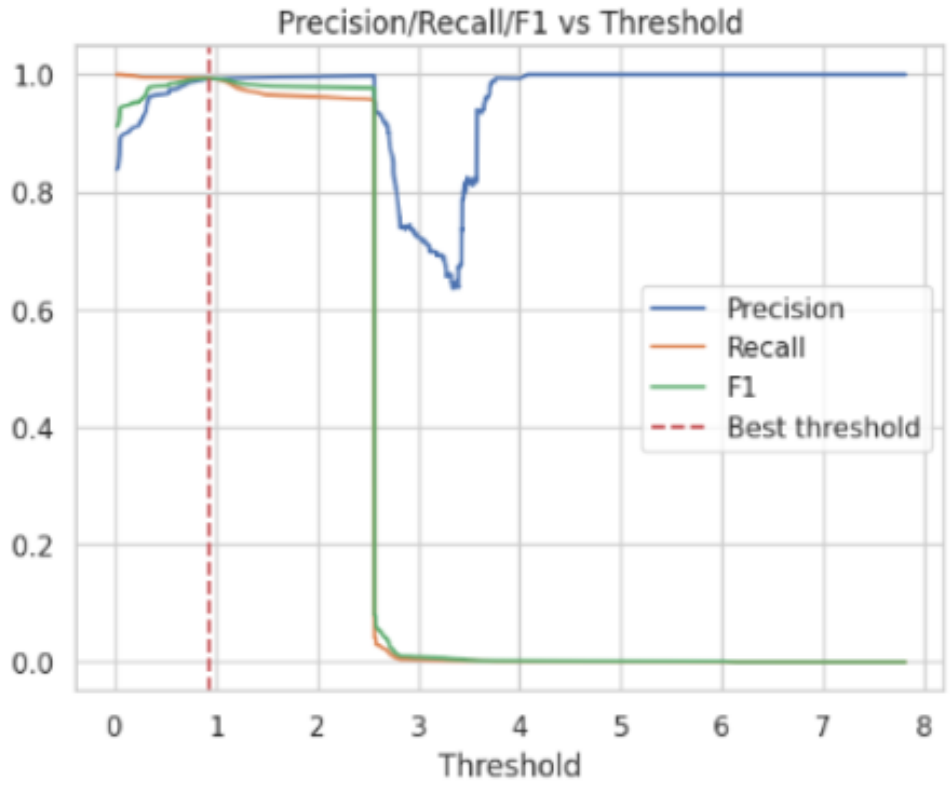


Figure 9: Precision Recall, F1 vs threshold plot

2.4 Classification Metrics

Once the best threshold was determined, we applied it to the test set. The model successfully identified anomalous sequences with high accuracy of 99%. The confusion matrix below shows that most benign and malicious events are correctly classified.

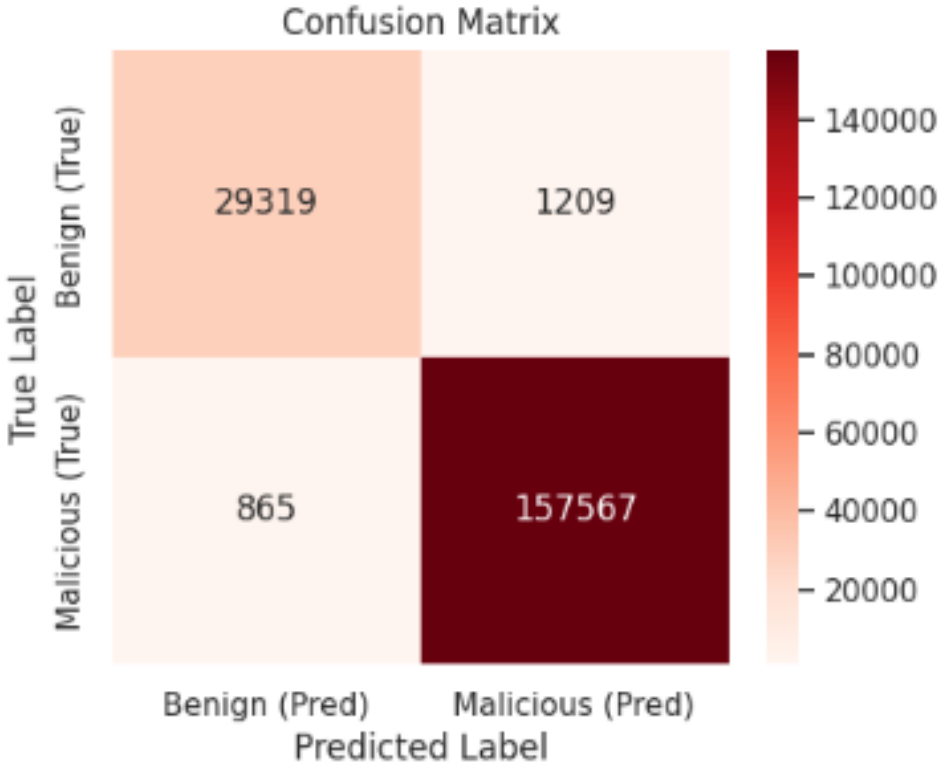


Figure 10: Confusion matrix for final predictions

- False Positive Rate: 0.040
- False Negative Rate: 0.005

The confusion matrix highlights the model's strength in correctly identifying malicious events (true positives), while maintaining a relatively low rate of false positives. Misclassifications were analyzed and often linked to sequences with novel process or host identifiers not seen during training further confirming the model's ability to flag unfamiliar temporal patterns.

The high recall suggests the model captures almost all malicious behavior, which is critical for a cybersecurity context. The small false positive rate (3.4%) indicates that few benign sequences are misclassified, making this model practical for real-time deployment where false alarms are costly. The ROC curve and AUC value confirm the model's excellent discriminatory power between normal and malicious activity.

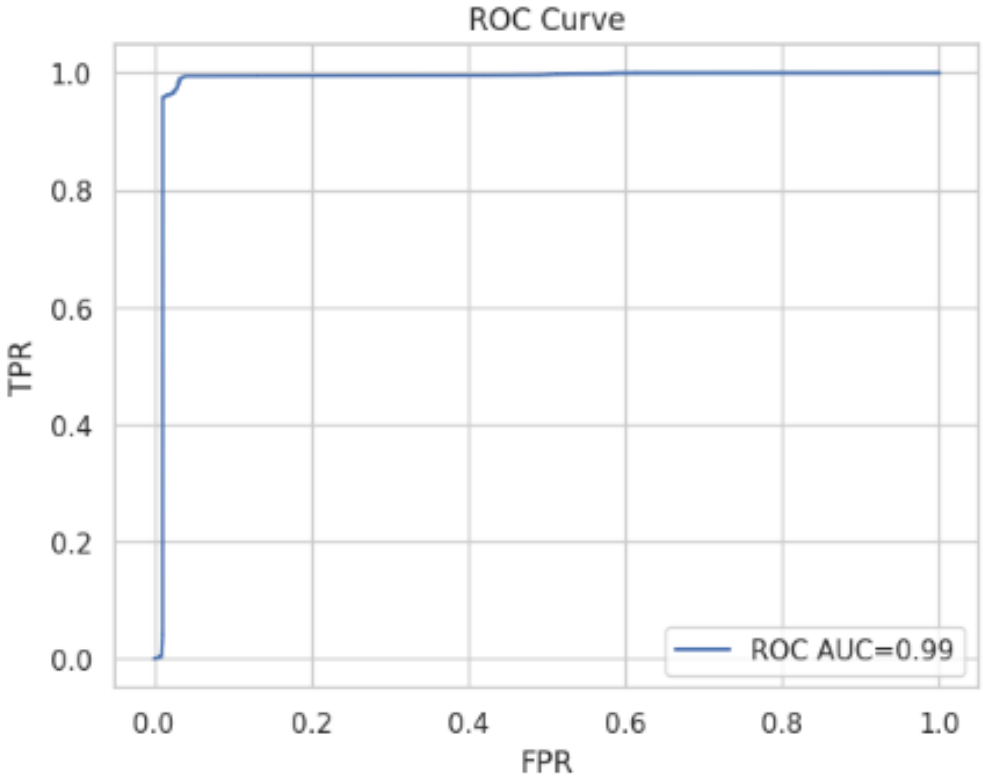


Figure 11: ROC Cuve

2.5 Overall Evaluation and Future Considerations

The LSTM autoencoder exhibited strong performance in terms of: Learning a stable representation of normal sequences. Flagging novel and anomalous patterns through unsupervised learning. Achieving a high F1-score despite class imbalance. Aligning with state-of-the-art performance in similar studies (Highnam et al. 2021; Kim et al. 2016). For future deployment, periodic retraining is recommended to adapt to system evolution. Additionally, incorporating attention mechanisms or hybrid ensembles may further enhance sensitivity to subtle anomalies.

IV-Conclusion

This project has demonstrated the substantial potential of LSTM autoencoders for detecting anomalous cybersecurity events in real world, process level system logs. By leveraging the BETH dataset one of the most realistic and contemporary benchmarks for intrusion detection, we were able to rigorously assess the effectiveness of deep sequence modeling for practical cyber defense. Our approach, grounded in unsupervised learning, involved training the LSTM autoencoder exclusively on benign sequences to learn the complex temporal dependencies that characterize normal system behavior. The model's ability to generalize from limited windows of typical activity allowed it to sensitively identify even subtle deviations linked to novel or sophisticated attacks. Through careful feature engineering, one-hot encoding, and sequence modeling, we prepared heterogeneous system logs for robust anomaly detection.

The evaluation of our model was comprehensive and multidimensional. Using metrics such as precision, recall, F1-score, ROC-AUC, and confusion matrices, we showed that the LSTM autoencoder achieved high accuracy (99%) and a strong ability to discriminate between benign and malicious activity, even in the face

of class imbalance and temporal data skew. The model's low false positive and false negative rates affirm its practicality for real-time deployment, where both missed attacks and false alarms carry significant costs.

A key strength of this work lies in its adaptability. Because the method is unsupervised and does not rely on prior knowledge of attack signatures, it remains robust against previously unseen or zero-day threats. This property, combined with rigorous threshold tuning and cross-validation, positions the approach as a promising candidate for deployment in dynamic, evolving cybersecurity environments.

Limitations and Future Work: While the results are compelling, some limitations remain. The dataset's temporal compression and event imbalance, as well as potential overfitting to benign process patterns, suggest the need for ongoing retraining as system behaviors evolve. Future enhancements could include integrating attention mechanisms, hybrid ensemble methods, or continual learning strategies to further improve detection sensitivity and adaptability.

In summary, this project bridges advanced deep learning research with operational cybersecurity needs. By demonstrating that LSTM autoencoders can reliably flag anomalies in complex system logs without extensive labeled attack data, we provide a practical, scalable, and interpretable framework for next-generation intrusion detection. Our findings underscore the critical role of sequence-aware models in defending modern digital infrastructures against a rapidly evolving threat landscapes

---

References

Cinque, Marcello, Raffaele Della Corte, and Antonio Pecchia. 2022. "Micro2vec: Anomaly Detection in Microservices Systems by Mining Numeric Representations of Computer Logs." *Journal of Network and Computer Applications* 208: 103515. https://doi.org/https://doi.org/10.1016/j.jnca.2022.103515.

Gökstorp, Simon, Johan Nyberg, Yong Kim, Pontus Johnson, and Gábor Dán. 2024. "Anomaly Detection in Security Logs Using Sequence Modeling." In *NOMS 2024-2024 IEEE Network Operations and Management Symposium*, 1–9. https://doi.org/10.1109/NOMS59830.2024.10575561.

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. "Speech Recognition with Deep Recurrent Neural Networks." In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. https://doi.org/10.1109/ICASSP.2013.6638947.

Greff, Klaus, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2017. "LSTM: A Search Space Odyssey." *IEEE Transactions on Neural Networks and Learning Systems* 28 (10): 2222–32. https://doi.org/10.1109/TNNLS.2016.2582924.

Hettich, S, and SD Bay. 1999. "The UCI KDD Archive." University of California, Irvine. http://kdd.ics.uci.edu.

Highnam, Kate, Kai Arulkumaran, Zachary Hanif, and Nicholas R Jennings. 2021. "BETH Dataset: Real Cybersecurity Data for Anomaly Detection Research." In *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 3095:1–12. https://ceur-ws.org/Vol-3095/paper1.pdf.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. "Long Short-Term Memory." *Neural Computation* 9 (8): 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.

Kim, Jihyun, Jaehyun Kim, Huong Le Thi Thu, and Howon Kim. 2016. "Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection." *International Journal of Advanced Computer Science and Applications* 7 (6). https://doi.org/10.1109/PlatCon.2016.7456805.

Lipton, Zachary C, David C Kale, Charles Elkan, and Randall Wetzell. 2016. "Learning to Diagnose with LSTM Recurrent Neural Networks." *arXiv Preprint arXiv:1511.03677*. https://www.researchgate.net/publication/283761671_Learning_to_Diagnose_with_LSTM_Recurrent_Neural_Networks.

Malhotra, Pankaj, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. 2016. "LSTM-Based Encoder-Decoder for Multi-Sensor Anomaly Detection." *CoRR* abs/1607.00148. http://arxiv.org/abs/1607.00148.

Nguyen, H. D., K. P. Tran, S. Thomassey, and M. Hamad. 2021. "Forecasting and Anomaly Detection Approaches Using LSTM and LSTM Autoencoder Techniques with the Applications in Supply Chain Management." *International Journal of Information Management* 57: 102282. https://www.sciencedirect.com/science/article/abs/pii/S026840122031481X.

Shiravi, Ali, Hadi Shiravi, Mahbod Tavallaee, and Ali A Ghorbani. 2012. "Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection." *Computers & Security* 31 (3): 357–74. https://doi.org/10.1016/j.cose.2011.12.012.

Sundermeyer, Martin, Ralf Schlüter, and Hermann Ney. 2012. "LSTM Neural Networks for Language Modeling." In *Interspeech*. https://doi.org/10.21437/Interspeech.2012-65.

Tavallaee, Mahbod, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. 2009. "A Detailed Analysis of the KDD Cup 99 Dataset." *IEEE Symposium. Computational Intelligence for Security and Defense Applications, CISDA* 2. https://doi.org/10.1109/CISDA.2009.5356528.

Trinh, Hoang Duy, Engin Zeydan, Lorenza Giupponi, and Paolo Dini. 2021. "Detecting Mobile Traffic Anomalies Through Physical Control Channel Fingerprinting: A Deep Semi-Supervised Approach." *IEEE Access* 9: 82564–80. https://doi.org/10.1109/ACCESS.2021.3087287.

Yin, Chuanlong, Yuefei Zhu, Jinlong Fei, and Xinzheng He. 2017. "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks." *IEEE Access* 5: 21954–61. https://doi.org/10.1109/ACCESS.2017.2762418.