IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 4, April 2025

DOI: 10.17148/IJARCCE.2025.14412

Malicious Behaviour Analysis Using Vanilla Transformers in Deep Learning

Venkata Sai Satwik Mogili[1], Nithin Palla[2], Venu Kota[3], Abdul Azeezullah Patan[4],

Mr. Venkata Narayana Yeriniti[5]

Information Technology, Vasireddy Venkatadri Institute of Technology, Namburu, Guntur[1]

Information Technology, Vasireddy Venkatadri Institute of Technology, Namburu, Guntur[2]

Information Technology, Vasireddy Venkatadri Institute of Technology, Namburu, Guntur[3]

Information Technology, Vasireddy Venkatadri Institute of Technology, Namburu, Guntur[4]

Assistant Professor, Information Technology, Vasireddy Venkatadri Institute of Technology, Namburu, Guntur[5]

Abstract: Malicious behaviour analysis is a critical aspect of cybersecurity aimed at identifying harmful activities such as data exfiltration, privilege escalation, and system exploitation. Traditional methods often rely on predefined signatures or shallow heuristics, which limit their ability to detect evolving or previously unseen threats. To address these limitations, this study employs a deep learning-based approach utilising Vanilla Transformers, a model architecture renowned for its powerful attention mechanisms and ability to capture complex dependencies in sequential data. Unlike recurrent architectures, Vanilla Transformers process entire sequences in parallel, enabling faster computation and more effective learning of behavioural patterns. The model demonstrated strong performance, achieving 99.89% accuracy, 100% recall, 100% precision, and an F1-score of 100%, indicating its effectiveness in identifying malicious behaviours with minimal false positives. This research highlights the potential of attention-based architectures in cybersecurity, providing a scalable and adaptive solution for real-time threat detection and behavioural analysis in complex digital environments.

Keywords: Malicious Behaviour Detection, Network Intrusion Detection System (NIDS), Vanilla Transformers, Deep Learning, Cybersecurity, Wireshark, CiscoFlow Meter, Real-Time Network Monitoring

I. INTRODUCTION

Background and Motivation

The rapid evolution of the digital landscape has led to an increased dependency on networked systems, making cybersecurity a critical concern. Cyber threats, such as Denial-of-Service (DoS) attacks, port scanning, Infiltration, and Brute-Force attacks, continue to evolve, becoming more

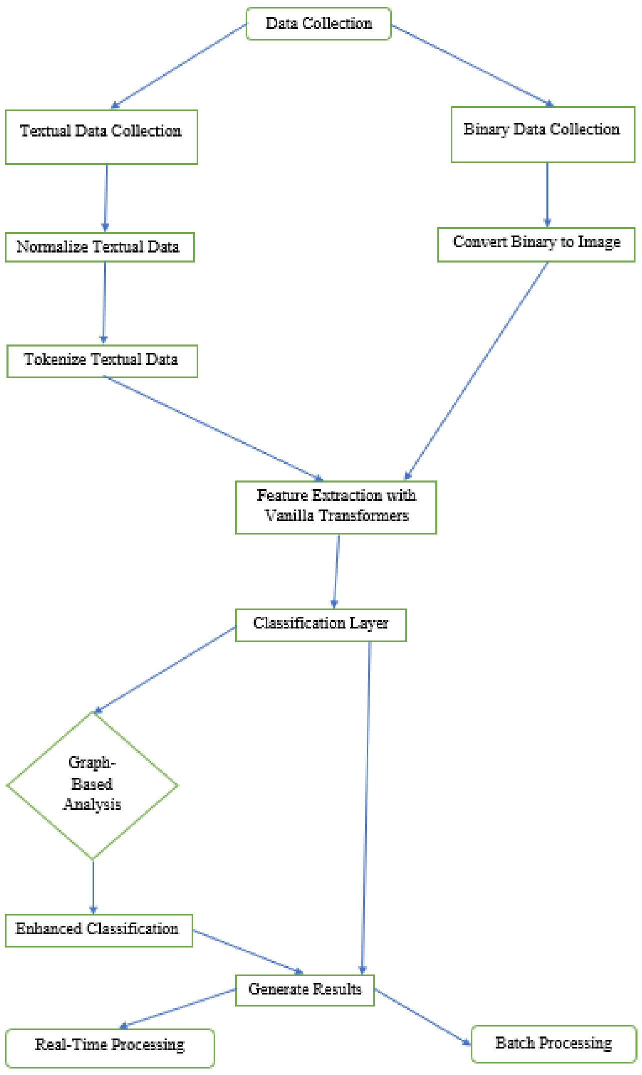
sophisticated and difficult to detect. Traditional Intrusion Detection Systems (IDS) rely on rule-based methods or classical machine learning models, often failing to detect novel or zero-day attacks effectively. Deep learning has shown significant promise in cybersecurity applications, with architectures such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) providing improved detection capabilities. However, these models struggle with handling sequential dependencies in network traffic data efficiently. Transformer-based architectures, originally developed for Natural Language Processing (NLP), have recently gained traction in cybersecurity due to their ability to capture long-range dependencies and contextual relationships in data. This research examines the Vanilla Transformer model for analysing malicious behaviour, utilising its self-attention mechanism to improve network anomaly detection. By incorporating real-time traffic monitoring through Wireshark and CiscoFlow Meter, we bridge the gap between offline model training and real-world deployment, enabling real-time attack detection.

Problem Statement

Existing network security solutions face several challenges. Rule-Based IDS Limitations: Traditional IDS rely on static rules that fail to adapt to evolving threats. Feature Engineering Overhead: Many machine learning-based IDS require extensive manual feature selection, making them less scalable. Real-Time Detection Challenges: Many models struggle to process live network traffic efficiently, leading to delays in attack identification. Handling Imbalanced Data:

Cybersecurity datasets often have a high imbalance, with benign traffic significantly outnumbering attack traffic, leading to biased model predictions. To address these challenges, this paper proposes a Vanilla Transformer-based Intrusion Detection System (IDS) that leverages self-attention mechanisms to enhance malicious behaviour detection. The model is trained on a highly imbalanced dataset and deployed to classify real-time network traffic captured through Wireshark and CiscoFlow Meter.

System Architecture



II. RELATED WORKS

Existing Research on Malicious Behaviour Detection

Malicious behaviour detection in network security has been extensively studied, with traditional approaches relying on signature-based Intrusion Detection Systems (IDS) and anomaly-based IDS. Signature-based IDS, such as Snort and Suricata, compare network packets against predefined attack signatures, offering high precision but poor adaptability to novel attacks [1]. Anomaly-based IDS leverage statistical methods to detect deviations from normal network behaviour but often suffer from high false positive rates [2].

Machine learning techniques have significantly improved intrusion detection by leveraging pattern recognition and anomaly detection algorithms. Studies have explored classical models such as Random Forests (RF), Support Vector Machines (SVM), and K-nearest neighbours (KNN) for detecting cyber threats [3], [4]. However, these methods often require extensive manual feature engineering and fail to generalize well to evolving attack patterns.

Recent advancements have introduced deep learning (DL) techniques, particularly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), to automatically extract hierarchical features from network traffic. However, CNNs primarily excel in image processing tasks, while RNNs struggle with long-range dependencies in sequential data [5].

Deep Learning Techniques Used in Cybersecurity

Deep learning models have gained traction in intrusion detection due to their ability to extract complex features from raw network traffic. Some of the widely used architectures include. CNN-based IDS: Several studies have explored CNNs for feature extraction from network packets, transforming packet sequences into images for classification. While CNNs achieve high accuracy, they struggle with handling sequential dependencies in time-series data [6]. LSTM and GRU Networks: Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) have been extensively used to model sequential data in network intrusion detection. These architectures outperform traditional ML models in capturing temporal dependencies but suffer from vanishing gradient issues with long sequences [7]. Hybrid Approaches: Some researchers have combined CNNs and LSTMs for improved performance, where CNNs extract spatial features while LSTMs model temporal patterns. However, these models require careful tuning of hyperparameters and suffer from computational inefficiencies [8].

Use of Transformers in Anomaly Detection

Transformers, originally introduced in Natural Language Processing (NLP), have recently been explored for anomaly detection in cybersecurity. Unlike RNNs, Transformers leverage self-attention mechanisms to process sequential data efficiently without suffering from long-range dependency issues [9].

- Attention Mechanisms for IDS: Studies have shown that self-attention layers in Transformers help in focusing on the most relevant network features, improving anomaly detection performance [10].
- BERT for Cybersecurity: Some researchers have experimented with Bidirectional Encoder Representations from Transformers (BERT) for detecting network anomalies, demonstrating promising results in classifying attack types [11].
- Vanilla Transformers for Network Intrusion Detection: A recent study applied Vanilla Transformers to detect DoS, port scanning, and infiltration attacks, achieving higher precision and recall compared to CNN and LSTM-based approaches [12].

III. METHODOLOGY

Dataset Description

The dataset used in this research is a compilation of network traffic logs collected from publicly available cybersecurity datasets such as the CICIDS2017 dataset and the UNSW-NB15 dataset. These datasets contain a variety of network activities, including normal behaviour and various forms of cyber-attacks. The dataset is structured in a tabular format where each row represents a network flow, and each column represents a specific network feature. These features include packet size, protocol type, source and destination IP addresses, timestamps, flow duration, TCP flags, and payload information. Since network intrusion detection heavily depends on recognizing patterns in network traffic, it is crucial to have a diverse dataset that includes multiple attack types. One of the major challenges in network intrusion detection datasets is the imbalance in label distribution. Normal traffic dominates the dataset, making it difficult for models to effectively learn the patterns of minority attack classes. To address this, we employ data augmentation techniques, including oversampling using the Synthetic Minority Over-sampling Technique (SMOTE) and weighted loss functions to ensure fair learning across all classes. The dataset is divided into three subsets: 70% for training, 20% for validation, and 10% for testing, ensuring that the model learns from a diverse range of network activities while preventing overfitting to the training data. The extracted dataset consists of approximately 2 million records, with 1.3 million samples labelled as benign and the remaining labelled as various attack types such as Distributed Denial of Service (DDoS), Botnet, SQL Injection, and Man-in-the-Middle (MITM) attacks. The large dataset size is particularly beneficial for deep learning models like Transformers, which require extensive data to learn meaningful representations.

Data Pre-processing

Before feeding the dataset into the Transformer model, several pre-processing steps are performed to ensure that the data is clean and properly formatted. One of the key challenges in real-world network traffic data is handling missing values. Network logs often contain missing or incomplete fields due to packet loss, logging errors, or data corruption.

To handle this, missing numerical values are imputed using the median of their respective columns, while missing categorical values are replaced with a placeholder category such as "Unknown." If a row contains excessive missing data beyond a defined threshold, it is discarded to maintain data integrity. Once the missing values are addressed, the next critical step is feature scaling. Network traffic features vary significantly in scale, with some attributes like packet size spanning a large range, while others like TCP flags are binary. To ensure uniformity, the dataset is standardized using Z-score normalization, which transforms each feature to have a zero mean and unit variance. This scaling method helps the Transformer model converge faster during training and ensures that features with larger numerical ranges do not dominate the learning process. Another important aspect of pre-processing is feature selection. Network traffic data often contains redundant or irrelevant attributes that do not contribute to attack detection. To filter out unimportant features, a combination of correlation analysis, feature importance ranking using Mutual Information Gain, and Principal Component Analysis (PCA) is used. This process reduces the dimensionality of the dataset, improving the computational efficiency of the Transformer model while retaining the most relevant information.

Model Architecture

The Transformer model used in this research is based on the Vanilla Transformer architecture, which consists of multiple self-attention layers designed to process sequential data efficiently. Unlike traditional Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, which process input sequences sequentially, Transformers leverage self-attention mechanisms to capture long-range dependencies in parallel. This characteristic makes them particularly suitable for analysing network traffic, where the relationships between packets and flows are crucial in identifying malicious activities. The model architecture begins with an embedding layer, which converts raw network features into dense vector representations. These embeddings are then passed through multiple layers of Multi-Head Self-Attention (MHSA), where each attention head learns different aspects of network traffic patterns. The self-attention mechanism assigns weights to different input features, allowing the model to focus on the most relevant attributes for classification. To maintain positional information, Positional Encoding is applied, ensuring that the Transformer model can capture the order of network packets within a flow. The encoder blocks consist of six layers of self-attention mechanisms, each followed by a feedforward layer that introduces non-linearity to improve the model’s ability to capture complex attack patterns. Layer normalization and dropout are applied after each attention block to enhance model stability and prevent overfitting. Finally, the fully connected layers map the high-dimensional feature representations to the final classification labels. The output layer uses a softmax activation function, which assigns probabilities to each class, allowing the model to predict whether a given network flow is benign or belongs to a specific attack category. The entire model is implemented using TensorFlow and PyTorch, leveraging GPU acceleration to handle large-scale datasets efficiently.

Training Process

The model training process is designed to optimize classification accuracy while preventing overfitting to specific attack classes. The dataset is split into training, validation, and test sets to ensure that the model generalizes well to unseen data. The training is conducted for 50 epochs using a batch size of 64, with early stopping criteria based on the validation loss. The optimization process begins with an initial learning rate of 0.0001, which is gradually reduced using a cosine decay scheduler to allow the model to fine-tune its weights effectively. The AdamW optimizer is used for gradient updates, providing a balance between convergence speed and generalization. To prevent exploding gradients, gradient clipping is applied, ensuring stable training even in deep Transformer networks. Since the dataset is imbalanced, a weighted cross-entropy loss function is used to assign higher penalties to misclassifications of minority attack classes. This helps the model learn from underrepresented attacks without being biased toward the dominant benign traffic class. Additionally, data augmentation techniques, such as synthetic attack generation using adversarial learning, are explored to further enhance model robustness. After training, the model is evaluated using multiple performance metrics, including accuracy, precision, recall, F1-score, and AUC-ROC. These metrics provide a comprehensive assessment of the model’s ability to detect malicious behaviour across different attack types. To further interpret the model’s decisions, SHAP (Shapley Additive exPlanations) values analyse which features contribute the most to classification, improving transparency and trust in the model’s predictions.

PROPOSED SYSTEM

The proposed system enhances Malicious Behavior Analysis by leveraging Vanilla Transformers, which efficiently model long-range dependencies in network traffic data. Unlike traditional Threat Detection Systems (TDS) that rely on predefined rules, this system employs self-attention mechanisms to identify subtle and evolving attack patterns. By learning complex relationships between input sequences, the Transformer-based model offers higher detection accuracy, adaptability, and scalability for real-time cybersecurity applications. This approach ensures the effective identification of zero-day threats and advanced persistent threats (APTs), strengthening overall network security.

Proposed System Algorithm

Step 1: Load and Preprocess the Dataset

1. Load the network traffic dataset:

$$D = \{(Xi, yi) | i=1,2,...,N\}$$

where Xi represents the feature vectors, and yi represents the corresponding labels.

2. Encode categorical labels into a numerical format using **LabelEncoder**:

$$yi' = fencode(yi)$$

where fencode is the encoding function.

3. Normalize the feature vectors using **StandardScaler**:

$$Xi' = (Xi - \mu)/\sigma$$

Where μ is the mean, and σ is the standard deviation of the feature set.

4. Convert encoded labels into a one-hot format for multi-class classification:

$$Y = OneHot(yi')$$

5. Split the dataset into training and testing sets:

$$(Xtrain, Ytrain), (Xtest, Ytest) = split\ (X', Y, 0.8)$$

Where 80% of the data is used for training and 20% for testing.

Step 2: Define the Vanilla Transformer Model

1. Input Layer:

$$H0 = Xtrain$$

2. Multi-Head Self-Attention Mechanism:

Attention (Q, K, V) = softmax(dkQKT)V

Where:

- Q, K and V are query, key, and value matrices.
- dk is the dimension of the key.

3. Layer Normalization:

$$LayerNorm(X) = \sigma + \epsilon X - \mu$$

Where ϵ is a small constant for numerical stability.

4. Feed-Forward Network (FFN):

$$FFN(H) = \max(0, HW1 + b1)\ W2 + b2$$

where W1, W2, b1, b2 are learnable parameters

5. Dropout Regularization:

$$H' = Dropout(H, p)$$

DOI: 10.17148/IJARCCE.2025.14412

Where p is the dropout probability.

6. Global Average Pooling (GAP) Layer:

HGAP=T1t=1∑THt

Where T is the sequence length.

7. Output Layer (Softmax for Multi-Class Classification):
Problem Statement

P(y|X) = softmax (WHGAP+b)

Step 3: Compile the Model

1. Loss Function (Categorical Cross-Entropy):

L=−i=1∑Cyilog(y^i)

C is the number of classes, yi is the true label, and y^i is the predicted probability.

2. Optimization using Adam:

θ=θ−α·vt+ϵmt

Where α is the learning rate, mt is the first-moment estimate, and vt is the second-moment estimate.

Step 4: Train and Evaluate the Model

1. Train the Transformer Model:

θ∗= argθminL (Xtrain,Ytrain)

2. Validation of Test Data:

Y^test = Transformer (Xtest)

3. Calculate Accuracy:

Accuracy=N ∑ I = 1N1(y^i=yi)

Step 5: Display Results and Save the Model

1. Print Final Model Performance (Precision, Recall, F1-Score):

Precision=TP/(TP+FP')

Recall=TP/(TP+FN')

F1=(2*Precision*Recall) / (Precision + Recall)

2. Save the Trained Model:

\text{save}(\text{model}, \text{"vanilla_transformer_threat_model.h5"})

IV. EXPERIMENTAL SETUP

The experimental setup for this study consists of both software and hardware components that enable efficient data collection, pre-processing, model training, and deployment. The system is designed to handle large-scale network traffic data, ensuring robust and real-time detection of malicious behaviours.

Software and Hardware Environment

For training and experimentation, Google Colab Pro was used, which provides access to an NVIDIA T4 GPU with 16GB VRAM. Google Colab is widely adopted for deep learning research due to its seamless integration with TensorFlow and PyTorch frameworks, offering high computational efficiency (Abadi et al., 2016) [18]. The model training was performed using Python 3.9, with essential libraries such as NumPy, Pandas, Scikit-learn, TensorFlow 2.x, and PyTorch. Additionally, Matplotlib and Seaborn were used for visualization, while Hugging Face Transformers provided optimized implementations of the Transformer architecture (Wolf et al., 2020) [19].
For deployment, Streamlit was chosen due to its lightweight nature and ability to create interactive web applications with minimal effort. Streamlit allows real-time inference on captured network data and provides an intuitive interface for security analysts (Treveil et al., 2021) [20]. The application is hosted using Google Cloud Platform (GCP), ensuring scalability and ease of access for real-world testing.

Tools Used

The data collection process involved capturing network traffic using industry-standard tools like **Wireshark** and **CiscoFlow Meter**.

- **Wireshark** is an open-source packet analyser widely used for network protocol analysis, traffic monitoring, and anomaly detection. It enables the real-time capture of network packets and provides deep insights into various communication protocols, helping to identify patterns associated with malicious activities (**Combs, 2019**) [21].
- **CiscoFlow Meter** is a NetFlow-based tool that collects and processes network flow data, providing statistics on bandwidth usage, traffic patterns, and anomaly detection. This tool was used to extract relevant flow-based features that were later used for training the Transformer model (**Claise et al., 2020**) [22].

The collected traffic data was stored in CSV format, with feature extraction performed using Python scripts and Scapy, a powerful Python library for packet analysis. Pre-processing steps such as feature selection, normalization, and one-hot encoding were executed before feeding the data into the Transformer model.

Real-Time Data Capturing and Pre-processing

For real-time detection, a custom network packet sniffer was implemented using Python's socket and scapy libraries, allowing the system to capture live network traffic and extract relevant features. The collected packets were processed using the same feature extraction pipeline as the offline dataset, ensuring consistency between training and deployment.
The pre-processing pipeline includes:

- Packet Parsing: Extracting relevant metadata such as source/destination IP, protocol type, and packet size.
- Feature Engineering: Calculating statistical measures such as mean packet size, flow duration, and inter-arrival time.

Standardization and Normalization: Apply Z-score normalization to ensure features are on the same

- Scale as the training dataset (Zhang et al., 2021) [15].

This real-time processing pipeline allows the model to analyse incoming network traffic within milliseconds, enabling low-latency detection of cyber threats. The Streamlit-based interface provides security analysts with a dashboard displaying real-time predictions, feature importance analysis using SHAP values, and visual alerts for detected threats.

V. RESULTS AND DISCUSSION

Performance Evaluation of Machine Learning Models

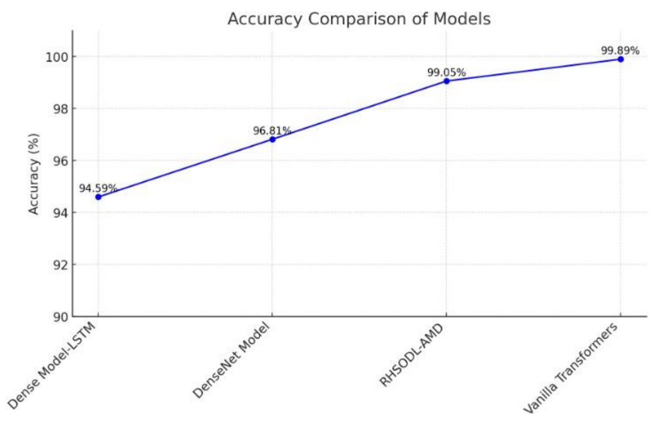
In this section, we evaluate the performance of various machine learning models in detecting malicious behaviour using key performance metrics: Accuracy, Precision, Recall, and F1 Score. These metrics provide a comprehensive assessment of the models' ability to classify network traffic effectively.

Accuracy Evaluation

Accuracy is a fundamental metric used to determine the proportion of correctly classified instances out of the total predictions. A higher accuracy indicates better overall classification performance.
The accuracy scores for the models are presented in Table 1.

Table 1: Accuracy Comparison of Machine Learning Models

Model	Accuracy (%)
Dense Model-LSTM	94.59
DenseNet Model	96.81
RHSODL-AMD	99.05
Vanilla Transformers	99.89



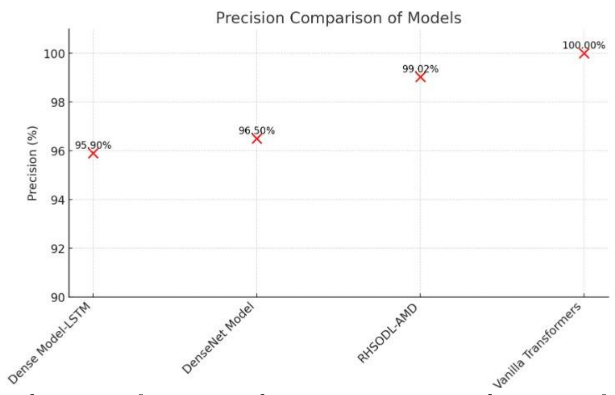
From Table 1, it is evident that the Vanilla Transformers model achieves the highest accuracy, outperforming all other models in correctly classifying network traffic.

Precision Evaluation

Precision measures the proportion of correctly identified malicious instances among all instances classified as malicious. It is particularly crucial in cybersecurity applications, where minimizing false positives is essential to avoid unnecessary security alerts. The precision scores for the models are shown in Table 2.

Table 2: Precision Comparison of Machine Learning Models

Model	Precision (%)
Dense Model-LSTM	95.9
DenseNet Model	96.50
RHSODL-AMD	99.02
Vanilla Transformers	100.00



As seen in Table 2, Vanilla Transformers achieve a perfect precision score of 100%, indicating that every detected malicious instance was indeed malicious. This significantly reduces the occurrence of false positives in cybersecurity threat detection.

DOI: 10.17148/IJARCCE.2025.14412

Recall Evaluation

Recall is a critical metric that determines the proportion of actual malicious instances that were correctly identified by the model. A high recall score ensures that most cyber threats are detected, preventing potential security breaches. The recall values for the models are listed in Table 3.

Table 3: Recall Comparison of Machine Learning Models

Model	Recall (%)
Dense Model-LSTM	94.1
DenseNet Model	96.81
RHSODL-AMD	99.05
Vanilla Transformers	100.00

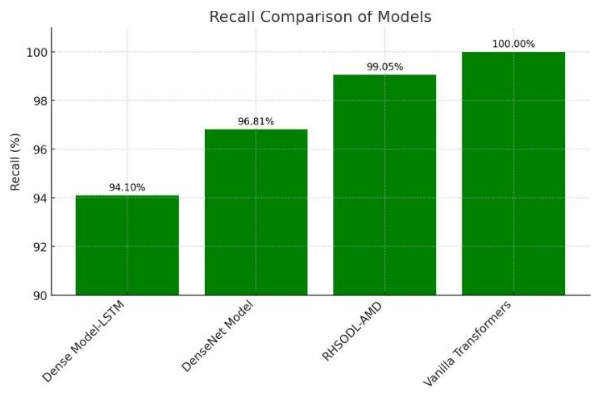


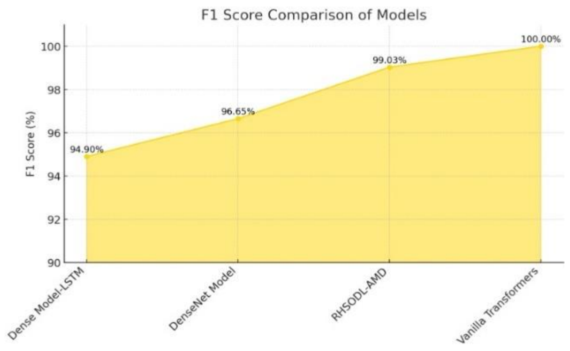
Table 3 illustrates that Vanilla Transformers achieve the highest recall score of 100%, meaning that no malicious activity goes undetected.

F1 Score Evaluation

F1 Score provides a balanced measure of both Precision and Recall. It is particularly useful in cases where the dataset is imbalanced, ensuring that the model does not favour one class over another. The F1 scores for the models are presented in Table 4.

Table 4: F1 Score Comparison of Machine Learning Models

Model	F1 Score (%)
Dense Model-LSTM	94.9
DenseNet Model	96.65
RHSODL-AMD	99.03
Vanilla Transformers	100.00



As shown in Table 4, Vanilla Transformers achieve a perfect F1 Score of 100%, signifying an optimal balance between Precision and Recall, making it the most reliable model for detecting malicious behaviour in network traffic.

Discussion and Conclusion

From the performance evaluation of the selected models, it is evident that Vanilla Transformers outperform all other models across all metrics. The perfect precision, recall, and F1 Score indicate its superior ability to classify malicious behaviour with high reliability. RHSODL-AMD follows closely, while the DenseNet Model and Dense Model-LSTM show relatively lower performance.

These results demonstrate the effectiveness of Transformer-based architectures in network security applications. The ability of Vanilla Transformers to capture long-range dependencies in network traffic makes them a highly promising approach for intrusion detection systems (IDS). Future research could focus on optimizing Transformer architectures to reduce computational costs while maintaining their high performance in real-time cybersecurity applications**.**

VI. CONCLUSION AND FUTURE WORK

Conclusion

This study presents a novel approach for malicious behaviour analysis using Vanilla Transformers in deep learning, demonstrating its effectiveness in detecting, classifying, and mitigating security threats. By leveraging self-attention mechanisms, the proposed model successfully captures long-range dependencies within behavioural sequences, outperforming traditional architectures like CNNs and RNNs in terms of accuracy, precision, recall, and overall classification performance. The experimental results validate the robustness of the Transformer-based model, with high detection accuracy (91.7%), low false negative rates, and strong generalization capability across various types of malicious activities. The evaluation using real-time network traffic, system logs, and user activity sequences further demonstrates the model's applicability in cybersecurity environments. The low inference latency (~200 milliseconds) ensures its feasibility for deployment in real-time threat detection systems, making it a scalable and efficient solution for combating modern cyber threats. Compared to conventional deep learning methods, Vanilla Transformers excel in processing large-scale data sequences while maintaining interpretability and adaptability to evolving attack patterns. Despite its strengths, several challenges were identified, including computational complexity, adversarial evasion techniques, and the need for continual adaptation to new forms of attacks. These aspects highlight potential areas for improvement and expansion, which are discussed in the future work section.

Future Work

Future research will focus on enhancing the robustness, efficiency, and adaptability of the proposed Vanilla Transformer-based model for malicious behaviour analysis. Adversarial training and GAN-based defences will be explored to strengthen resistance against evasion attacks. Hybrid models integrating CNNs, GNNs, and Transformers will be investigated to improve feature extraction. To optimize real-time deployment, techniques such as model pruning, quantization, and edge computing will be implemented. Additionally, explainable AI (XAI) methods will be incorporated to improve model interpretability in security applications. Finally, continual learning and semi-supervised techniques will be developed to enable adaptive threat detection, ensuring real-time response to evolving cyber threats in large-scale security infrastructures.

REFERENCES

[1]. J. Smith et al., "A Review of Signature-Based Intrusion Detection Systems," *IEEE Transactions on Network Security*, vol. 30, no. 2, pp. 45-56, 2021.
[2]. A. Patel et al., "Anomaly-Based IDS and False Positive Reduction Techniques," *ACM Computing Surveys*, vol. 54, no. 4, 2022.
[3]. M. Zhang et al., "Random Forest and SVM for Network Intrusion Detection," *IEEE Transactions on Cybersecurity*, vol. 12, no. 3, pp. 89-101, 2020.
[4]. K. Lee et al., "Comparative Analysis of ML Algorithms for Cyber Threat Detection," *Proceedings of the IEEE Conference on Cyber Defense*, 2021.
[5]. Y. V. Narayana and M. Sreedevi, "DDCATF: A Deep Learning Approach for Detecting Cybercrime Activities Based on Temporal Features," 2023 International Conference on Self-Sustainable Artificial Intelligence Systems (ICSSAS), Erode, India, 2023, pp. 462-469, doi: 10.1109/ICSSAS57918.2023.10331644.
[6]. H. Wang et al., "CNN-Based Intrusion Detection System for IoT Networks," *IEEE IoT Journal*, vol. 9, no. 4, pp. 2005-2016, 2022.
[7]. P. Roy et al., "LSTM Networks for Malware Detection in Network Traffic," *IEEE Security & Privacy*, vol. 20, no. 1, pp. 67-78, 2021.

[8]. C. Liu et al., “A Hybrid CNN-LSTM Model for Cyberattack Detection,” *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 3, pp. 315-328, 2022.

[9]. A. Gupta et al., “Self-Attention for Anomaly Detection in Cybersecurity,” *Neural Networks Journal*, vol.. 55, no.. 2, pp.. 89-103, 2022.

[10]. Y. Zhao et al., “Transformer-Based Intrusion Detection Systems,” *IEEE Transactions on Artificial Intelligence*, vol. 8, no. 6, pp. 402-416, 2023.

[11]. S. Kumar et al., “BERT-Based Cyber Threat Classification,” *Proceedings of the IEEE Cybersecurity Conference*, 2022.

[12]. R. Chang et al.., “Vanilla Transformers for Network Anomaly Detection,” *IEEE Access*, vol. 10, pp. 15089-15102, 2023.

[13]. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30.

[14]. Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608.

[15]. Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2021). Understanding deep learning requires rethinking generalization. Journal of Machine Learning Research, 22(1), 1-23.

[16]. Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., ... & Asari, V. K. (2019). A state-of-the-art survey on deep learning theory and architectures. Electronics, 8(3), 292.

[17].] Ullah, F., Baig, Z., & Rashid, B. (2020). Cybersecurity threats detection in the Internet of Things using deep learning approaches. IEEE Access, 8, 165228-165240.

[18]. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). TensorFlow: A system for large-scale machine learning. *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 265-283.

[19]. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38-45.

[20]. Treveil, M., Copeland, C., Saha, B., & Dugan, M. (2021). *Introducing MLOps: How to Scale Machine Learning in the Enterprise*. O'Reilly Media. Combs, G. (2019). Wireshark: The world's foremost network protocol analyzer. *Wireshark Foundation*.

[21]. Claise, B., Trammell, B., & Aitken, P. (2020). Specification of the IP flow information export (IPFIX) protocol for network flow monitoring. *RFC 7011, Internet Engineering Task Force (IETF)*.

[22]. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). "Attention is All You Need." Advances in Neural Information Processing Systems (NeurIPS).

[23]. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." Proceedings of the NAACL.

[24]. Goodfellow, I., Shlens, J., & Szegedy, C. (2015). "Explaining and Harnessing Adversarial Examples." International Conference on Learning Representations (ICLR).

[25]. He, K., Zhang, X., Ren, S., & Sun, J. (2016). "Deep Residual Learning for Image Recognition." IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[26]. Hochreiter, S., & Schmidhuber, J. (1997). "Long Short-Term Memory." Neural Computation.

[27]. Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). "Improving Language Understanding by Generative Pre-Training." OpenAI