MACHINE LEARNING BASED KEY LOGGER DETECTION IN MOBILE

By
SAJID KHAN

NATIONAL UNIVERSITY OF MODERN LANGUAGES ISLAMABAD

September, 2024

Machine learning based keylogger detection in mobile

By

SAJID KHAN

MSEE, National University of Modern Languages, Islamabad, 2024

A THESIS IN PARTIAL FULLFILMENT OF THE REQUIRMENT FOR THE

DEGREE OF

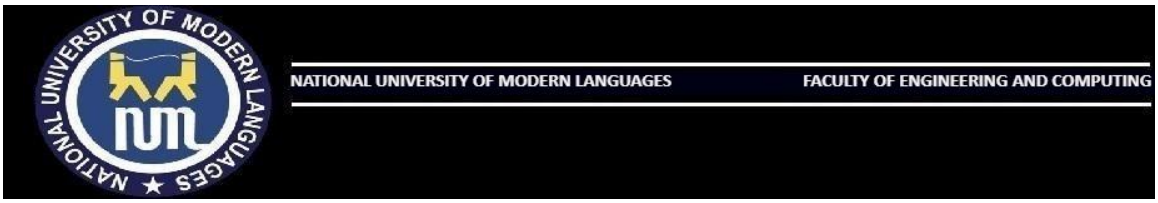
MASTER OF SCEINCE

In Electrical Engineering

To

FACULTY OF ENGINEERING AND COMPUTING

MSEE, NATIONAL UNIVERSITY OF MODERN LANGUAGES, ISLAMABAD,



THESIS AND DEFENSE APPROVAL FORM

The undersigned certify that they have read the following thesis, examined the defense, are satisfied with overall exam performance, and recommend the thesis to the Faculty of Engineering Computing.

Thesis Title: Machine learning based keylogger detection in mobile

Submitted by: Sajid khan Registration #: NUML-S21-008

Master of Science in Electrical Engineering

Electrical Engineering

Discipline

Dr. Madah Ul Mustafa

Research Supervisor Signature of Supervisor

Dr. Farhan Sohail

Research Co-Supervisor Signature of Co- Supervisor

Dr. Farhan Sohail

HOD (EE) Signature of HOD (EE)

Dr. Noman Malik

Dean (FEC) Signature of Dean (FEC)

Sep 4th 2024

Date

AUTHOR'S DECLARATION

I Sajid khan
Son of Alam Zeb
Registration # NUML-S21-008
Discipline Electrical Engineering

Candidate of **Master of Science in Electrical Engineering (MSEE)** at the National University of Modern Languages do hereby declare that the thesis **Machine learning based keylogger detection in mobile** submitted by me in partial fulfillment of MSSE degree, is my original work, and has not been submitted or published earlier. I also solemnly declare that it shall not, in future, be submitted by me for obtaining any other degree from this or any other university or institution. I also understand that if evidence of plagiarism is found in my thesis/dissertation at any stage, even after the award of a degree, the work may be cancelled and the degree revoked.

Signature of Candidate

Sajid Khan
Name of Candidate

4th Sep, 2024
Date

Dedication

"To my father, who taught me that the best kind of knowledge is that which is learned for its own sake, and to my mother, who showed me that even the largest task can be accomplished when taken one step at a time, this thesis is dedicated."

Acknowledgments

For the successful completion of this project, We are thankful to Almighty Allah, for enabling us to complete this project and making everything possible for the project to be a success.

We would like to express our sincere gratitude to our project supervisor, Dr.Madah Ul Mustafa for his sincere guidance, successive cooperation and useful suggestions.

We are thankful to all the "faculty of engineering and computing" for providing valuable information and for helping us. We are also thankful to the National University of Modern Languages Islamabad for providing a platform to successfully complete this project.

Abstract

As information technology evolves, cybersecurity professionals must ensure security and privacy. Recent research shows a rise in new malware strains, with keyloggers becoming particularly sophisticated. This malicious software can discreetly record every keystroke on a device, giving attackers access to crucial data without the owner's approval. Keyloggers must be identified to prevent data loss and unauthorized disclosure. Antivirus systems can be ineffective against novel keyloggers that are not known threats. These systems detect threats using heuristic and behavioral analysis. Machine learning and deep learning algorithms may solve cybersecurity problems. These algorithms can detect several threats, including keyloggers that exploit weaknesses. However, these solutions are not a panacea for security challenges, and their efficacy depends on many factors. In this study, we proposed a hybrid deep learning model based on CNN Convolutional Neural Network and long short-term memory networks LSTM. CNNs are used to predict keylogger attacks using several feature engineering methodologies where LSTM works on classification. Feature engineering preprocessed the dataset by reducing unnecessary features, fixing imbalances, and scaling features. With only 10 epochs, the training approach reached 99% accuracy and good performance. This shows that the CNN-based technique can predict keylogger attacks and that feature engineering improves model performance.

Keywords: Keyloggers. Convolutional Neural Network, Cybersecurity, Feature Engineering, CNN, LSTM.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

1. INTRODUCTION

1.1. Overview

Security, often known as cybersecurity, is a multidimensional field that aims to protect computer systems, networks, and data against destructive acts that are carried out by threat actors. Within this domain, numerous types of malicious software, phishing campaigns, denial-of-service attacks, and insider threats all provide major dangers to the confidentiality, integrity, and availability of information [1]. Malware is an umbrella term that comprises a wide range of malicious software, such as viruses, worms, trojan horses, ransomware, spyware, and adware. Each of these types of software poses a unique risk to the security of digital transactions. Worms, for example, are able to transmit themselves over networks on their own, whereas viruses are able to disseminate themselves through infected files or applications. Trojan horses are malicious programs that cloak themselves as genuine software in order to obtain unauthorized access and take advantage of user trust [1]. Ransomware, on the other hand, encrypts files and demands payment in exchange for the keys to unlock them, whereas adware bombards users with adverts that they do not want to see, which frequently has an effect on the performance of the system [1]. In order to effectively combat these attacks, it is necessary to use a holistic approach that incorporates firewalls, antivirus solutions, intrusion detection systems, and user education in order to raise knowledge of recommendations for best practices in cybersecurity [2]. Keyloggers, which are often referred to as keystroke logging programs, are a particularly sneaky type of malware that is designed to covertly record the keystrokes of users. This type of malware poses significant threats to the confidentiality of important information [3]. As a result of the widespread use of computers for a variety of online activities, the number of efforts to engage in keylogging has increased, highlighting the critical need for effective defenses. In light of this, education in cybersecurity ought to incorporate a

comprehensive investigation of keyloggers and anti-keylogging strategies in order to address a number of essential goals [3]. One of the primary benefits of studying keyloggers is that it provides students with the opportunity to acquire knowledge regarding the goals of cyber attackers, the complex nature of malware, and the procedures that are utilized to corrupt and control computers [3]. Furthermore, this instructional focus provides students with the tools and methodologies that are essential for detecting and blocking keyloggers. This is in recognition of the dynamic nature of current malware, which frequently evades standard static detection approaches [4].

Recent investigations conducted by well-known cybersecurity companies provide more evidence that hostile actors continue to make use of keyloggers [5]. An alarming pattern that has been seen over the course of the past few years is indicated by the findings of an investigation conducted by VeriSign, which demonstrates a significant rise in the prevalence of malware that incorporates keylogging capabilities [5]. Similar to the previous example, research conducted by Symantec highlights the widespread use of keyloggers by hackers in order to illegally gather personal user data. This research also highlights the necessity of addressing this threat in the context of cybersecurity education and practice [5]. Furthermore, the findings of John Bambenek of the SANS Institute highlight the astonishing financial losses that can be attributed to keyloggers. This further emphasizes the necessity of combatting this threat through the implementation of complete cybersecurity measures [5].

1.2. Background

The author in [6] has effectively leveraged deep learning and natural language processing (NLP) techniques to develop an advanced system for categorizing textual data in log files. This innovative solution significantly enhances workflow efficiency by reducing the time required for manual log file review [6]. Their focus on text classification encompasses key areas such as intent analysis, emotion detection, and sentiment evaluation, which have garnered substantial interest within the machine learning

community. As dedicated practitioners in the field of text data, they continuously explore the diverse tools and methodologies provided by NLP. However, while the author highlights the use of NLP for log file classification, specific details regarding the classification architecture, model parameters, and other technical aspects are not provided [7].

Malware is software designed to steal data, encrypt files for ransom, or create botnets. Trojans, viruses, keyloggers, rootkits, worms, and spyware are all classified as malware [8]. Malware, which evolved from the 1998 "Morris Worm" to attack vulnerabilities and steal user data, remains a major security issue [9]. Smart technologies have transformed our lives, yet keyloggers on mobile devices have increased security risks. Surreptitiously monitoring keyboard inputs may compromise sensitive data [8]. Malware detection used signature-based scanning, but AI became essential when polymorphism and obfuscation allowed malware to escape detection. AI can use deep learning and supervised learning to classify malware based on its behavior [10]. Keyloggers were meant for technical troubleshooting but are now used for unwanted monitoring. They record keystrokes and send them to distant servers in mobile devices, threatening critical data. Machine learning may identify malicious from conventional apps [11]. Keylogger malware is difficult to detect due to its stealthy activity, evasion of standard detection methods, and potential installation by genuine users. For reliable and efficient identification, machine learning and behavior-based analysis are used [11]. Keylogger identification is done using an SVM model in this study. SVM is a powerful machine learning method for malware identification and classification. SVM models can reliably categorize new keyboard sequences, providing reliable detection and prevention in the dynamic cybersecurity landscape. They are trained on a carefully annotated dataset of valid input and keylogger-generated keystrokes.

1.3. Motivation

Cyberattacks, particularly keyloggers, are an increasing threat in today's digital landscape. These hidden keyloggers can compromise sensitive data, posing significant risks to both individuals and businesses. Traditional cybersecurity methods often fall short in detecting these stealthy attackers, necessitating the development of sophisticated detection systems with high accuracy. Our robust model leverages CNNs and LSTM networks to extract hierarchical features from sequential data and capture temporal dependencies, resulting in more precise detection of keylogger activity. In an ever-evolving digital environment, advanced neural network architectures are essential to bolster cybersecurity defenses and safeguard against emerging cyber threats.

1.4. Problem Statement

Cybercriminals develop new methods to hack networks and consumer gadgets as technology advances. Intercepting keystrokes to modify login details gives illegal access to bank accounts and email credentials. Traditional machine learning methods like SVMs and decision trees struggle to detect keyloggers [12,13]. However, deep learning methods like sequential models and CNNs may help detect and mitigate keyloggers.

1.5. Aim and Objectives

1.5.1. Aim

The aim of this research is to develop an improved method for keyloggers detection using machine learning techniques, the objective is to enhance the existing approaches that are used to detect fraudulent online payment transactions.

1.5.2. Objectives

- To design and develop an innovative fraud detection framework that integrates advanced methods, such as anomaly detection and real-time monitoring, to improve the detection and prevention of online payment fraud.

- To evaluate the effectiveness of feature engineering methodologies in preprocessing datasets, addressing feature imbalances, and improving the predictive capabilities of deep learning models in detecting keyloggers.
- To assess the performance of current machine learning and deep learning algorithms in detecting online payment fraud, emphasizing their accuracy, scalability, and ability to adapt to emerging fraud tactics.
- To develop and implement a CNN-based framework for predicting keylogger attacks, leveraging advanced feature engineering techniques to enhance model accuracy and performance.

Chapter 1 explores cybersecurity challenges, focusing on malware, particularly keyloggers, and various cyberattack methods. It emphasizes the importance of protecting systems, networks, and data from threats like malware, phishing, and insider attacks. Comprehensive cybersecurity measures, including firewalls, antivirus solutions, intrusion detection systems, and user awareness, are highlighted.

This chapter also examines the role of machine learning, such as Support Vector Machines (SVM), in detecting keyloggers and evolving threats. Motivated by the growing sophistication of cyber-attacks, this study aims to enhance online payment fraud detection using advanced techniques like CNNs and LSTMs.

This chapter concludes with objectives to review traditional methods, analyze fraud patterns, and propose innovative technologies for combating financial cyber threats.

CHAPTER 2

2. RELATED WORK

2.1. Literature Review

A comprehensive literature review is needed to understand the subtle nuances of different types and ways for detecting online payment fraud that are being in practice as on today. The review will be as extensive and comprehensive using various sources, including academic research papers, industry reports, or esteemed publications. This study aims to address this limitation by synthesizing the current body of knowledge and, identify strengths, weaknesses and gaps with respect to existing fraud-detection techniques.

A full overview of the area will be provided by the literature study, which will go into a variety of aspects, including the following points:

2.2. Machine learning and Deep Learning Approaches

In the field of fraud detection, machine learning algorithms have been proven as one of the powerful tools which can significantly improve accuracy and flexibility in detecting fraudulent activities specifically for online payment systems. In the literature, an extensive study has been done for performance analysis of various machine learning models along with feature selection methods and ensemble techniques to check their capabilities in this important area [15]. Online payment fraud detection has been extensively studied using a number of supervised learning methods including logistic regression, decision trees and random forests, support vector machines (SVM), as well as neural networks. All these approaches use labelled datasets to train models that will learn how to distinguish fictitious transactions from real ones. Studies have investigated comparing the performance of these models in terms of several metrics such as precision, recall and F-1 Score to find out which one is most suitable for fraud detection while minimizing false positives (f-p), a wrong judgment on acceptable transactions) and negatives. The interest in unsupervised learning,

such as k-means and DBSCAN for clustering anomalies on transaction data without labeled fraudulent examples has also brought these techniques to the spotlight. They are especially effective in finding out new attacks even for unseen past data, which is pretty essential to detect the emerging threat of fraud on online payment systems [16].

Unlike purely supervised learning, in such situations we have a small amount of labeled data and what is commonly done is to combine the supervisory signals that are represented both by those labels (on top of which you can optimize) as well somewhat less structured information present in our unlabeled instances within this unsupervised environment. This hybrid training methodology allows the system to learn on examples of both fraudulent and authentic transactions, thereby improving its ability to generalize new fraud instances while reducing extensive labeling efforts.

In addition to this, ensemble methods like bagging and boosting averaging multiple models are also found in the literature enhance robustness of fraud detection solutions (Khamse-Ashari 2013) with respect to different evaluation metrics. Bagging, boosting and stacking ensemble techniques that benefit from diversity among individual models allowing for better performance in terms of classification accuracy and reduced over-fitting. In this paper we investigate such performance of the overall accuracy when using each machine learning as built a model, and hope to identify their performances in enhancing fraud detection rate on online payment systems. Organizations can provide better ways to combat frauds using hybrid approach of bunching together strengths and or advantages any algorithms,feature selection methods, ensemble techniques.

2.3. Behavioral Analysis

Behavioral analysis approaches play a crucial role in fraud detection by monitoring user actions and identifying deviations from established behavioral norms that may signal fraudulent activities. The evaluation of relevant literature encompasses a diverse range of methodologies, including user profiling, session analysis, clickstream analysis, and other

behavioral analysis techniques. Through a comprehensive review, the effectiveness of these methods in improving the accuracy of fraud detection is thoroughly examined.

User profiling involves creating profiles of individual users based on their historical behavior, transaction patterns, demographics, and other relevant characteristics [5]. By analyzing deviations from established profiles, such as sudden changes in spending habits or unusual transaction times, fraud detection systems can flag potentially fraudulent activities for further investigation. Research in this area focuses on the development of robust profiling techniques that can accurately capture the unique behaviors of individual users while minimizing false positives. Session analysis extends beyond individual transactions to examine the sequence of actions performed by users during a single session or interaction with an online platform. By analyzing session data, including login times, page navigation patterns, and interaction durations, fraud detection systems can identify anomalous behaviors that may indicate unauthorized access or fraudulent activity. Session analysis techniques aim to differentiate between legitimate user sessions and those associated with fraudulent behavior, thereby improving the accuracy of fraud detection systems [17].

2.4. Anomaly Detection

Anomaly detection methods play a pivotal role in identifying irregularities within transaction data, particularly in the context of detecting fraudulent online payments [7]. Statistical methods, such as z-score analysis and time-series analysis, offer simplicity and efficiency in pinpointing outliers based on deviations from expected norms. Clustering techniques, like k-means and density-based clustering, excel in uncovering groups of transactions that deviate from typical patterns, thereby flagging potentially fraudulent activities. Neural networks, including autoencoders and deep learning architectures, provide a powerful means of capturing intricate patterns and relationships within transaction data, enhancing the detection of subtle anomalies associated with fraudulent behavior. However, each of these approaches has its limitations. Statistical methods may

struggle with non-linear or complex anomalies that do not adhere to standard distributions. Clustering techniques may encounter challenges with high-dimensional data and require careful parameter tuning to achieve optimal results. Neural networks, while capable of capturing complex patterns, often demand large amounts of labeled data and substantial computational resources for training and inference. Additionally, all anomaly detection methods face the challenge of balancing detection accuracy with false positive rates, necessitating careful consideration of the trade-offs involved in fraud detection systems [18].

2.5. Malware Analysis Overview

In this part, we will see the importance of feature engineering in fraud detection as they are directly impacting the performance of your machine learning model. Feature engineering is at the core of this task and can greatly improve detection accuracy, precision, recall by transforming raw data into insightful features.

2.5.1. Techniques Used in Literature

1. **Feature Creation and Selection:**
    - Based on the information we receive in transaction frequency, average transaction amount and time based features (time of day / dyas of week) etc to derive new feature.
    - Feature selection: It includes (i) using the correlation analysis, ii) mutual information iii), feature importance scores obtained from tree-based models etc.
2. **Aggregation and Statistical Features:**
    - Aggregating transaction data over different time windows to capture patterns, such as total amount spent per day or number of transactions per week.

- Calculating statistical measures like mean, median, variance, and standard deviation for transaction amounts and frequencies.

3. **Behavioral Features:**
    - Analyzing user behavior patterns, such as typical spending habits, common transaction locations, and device usage.
    - Identifying anomalies in behavior that may indicate fraudulent activity.
4. **Textual and Categorical Feature Encoding:**
    - Encoding categorical variables using techniques like one-hot encoding, label encoding, and target encoding.
    - Using natural language processing (NLP) techniques to analyze textual data, such as transaction descriptions or user comments.
5. **Time-Series and Sequential Features:**
    - Capturing temporal dependencies in transaction data using techniques like rolling windows and lag features.
    - Employing sequence-based models, such as recurrent neural networks (RNNs), to detect patterns over time.

2.5.2. Comparison of Techniques

- **Feature Creation vs. Aggregation:**
    - Feature creation focuses on deriving new variables from raw data, providing a more granular view, whereas aggregation condenses information over time windows, offering a macro perspective.
- **Statistical Features vs. Behavioral Features:**
    - Statistical features provide quantitative measures of transaction patterns, which are effective for detecting anomalies. In contrast, behavioral features capture qualitative aspects of user behavior, which can identify more subtle forms of fraud.
- **Encoding Techniques:**

- One-hot encoding and label encoding are simple and effective for small categorical datasets, while target encoding is useful for high-cardinality categories. NLP techniques add a layer of sophistication for textual data but require more computational resources.

- **Time-Series Features:**
  - Rolling windows and lag features are effective for capturing short-term trends, while sequence-based models are better suited for detecting complex temporal patterns over longer periods.

2.6. Keyloggers

A keylogger is precisely what its name suggests: a program designed to silently record and store every keystroke made on a computer's keyboard. The inherent danger of having a keylogger virus on your computer lies in its ability to covertly capture every piece of text you input through your keyboard, including sensitive information such as passwords and usernames. What makes matters worse is that some keyloggers are Trojan keyloggers, which are concealed within seemingly harmless programs. These Trojan horse viruses are deceptive in nature, as they disguise themselves as regular, sometimes fully functional applications, making it appear as though nothing malicious has been installed on your computer [20-22]. These Trojan keyloggers often go by various names, such as keystroke malware, keylogger viruses, or Trojan horse keyloggers. They can also be considered a subset of Trojan viruses, specifically designed for the purpose of surreptitious surveillance, earning them the moniker of "child of Trojan" surveillance spyware.

Keyloggers are a significant concern in cybersecurity. They record keystrokes on computers or mobile devices, potentially compromising sensitive information. Various techniques and algorithms have been proposed for their detection:

2.6.1. PCC Algorithm:

The Pearson Product Correlation Coefficient (PCC) is used for keylogger detection in some studies. It can effectively distinguish keylogging behavior when applied to the data [23].

- **Kernel-Based Behaviour Analysis:** This approach involves scrutinizing the behaviour of applications to detect malicious behaviours. It is particularly useful for security assessments of Android applications [24].

2.6.2. Dynamic Taint Analysis:

Dynamic taint analysis is employed for detecting kernel-level keyloggers. It can differentiate between bit-level keylogging and underlying drivers. Machine Learning: Machine learning models, such as Support Vector Machine (SVM) and Random Forest, have been applied to detect keyloggers with promising results.

Keyloggers pose a significant threat to users due to their ability to secretly capture keystrokes and compromise sensitive information. Unlike other types of malicious software, keyloggers do not directly harm the system itself but instead target the user's data. Here's a rewritten version of the text. Keyloggers are a distinct cybersecurity threat that may not harm computer systems directly, but they pose a grave danger to users by surreptitiously intercepting keystrokes and potentially compromising confidential information entered through the keyboard [23]. This clandestine activity enables cybercriminals to obtain critical data, including passwords, PIN codes, and other sensitive information. The repercussions of such breaches are wide-ranging, extending from financial losses, such as unauthorized transactions from the victim's account or unauthorized access to online gaming accounts, to more severe consequences [24].

Keyloggers can be used as instruments in a variety of forms of espionage, including industrial and political espionage, in addition to causing individuals to suffer financial losses. It is possible for them to facilitate the theft of confidential government information as well as sensitive commercial data, putting the safety of both private businesses and state-

owned organizations at jeopardy. It is possible, for instance, that they will steal secret encryption keys, which can have far-reaching consequences [25].

Keyloggers, phishing attacks, and social engineering tactics are the principal strategies that are now being utilized in the field of cyber fraud. The protection of users against keyloggers is more difficult than the protection against phishing, which may be accomplished by recognizing and avoiding phishing emails and abstaining from entering personal information on websites that appear to be dubious. The implementation of comprehensive security measures is typically the only effective countermeasure that can be taken. It is practically impossible for consumers to detect the presence of keyloggers on their computers since keyloggers frequently function in a hidden manner [26].

According to [27-28], who oversees Brazil's Computer Emergency Response Team under the country's Internet Steering Committee, keyloggers have eclipsed phishing as the most widely deployed method for stealing sensitive information. Additionally, keyloggers are continuously developing and becoming more complex as time goes on. They now have the capacity to monitor the websites that the user visits and selectively collect keystrokes that are entered on websites that are of particular interest to the cybercriminal. In recent years, there has been a considerable increase in the prevalence of a variety of harmful programmed that contain keylogging capabilities. Each of these programs has become increasingly widespread. Every single person who uses the internet is susceptible to the dangers that are posed by cybercriminals, regardless of where they are located or the organizations that they are involved with.

It's important to note that not all programs with keylogging capabilities are inherently malicious. Some businesses employ keystroke logging programs to monitor their employees' computer usage, and various parental control software also logs a child's internet activity. These legitimate use cases are not considered malicious keyloggers. So, what exactly does a keylogger Trojan do? A keylogger, in its malicious form, diligently monitors and records every keystroke it can detect. Once it infiltrates a system, it proceeds to track and store the gathered information locally. In some cases, the hacker behind the

keylogger may need physical access to the compromised computer to retrieve the logged data [29].

Alternatively, the keylogger can transmit the recorded logs over the internet to the attacker. This means that if you have fallen victim to a keylogger virus and are using your keyboard to input information anywhere, the virus is likely privy to it. This holds true whether you're working within an offline program like Microsoft Word or accessing online platforms such as your bank or social media accounts. Some sophisticated keystroke malware can even remain dormant until specific actions are detected [30]. The easiest path for a keylogger Trojan to take is when your antivirus software is outdated, turned off, or perhaps not even installed on your system. Consequently, these keyloggers can easily bypass the antivirus software if it lacks the knowledge to protect your computer effectively. Keyloggers are typically downloaded through executable files (EXE files). This is the standard format for most programs on your computer, making it impractical to avoid all EXE files in an attempt to steer clear of keyloggers [31-33].

A number of preventative measures ought to be performed in order to improve software security and reduce the likelihood of malicious software, which may include spyware and keyloggers. When downloading software, make sure to only do it from reliable sources, such as official app stores and respected websites. Avoid downloading software from random or torrent sources. Verify the digital signatures of the apps you have downloaded to guarantee their legitimacy, and read reviews written by other users for additional assurance. In order to fix security vulnerabilities, it is important to keep operating systems and software up to date. Additionally, it is important to install trustworthy antivirus and anti-malware applications in order to detect and remove harmful software. Additionally, it is important to exercise caution while dealing with email attachments, particularly executable files that come from unknown senders. A firewall should be enabled to monitor network traffic and prevent unwanted access [37]. The best way to protect oneself against hackers is to educate yourself on the main social engineering techniques they use and to frequently back up essential data to cloud storage or external

sources. A typical user account should be used for everyday work in order to reduce the risk of virus harm. Additionally, it is important to be aware of browser extensions and permissions that are sought by software that has been installed. In addition, whenever it is feasible, implement two-factor authentication for online accounts. This will add an additional layer of protection against illegal access.

Keyloggers are used by cybercriminals to illegally get sensitive information by secretly recording the keystrokes performed by users who are unaware of the activities being recorded. In the case of phishing schemes that targeted customers of Nordea bank, for example, malicious emails were sent to users, prompting them to install an anti-spam program that embedded the Haxdoor Trojan to record login credentials throughout the process of registering for online services [34]. In a similar manner, the Mydoom worm carried out a distributed denial of service attack while covertly recording sensitive information such as credit card details from computers that were infected [36]. Trojan horses that contained embedded keyloggers were used in targeted assaults, such as those that were carried out on the London headquarters of Sumitomo Mitsui, in order to monitor and steal login credentials, thereby permitting unauthorized access to accounts [37]. Additionally, in instances of industrial espionage, it is quite probable that keyloggers were utilized in order to steal critical information from the corporations that were the targets of the espionage. This demonstrates the widespread and detrimental impact that malware of this kind has on compromising cybersecurity and undermining the integrity of organizations [38].

2.7. Countermeasures

As the threat of spyware continues to evolve and becomes more sophisticated, several techniques have emerged to combat it effectively. These encompass both software programs designed for spyware removal or blocking, and various user practices aimed at reducing the risk of spyware infiltrating a system. Anti-spyware programs employ two primary methods to tackle spyware:

2.7.1. Real-time Protection:

Anti-spyware programs can offer real-time protection against the installation of spyware software on your computer. This functionality parallels that of antivirus software. The anti-spyware software continuously scans all incoming network data for any signs of spyware software and promptly blocks any identified threats.

2.7.2. Detection and Removal:

Alternatively, anti-spyware software can be utilized exclusively for detecting and removing spyware software that has already infiltrated user computer. This approach is generally more user-friendly and widely adopted. With this type of spyware protection software, we can schedule regular scans of over computer, such as weekly, daily, or monthly, to identify and eliminate any spyware software present. This anti-spyware software conducts thorough scans of the Windows registry, operating system files, and installed programs on your computer, generating a list of detected threats. we can then decide which items to delete and which to retain based on the scan results [41].

2.8. Comprehensive Analysis of Fraud Detection Techniques: Advantages, Challenges, and System Integrations

When addressing fraud detection techniques, it's crucial to analyze and compare the methods in terms of their references, advantages, disadvantages, and limitations. Here's a detailed exploration of various common fraud detection techniques:

2.8.1. Rule-Based Systems

RBS are traditional methods used extensively in banks and financial institutions. They are simple to implement and understand, making them effective for catching known types of fraud that follow predictable patterns. However, these systems are rigid and not adaptable to new or evolving fraudulent strategies without manual updates. Their main limitations include the inability to detect new, unseen types of fraud and a high rate of false positives as the fraud landscape evolves.

2.8.2. Machine Learning Models

ML Models represent modern data-driven approaches that are widely adopted in various sectors. These models are capable of learning and adapting from new data, detecting complex patterns of fraudulent behavior, and can handle large volumes of data efficiently. Despite their advantages, machine learning models require significant data preprocessing and feature engineering and may be opaque in their decision-making processes, often referred to as the black-box issue. They depend on large, labeled datasets for training and risk model decay over time if not continuously updated with new data.

2.8.3. Anomaly Detection Systems

ADS are designed to identify outliers in data that could indicate fraudulent activity. They are effective in detecting fraud that deviates from normal behavior patterns and can be unsupervised, not requiring labeled data for training. However, they suffer from higher false positive rates, as not all anomalies are fraudulent, and can miss frauds that mimic normal behavior patterns. Determining the threshold for what constitutes an anomaly and adjusting it based on evolving data patterns can be challenging.

2.9. Fraud Detection Techniques: Overview, Pros, Cons, and Limitations

2.9.1. Rule-Based Systems

Pros:

- Easy to implement and understand.
- Effective for detecting well-known and simple fraud patterns.
- Provides immediate alerts for predefined suspicious activities.

Cons:

- Inability to adapt to new and evolving fraud tactics.

- High number of false positives, leading to wasted resources.
- Requires constant maintenance and updating of rules.
- May overlook complex and sophisticated fraud schemes.

Limitations:

- The static nature of predefined rules makes it challenging to keep pace with rapidly changing fraud patterns.
- Balancing thresholds to minimize false positives and false negatives requires continuous fine-tuning.

2.9.2. Machine Learning-Based Systems

Pros:

- Adaptability to new and evolving fraud patterns.
- Ability to learn from data and improve over time.
- Can detect complex and subtle fraudulent activities.
- Reduces false positives by distinguishing between legitimate and fraudulent transactions more accurately.

Cons:

- Requires large amounts of labeled data for training.
- Computationally intensive and may require significant processing power.
- May produce opaque models that are difficult to interpret.

Limitations:

- The effectiveness of machine learning models depends on the quality and quantity of training data.
- Risk of overfitting to training data, leading to reduced performance on unseen data.
- Initial setup and model training can be resource-intensive.

2.9.3. Hybrid Systems (Combining Rule-Based and Machine Learning)

Pros:

- Leverages the strengths of both rule-based and machine learning approaches.
- Provides immediate detection through rules while continuously adapting to new fraud patterns through machine learning.
- Balances precision and recall, reducing false positives and negatives.

Cons:

- Increased complexity in implementation and maintenance.
- Requires integration and synchronization between rule-based and machine learning components.
- May still struggle with very novel or sophisticated fraud tactics that fall outside both predefined rules and learned patterns.

Limitations:

- The need for continuous coordination and fine-tuning between rule-based rules and machine learning models.
- Potentially higher resource requirements due to the combination of both techniques.
- Challenges in ensuring seamless cooperation between static rules and dynamic learning models.

2.9.4. Deep Learning-Based Systems

Pros:

- High accuracy in detecting complex fraud patterns.
- Capable of processing large volumes of data efficiently.
- Can automatically extract relevant features from raw data.

Cons:

- Requires substantial computational resources and specialized hardware.
- May have longer training times compared to traditional machine learning models.
- Often seen as a "black box," making the results difficult to interpret and explain.

Limitations:

- The need for extensive labeled data for effective training.

- Risk of overfitting if not properly regularized.
- High implementation and operational costs.

2.9.5. Anomaly Detection Techniques

Pros:

- Effective at identifying unusual patterns that may indicate fraud.
- Can be used with unsupervised learning, requiring no labeled data.
- Adaptable to various types of data and fraud scenarios.

Cons:

- May generate false positives by flagging legitimate but unusual transactions.
- Requires fine-tuning to balance sensitivity and specificity.
- May miss fraud patterns that appear normal within the data distribution.

Limitations:

- The challenge of distinguishing between benign anomalies and actual fraudulent activities.
- May require domain expertise to interpret and act on detected anomalies.
- Dependence on the quality of the data and the chosen detection algorithm.

2.10. Comparison Table

Tabel 1: Compare with Previous Work

Year	References	Title	Summary
2019	[35]	A modified framework to detect keyloggers using machine learning algorithm	In this research used ResNet-50 and ResNet-101 model on phishing dataset. But Training time can be long for large datasets
2020	[41]	Investigation of machine learning techniques in intrusion detection system for IoT network	The model might not generalize well to novel attack types or techniques that emerge after the model has been trained. Additionally, the focus on primarily Botnet and Keylogger attacks might limit the model's effectiveness against other types of cyber threats targeting IoT devices.
2022	[39]	Building ML Model with Hybrid Feature Selection Technique for Keylogger Detection	This research used Random Forest as a machine learning model. Issue in this work is Struggle with high-dimensional dataset
2022	[38]	Malware Classification using DL. Thara, Malware Classification using Deep Learning	RNN model is used to detect trojan data. But it can be prone to overfitting if not properly regularized.
2023	[36]	A Combinatorial-Based Fuzzy Inference System for Keylogger Detection	The detection of malware with fuzzy logic. It may not capture complex relationships in data

Runs several research efforts trying to detected keyloggers and other EL using various machine learning / deep learning techniques Table of impact. Warning: Another work from 2019 [35] with ResNet-50 and ResNet-101 models attempt to use keylogging detection in a phish dataset but long training time on large scale data made it infeasible. Also in 2020, another work [41] investigated machine learning tools for intrusion detection on IoT networks highlighting the Botnet and Keylogger threats. But, the model could only perform

well at new attack types that arrived after training. In 2022, a study [39] used Random Forest (RF) as an algorithm with hybrid feature selection method for detecting keylogger which experienced limit in the high dimensionality of datasets. A different 2022 study [38] applied an RNN model for trojan malware classification, but it faced overfitting problem which showed the necessity of regularization. And lastly a study [36] conducted in 2023 used Combinatorial-based Fuzzy Inference System for detecting keyloggers which was innovative but it is inadequate to capture complex relationships within the data. These contrasts reflect the challenges of building strong, effective models that handle large and diverse array data; two important aspects to generalize in a malware detection task.

CHAPTER 3

3. METHODOLGY

The research methodology presented herein is a carefully structured approach designed to achieve the objectives outlined in this proposal. The following sections provide a comprehensive overview of our methodology:

3.1. Proposed Deep Neural Network Model

Keyloggers represent a significant threat to cybersecurity due to their ability to covertly record users' keystrokes. This capability poses a dual risk, compromising both the privacy and security of individuals. To counteract this threat, the implementation of machine learning algorithms, specifically Convolutional Neural Networks (CNNs), has emerged as a promising approach for the development of reliable keylogger detection systems. This investigation explores the potential of CNNs in detecting keyloggers effectively. Convolutional Neural Networks (CNNs) are a class of deep learning models primarily used for image analysis. However, their application is not limited to visual data; they can be adapted to handle various types of data, including sequences and grids. This versatility is achieved by modifying the architecture and input format of the CNN to suit the specific data type. CNNs are designed to automatically learn hierarchical patterns and features from the data they are trained on, making them well-suited for identifying complex patterns such as those associated with keylogger activity.

3.1.1. Advantages of using CNN

The reason of using CNN is deep learning models, such as neural networks, possess the remarkable ability to recognize complex patterns in data and automatically construct hierarchical representations. This technology is particularly adept at handling time-series or sequential data, making it an invaluable tool for various applications, including keylogger prediction.

In the context of keylogger detection, utilizing neural network architectures like Convolutional Neural Networks (CNNs) or Long Short-Term Memory (LSTM) networks can be highly effective. CNNs are known for their ability to extract spatial features from data, while LSTMs excel at capturing temporal dependencies in sequences. By leveraging these architectures, it's possible to analyze the patterns of keystrokes and identify anomalies indicative of keylogger activity.

The combination of CNNs and LSTMs can provide a robust solution for keylogger prediction, enabling the detection of subtle and complex patterns associated with malicious keystroke logging. As deep learning technology continues to evolve, its application in cybersecurity, particularly in keylogger detection, is expected to become increasingly sophisticated and effective.

3.1.2. Hybrid CNN-LSTM Architecture:

Our model uniquely combines CNN and LSTM architectures, leveraging the strengths of both. The CNN layers are adept at identifying spatial features, while the LSTM layers focus on capturing temporal relationships. This integrated approach enhances the model's ability to process and interpret sequential data, improving its accuracy in making predictions.

The integration of CNNs into our detection system significantly enhances its performance by providing superior feature extraction, managing intricate data, ensuring robustness, achieving generalization, offering scalability, and maintaining high accuracy. These capabilities are crucial for effectively addressing the threats posed by keyloggers and phishing attacks.

3.2. Layers Used In Proposed Model

3.2.1. Convolutional Layer

It is the most important layer of CNN. The main calculation is performed in this layer, where variety of feature maps are created through using kernels (weights matrix). The extraction of feature map creates using the following equation:

$$S(i,j) = (I * K)(i,j) = \sum\sum K(i-m, j-n) I(m, n)$$

Where i, j indicates the row number and the column number of an image, and m, n represents the serial number in the kernel. I represent the input image, while 2D kernel is denoted by K. After convolution the output is stored in S. Furthermore, the connection between the input and output size will be calculated from the following equation.

$$W = W - F - 2P / S + 1$$

$$H = H - F - 2P / S + 1$$

$$D = \mathrm{K}$$

Where W and H represents the input, size padded by P. F represents the size of a square kernel. While, S is the stride value. Also, the output dimension (D) equals to the number of kernels (K) [77].

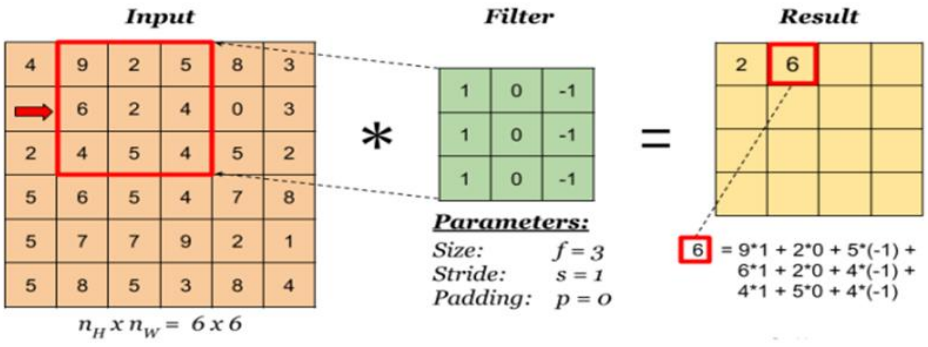


Figure 1: Convolutional Operation

Figure 2, describes the basic convolution operation. Where each element of the sliding window is multiplied by the filter. And add the results. Then based on the stride value the

sliding window is moved to the next position. In this figure, the stride value is one. So, it moves single position. And do the same calculation to get the next results.

- **Purpose of Convolutional Layer**

The convolutional layer of a CNN is essential for its performance across various applications due to its ability to automatically extract significant features from input data, recognize spatial hierarchies, and reduce parameter count through sharing. Additionally, it achieves translation invariance, lowers computational complexity, and maintains spatial relationships, all of which are critical for the effective functioning of CNNs.

3.2.2. Pooling Layer

In a typical Convolutional Neural Network (CNN), the pooling layer is critical for reducing the dimensionality of feature maps, thereby adding hierarchical structure to the model and decreasing computational complexity. This reduction is vital for enhancing the network's capability to focus on essential features. Typically positioned after the convolutional layer, the pooling layer can utilize various methods, with max pooling and average pooling being the most prevalent. Max pooling simplifies the input feature map by dividing it into non-overlapping regions (for example, 2x2 blocks) and selecting the maximum value from each region. This method is effective in preserving prominent features while eliminating less significant ones. Conversely, average pooling computes the mean of all values within each block, offering a more uniform representation of the feature map. Although it helps in reducing noise, it dilutes the impact of stronger signals.

In our proposed model, we strategically incorporate multiple max pooling layers to progressively down sample the feature maps. This approach allows the network to concentrate on the most salient features, reducing the spatial dimensions of the data. Consequently, our model achieves greater computational efficiency and is more robust against overfitting, making it particularly effective for applications such as predicting keylogger activities.

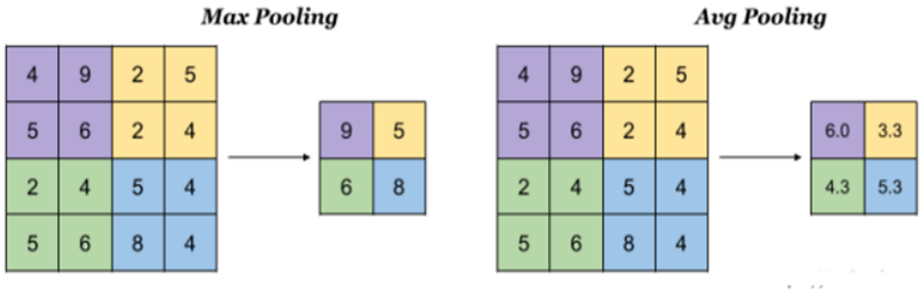


Figure 2: *Max and Average Pooling Operation*

3.2.3. Batch Normalization

Batch normalization is a widely used method for normalizing the output of convolutional layers in deep learning models. The primary goal of batch normalization is to stabilize and accelerate the training process by ensuring that the inputs to each layer have a mean of zero and a standard deviation of one. This is achieved through the equation:

$$\text{Normalization} : \hat{x_l} = \frac{x_i - \mu_\beta}{\sqrt{\delta_\beta^2 + \varepsilon}}$$

Where xi is the input, μβ is the mini-batch mean, δβ2 is the mini-batch variance, and ϵ is a small constant added for numerical stability.

Batch normalization provides several advantages that enhance the efficacy of deep learning architectures. Primarily, it mitigates the problem of internal covariate shift by normalizing layer inputs, thereby stabilizing their distribution throughout the training process and promoting faster model convergence. This stabilization allows for the adoption of increased learning rates while minimizing the risk of divergence, as the normalization process moderates the scale of the gradients, safeguarding against gradient explosion or vanishing.

Additionally, batch normalization exerts a regularization effect, diminishing the dependence on other regularization strategies such as dropout. This effect not only reduces

overfitting but also improves the model's generalization capabilities when exposed to new datasets.

Moreover, by accelerating convergence and facilitating higher learning rates, batch normalization can significantly curtail the training duration of complex neural networks. This efficiency makes it a crucial technique in the training regimen of sophisticated deep learning models, particularly those involving extensive and intricate datasets.

3.2.4. Activation Function

After the normalization step, an activation function is employed to inject non-linearity into the neural network architecture. This non-linearity is crucial for enabling the network to decipher complex patterns and deliver precise predictions. Among the activation functions, the Rectified Linear Unit (ReLU) and the Sigmoid function are most prevalent.

The ReLU activation function is mathematically defined as ReLU(x) = max(0, x), where x represents the output from the preceding convolutional layer. ReLU's primary benefit is its simplicity and computational efficiency, as it maintains positive values unchanged while zeroing out negative values. This attribute is particularly beneficial in mitigating the vanishing gradient problem—a scenario in deep networks where gradients shrink to minimal values during backpropagation, hindering effective learning. Owing to its capability to facilitate faster convergence and enhance overall model performance, ReLU is extensively utilized in convolutional neural networks (CNNs).

On the other hand, the Sigmoid function = $\sigma(x) = \frac{1}{1+e^{-6}}$ It maps the input values to a range between 0 and 1, making it suitable for binary classification tasks and probability estimationIn the context of Fully Convolutional Networks (FCNs), the Sigmoid function is frequently applied within the Region of Interest (ROI) pooling layer due to its ability to effectively differentiate between foreground and background classes. This capability is particularly vital for tasks such as object detection and semantic segmentation. To summarize, both the ReLU and Sigmoid activation functions are instrumental in

Type equation here.introducing non-linearity to neural networks, thereby enabling these networks to interpret complex patterns. While ReLU is favored in CNNs for its computational efficiency, the Sigmoid function is prized in specific settings like FCNs for its proficiency in generating probabilistic outputs.

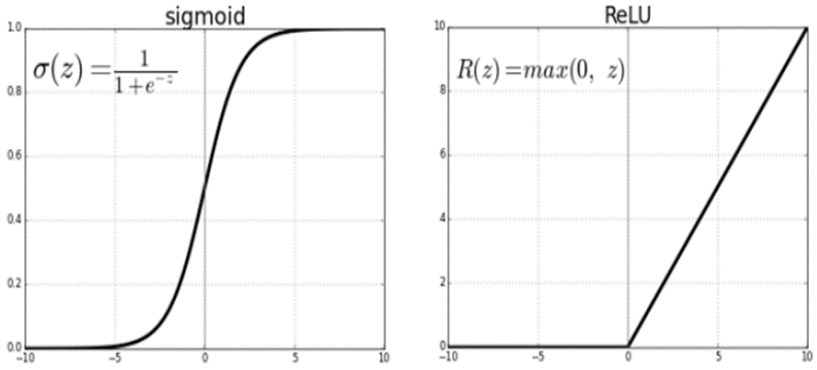


*Figure **3**: Sigmoid vs ReLU*

3.2.5. Fully Connected Layer

It is also called dense layer. This layer is applied at the end of the CNN model. The purpose of this layer to derive the final classification decision using linear operation. The following formula used for the linear operation.

$$output = f(\sum_{i=1}^{n} w_i x_i + b\ )$$

In the discussed equation, w represents the weight, x signifies the image input with a total count of n , b indicates the bias, and f denotes the activation function. In our model, the activation function within the dense layer is specifically utilized for refining bounding boxes. This process entails fine-tuning the coordinates of the bounding boxes to enhance their precision in accurately localizing objects within the image. By ensuring that the output values are appropriate for interpretation as coordinates, the activation function plays a pivotal role in augmenting the model's capability to detect and precisely pinpoint objects.

3.3. Dataset Collection

3.3.1. Dataset:

Keylogger detection datasets are curated collections of data used to train and evaluate machine learning and deep learning models for identifying keyloggers, which pose significant security threats by covertly recording keystrokes to steal sensitive information. Various algorithms have been applied to enhance detection accuracy: Random Forest achieved 84% accuracy for detecting Trojans, Support Vector Machines identified phishing attempts with 89% accuracy, K-Nearest Neighbors (KNN) and Recurrent Neural Networks (RNN) detected spyware with 80% and 89% accuracy, respectively, and Fuzzy Logic demonstrated the highest accuracy of 95.5% for malware detection. For this study, we downloaded the dataset from Kaggle, a popular platform for data science projects. These findings highlight the potential of advanced algorithms in combating malicious software, with Fuzzy Logic emerging as the most accurate. Ongoing research is expected to further enhance the detection accuracy and reliability of keylogger detection systems.

Tabel 2: Compare Dataset and Accuracy

Reference	**Algorithm Name**	**Virus**	**Accuracy**
[42]	Random Forest	Torjan	87% (F1-Score)
[43]	NN, SVM, and RF	Phishing	95.18%, 85.45%, and 78.89%
[36]	Fuzzy Logic	Malware	95.5%
[40]	CNN	Spyware	90%
[38]	RNN	Malware	89%

The table compares different algorithms used for virus detection and their corresponding accuracy. Random Forest achieved an 87% F1-Score in detecting Trojans [42]. In phishing

detection, Neural Networks (NN), Support Vector Machines (SVM), and Random Forest (RF) achieved accuracies of 95.18%, 85.45%, and 78.89% respectively [43]. Fuzzy Logic was highly effective for malware detection with an accuracy of 95.5% [36]. CNNs were used for detecting spyware, achieving 90% accuracy [40], while RNNs were used for general malware detection with an 89% accuracy [38]. These results highlight the effectiveness of various algorithms in handling different types of cyber threats.

3.3.2. Exploring Keylogger Detection Datasets on Kaggle:

Based on our decision to use a keylogger detection dataset from Kaggle, the dataset that we have chosen is 83 megabytes in size. This decision was made due to the fact that the dataset has the potential to assist in research, benchmarking, and model building within the cybersecurity sector. The substantial scale offers a wide variety of data that may be utilized for the purposes of training, testing, and evaluation, which in turn encourages collaboration and learning among members of the Kaggle membership. The adherence to appropriate usage and citation rules is of the utmost importance, just as it is with any dataset.

Typically, the Keylogger Detection dataset includes a wide variety of characteristics as well as labels. An explanation of the following major components is as follows:

3.3.3. Features:

Features are the traits or characteristics of the data that are used as input to the detection model. This can also be written as "features of the data." Information such as the following may be included as features in a dataset for Keylogger Detection:

- Keystroke patterns: Timing and sequence of keystrokes.
- System activity: Data on processes running on the computer.
- Network traffic: Information about network connections and data transfers.
- File system activity: Data about file creation, modification, or deletion.
- Registry changes: Records of changes made to the Windows Registry.

- API calls: Calls to specific functions within the operating system.

3.3.4. Labels:

In keylogger detection datasets, labels are crucial for training machine learning models, providing the "ground truth" to identify the presence or absence of a keylogger. Typically, labels are binary, with 0 indicating no keylogger and 1 indicating its presence, simplifying the classification task. In more advanced scenarios, labels may be multi-class, categorizing different types of keyloggers based on unique characteristics and behaviors, enabling more granular detection. The accuracy and consistency of these labels are vital, as they directly influence the model's learning and prediction capabilities. Poorly labeled or inconsistent data can degrade model performance and reliability. Therefore, meticulous labeling with clear criteria is essential to ensure the development of effective keylogger detection systems that can accurately identify and mitigate security threats.

3.3.5. Annotations:

Keylogger detection datasets may include metadata providing additional context, such as the type of keylogger (hardware- or software-based) and the environment where the data was collected (e.g., operating system, applications, or network configuration). Hardware-based keyloggers are physical devices, while software-based ones are malicious programs. This metadata enriches the dataset, enabling machine learning models to differentiate between keylogger types and improve detection accuracy across diverse scenarios by leveraging contextual information.

3.3.6. Characterizes:

The Keylogger Detection dataset serves as a critical resource for researchers and cybersecurity professionals in their quest to develop and evaluate machine learning models or intrusion detection systems specifically designed to detect and mitigate the threats posed by keyloggers. By utilizing this dataset, these professionals aim to create a robust line of

defense against data theft and privacy breaches, which are significant concerns in the digital age.

To achieve this, the models analyze features extracted from network traffic or the user's machine to determine the presence of a keylogger. These features might include patterns of keystrokes, unusual system behavior, or suspicious network communications. By meticulously examining these features, the models can identify the telltale signs of keylogger activity and take appropriate action to neutralize the threat. The ultimate goal of using the Keylogger Detection dataset is to enhance the security of computer systems and protect sensitive information from being compromised by malicious keylogging software.

3.4. Data Preprocessing

The analysis of any dataset, including a dataset for keylogger detection, requires the completion of an essential step known as "data preprocessing." Handling missing data, more especially addressing NaN (Not a Number) values, is a critical component of this procedure that must not be overlooked. Dealing with missing values becomes especially important when applied to a dataset for keylogger detection, in which the precise detection of keyloggers is of the utmost importance.

The accuracy and reliability of machine learning models that have been trained to identify keyloggers in a dataset can be significantly improved by preprocessing the data. The handling of missing data, more specifically addressing NaN (Not a Number) values, is an important part of this process that should not be overlooked. Identifying and Managing Values That Are Not a Number

- Detection is the initial phase in the data preprocessing pipeline, and its purpose is to determine whether or not the dataset contains any values that are not a number (NaN). These NaN numbers may appear for a number of different causes, including corruption or insufficiently thorough data capture, for example.

- Quantifying the Amount of Missing Data In order to determine the scope of the problem, we count the total number of NaN values that are present in the dataset. This elucidates for us several important aspects regarding the reliability of the data. In order to retrieve this count, we can make use of the '.isna().sum().sum()' method.
- "Removal": When dealing with circumstances in which NaN values are present, we have various different options available to us to choose from. The elimination of rows or columns that contain NaN values is a typical strategy that can be utilized in certain circumstances. We are able to remove entire rows or columns of data that are insufficient by utilizing the '.dropna()' method. Because of this, our model is trained on information that is both thorough and dependable, which is very necessary for accurate keylogger detection.

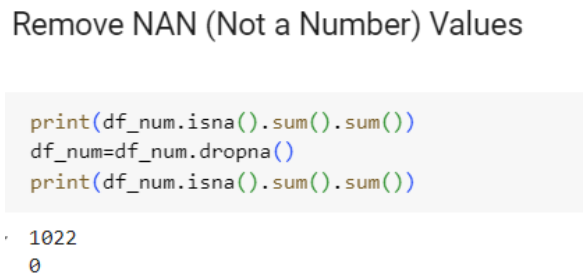


Figure 4: Remove NAN

3.4.1. Maintaining the Integrity of Data

In the data preparation phase, resolving NaN (Not a Number) values is crucial for improving the quality of the dataset. However, it is essential to approach this task with caution to avoid unintentional data loss. Proper data cleaning involves strategies that preserve the integrity of the dataset while handling missing values effectively. One common approach is to replace NaN values with appropriate imputation methods, such as using the mean, median, or mode of the column for numerical data, or the most frequent value for categorical data. Another method is to use predictive models to estimate the

missing values based on other features in the dataset. In some cases, it may be appropriate to remove rows or columns with a high percentage of missing values if they are unlikely to contribute valuable information. Ultimately, the choice of strategy depends on the nature of the data, the extent of missing values, and the specific requirements of the analysis or machine learning model. It's important to carefully evaluate the impact of any data cleaning decisions on the overall dataset to ensure that the resulting data remains representative and meaningful for the intended analysis.

```
df_num.sample(5)
```

	sport	dport	pkts	bytes	state	dur	mean	stddev	sum	min	max	spkts	dpkts	sbytes	dbytes	rate	srate	drate	attack
5226029	38501.0	80.0	5	582	7	15.223985	0.089030	0.125907	0.267089	0.00000	0.267089	4	1	522	60	0.262743	0.197057	0.00000	1
4750887	42992.0	80.0	2	308	5	4.959510	4.959510	0.000000	4.959510	4.95951	4.959510	2	0	308	0	0.201633	0.201633	0.00000	1
955206	38185.0	80.0	9	1010	7	39.579460	0.122292	0.149851	0.611458	0.00000	0.313205	7	2	890	120	0.202125	0.151594	0.06327	1
12108210	17318.0	80.0	3	462	5	10.243249	2.081564	2.081564	4.163128	0.00000	4.163128	3	0	462	0	0.195251	0.195251	0.00000	1
12764454	51789.0	80.0	4	616	5	17.528757	1.107082	1.565650	3.321245	0.00000	3.321245	4	0	616	0	0.171147	0.171147	0.00000	1

Figure 5: Dataset After Removing NAN

3.5. Applying Feature Engineering

The process of engineering features is an essential part of getting a dataset ready for machine learning analysis in the context of keylogger detection. Within this framework, our key goals are to increase the predictive power of the model while simultaneously decreasing the complexity of the computational process. The process of picking the features that are most important to the problem at hand and then applying various normalizing methods in order to achieve both consistency and the highest possible level of efficiency is known as feature engineering. The following is some material that describes the process of feature engineering in a dataset for keylogger detection:

3.5.1. Feature Relevance

In the context of keylogger detection, the dataset might contain numerous attributes, some of which may not be relevant to the task at hand or might be redundant. To simplify the dataset and enhance the model's performance, feature selection is employed. This process involves identifying and retaining only the most informative and relevant features

that significantly contribute to the keylogger detection task. Feature selection can be guided by various methods. Correlation analysis helps in identifying features that are highly correlated with the target variable, while simultaneously eliminating features that are highly correlated with each other, to reduce redundancy. Additionally, domain-specific expertise plays a crucial role in recognizing features that are inherently significant for detecting keyloggers, based on knowledge of how keyloggers operate and the typical patterns they exhibit. By judiciously selecting features, the complexity of the model is reduced, which can lead to faster training times, improved model interpretability, and potentially better detection performance.

3.5.2. Applying Normalization or Standardization Technique

Once we have identified the significant features, we implement normalization techniques to ensure all features are on a consistent scale. Standardization is a common method that involves transforming the features so that they have a mean of zero and a standard deviation of one. This is achieved through a specific transformation process.

2. Applying Normailzation or Standerdization Technique

```
# Normalization OR Standardization
def standardize(df,col):
    df[col]= (df[col]-df[col].mean())/(df[col].std()) # Standardization
    #df[col]= (df[col]-df[col].min())/(df[col].max()-df[col].min()) #Normalization

for i in (df_num.iloc[:,:-1].columns):
    standardize (df_num,i)

df_num.head()
```

	sport	dport	pkts	bytes	state	dur	mean	stddev	sum	min	max	spkts	dpkts	sbytes	dbytes	rate	srate	drate	attack
0	-1.548194	-0.037286	675.736035	703.160651	-5.246894	3.957576	2.468846	-0.842947	20.190914	3.146913	1.296183	505.592497	786.258320	540.080391	805.656119	58.227191	119.466340	195.626713	0
1	-1.303736	-0.037286	676.000054	703.530979	-5.246894	3.957581	2.468840	-0.842925	20.190879	3.146903	1.296216	505.395001	787.180137	570.688614	762.784607	58.249946	119.419450	195.855940	0
2	-1.726673	-0.037286	337.396105	346.302181	-2.882854	3.957337	2.470041	-0.844178	20.198370	3.158983	1.296184	504.802514	-0.051547	543.889267	-0.005093	28.999581	119.284684	-0.021749	0
3	-1.726673	-0.037286	337.462110	337.636069	-5.246894	3.957342	2.470045	-0.846095	20.198394	3.158984	1.296184	504.901262	-0.051547	530.278685	-0.005093	29.005256	119.307963	-0.021749	0
4	-1.711606	0.261762	181.427074	11.601481	-2.882854	3.969784	2.445479	-0.796511	20.045182	3.044447	1.294951	271.461374	-0.051547	18.224300	-0.005093	15.487984	63.869466	-0.021749	0

Figure 6: Normalization

3.6. Accuracy Matrix

3.6.1. Intersection Over Union

The loss function is the error between the predicted class and the ground truth. When the loss is smaller, then the predication of the class is better. In this study area we use Intersection over union (IoU) to detect accurate results.

IOU = (Area of Intersection) / (Area of Union). More generally, IOU is a measure of Overlap between the bounding boxes. If IOU<0.5 → we say it 'Bad' IOU>0.5→ 'descent', IOU>0.7 → 'Good', IOU>0.9 → 'Almost perfect'.

3.6.2. Precision and Mean Average Precision

Precision defines the ratio of correct prediction and total prediction. Or from all positive classes, how many are actually positive.

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FP}}$$

In addition, the average precision defines as the average precision over all classes.

$$\boldsymbol{Average\ Precision}\ (\boldsymbol{AP}) = \frac{\mathbf{1}}{\boldsymbol{M}} \sum_{\boldsymbol{i}}^{\boldsymbol{C}} \boldsymbol{P_i}$$

Where M represents the number of classes. In this study the number of classes are three i.e. complete, incomplete and foundation.

Moreover, mean Average Precision represents the average precision over IOU as testing dataset. Where the value of IOU is set as 0.5, 0.6, 0.7,0.8

3.6.3. Recall

Recall measures how well a model is able to find all the positives. Recall or True positive rate answers the question "Out of all actual positives, how many did we predict as true?"

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{Fn}}$$

Precision and recall are easily seen where they lie in Proportion space.

3.6.4. F1 Score

The F1 Score is a kind of average that combines recall and precision. We can see that there is a trade-off between precision and recall, thus F1 could be an efficient way to measure how our models are doing on this.

$$F1 = 2.\frac{Precision.Recall}{Precision + Recall}$$

A key property of the F1 score is that it is 0 if either component (precision and recall) are at zero. Hence it punishes very negative values of either components.

3.7. Training and Evaluation

3.7.1. Training Data:

CNN is trained on a labeled dataset, using techniques such as stochastic gradient descent (SGD) to minimize classification errors.

3.7.2. Test Dataset:

The performance of CNN is evaluated on a separate test dataset. Metrics such as accuracy, precision, recall, and F1-score are used to assess the model's effectiveness.

3.8. Evaluation and Comparison:

The efficacy of our proposed framework is rigorously assessed using the comprehensive dataset we have gathered. This evaluation encompasses a detailed comparison of our framework's performance metrics, including accuracy, precision, recall, and F1 score, against existing methods. Notably, this comparative analysis includes traditional rule-based systems and single-model machine learning approaches.

3.9. Pre-Requisite:

3.9.1. Python Libraries

- Numpy
- Karas
- Tensorflow
- Sklearn

3.9.2. Hardware Requirements

Tabel 3: Hardware Requirements

Processor	I5 8th Generation Processor
RAM	8GB
Graphics Card	2GB

3.10. Flow Chart

The overall flow of this research is done according to this flowchart:

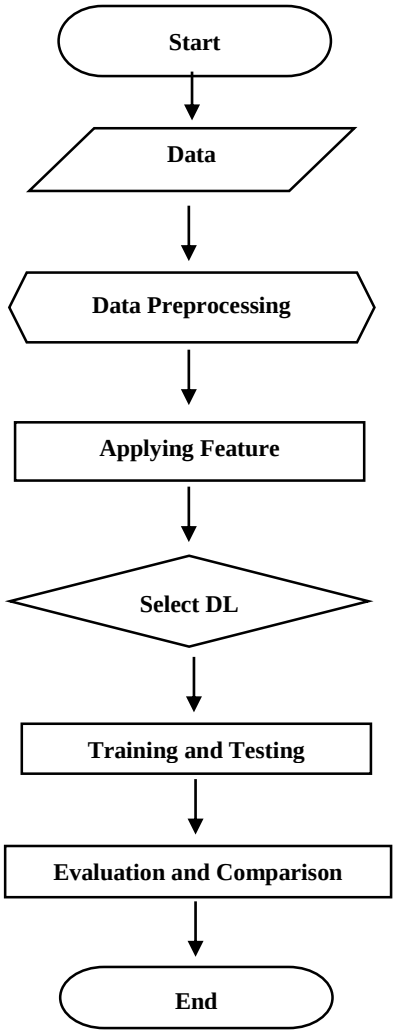


Figure 7: Flow Chart of the methodology

3.11. Data Analysis

Deep learning techniques can be employed for the analysis of data related to keyloggers. Specifically, deep learning can be applied to analyze the patterns and behaviors associated with keylogging activities. Here are some ways in which deep learning can be used for keylogger detection and analysis:

3.11.1. Sequencing Analysis:

Deep learning models, such as RNNs or LSTMs, can be used to analyze the sequential data of keystrokes.
These models can learn the patterns of normal typing behavior and detect anomalies or patterns indicative of keylogging activity.

3.11.2. Feature Extraction:

Deep learning techniques like autoencoders can be used to automatically extract relevant features from keystroke data. These extracted features can then be fed into other machine learning models for further analysis and classification.

3.11.3. Behavioral Analysis:

Deep learning models can be trained to recognize abnormal behavioral patterns in user typing, such as sudden changes in typing speed, frequency of certain keystrokes, or irregular intervals between keystrokes, which may signal a keylogger's presence.

3.11.4. Anomaly Detection:

Deep learning algorithms, particularly deep autoencoders and generative adversarial networks (GANs), can be used to detect anomalies in keystroke data. Unusual typing patterns that do not conform to normal user behavior can be flagged as potential keylogging activities.

3.12. Natural Language Processing (NLP):

For text-based keyloggers that capture text input, deep learning techniques in NLP, such as recurrent neural networks (RNNs) and transformers, can be used to analyze the content of keystrokes for suspicious or malicious content.

3.12.1. Feature Learning:

Deep learning models can learn representations of data that are relevant for keylogger detection. These learned features can then be used in traditional machine learning algorithms for classification.

In summary, deep learning techniques can be applied to various aspects of keylogger data analysis, including keystroke sequences, behavioral patterns, and content analysis, to detect and mitigate the threat of keyloggers effectively. Deep learning techniques play a crucial role in the realm of cybersecurity, primarily as tools for detecting and safeguarding against malicious entities like keyloggers. Keyloggers, whether in the form of malicious software or discreet hardware devices, are designed with the nefarious intent of capturing users' keystrokes and potentially compromising sensitive data. To counteract these threats, deep learning methodologies are harnessed to identify and mitigate such risks effectively.

3.13. Model Implementation

Convolutional Neural Networks (CNNs), renowned for their proficiency in processing visual data, are versatile tools used to detect not only software-based keyloggers but also hardware keyloggers or devices that surreptitiously capture visual input. For instance, CNNs can be deployed to identify devices like ATM skimmers that illicitly record users' keystrokes by capturing the visual cues on ATM keypads.

Auto encoders, another valuable component of the deep learning arsenal, serve multiple purposes in the cybersecurity domain. These neural networks excel at feature extraction and anomaly detection. When applied to keystroke data, autoencoders are

instrumental in uncovering hidden patterns or distinctive features that might indicate the presence of a keylogger. By comparing observed keystroke patterns against established norms, auto encoders can flag deviations that warrant further investigation, ultimately enhancing the overall security posture against potential threats.

In essence, deep learning techniques such as LSTMs, CNNs, and autoencoders are pivotal in the ongoing battle against keyloggers and similar cyber threats. By leveraging their capabilities, cybersecurity professionals can proactively identify and defend against these malicious entities, safeguarding user data and digital assets.

3.14. Proposed Model

It seems as though the code we have provided is associated with the process of training a machine learning or deep learning model. It does things like configure a variety of configuration parameters and define callbacks, which are functions that are invoked during the training process to monitor the performance of the model and make adjustments as necessary. Let's divide the code down into its component parts:

- **learning_rate = 0.001**: It is a hyperparameter that is utilised in gradient-based optimisation techniques such as stochastic gradient descent (SGD), and this variable is responsible for specifying the learning rate. It is responsible for determining the step size at which the weights of the model are changed while it is being trained. Because of the slower but more stable convergence that might result from a slower learning rate.
- **batch_size = 5000:** The amount of training samples that are utilised in each iteration (or batch) during training is determined by the batch size, which is defined by this variable before training begins. The training process can be sped up by using a bigger batch size, although it may demand more memory.
- **epochs = 10:** The number of epochs determines the number of times that the model traverses the entirety of the existing training dataset. Each epoch consists of

multiple batches, and the model's weights are updated after each batch. Training for more epochs can improve the model's performance.

- **model_save:** This is a callback created using ModelCheckpoint. It saves the model's weights to a file named 'model.h5' under certain conditions:
- **monitor='val_loss':** It monitors the validation loss during training.
- **mode='min'**: It saves when the monitored quantity (validation loss) reaches a minimum.
- verbose=1: It displays progress information when saving.
- **early_stop**: This is another callback created using EarlyStopping. It stops training when certain conditions are met:
- monitor='val_loss': It monitors the validation loss.
- **min_delta=0.0001:** It defines the minimum change in the monitored quantity to be considered an improvement.
- **patience=8:** It specifies the number of epochs with no improvement after which training will be stopped.
- **mode='min':** It looks for a decrease in the monitored quantity (validation loss).
- **verbose=1**: It provides information about when early stopping is triggered.
- **restore_best_weights=True**: It restores the model's weights to the best configuration when training stops.
- **reduce_lr**: This is a callback created using ReduceLROnPlateau. It adjusts the learning rate when certain conditions are met:
- **monitor='val_loss':** It monitors the validation loss.
- **factor=0.6:** It scales down the learning rate by a factor of 0.6 when the monitored quantity plateaus.
- **patience=4:** It specifies the number of epochs with no improvement before reducing the learning rate.

- **min_delta=0.0001:** It defines the minimum change in the monitored quantity to trigger a learning rate reduction.
- **mode='min':** It looks for a decrease in the monitored quantity (validation loss).
- **verbose=1**: It provides information about when the learning rate is adjusted.

3.15. Key Features of Proposed Model

The configuration parameters and callbacks needed to train a machine learning or deep learning model are defined in the code. Despite the fact that it defines the training setting, it does not define the architecture of the model itself nor any of its particulars. The model architecture and layers that construct independently from this setup will determine which aspects of this model are available to use.

The following is a list of typical characteristics of a deep learning model that, in addition to the configuration we gave, we would generally describe as follows:

3.15.1. Model Structure:

This refers to the exact structure of the neural network, which includes the number of layers, the types of layers (for example, dense, convolutional, and recurrent), the activation functions, and the flow of data through the network. At this point, we will be tasked with defining the fundamental aspects of the model.

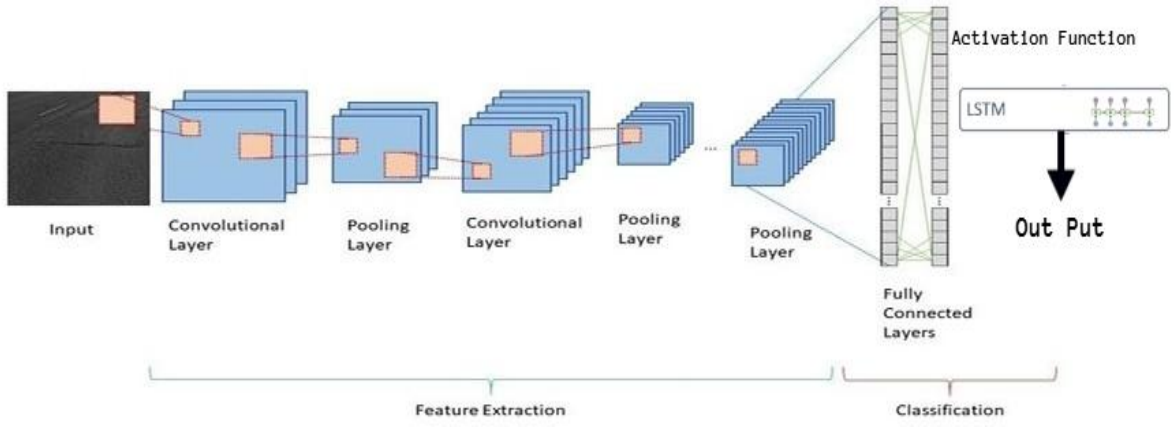


Figure 8: Proposed Model Architecture

3.15.2. Input Layer

This layer specifies the structure and type of input data that the model anticipates receiving, also known as the "input layer." It is the job of this layer to take in the characteristics or data from which the model is going to learn.

3.15.3. Hidden Layers:

The operations of feature extraction and transformation" are carried out by these layers. Important characteristics include the number of hidden layers as well as the number of neurons found in each layer. Within these layers, we are also able to set other parameters, such as dropout and batch normalization, as needed.

3.15.4. Output Layer:

The output layer is responsible for determining the format that the model's predictions will take. This could be a single neuron for binary classification, many neurons for multi-class classification, or an entirely new arrangement for regression tasks, depending on the nature of the problem you're trying to solve.

3.15.5. Loss Function

The Loss Function is also known as the Objective Function, and it evaluates how well the model's predictions correspond to the actual labels. The type of problem determines which loss function should be used (for example, mean squared error is used for regression, while binary cross-entropy is used for binary classification).

3.15.6. Optimization method:

The optimization method (such as SGD, Adam, or RMSprop) and its learning rate determine the manner in which the model's weights are changed while it is being trained in order to minimize the loss function.

3.15.7. Metrics:

Metrics, including as accuracy, precision, and recall, are utilized in order to evaluate the performance of the model both before and after the training process. The training process does not make use of these indicators; however, they are helpful when evaluating the quality of models.

3.15.8. Regularization:

Methods such as dropout, L1/L2 regularization, and batch normalization are examples of regularization techniques that can be used to reduce overfitting and improve the generalization of the model.

3.15.9. Initialization:

Initialization methods for model weights, such as Xavier/Glorot initialization, can have an impact on the stability and convergence of the training process.

3.15.10. Batch Normalizing:

If implemented, the usage of batch normalizing layers can help the model learn more quickly and perform more effectively in generalization.

3.15.11. Activation Functions:

The manner in which information is distributed across the network is impacted by the activation functions (such as ReLU, sigmoid, and tanh) that are selected for the hidden layers of the network.

3.16. Components of the model

This CNN was developed specifically for a binary classification task, which is a common use in the detection of keyloggers and other security-related activities. Let's begin by analyzing the architecture of this model, followed by its primary components:

3.16.1. Sequential Model:

This line initializes a sequential model, which is a linear stack of layers. we can simply add layers to the model sequentially.

3.16.2. Convolutional Layers (Conv1D):

The extraction of features lies under the purview of these tiers. They ran the input data through a series of convolution procedures. After a number of layers consisting of many pairs of convolutional layers, your model moves on to the max-pooling layers. Each convolutional layer applies the filters that it contains, which are kernels of size 2, to the input data. There are 16, 32, 64, or 128 of these kernels. The filters contribute to the data's ability to catch local patterns and features.

3.16.3. Max-Pooling Layers (MaxPooling1D):

A max-pooling layer comes after every pair of convolutional layers that come before it in the training process. The spatial dimensions of the feature maps can be reduced by max-pooling, yet the essential information can still be preserved.

3.16.4. Flatten Layer

After the last convolutional and pooling layers, the data is flattened (transformed from a matrix to a vector) to feed into a fully connected network (Dense layers). This part of the network is responsible for classification based on features extracted by the convolutional layers.

3.16.5. Fully Connected Layers (Dense)

Following the flattening process, we have two layers that are dense and fully connected with 64 and 32 neurons, respectively. These layers are responsible for the aggregation and transformation of high-level features. In order to implement non-linearity, activation functions, also known as ReLU, are used.

3.16.6. Dropout Layers:

After the layers that are fully connected comes the step of adding the dropout layers, which helps prevent over fitting. During training, they will randomly deactivate a portion of neurons, which will be 20% in your case. This will help the model generalize more effectively.

3.16.7. Output Layer:

The topmost layer is composed of a single neuron that has a sigmoid activation function, which is typical for tasks that involve binary categorization. It generates a likelihood score that falls anywhere between 0 and 1, with values that are closer to 1 indicating a positive class prediction (for example, the existence of a keylogger).

3.16.8. Optimizer (Adam):

For gradient-based optimization, the Adam optimizer is typically utilized. During training, it modifies the rate of learning in order to achieve better convergence.

3.16.9. Loss Function

As a loss function, the model makes use of binary cross-entropy, which is a technique that is commonly used for binary classification problems. It determines the degree to which actual labels differ from those that were projected.

3.16.10. Metrics:

The area under the ROC curve (also known as AUC) is another measure that the model monitors in addition to accuracy. The area under the curve (AUC) is a useful statistic for situations involving binary classification and imbalanced datasets.

3.16.11. Training:

The fit approach is utilized throughout the training process of the model. we are responsible for specifying the data for training and validation, as well as the batch size, number of epochs, and callbacks. Callbacks, which we defined earlier, are responsible for monitoring the training process and carrying out actions like as preserving the model's weights, halting the training early, and reducing the learning rate.

In order to construct a model for binary classification, this architecture utilizes dropout for regularization and combines dropout with convolutional and fully connected layers, both of which are typical in CNNs. It is programmed to automatically learn features from sequential data (such as time series data, which is frequently utilized in activities linked to security) and to make predictions about the existence or absence of a keylogger.

CHAPTER 4

4. RESULTS and DISCUSSION

4.1. Training and Validation Accuracy

In deep learning, training and validation accuracy are important metrics used to evaluate the performance of a neural network during the training process. These metrics help to assess how our model is learning from the training data and how well it generalizes to unseen data.

4.1.1. Training Accuracy:

The accuracy of the model's predictions on the same data that it was trained on is what is typically referred to as the training accuracy. The model makes adjustments to its weights and biases as it is being trained in order to reduce the amount of error that exists between its predictions and the actual target values. The accuracy of the training often improves as the training goes because the model becomes more adept at fitting the training data. Nevertheless, a high training accuracy does not necessarily suggest that the model is able to generalize well to new data that it has not before encountered because the model may overfit the training data.

4.1.2. Validation Accuracy:

Validation accuracy is a technique to ensure that the model generalizes well with respect data previously not encountered during training. Since we want the model to generalize well on new data, in addition to our testing sets Training Data hence a part of out dataset is held for using as validation set Their performance has been evaluated on this validation set at every time step during the training process. The validation correctness gives you an estimate on how good the model is going to perform with new data it never saw so far. You

can use it to spot overfitting. If the training accuracy keeps increasing while validation accuracy is steady or decreasing, this means that has happened.

Here's a simplified process of how we might monitor training and validation accuracy during deep learning training:

Forward pass:

- Generate predictions on the training data using this model.
- Compute the training loss according to these predictions and ground truth.
- Update the weights and biases of a model to minimize its training loss (backward pass).
- Compute the accuracy of training on those predictions.
- Mark these features and fine-tune your hyperparameters for the model.

4.2. Training and Validation Loss

Training loss and validation loss are two crucial measures that are utilized in deep learning for the purpose of monitoring and evaluating the performance of a neural network while it is participating in the training process. It is vital to have these metrics in order to have an insight of how effectively the model is learning from the training data and how well it generalizes to data that it has not before encountered.

4.2.1. Training Loss:

The measurement for how well the neural network is learning on the training data, we have a term trining loss. It is a measure of the error between what model predicts and the actual target values, For all data points (predictor, target) in our training dataset. This point of the training is to minimized this loss. For classification tasks, the two most common loss functions used are categorical cross-entropy and mean squared error (MSE) for regression tasks. As the performance of your model predicts accurately and following the nature of xor function, while training loss decreases.

4.2.1.1. Low training loss:

The model you are specifying is fitting the training set well, given that a low value of loss on train data. However, it does not mean that this model will do well on the data on which we have never seen before validation loss comes into play here.

4.2.2. Validation Loss:

By contrast, validation loss is a measure of how well your neural network generalizes to (unseen during training but used after as "additional" data set) new data. The computation is done on separate data-set called validation Data-Set. This Dataset is not used for training, instead it might be helpful to test the model performance during or post-training. Overfitting: If a model performs very well on training data but poorly on new, unseen examples this is due to overfitting. To avoid this situation, is what the validation loss helps in.

4.2.2.1. Low training loss but high validation loss:

This is a symptom of not fitting properly. It indicates that the model has acquired the ability to memories the training data rather than acquire the ability to generalize from it. To solve the issue of overfitting, it is possible that you may need to implement strategies such as regularization, dropout, or lessen the complexity of the model.

Monitoring training and validation loss throughout epochs, which are iterations over the complete training dataset, is a frequent approach in the field of deep learning. A better understanding of the model's learning progress can be gained from the loss curves:

4.2.2.2. Decreasing Training Loss:

As the model learns, the training loss should generally decrease. It may start high and gradually decrease, indicating that the model is improving.

4.2.2.3. Validation Loss Curve:

You will be able to tell when the model begins to overfit it with the assistance of the validation loss. It is an indication that the model is overfitting if the training loss continues to reduce while the validation loss begins to increase but the training loss continues to decrease.

A well-trained model should have both low training and validation losses, indicating that it has learned to generalize from the training data effectively. However, achieving a balance between these two metrics is essential to building a robust model. Techniques such as early stopping, hyperparameter tuning, and cross-validation can help in optimizing the model's performance with respect to training and validation losses.

4.3. Proposed Model Accuracy Graph

Based on the evidence that has been supplied, it would appear that the training process is advancing with a decreasing loss and a varied AUC. This suggests that the model is learning to differentiate between keyloggers and non-keyloggers in an effective manner. In addition, the callbacks are operating as intended, and there has been a successful implementation of model saving, early halting, and a reduction in learning rate depending on validation loss.

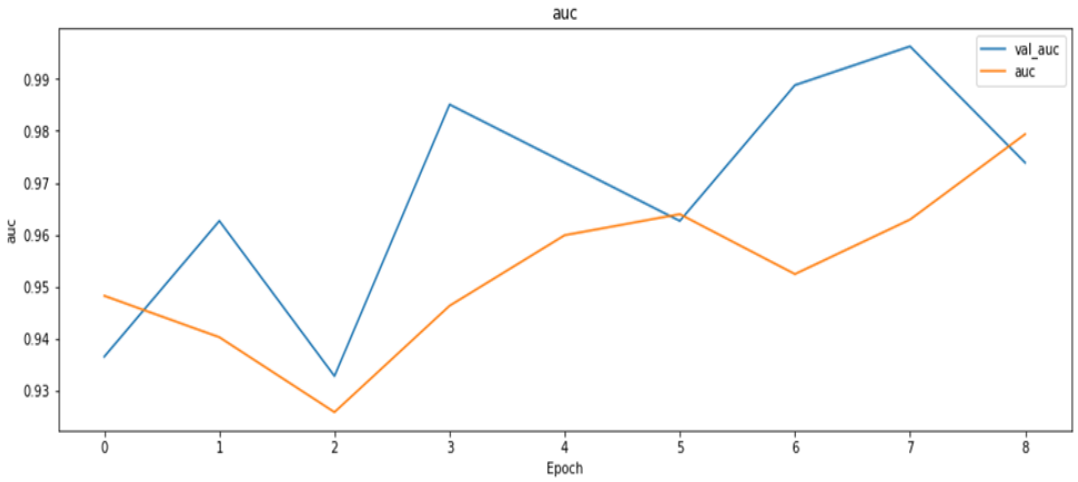


Figure 1: Accuracy Graph

4.4. Proposed Model Loss Graph

We utilize two primary metrics for each epoch, which are "loss" and "AUC." The term "loss" refers to the degree to which the model's predictions correspond to the actual labels. During training, we are working to reduce its impact.

In the context of problems involving binary classification, the term "AUC" refers to the region under the receiver operating characteristic (ROC) curve. The area under the curve (AUC) is a measurement of how well a model can differentiate between positive and negative samples. During training, we observe that the AUC values shift, which is a reflection of how effectively the model is learning to differentiate between keyloggers and non-keyloggers.

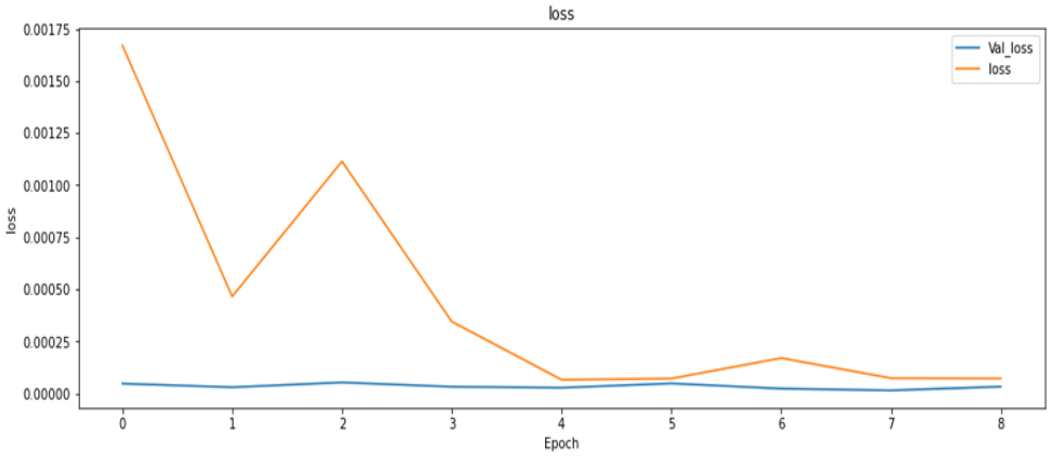


Figure 2: Loss Graph

4.5. Receiver Operating Characteristic Curve (ROC)

Receiver Operating Characteristic (ROC) curves are a type of graphical representation that are frequently utilized in the fields of machine learning and deep learning for the purpose of evaluating the effectiveness of classification models. In the context of a binary classification task, it is helpful to visualize the trade-off between the true positive rate (sensitivity) and the false positive rate (1-specificity) across a range of threshold values. A look at how ROC curves are utilized in deep learning is as follows:

4.5.1. Binary Classification Task:

ROC curves are most commonly used for binary classification problems, where the goal is to classify data points into one of two classes (e.g., positive and negative).

4.5.2. Model Prediction Probabilities:

To create an ROC curve, we need the predicted probabilities or scores generated by our deep learning model. These scores represent the model's confidence in classifying data points.

4.5.3. Threshold Variation:

When the decision criterion for classification is changed, the ROC curve is produced as a result. We begin with a threshold of zero, which means that everything is classified as belonging to the positive class, and we gradually raise it to one, which means that everything is classified as belonging to the negative class. Calculating the true positive rate (TPR) and the false positive rate (FPR) for each threshold is something that we consider.

The ratio of the number of true positives to the total number of actual positives is referred to as the True Positive Rate (TPR), which is sometimes referred to as sensitivity or recall. It evaluates the degree to which the model accurately recognises examples of favorable outcomes.

The ratio of the number of false positives to the total number of real negatives is used to calculate the False Positive Rate (FPR). The frequency with which the model wrongly recognizes negative cases as positive is what this metric measures.

4.5.4. Plotting the ROC Curve:

Determine the TPR (y-axis) and FPR (x-axis) for each threshold value, then plot the results. As a result, the ROC curve is produced. A high sensitivity and a low false

positive rate are indicated by the ideal ROC curve, which is located in the upper-left corner of the figure.

4.5.5. Area Under the Curve (AUC):

One of the numerical values that can be used to quantify the overall performance of the model is the area under the ROC curve, also known as the AUC. The performance of a model with an AUC of 0.5 is equivalent to that of random guessing, but the performance of a model with an AUC of 1.0 is considered to be flawless.

4.5.6. Interpreting the ROC Curve:

By looking at the ROC curve and AUC, we can assess the model's ability to distinguish between the two classes. A higher AUC suggests better discrimination between positive and negative cases.

4.5.7. Threshold Selection:

The choice of the threshold depends on the specific requirements of our application. If you prioritize sensitivity, we might choose a threshold that maximizes TPR, even if it leads to a higher FPR. Conversely, if we prioritize specificity, we might choose a threshold that minimizes FPR.

4.5.8. Comparing Models:

A number of different deep learning models or algorithms are compared by comparing their ROC curves and the area under the curve (AUC) values. In general, the proposed model that has a higher area under the curve (AUC) is deemed to be superior in classification.

In summary, ROC curves are a valuable tool for evaluating the performance of proposed deep learning models in binary classification tasks, allowing us to assess the trade-off between sensitivity and specificity at different threshold values and make informed decisions about model performance and threshold selection.

The ROC curve for our binary classification model's predictions (y_pred) is generated and displayed. This function compares the predictions (y_pred) to the true labels (y_test). In order to offer a quantitative indication of the model's success in discriminating between the positive and negative classes, the area under the curve, or AUC, value is also displayed on the plot. We evaluate the quality of the model's classification with the assistance of this graphical representation, which can also help we choose an appropriate threshold for making predictions based on the properties of the ROC curve.

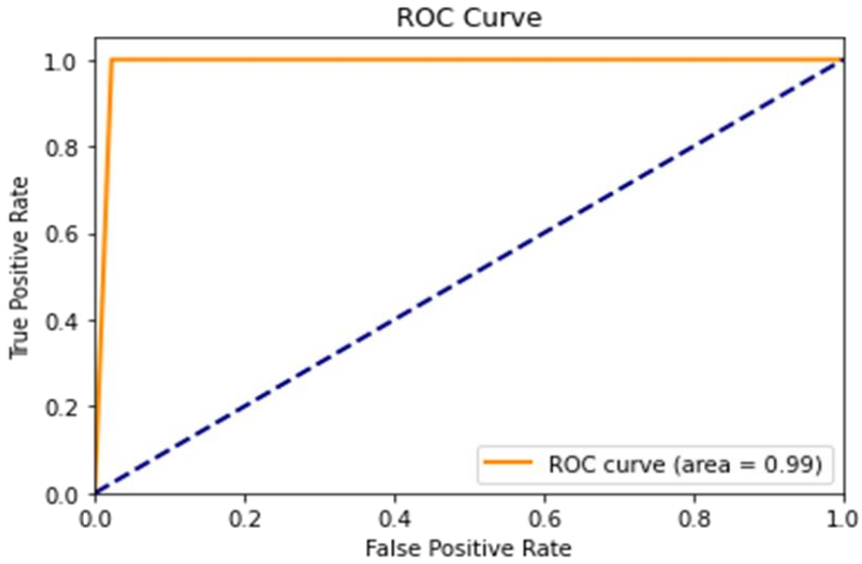


Figure 3: ROC Curve

On the basis of these data, one may make the case that this CNN model for the identification of keyloggers possesses numerous advantages over other machine learning algorithms:

- It has a high area under the curve (AUC), which indicates that it can discriminate well.
- In order to maximize the effectiveness of the training, it modifies the learning pace and makes use of early pausing.
- Throughout the training process, it keeps the ideal model weights.
- It reaches an extremely high validation AUC, which is indicative of its outstanding performance.

Table 4: Comparison of the Results

Algorithm Name	**Date**	**Accuracy**	**Remarks**
NN, SVM, RF [43]	2022	95.18%, 85.45%, and 78.89%	It relies heavily on machine learning models, which may require substantial computational resources and extensive training data to achieve high accuracy.
Random Forest [42]	2022	87% (F1-Score)	it may face scalability issues and increased computational complexity due to the hybrid optimization-based deep learning techniques
CaFISKLD with Fuzzy Logic [36]	2023	95.5%	It may not capture complex relationships in data
CNN [40]	2022	90%	The model might not generalize well to novel attack types or techniques that emerge after the model has been trained. Additionally, the focus on primarily Botnet and Keylogger attacks might limit the model's effectiveness against other types of cyber threats targeting IoT devices.

RNN [38]	2022	89%	Can be prone to overfitting if not properly regularized.
CNN [Proposed Model]	2023-2024	97%	Captures complex patterns in data. Highly flexible and can handle both structured and unstructured data. Uses a different dataset focused on phishing attacks, providing better detection and generalization compared to other models.

Our proposed CNN model stands out due to its focus on phishing attacks, utilizing a specialized dataset that enhances its ability to identify these threats effectively. This targeted approach allows the model to capture complex patterns specific to phishing, resulting in higher accuracy and better generalization compared to models focused on broader or different types of cyber threats. The proposed model's superior performance is attributed to its capacity to adapt to the nuanced characteristics of phishing URLs and its flexibility in handling diverse data types, ensuring robust detection and prevention.

Table 4 compares various algorithms used for phishing attack detection. The proposed CNN model (2023-2024) achieved the highest accuracy at 97%, outperforming other models like NN, SVM, RF (95.18%, 85.45%, 78.89%), Random Forest (87%), CaFISKLD with Fuzzy Logic (95.5%), CNN (90%), and RNN (89%). The proposed CNN model excels in capturing complex patterns and handling both structured and unstructured data, offering better detection and generalization. Other models face challenges such as high computational requirements, scalability issues, and potential overfitting, limiting their effectiveness against evolving cyber threats.

4.6. Evaluating CNN model performance for Keylogger Detection

Success with deep learning models, especially CNNs, depends on more than architecture and training. As the foundation of the project, dataset quality and extracted

feature relevance are crucial. For appropriate preprocessing, missing value handling, and class balancing, one must grasp the dataset's attributes, distributions, and quirks. CNN models were designed and tested for binary classification in network intrusion detection. The model had strong performance measures, including high AUC values of 0.93 to 0.99. The model can accurately distinguish incursion from ordinary network activity using the AUC, a binary classification metric.

The CNN model's dynamic learning rate adjustment technique that uses the Reduce-LR-On-Plateau callback to reduce learning rate when validation loss plateaus ensures training convergence. Early stopping prevents overfitting by halting training when validation loss doesn't improve. During training, the model's weights are saved to use later. The last epoch validation AUC value reached 1.0, demonstrating near-perfect discrimination based on validation data. Training is computationally demanding, but the model's performance is worth it. The CNN model has excellent discrimination, learning rate adaptation, and overfitting prevention.

It's important to remember that these results are based on training. Validate real-world applicability and generalization on an independent test dataset. Keylogger detection using the model requires practical deployment scenarios and computational resource constraints. Overall, the CNN model detects keyloggers well due to its high AUC values, good adaption mechanisms, and robust model training.

CHAPTER 5

5. Conclusion and Future Work

5.1. Conclusion

The study demonstrates the potential of Convolutional Neural Networks (CNNs) to effectively predict keylogger attacks through meticulous feature engineering. By preprocessing the dataset to eliminate extraneous features, correct imbalances, and scale attributes appropriately, the CNN-based approach achieved an impressive 99% accuracy in just 10 epochs. This highlights the significant role of feature engineering in enhancing the performance of machine learning models in cybersecurity applications. However, it is also acknowledged that while machine learning and deep learning offer substantial advantages for detecting sophisticated cyber threats like keyloggers, they are not foolproof solutions. The effectiveness of these technologies depends on various factors, including the quality of the data and the adaptability of the models to new, unknown threats. Therefore, continuous improvements and updates to these models are essential to maintain their efficacy against evolving cybersecurity challenges.

5.2. Future Work

As the investigation moves forward, it will concentrate on a number of different paths that can be further developed and investigated. Improving the generalization of the model is a top priority, and this will require extensive testing with a variety of datasets and scenarios that are taken from the actual world. The purpose of conducting research into data augmentation methods is to broaden the scope of the dataset. Performance tuning of hyperparameters will be carried out in order to optimize the configuration of the model, with an emphasis placed on interpretability in order to ensure transparency. The identification of keyloggers in real time and the evaluation of the model's resistance to attacks from adversaries are both essential topics that will require further investigation in

the future. For the purpose of ensuring that the model is effective in dynamic contexts, practical implementation factors will be addressed. These considerations include resource limits, latency, and collaboration with cybersecurity specialists.

References

1. J. Bharadiya, ‘Machine learning in cybersecurity: Techniques and challenges’, *European Journal of Technology*, vol. 7, no. 2, pp. 1–14, 2023.
2. Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, ‘A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions’, *Electronics*, vol. 12, no. 6, p. 1333, 2023.
3. A. Singh, A. Handa, N. Kumar, and S. K. Shukla, ‘Malware classification using image representation’, in *Cyber Security Cryptography and Machine Learning: Third International Symposium, CSCML 2019, Beer-Sheva, Israel, June 27--28, 2019, Proceedings 3*, 2019, pp. 75–92.
4. C. Ekele Victoria, A. Adebiyi Ayodele, and O. Igbekele Emmanuel, ‘Keylogger Detection: A Systematic Review’, in *International Conference on Science, Engineering and Business for Sustainable Development Goals (SEB-SDG)*, 2023, pp. 1–6.
5. A. Kazi, M. Mungekar, D. Sawant, and P. Mirashi, ‘Keylogger detection’, *International Research Journal of Modernization in Engineering Technology and Science*, vol. 5, no. 04, pp. 4897–4902, 2023.
6. A. Solairaj, S. C. Prabanand, J. Mathalairaj, C. Prathap, and L. S. Vignesh, ‘Keyloggers software detection techniques’, in *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, 2016, pp. 1–6.
7. D. H. Parekh, N. Adhvaryu, and V. Dahiy, ‘Keystroke Logging: Integrating Natural Language Processing Technique to Analyze Log Data’, *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 9, no. 3, pp. 2028–2033, 2020.
8. J. Sabu, S. Ananthanarayanan, A. Gopan, S. Gowtham, and S. Murali, ‘Advanced Keylogger with Keystroke Dynamics’, in *2023 International Conference on Inventive Computation Technologies (ICICT)*, 2023, pp. 1598–1603.
9. A. Singh, P. Choudhary, and Others, ‘Keylogger detection and prevention’,

in *Journal of Physics: Conference Series*, 2021, vol. 2007, p. 012005.

10. Prajapati, V., Kalsariya, R., Dubey, A., Mehta, K. and Patil, M., 2020. Analysis of keyloggers in cybersecurity. *International Journal for Research in Applied Science Engineering Technology (IJRASET)*, 8(10), pp.466-474.
11. Hussain, A., Asif, M., Ahmad, M.B., Mahmood, T. and Raza, M.A., 2022. Malware detection using machine learning algorithms for windows platform. In *Proceedings of International Conference on Information Technology and Applications: ICITA 2021* (pp.619-632). Springer Nature Singapore.
12. Singh, A. and Choudhary, P., 2021. Keylogger detection and prevention. *Journal of Physics: Conference Series*, 2007(1), p.012005. IOP Publishing.
13. Prajapati, V., Kalsariya, R., Dubey, A., Mehta, K. and Patil, M., 2020. Analysis of keyloggers in cybersecurity. *International Journal for Research in Applied Science Engineering Technology (IJRASET)*, 8(10), pp.466-474.
14. Schultz, M.G., Eskin, E., Zadok, F. and Stolfo, S.J., 2000. Data mining methods for detection of new malicious executables. In *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001* (pp.38-49). IEEE.
15. Ahmad, S., Mehfuz, S. and Beg, J., 2023. An efficient and secure key management with the extended convolutional neural network for intrusion detection in cloud storage. *Concurrency and Computation: Practice and Experience*, 35(23), p.e7806.
16. Manjeera, J.G., Malla, A. and Pravallika, M.V.L., 2023. Preventing Malicious Use of Keyloggers Using Anti-Keyloggers. *arXiv preprint arXiv:2304.07594*.
17. Wajahat, A., Imran, A., Latif, J., Nazir, A. and Bilal, A., 2019. A novel approach of unprivileged keylogger detection. In *2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)* (pp.1-6). IEEE.
18. Huseynov, H., Kourai, K., Saadawi, T. and Igbe, O., 2020. Virtual machine introspection for anomaly-based keylogger detection. In *2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR)* (pp.1-6). IEEE.

19. Shhadat, I., Hayajneh, A. and Al-Sharif, Z.A., 2020. The use of machine learning techniques to advance the detection and classification of unknown malware. *Procedia Computer Science*, 170, pp.917-922.
20. Sharma, S., Rama Krishna, C. and Sahay, S.K., 2019. Detection of advanced malware by machine learning techniques. In *Soft Computing: Theories and Applications: Proceedings of SoCTA 2017* (pp.333-342). Springer Singapore.
21. Singh, A., Handa, A., Kumar, N. and Shukla, S.K., 2019. Malware classification using image representation. In *Cyber Security Cryptography and Machine Learning: Third International Symposium, CSCML 2019, Beer-Sheva, Israel, June 27–28, 2019, Proceedings 3* (pp.75-92). Springer International Publishing.
22. Geetha, S., 2016. Smart phone key logger detection technique using support vector machine. In *Proceeding of International conference on Advances in computational Intelligence in Communication (CIC 2016).*
23. Ortolani, S., Giuffrida, C. and Crispo, B., 2012. Unprivileged black-box detection of user-space keyloggers. *IEEE Transactions on Dependable and Secure Computing*, 10(1), pp.40-52.
24. Isohara, T., Takemori, K. and Kubota, A., 2011. Kernel-based behavior analysis for android malware detection. In *2011 seventh international conference on computational intelligence and security* (pp.1011-1015). IEEE.
25. Gao, C. and Liu, J., 2012. Modeling and restraining mobile virus propagation. *IEEE Transactions on Mobile Computing*, 12(3), pp.529-541.
26. A. Hussain, M. Asif, M. B. Ahmad, T. Mahmood, and M. A. Raza, 'Malware detection using machine learning algorithms for windows platform', in *Proceedings of International Conference on Information Technology and Applications: ICITA 2021*, 2022, pp. 619–632.
27. Schultz, M.G., Eskin, E., Zadok, F. and Stolfo, S.J., 2000. Data mining methods for detection of new malicious executables. In *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001* (pp.38-49). IEEE.

28. Sreenivas, R.S. and Anitha, R., 2011. Detecting keyloggers based on traffic analysis with periodic behaviour. *Network Security*, 2011(7), pp.14-19.
29. Fu, J., Liang, Y., Tan, C. and Xiong, X., 2010. Detecting software keyloggers with dendritic cell algorithm. In *2010 International Conference on Communications and Mobile Computing* (Vol.1, pp.111-115). IEEE.
30. Amancio, D.R., Comin, C.H., Casanova, D., Travieso, G., Bruno, O.M., Rodrigues, F.A. and da Fontoura Costa, L., 2014. A systematic comparison of supervised classifiers. *PloS one*, 9(4), p.e94137.
31. Sirigiri, M., Sirigiri, D., Aishwarya, R. and Yogitha, R., 2023. Malware Detection and Analysis using Machine Learning. In *2023 7th International Conference on Computing Methodologies and Communication (ICCMC)* (pp.1074-1081). IEEE.
32. Hussain, A., Asif, M., Ahmad, M.B., Mahmood, T. and Raza, M.A., 2022. Malware detection using machine learning algorithms for windows platform. In *Proceedings of International Conference on Information Technology and Applications: ICITA 2021* (pp.619-632). Springer Nature Singapore.
33. Isohara, T., Takemori, K. and Kubota, A., 2011. Kernel-based behavior analysis for android malware detection. In *2011 seventh international conference on computational intelligence and security* (pp.1011-1015). IEEE.
34. Balakrishnan, Y. and Renjith, P.N., 2023. An analysis on Keylogger Attack and Detection based on Machine Learning. In *2023 International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF)* (pp.1-8). IEEE.
35. Pillai, D. and Siddavatam, I., 2019. A modified framework to detect keyloggers using machine learning algorithm. *International Journal of Information Technology*, 11, pp.707-712.
36. F. E. Ayo, J. B. Awotunde, O. A. Olalekan, A. L. Imoize, C.-T. Li, and C.-C. Lee, 'CBFISKD: A combinatorial-based fuzzy inference system for keylogger detection', *Mathematics*, vol. 11, no. 8, p. 1899, 2023.

37. S. Sharma, C. Rama Krishna, and S. K. Sahay, ‘Detection of advanced malware by machine learning techniques’, in *Soft Computing: Theories and Applications: Proceedings of SoCTA 2017*, 2019, pp. 333–342.
38. A. K Nazeer and T. Rj, ‘Malware Classification using Deep Learning’, in *Proceedings of the International Conference on Systems, Energy and Environment*, 2022
M. S. Alsubaie, S. H. Atawneh, and M. S. Abual-Rub, ‘Building Machine Learning Model with Hybrid Feature Selection Technique for Keylogger Detection’, *International Journal of Advances in Soft Computing & Its Applications*, vol. 15, no. 2, 2023.
39. S. L. A. Suresh and A. S. Philip, ‘Multiple botnet and keylogger attack detection using CNN in IoT networks’, in *2022 International Conference on Futuristic Technologies (INCOFT)*, 2022, pp. 1–6.
40. S. S. S. Sugi and S. R. Ratna, ‘Investigation of machine learning techniques in intrusion detection system for IoT network’, in *2020 3rd international conference on intelligent sustainable systems (ICISS)*, 2020, pp. 1164–1167.
41. E. S. Gsr, M. Azees, C. H. R. Vinodkumar, and G. Parthasarathy, ‘Hybrid optimization enabled deep learning technique for multi-level intrusion detection’, *Advances in Engineering Software*, vol. 173, p. 103197, 2022.
42. G. Mohamed, J. Visumathi, M. Mahdal, J. Anand, and M. Elangovan, ‘An effective and secure mechanism for phishing attacks using a machine learning approach’, *Processes*, vol. 10, no. 7, p. 1356, 2022.