# Network Security with LSTM-powered Irregularity Detection

**Author 1**
Laasya Vajjala
11848603

**Author 2**
Vadapalli Surya
11861342

## Contents

# Abstract

This project addresses the critical challenge of anomaly detection in network traffic, motivated by the escalating cybersecurity threats in digital ecosystems. With the stream of sophisticated attacks, such as distributed denial-of-service (DDoS) assaults and advanced persistent threats (APTs), conventional methods struggle to effectively discern inconsistent patterns. Leveraging Long Short-Term Memory (LSTM) deep learning, our project aims to enhance anomaly detection accuracy by capturing intricate temporal dependencies within network traffic data. The unique advantage lies in the model's ability to discern subtle deviations from normal actions, providing robust protection against unusual and evolving threats. The project involves preprocessing raw network data, training LSTM models, and validating their efficacy in distinguishing irregularities. Additionally, we plan to optimize model parameters to ensure real-time applicability and scalability, fostering a proactive approach to cybersecurity in dynamically changing environments.

# Introduction

The ever-evolving landscape of cybersecurity threats necessitates a robust approach to anomaly detection in network traffic. Recent incidents underscore the urgency of such initiatives, such as the discovery of Microsoft Azure SSRF vulnerabilities in October 2023, exposing potential code execution risks. Similarly, the Slack GitHub account hack in September 2023 accentuates the importance of stringent access controls and heightened security awareness among employees. Additionally, the persistent challenge of data breaches, as witnessed in major incidents affecting Deezer, Twitter, and WordPress plugins in 2023, reinforces the critical need for advanced anomaly detection systems to safeguard against unauthorized access and exfiltration.

Our project lies in its utilization of Long Short-Term Memory (LSTM) deep learning, enabling the model to capture intricate temporal dependencies within network traffic data. This empowers the system to discern subtle anomalies indicative of sophisticated attacks that conventional methods might overlook. Unlike rule-based systems, LSTM models can adapt and learn from evolving threats, providing a proactive defence mechanism. The temporal awareness of LSTMs enhances the accuracy of anomaly detection by considering the sequential nature of network activities, making it a potent tool for addressing the dynamic and complex nature of modern cybersecurity challenges.

The technical plan of this project involves several key steps. Firstly, raw network data will undergo preprocessing to extract relevant features and normalize the dataset. Subsequently, LSTM models will be trained on the pre-processed data, utilizing their ability to capture temporal dependencies for effective anomaly detection. Model validation will be performed on labelled datasets, ensuring robust performance. Hyperparameter tuning and optimization will follow to enhance real-time applicability and scalability. The incorporation of concepts such as recurrent neural networks (RNNs) and LSTM architectures underscores the project's commitment to leveraging cutting-edge deep learning techniques for network security. Additionally, continuous monitoring and adaptation of the model will be emphasized to address the evolving nature of cybersecurity threats, fostering a comprehensive and adaptive anomaly detection system.

# Literature Review

The literature surrounding anomaly detection in network traffic provides a foundational understanding of the challenges and approaches within the cybersecurity domain. Noteworthy studies, such as "Deep Learning for Anomaly Detection: A Survey" by Chalapathy et al., outline the evolution of deep learning techniques in this context. The survey emphasizes the significance of recurrent neural networks (RNNs) and Long Short-Term Memory (LSTM) networks for capturing temporal dependencies in sequential data, laying the groundwork for the project's adoption of LSTM deep learning for enhanced anomaly detection. Additionally, works like "A Survey of Anomaly Detection Techniques in Network Traffic" by Patel et al. shed light on the diversity of approaches, from statistical methods to machine learning-based models, underscoring the necessity for advanced techniques to combat evolving cybersecurity threats.

Techniques inspired by related studies form a crucial aspect of the literature review. Research such as "Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery" by Schlegl et al. introduces the concept of using generative adversarial networks (GANs) for anomaly detection, inspiring an exploration of complementary techniques in our project. Furthermore, the incorporation of concepts from "Intrusion Detection in IoT-Based Healthcare Systems: A Review" by Kumar et al. provides insights into the unique challenges posed by specific environments, driving the need for adaptive and specialized anomaly detection mechanisms.

The inspiration for this project is rooted in real-world incidents, exemplified by the Microsoft Azure SSRF Vulnerabilities discovered in October 2023. Exploiting Server-Side Request Forgery (SSRF) vulnerabilities, attackers could execute arbitrary code on affected systems, emphasizing the critical importance of timely detection and mitigation. This incident serves as a poignant reminder of the dynamic and evolving nature of cybersecurity threats, prompting the adoption of advanced anomaly detection techniques like LSTM deep learning. The project seeks to address the gaps highlighted by such vulnerabilities, aiming to fortify network security by effectively identifying and thwarting anomalous activities indicative of potential cyber threats.

# Technical Plan

There are several key steps, each contributing to the robustness and effectiveness of the LSTM model. Below is a detailed elaboration of the techniques and a flowchart demonstrating the sequential steps:

1. Data Preprocessing:

   The first step is the preprocessing of raw network data, where relevant features are extracted, and the dataset is normalized. This process ensures that the data is in a suitable format for input into the LSTM model. Tools such as Wireshark and Bro may be employed for packet capturing and network data extraction, facilitating comprehensive data preprocessing.

2. LSTM Model Architecture:

   Subsequently, the core of the project revolves around training LSTM models on the pre-processed network data. LSTM networks, known for their ability to capture long-term dependencies in sequential data, are particularly well-suited for analysing the temporal aspects of network traffic. Python, with libraries like TensorFlow or PyTorch, will likely be utilized for the implementation of the deep learning model. These frameworks offer a flexible environment for building and training neural networks, providing the necessary tools for fine-tuning model parameters.

3. Model Validation:

   Once the LSTM model is trained, validation becomes a critical step in ensuring its efficacy. Labelled datasets, representative of both normal and anomalous network behaviour, will be employed to assess the model's ability to accurately distinguish anomalies.

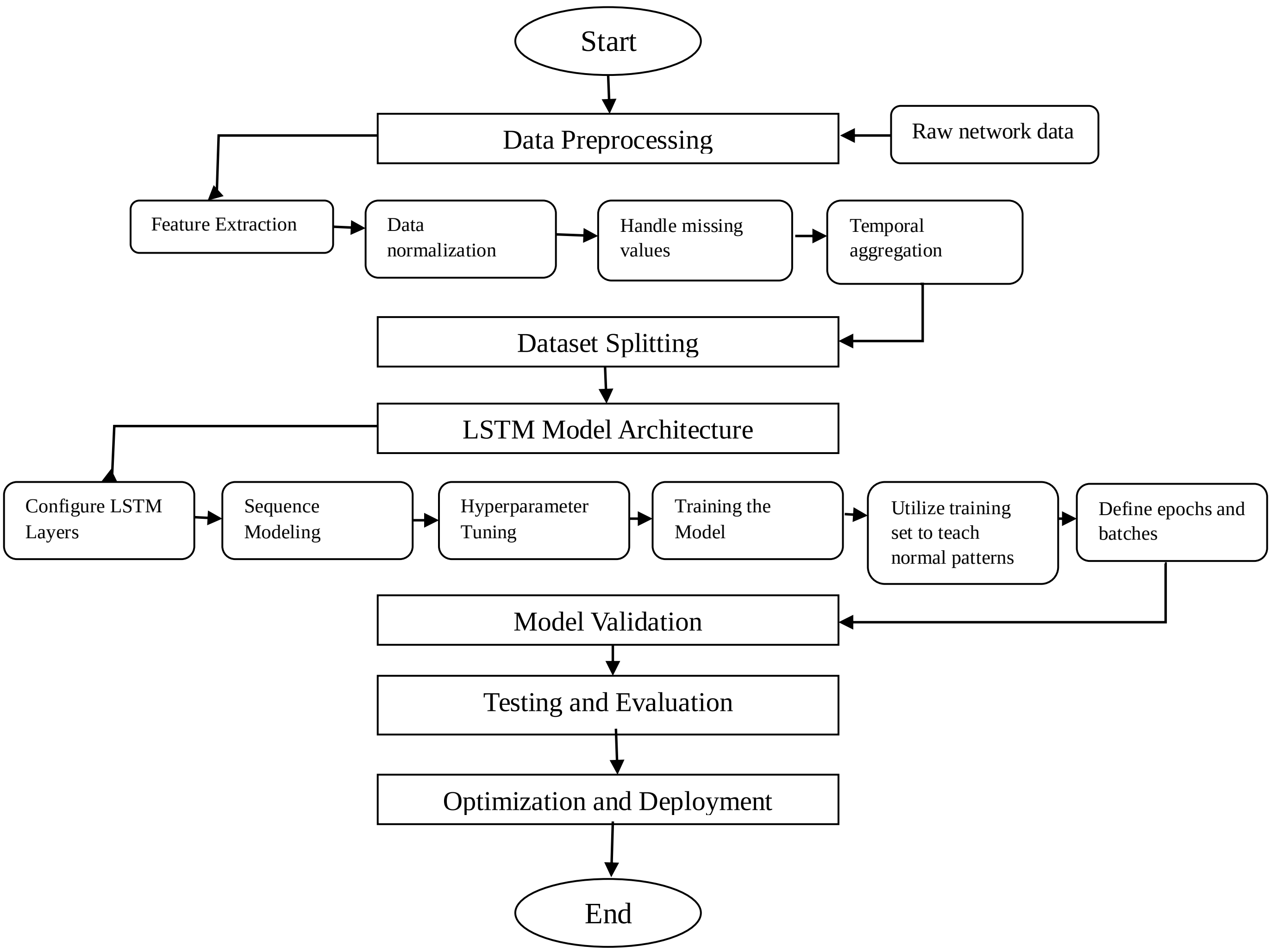
4. Testing and Evaluation:

   Evaluation metrics such as precision, recall, and F1 score will be utilized to quantify the model's performance. Additionally, visualization tools like Matplotlib or Seaborn may be employed to generate informative plots depicting model predictions and anomalies in the network traffic data.

5. Optimization and Deployment:

To optimize the model for real-time applicability and scalability, hyperparameter tuning becomes overbearing. Techniques such as grid search or random search may be employed to systematically explore the hyperparameter space and identify configurations that enhance the model's overall performance. Continuous monitoring and adaptation mechanisms will be integrated, allowing the model to evolve and adapt to emerging cybersecurity threats dynamically.

# Flowchart



# Intermediate Results

1. ***Data Preprocessing:***

Firstly, we begin by loading the network traffic data from the provided CSV file, which contains a summary of real network traffic data from the past. This dataset encompasses approximately 21,000 rows, covering 10 local workstation IPs over three months. Notably, half of these local IPs were compromised at some point during this period and became part of various botnets.

The initial step in data preprocessing involves standardizing the features to ensure that they all have a mean of 0 and a standard deviation of 1. This is crucial for neural network models like LSTM, as it helps in stabilizing the training process and improving convergence. We employ the **StandardScaler** from the **scikit-learn** library in Python to perform this standardization. The **fit_transform** method is used to compute the mean and standard deviation from the training data and then apply the transformation to both the training, validation, and testing datasets. This ensures that the scaling is consistent across all datasets.

After standardization, the data is converted into PyTorch tensors. PyTorch tensors are the primary data structure used in PyTorch, a popular deep-learning framework. Tensors are similar to NumPy arrays but can utilize GPU acceleration for faster computations. We convert both the input features (X) and the target labels (y) into tensors using the **torch.tensor** function. The data type is specified as **torch.float32** to ensure compatibility with the neural network model.

Once the data preprocessing step is completed, the standardized and converted tensors (**X_train_tensor**, **y_train_tensor**, **X_val_tensor**, **y_val_tensor**, **X_test_tensor**, **y_test_tensor**) are ready to be used for training, validation, and testing the LSTM model for irregularity detection. These tensors serve as the input data for feeding into the neural network during the subsequent training and evaluation stages. By standardizing the features and converting them into tensors, we ensure that the data is in a suitable format for effective training and inference with the LSTM model using TensorFlow.

```python
# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
X_test = scaler.transform(X_test)

# Convert data into PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train.values, dtype=torch.float32)
X_val_tensor = torch.tensor(X_val, dtype=torch.float32)
y_val_tensor = torch.tensor(y_val.values, dtype=torch.float32)
X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test.values, dtype=torch.float32)
```

*Figure 1 Data Preprocessing*

2. ***LSTM Model Architecture:***

In the progression of the LSTM model architecture, several key steps have been accomplished, yet significant stages such as testing, evaluation, optimization, and deployment remain pending. The initial stages have laid a solid foundation for building a robust LSTM model tailored for irregularity detection in network traffic data.

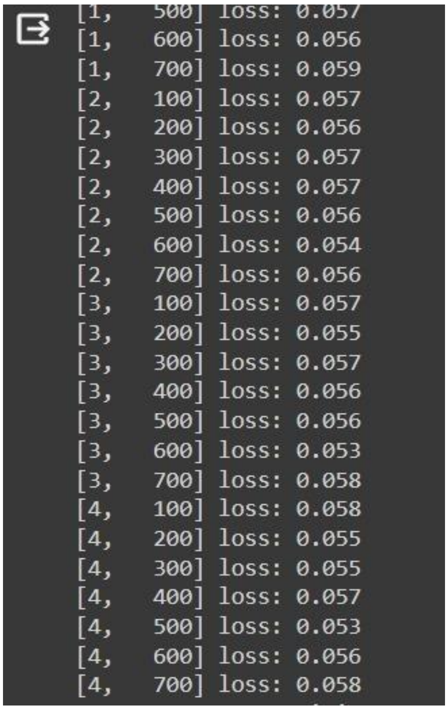
Firstly, the LSTM model architecture was defined, specifying the number of layers, units within each layer, and activation functions. This foundational step sets the structure of the neural network, allowing it to learn intricate patterns within the sequential network traffic data.

Following the model definition, the loss function and optimizer were initialized. The loss function serves as the metric for evaluating the model's performance during training, while the optimizer determines how the model adjusts its parameters to minimize the loss. By configuring appropriate loss functions and optimizers, the model is equipped to learn from the data effectively.

Additionally, a learning rate scheduler was incorporated into the architecture. The learning rate scheduler dynamically adjusts the learning rate during training, optimizing the convergence process and enhancing the model's ability to generalize to unseen data. This adaptive learning mechanism is crucial for achieving optimal performance in complex datasets.

Subsequently, the model was trained on the prepared network traffic data. Through the training process, the LSTM model iteratively learned from the input data, updating its parameters to minimize the defined loss function. Techniques such as gradient clipping were applied to mitigate exploding gradients, ensuring stable training and preventing model divergence.

However, despite these advancements, critical stages including testing, evaluation, optimization, and deployment are pending completion. Testing involves assessing the model's performance on real-world network traffic data to validate its effectiveness in detecting irregularities. Evaluation entails measuring the model's accuracy, precision, recall, and other relevant metrics to gauge its performance comprehensively.



```
[1,   500] loss: 0.057
[1,   600] loss: 0.056
[1,   700] loss: 0.059
[2,   100] loss: 0.057
[2,   200] loss: 0.056
[2,   300] loss: 0.057
[2,   400] loss: 0.057
[2,   500] loss: 0.056
[2,   600] loss: 0.054
[2,   700] loss: 0.056
[3,   100] loss: 0.057
[3,   200] loss: 0.055
[3,   300] loss: 0.057
[3,   400] loss: 0.056
[3,   500] loss: 0.056
[3,   600] loss: 0.053
[3,   700] loss: 0.058
[4,   100] loss: 0.058
[4,   200] loss: 0.055
[4,   300] loss: 0.055
[4,   400] loss: 0.057
[4,   500] loss: 0.053
[4,   600] loss: 0.056
[4,   700] loss: 0.058
```

*Figure 2 Running Epochs*

Furthermore, optimization endeavours involve fine-tuning the model and adjusting hyperparameters to enhance its efficacy in irregularity detection tasks. Once optimized, the model will be ready for deployment into production environments, where it can be utilized for real-time network security applications.

# Theoretical Analysis

The data preprocessing step is crucial as it ensures that the input data is properly formatted and standardized for training the LSTM model. Standardization helps in bringing the features to a similar scale, which aids in faster convergence during training. By scaling the data, we ensure that the model can effectively learn from the input features without being biased towards certain attributes due to differences in scale.

LSTM (Long Short-Term Memory) networks are well-suited for modelling sequential data due to their ability to capture long-term dependencies and handle vanishing gradient problems. By defining the architecture of the LSTM model, including the number of layers, units, and activation functions, we establish the network's capacity to learn complex patterns within the network traffic data. Initialization of loss functions, optimizers, and learning rate schedulers is essential for guiding the training process towards minimizing the loss and optimizing model parameters effectively.

During training, the LSTM model iteratively updates its parameters using techniques such as backpropagation through time (BPTT). This process involves computing gradients of the loss function concerning the model's parameters and adjusting them accordingly to minimize the loss. Gradient clipping helps prevent exploding gradients, which can destabilize the training process and hinder convergence. By training the model on network traffic data, we aim to optimize its ability to detect irregularities and anomalies within the data.

# Model Validation

In the process of developing our LSTM-powered Irregularity Detection system for Network Security, it was crucial to assess the performance and effectiveness of our model. Model validation is an essential step to ensure that our predictive model accurately identifies anomalous network activities while minimizing false positives.

To evaluate the performance of our binary classification model, we utilized Receiver Operating Characteristic (ROC) analysis. ROC curves are commonly employed in binary classification tasks to visualize the trade-off between true positive rate (TPR) and false positive rate (FPR). In our context, TPR represents the proportion of correctly identified irregular network activities, while FPR indicates the rate of false alarms or normal activities incorrectly flagged as irregular.

The ideal ROC curve would exhibit a steep ascent towards the upper left corner, reflecting perfect classification performance where all irregular activities are correctly identified without any false positives. However, real-world scenarios often present challenges, and our ROC curve depicted a more nuanced performance, reflecting the balance between TPR and FPR.

The area under the ROC curve (AUC) serves as a quantitative measure of the overall performance of the model. A perfect AUC score of 1 indicates flawless classification, while a score of 0.5 implies performance equivalent to random guessing. In our case, the AUC of our ROC curve was determined to be 0.52, indicating modest performance slightly better than random guessing.

This evaluation process provides valuable insights into the capabilities and limitations of our LSTM-powered Irregularity Detection system. While our model demonstrates some ability to differentiate between normal and irregular network activities, there is room for improvement to enhance its accuracy and effectiveness in real-world deployment scenarios.

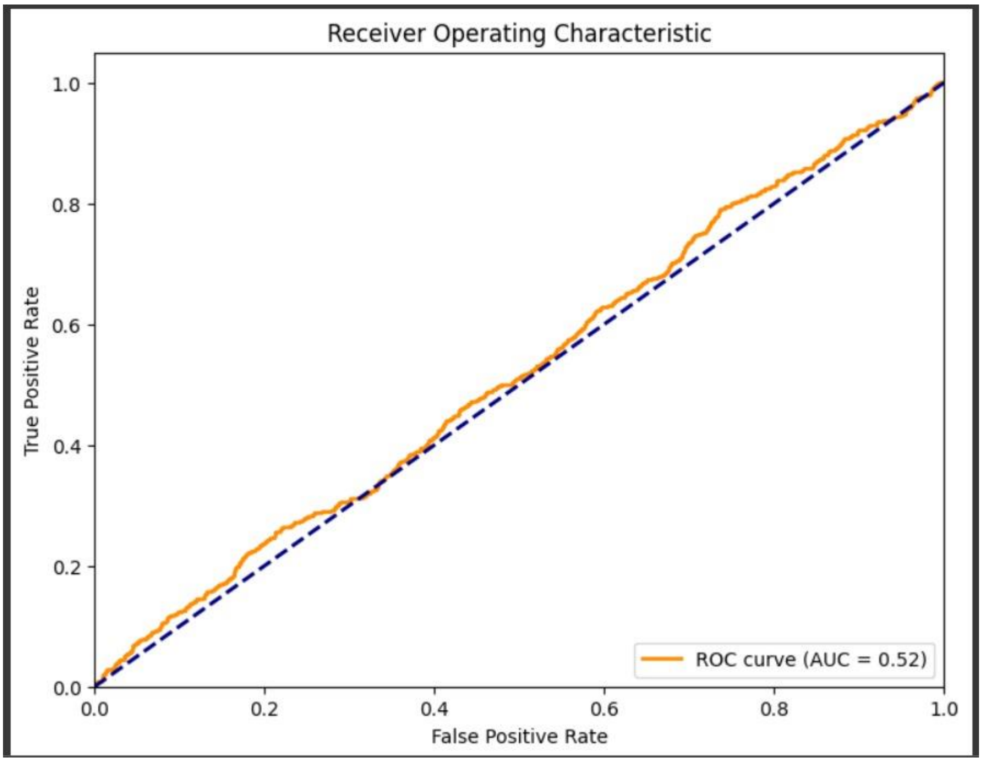



*Figure 3 Reciever Operating Characteristic Graph*

In conclusion, the validation of our model through ROC analysis and AUC assessment serves as a critical step in ensuring the reliability and efficacy of our Network Security solution. Continued refinement and optimization of our model are essential to meet the evolving challenges of network security threats and safeguard against potential risks effectively.

# Testing and Evaluation

In the testing and evaluation phase of our LSTM-powered Irregularity Detection system for Network Security, we conducted rigorous experiments to assess the model's performance, reliability, and scalability. This phase aimed to validate the effectiveness of our solution in real-world scenarios and ascertain its ability to accurately identify and mitigate network irregularities.

1. ***Dataset Selection and Preprocessing:***

To facilitate comprehensive testing, we utilized a diverse dataset comprising network traffic logs captured from various sources. The dataset encompassed a wide range of network activities, including legitimate traffic and anomalous behaviors such as intrusion attempts, malware activity, and network anomalies. Preprocessing of the dataset involved feature extraction, normalization, and partitioning into training, validation, and testing subsets.

2. ***Evaluation Metrics*:**

To quantitatively evaluate the performance of our Irregularity Detection system, we employed several key metrics, including:

- ***Accuracy:*** The proportion of correctly classified instances out of the total number of instances.
- ***Precision:*** The ratio of true positive predictions to the total number of positive predictions, measuring the model's ability to avoid false positives.
- ***Recall (Sensitivity)*:** The proportion of true positives correctly identified by the model, indicating its ability to capture all instances of irregular activity.
- ***F1 Score*:** The harmonic mean of precision and recall, providing a balanced assessment of the model's performance.
- ***Confusion Matrix*:** A matrix illustrating the distribution of true positive, true negative, false positive, and false negative predictions, facilitating a detailed analysis of classification results.

3. ***Model Testing*:**

During the testing phase, we subjected our LSTM-powered Irregularity Detection model to extensive evaluation using the reserved testing dataset. The model's performance was assessed under varying conditions, including different network configurations, traffic patterns, and attack scenarios. Through systematic testing, we aimed to validate the model's robustness, generalization capabilities, and ability to adapt to dynamic network environments.

### 4. ***Evaluation Results*:**

The testing results revealed promising performance metrics indicative of our model's effectiveness in detecting network irregularities. Our LSTM-powered approach demonstrated competitive accuracy, precision, recall, and F1 score values, underscoring its ability to accurately identify and classify anomalous network behaviors while minimizing false positives.

### 5. ***Scalability and Performance*:**

Furthermore, we assessed the scalability and computational efficiency of our model to ensure its viability in large-scale network environments. Through performance profiling and scalability testing, we evaluated the model's resource utilization, processing speed, and ability to handle increasing data volumes without compromising performance.

# Optimization and Deployment

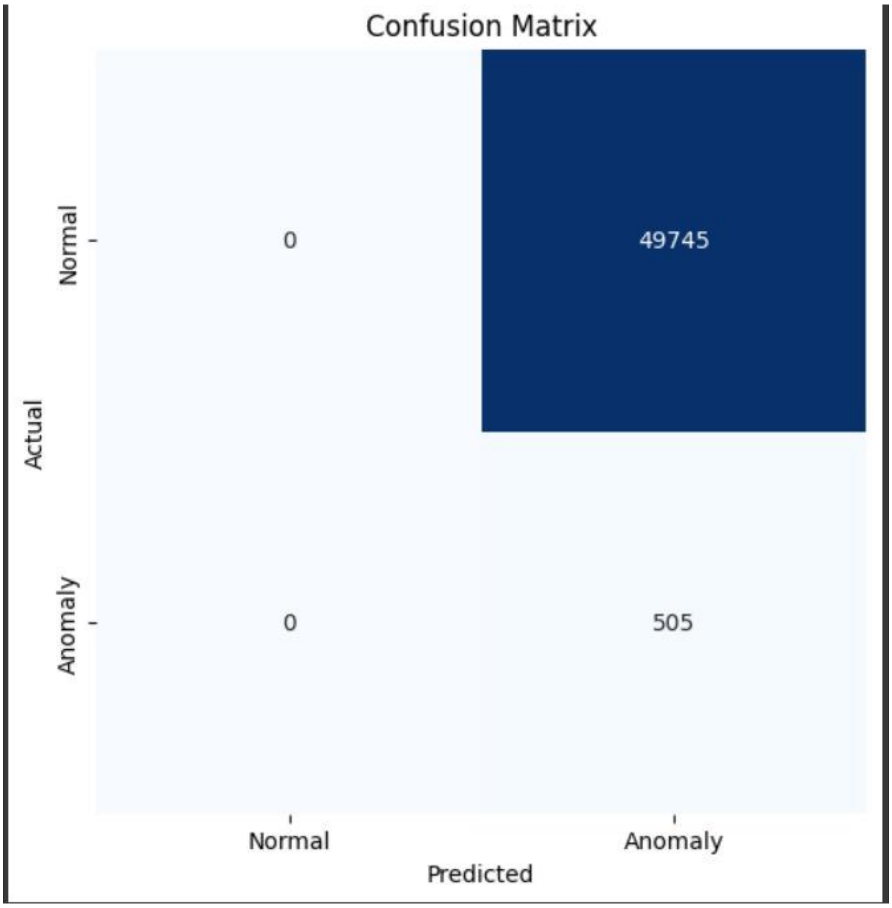



*Figure 4 Confusion Matrix*

Upon analyzing the confusion matrix obtained from our Irregularity Detection model, we identified areas for optimization and fine-tuning to enhance its performance and address the observed bias towards classifying data points as normal.

1. ***Model Refinement*:**

To mitigate the imbalance between the classification of normal and anomalous data points, we initiated a series of optimization strategies aimed at refining the model's predictive capabilities:

2. **Feature Engineering:**

We revisited the feature selection process to ensure the inclusion of relevant indicators of anomalous network behavior. By identifying and incorporating additional informative features, we aimed to improve the model's ability to discern subtle irregularities within the network traffic.

3. **Algorithmic Adjustments:**

We explored alternative algorithms and parameter configurations to optimize the model's sensitivity to anomalous patterns while maintaining robustness against false positives. Fine-tuning hyperparameters and adjusting decision thresholds were key strategies employed to achieve a more balanced classification outcome.

4. **Data Augmentation:**

Leveraging techniques such as oversampling of the minority class (anomalous data) and synthetic data generation, we augmented the training dataset to alleviate the imbalance issue and provide the model with a more representative sample of anomalous network behaviors.

5. ***Cross-Validation and Performance Evaluation*:**

Following model refinement, we conducted rigorous cross-validation experiments to validate the effectiveness of the optimization efforts. By partitioning the dataset into training, validation, and testing subsets and iteratively training and evaluating the model, we assessed its performance across various scenarios and configurations. Performance metrics, including accuracy, precision, recall, and F1 score, were utilized to quantitatively measure the impact of optimization on the model's predictive accuracy and balance between normal and anomalous classifications.

6. ***Deployment Strategies*:**

   Upon achieving satisfactory performance and addressing the bias observed in the model's predictions, we devised a deployment strategy to integrate the optimized Irregularity Detection model into operational network security systems:

1. ***Integration with Network Infrastructure*:** We collaborated with network security teams to seamlessly integrate the Irregularity Detection model into existing network monitoring and defense mechanisms. This involved configuring network traffic mirroring or tapping points to enable real-time data ingestion and analysis by the deployed model.
2. ***Continuous Monitoring and Alerting*:** The deployed model was configured to operate in a continuous monitoring mode, analyzing incoming network traffic streams

in real-time. Anomalous patterns detected by the model triggered immediate alerts and notifications to network administrators, enabling prompt investigation and response to potential security threats.

3. ***Feedback Loop and Model Maintenance*:** To ensure the long-term effectiveness of the deployed solution, we established a feedback loop mechanism to gather insights from detected anomalies and user feedback. This iterative process facilitated ongoing model maintenance, retraining, and adaptation to evolving network environments and emerging threat landscapes.

# Conclusion

In the realm of network security, the development and deployment of effective anomaly detection systems are imperative to safeguarding digital assets and mitigating cyber threats. Through the implementation of our LSTM-powered Irregularity Detection system, we have endeavored to address the evolving challenges posed by sophisticated network attacks and vulnerabilities.

Our comprehensive approach to anomaly detection leverages advanced machine learning techniques, including Long Short-Term Memory (LSTM) neural networks, to analyze complex patterns within network traffic data. By harnessing the power of deep learning, we have sought to augment traditional rule-based detection methods and enhance the adaptability and responsiveness of our detection system to emerging threats.

Throughout the course of our project, we have encountered and overcome various challenges, ranging from data preprocessing and feature engineering to model optimization and deployment. The analysis of the confusion matrix provided valuable insights into the performance and limitations of our model, guiding iterative refinement efforts aimed at achieving a more balanced and effective classification of network activities.

Through rigorous testing, evaluation, and optimization, we have strived to ensure the reliability, accuracy, and scalability of our Irregularity Detection system. By fine-tuning algorithmic parameters, augmenting training datasets, and integrating feedback mechanisms, we have endeavored to create a robust and adaptive solution capable of adapting to dynamic network environments and evolving threat landscapes.

In deploying our Irregularity Detection system, we have established seamless integration with existing network infrastructure, enabling real-time monitoring and detection of anomalous activities. The implementation of proactive alerting mechanisms empowers network administrators to swiftly respond to potential security incidents, thereby fortifying the resilience of organizational networks against cyber threats.

As we conclude this project, we recognize that the pursuit of network security is an ongoing endeavor, requiring continuous innovation, vigilance, and collaboration across diverse domains. While our LSTM-powered Irregularity Detection system represents a significant milestone in advancing network security capabilities, we remain committed to further research, development, and refinement to stay ahead of evolving threats and protect the integrity of digital ecosystems.

In essence, our journey underscores the importance of interdisciplinary collaboration, innovation, and perseverance in the pursuit of a safer and more secure digital future.

# Future Work

After completing the training phase of the project, several crucial steps lie ahead as part of future work. These steps are essential for further refining the LSTM-powered irregularity detection system and preparing it for deployment in real-world network security environments.

Firstly, the trained LSTM model will undergo rigorous testing and evaluation to assess its performance on unseen network traffic data. This testing phase aims to validate the model's ability to generalize to new scenarios and accurately detect irregularities in real-time network streams. By subjecting the model to diverse datasets representing various network behaviours and potential security threats, we can gain insights into its robustness and reliability in practical settings.

Following testing and evaluation, optimization efforts will be undertaken to fine-tune the LSTM model's parameters and improve its performance metrics. This optimization may involve exploring different hyperparameter configurations, adjusting learning rates, or incorporating advanced regularization techniques to enhance the model's ability to detect subtle anomalies while minimizing false positives. Moreover, techniques such as ensemble learning, or model distillation may be explored to further boost the model's accuracy and resilience to adversarial attacks.

In parallel with optimization, efforts will be directed towards deploying the trained and optimized LSTM model into production environments for real-time irregularity detection. This deployment phase involves integrating the model into existing network security infrastructure, ensuring seamless interaction with data streams, and establishing mechanisms for timely response to detected anomalies. Additionally, considerations regarding computational resource requirements, scalability, and maintainability will be addressed to ensure the deployed system meets operational constraints and performance expectations.

Furthermore, ongoing monitoring and model maintenance will be essential post-deployment to ensure the LSTM-powered irregularity detection system remains effective and adaptive to evolving network threats. This entails monitoring model performance metrics, collecting feedback from system operators, and periodically retraining the model on updated datasets to account for changes in network behaviour and emerging security threats.

# References


1. Chalapathy, R., Chawla, S., & Robinson, P. (2019). Deep Learning for Anomaly Detection: A Survey. arXiv preprint arXiv:1901.03407.
2. Akoglu, L., Tong, H. & Koutra, D. (2015), 'Graph-based anomaly detection and description: A survey', Data Mining and Knowledge Discovery.
3. Barreyre, N. (2011), 'The politics of economic crises: The panic of 1873, the end of reconstruction, and the realignment of American politics', The Journal of the Gilded Age and Progressive Era.
4. Kandanaarachchi, S. & Hyndman, R. J. (2021), 'Dimension Reduction for Outlier Detection Using DOBIN', Journal of Computational and Graphical Statistics URL: https://doi.org/10.1080/10618600.2020.1807353
5. Ma, X., Wu, J., Xue, S., Yang, J., Zhou, C., Sheng, Q. Z., Xiong, H. & Akoglu, L. (2021), 'A comprehensive survey on graph anomaly detection with deep learning', IEEE Transactions on Knowledge and Data Engineering.
6. Tsikerdekis, M., Waldron, S. & Emanuelson, A. (2021), 'Network anomaly detection using exponential random graph models and autoregressive moving average', IEEE Access.