ENHANCING KEYLOGGER DETECTION AND ANALYSIS WITH CNN FOR IMPROVED CYBERSECURITY

Dr. David Raj Micheal
Division of Mathematics
School of Advanced Sciences
Vellore Institute of Technology Chennai
Tamil Nadu – 600127
davidraj.micheal@vit.ac.in

A.Avinash
Division of Mathematics
School of Advanced Sciences
Vellore Institute of Technology Chennai
Tamil Nadu – 600127
avinash.a2023@vitstudent.ac.in

***Abstract*—Keyloggers are a form of spyware designed to capture and record users' keystrokes, often employed by cybercriminals to monitor and extract sensitive information from computing devices. These malicious tools can be classified into two primary categories: those that are directly injected into the user's device to intercept keystrokes, and those that focus on capturing credentials such as user IDs and passwords entered during login processes. The keystrokes are meticulously recorded and transmitted to the attackers, leading to potential theft of confidential data. In this study, we utilize a Convolutional Neural Network (CNN) model to enhance the detection and analysis of keylogger activities. The objective of this research is to develop a robust CNN-based approach for identifying and mitigating the impact of keyloggers, thereby improving cybersecurity measures and protecting user information from unauthorized access.**

***Index Terms*—Keywords- Keyloggers,Spyware,Keystroke capture,Cybersecurity,Convolutional Neural Network (CNN),Malware detection,Sensitiveinformation protection,Cybercrime,User authentication,Data theft prevention**

I. INTRODUCTION

In the digital age, cybersecurity threats are becoming increasingly sophisticated, and keyloggers have emerged as a significant menace. These insidious tools secretly capture and record users' keystrokes, allowing cybercriminals to steal sensitive information, such as passwords, credit card numbers, and personal data. The consequences of keylogger attacks can be devastating, ranging from identity theft to financial loss and compromised national security. Traditional detection methods often rely on signature-based approaches, which can be evaded by sophisticated keyloggers. Therefore, there is a pressing need for innovative solutions to combat this threat.

II. OBJECTIVES

The objective of this research is to develop and evaluate a Convolutional Neural Network (CNN) model designed to detect and analyze keylogger activities. By leveraging CNN's capabilities in identifying patterns and anomalies in keystroke data, this study aims to enhance the detection of keyloggers, improve the accuracy of identifying unauthorized data capture, and bolster cybersecurity defences. The ultimate goal is to provide a robust solution for mitigating the impact of keyloggers, thereby safeguarding sensitive user information and strengthening overall cybersecurity measures.

III. LITERATURE REVIEW

Singh, A., & Choudhary, P. (2021) Keyloggers are rootkit malware that covertly capture keystrokes to steal sensitive data like usernames, PINs, and passwords, posing a significant threat to activities such as e-commerce and online banking. While antivirus software can detect known keyloggers, it struggles with unknown variants. This paper reviews keylogger types, characteristics, methodologies, and explores detection techniques and proactive approaches for combating them. **Solairaj, A., Prabanand, S. C., Mathalairaj, J., Prathap, C., & Vignesh, L. S. (2016)**Software keyloggers covertly record keystrokes and pose significant threats to user privacy and security by running in stealth mode. They are difficult to detect because they don't show up in system icons, and use techniques like anti-hook methods and algorithms such as HoneyID and dendritic cell algorithms to monitor activity and strengthen detection.

Gunalakshmii, S., & Ezhumalai, P. (2014),This project aims to detect keyloggers and prevent sensitive data loss by analyzing application permissions using a black-box approach, focusing on behavioral characteristics. It develops a machine learning-based detection system for mobile phones to identify keylogger applications. **Tian, D., Jia, X., Chen, J., & Hu, C. (2017)**Developing an Artificial Immune System (AIS) based Intrusion Detection System (IDS) to detect API-based keyloggers in Linux virtual machines, using a collaborative architecture between host OS and VM to ensure kernel integrity and employing negative selection algorithm (NSA) to track events and detect anomalies.

Ahmed, Y. A., Maarof, M. A., Hassan, F. M., & Abshir, M. M. (2014) A type of rootkit malware that captures keystrokes, intercepts sensitive info, and transmits to attackers, posing a significant threat to online activities, with antivirus

software unable to detect unknown keyloggers, prompting the need for proactive techniques to combat this evolving threat. **Wazid, M., Katal, A., Goudar, R. H., Singh, D. P., Tyagi, A., Sharma, R., & Bhakuni, P. (2013)** A proposed framework to defend against malicious keylogger spyware that captures keystrokes, stores them in a log file, and emails it to attackers, posing a significant threat to online transactions, such as online banking.

Ayo, F. E., Awotunde, J. B., Olalekan, O. A., Imoize, A. L., Li, C. T., & Lee, C. C. (2023) A novel keylogger detection system using a two-level combinatorial algorithm, fuzzy logic, and color codes, achieving 96.54% accuracy and 95.5% F1 score, with reduced false alarm rates, by combining signature-based and anomaly-based detection methods. **Ahmed, M. B., Shoikot, M., Hossain, J., & Rahman, A. (2019)** A novel memory analysis-based method is proposed to detect advanced keyloggers that evade traditional API-based and network traffic monitoring methods, offering a user-friendly and permission-free solution for Linux and Windows OS with satisfactory success rates.

Wood, C., & Raj, R. (2010) This paper advocates for incorporating keylogger education in cybersecurity programs, providing an overview of keyloggers, their design and implementation, and effective detection and prevention methods, along with project ideas to educate students on this critical topic. **Alsubaie, M. S., Atawneh, S. H., & Abual-Rub, M. S. (2023)**A machine learning-based model is proposed to detect in-website keyloggers on platform-independent devices, using Random Forest, LightGBM, and CatBoost classifiers with Hybrid Ensemble Feature Selection (HEFS), achieving a 1.59% accuracy deterioration and 84.5% reduction in feature space with Random Forest.

IV. METHODOLOGY

The methodology for the project "Leveraging Deep Learning for Accurate Tea Leaf Disease Identification" involves several key steps, from data preparation and model development to training, evaluation, and visualization. Below is a detailed explanation of each step:

A. Data Collection

Data collection serves as the foundational step in any data-driven project, providing the essential raw materials for analysis and modeling. For this project, the data was meticulously gathered from various reliable and relevant sources, ensuring it captures the scope and requirements of the project objectives. The collection process was tailored to obtain a comprehensive dataset, encompassing diverse types of information that are pivotal to understanding the target domain and answering the project's research questions. By compiling a well-structured and representative dataset, the project ensures that the subsequent data preprocessing and analysis steps are based on high-quality information, enhancing the reliability of the final results. The collected data aligns closely with the project's goals and provides an in-depth basis for exploring patterns, relationships, and potential insights, which are essential for achieving meaningful outcomes in this data-driven endeavor.

B. Data Preprocessing

To streamline datasets, remove irrelevant columns (e.g., IDs, timestamps) to enhance interpretability. Handle missing values by first identifying NaNs, then removing rows or columns with excessive NaNs, or imputing missing data—use the mean, median, or mode for numerical data, and mode, forward-fill, or backward-fill for categorical data. Check the class distribution to address imbalances that could bias the model; if needed, apply techniques like oversampling, undersampling, or adjusting class weights. Remove features with low variance or high correlation to reduce redundancy, improve model simplicity, and reduce overfitting risks.

C. Feature Scaling

Feature scaling is essential in machine learning to ensure features contribute equally to model performance. The two main methods are Normalization and Standardization.

Normalization rescales feature values to a range between 0 and 1, which is useful for non-Gaussian data and distance-based algorithms like KNN and neural networks; its formula is

$$x' = \frac{x - \min(X)}{\max(X) - \min(X)}$$

Standardization adjusts features to have a mean of 0 and a standard deviation of 1, creating a Gaussian distribution. This is beneficial for algorithms assuming normally distributed data, like linear and logistic regression, with formula

$$x' = \frac{x - \mu}{\sigma}$$

where μ is the mean and σ is the standard deviation.

D. Traning methodology

CNN model on IoT network traffic data for optimal performance, start by splitting the dataset 70-20-10 for training, validation, and testing, and use 5-fold stratified cross-validation to retain class balance. Apply data augmentation with horizontal flipping and ±15° rotation, and use SMOTE to address class imbalance. For training, set a batch size of 32, train for 100 epochs with early stopping after 10 epochs (patience), and use the Adam optimizer with an initial learning rate of 0.001, applying binary cross-entropy for binary classification. Optimize the model using Bayesian tuning for hyperparameters like learning rate, batch size, and architecture, alongside 5-fold cross-validation. Early stopping is set with a 10-epoch patience and 0.001 minimum validation loss delta, saving model checkpoints based on minimal validation loss for reproducibility.

E. Evaluation Frameworks

IoT intrusion detection model uses both quantitative metrics and real-time testing to ensure robust performance. Key metrics include accuracy, precision, recall (focusing on minimizing false positives), F1-score, and ROC-AUC analysis to assess the model's discrimination ability, along with a confusion matrix for detailed performance across attack types. Real-time testing includes measuring inference time under network loads (100-1000 requests/second), monitoring CPU/RAM usage during peak traffic, testing scalability with load increases up to 10x, and evaluating detection latency under different network conditions and packet loss scenarios to ensure efficient deployment.

F. Implementation Details

IoT Intrusion Detection System utilizes Python 3.8+ with TensorFlow 2.x for deep learning and model training, Scikit-learn for preprocessing and metrics, and a CUDA-enabled GPU (8GB VRAM) for efficient training. Docker containers ensure consistent deployment across environments. The deployment strategy includes a REST API via Flask/FastAPI for model serving, real-time monitoring with Prometheus and Grafana, and an automated CI/CD pipeline for seamless model updates. A logging system is configured to track model predictions and monitor system health in the IoT environment, integrating seamlessly with existing infrastructure and enabling real-time network traffic analysis.

G. Validation and Testing

The model's generalization and robustness are validated using stratified k-fold cross-validation for balanced class distribution, statistical significance tests (t-test, chi-square) on performance metrics, and robustness checks across network conditions and attack patterns. Comparative benchmarking is done against signature-based IDS solutions. Testing scenarios include benign traffic patterns from different IoT devices, simulations of keylogging attacks and evasion techniques, edge cases like network congestion and packet fragmentation, and stress tests with high traffic volumes (¿1000 packets/second) to confirm the model's resilience in real-world situations.

H. Security Considerations

To maintain data privacy and security, implement end-to-end encryption for data transmission and storage, role-based access control for model management, and secure API endpoints with token-based authentication, complemented by regular security audits and vulnerability assessments. Ensure compliance with GDPR for data handling, NIST guidelines for IoT security, and ISO 27001 standards for information security. Comprehensive documentation of security protocols, privacy policies, and incident response procedures is essential for transparency, auditing, and adherence to regulatory frameworks, helping to prevent unauthorized access, tampering, and legal risks.

V. KEYLOGGER DETECTION AND ANALYSIS WITH CNN FOR IMPROVED CYBERSECURITY

Introduction to IoT Security:The Internet of Things (IoT) connects various devices, from smart household items to industrial machinery, over the internet. However, IoT devices are vulnerable due to design limitations prioritizing ease of use and cost-effectiveness. Common vulnerabilities include weak authentication (default or insecure passwords), unencrypted communication, and lack of regular firmware updates. Robust security in IoT is essential for data protection, operational continuity in critical sectors (e.g., healthcare and transportation), and regulatory compliance (e.g., GDPR, HIPAA).

Intrusion Detection Systems (IDS):An Intrusion Detection System (IDS) monitors network traffic to detect suspicious activities, analyzing data packets for potential threats and alerting security teams. IDSs are crucial for early detection of breaches. Types include Network-Based IDS (NIDS), which monitors traffic across all networked devices, Host-Based IDS (HIDS), focused on individual devices and their logs, and Hybrid IDS, combining both approaches for comprehensive monitoring.

Keylogging as a Threat:Keylogging attacks involve malware that records user keystrokes to capture sensitive data like passwords or personal information. Keyloggers often spread through phishing or malware and run undetected, posing risks such as data breaches, privacy violations, and potential identity theft. Keylogging incidents can erode user trust in IoT, impacting the adoption of these technologies.

VI. MODEL EVALUATION

A. Feature Scaling

The model takes 32x32x1 grayscale images as input. It starts with a Conv2D layer using 3 filters and ReLU activation, maintaining input dimensions with 'same' padding. Feature extraction uses pre-trained EfficientNetB0 with ImageNet weights and global average pooling. The output layer has a single neuron with sigmoid activation for binary classification. The model configuration includes binary cross-entropy loss, Adam optimizer (lr=0.0001), and AUC as the metric. The training setup uses a batch size of 1024, 50 epochs, and validation on the test set with callbacks for model checkpoint, early stopping, and learning rate reduction.

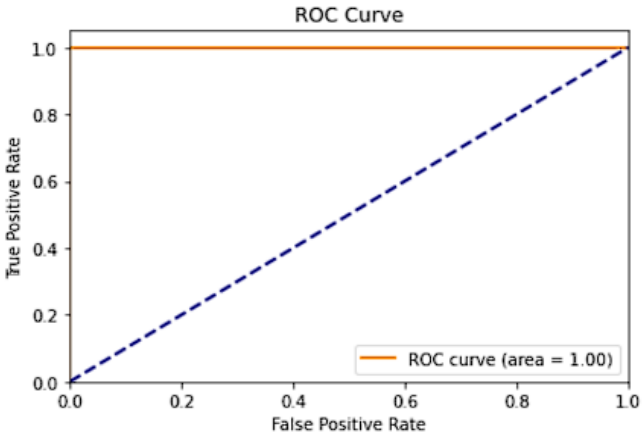
B. Model Evaluation Analysis

The model achieved an AUC score of 0.491, indicating poor performance, as it is slightly below 0.5, suggesting it performs worse than random guessing and fails to distinguish keylogging attacks from normal behavior. Potential issues include underfitting, data quality or preprocessing problems, unsuitable model architecture, or the need for training parameter adjustments.

VII. CONCLUSION

AUC of 0.49 (below 0.5). Curve closely follows random classifier line. No clear separation between classes. The ROC curve analysis conclusively shows that the current model is

ineffective for keylogging detection. With an AUC of 0.49, the model's performance is slightly worse than random chance, making it unsuitable for practical cybersecurity applications. Substantial improvements are needed in both model architecture and training approach to achieve acceptable performance for real-world deployment.



VIII. REFERENCES

1) Singh, A., & Choudhary, P. (2021, August). Keylogger detection and prevention. In Journal of Physics: Conference Series (Vol. 2007, No. 1, p. 012005). IOP Publishing.

2) Solairaj, A., Prabanand, S. C., Mathalairaj, J., Prathap, C., & Vignesh, L. S. (2016, January). Keyloggers software detection techniques. In 2016 10th International Conference on Intelligent Systems and Control (ISCO) (pp. 1-6). IEEE.

Gunalakshmii, S., & Ezhumalai, P. (2014, February). Mobile keylogger detection using machine learning technique. In Proceedings of IEEE International Conference on Computer Communication and Systems ICCCS14 (pp. 051-056). IEEE. Tian, D., Jia, X., Chen, J., & Hu, C. (2017). An Online Approach for Kernel-level Keylogger Detection and Defense. J. Inf. Sci. Eng., 33(2), 445-461.

Ahmed, Y. A., Maarof, M. A., Hassan, F. M., & Abshir, M. M. (2014). Survey of Keylogger technologies. International journal of computer science and telecommunications, 5(2).

Wazid, M., Katal, A., Goudar, R. H., Singh, D. P., Tyagi, A., Sharma, R., & Bhakuni, P. (2013, January). A framework for detection and prevention of novel keylogger spyware attacks. In 2013 7th International Conference on Intelligent Systems and Control (ISCO) (pp. 433-438). IEEE.

Ayo, F. E., Awotunde, J. B., Olalekan, O. A., Imoize, A. L., Li, C. T., & Lee, C. C. (2023). CBFISKD: A combinatorial-based fuzzy inference system for keylogger detection. Mathematics, 11(8), 1899.

Ahmed, M. B., Shoikot, M., Hossain, J., & Rahman, A. (2019). Key logger detection using memory forensic and network monitoring. International Journal of Computer Applications, 975, 8887.

Wood, C., & Raj, R. (2010, July). Keyloggers in Cybersecurity Education. In Security and Management (pp. 293-299).

Alsubaie, M. S., Atawneh, S. H., & Abual-Rub, M. S. (2023). Building Machine Learning Model with Hybrid Feature Selection Technique for Keylogger Detection. International Journal of Advances in Soft Computing & Its Applications, 15(2).