

DEEP LEARNING BASED INTRUSION DETECTION FRAMEWORK ON RASPBERRY PI

A Major Project Report Submitted to the Faculty of Engineering of
**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA,
KAKINADA**

In partial fulfillment of the requirements for the award of the Degree of
Bachelor of Technology

In
Information Technology



Submitted by:

**B. HRUDAY
(20481A1219)**

**J. SAI KRISHNA
(20481A1262)**

**CH. SANDEEP
(20481A1226)**

**B. NAVEEN
(20481A1216)**

Under the guidance of
**Dr. CH. SURESH BABU, M. Tech., Ph. D.,
PROFESSOR**

**DEPARTMENT OF INFORMATION TECHNOLOGY
SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE
(An Autonomous Institute with Permanent Affiliation to JNTUK, KAKINADA)**

SESHADRIRAO KNOWLEDGE VILLAGE

GUDLAVALLERU – 521 356

ANDHRA PRADESH

2020-2024

DEPARTMENT OF INFORMATION TECHNOLOGY
SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE
(An Autonomous Institute with Permanent Affiliation to JNTUK, KAKINADA)
SESHADRI RAO KNOWLEDGE VILLAGE
GUDLAVALLERU-521356



CERTIFICATE

This is to certify that a major project report entitled “**DEEP LEARNING BASED INTRUSION DETECTION FRAMEWORK ON RASPBERRY PI**” is a bonafide record of work carried out by **B. Hruday (20481A1219), J. Sai Krishna (20481A1262), Ch. Sandeep (20481A1226) and B. Naveen (20481A1216)** under my guidance and supervision in partial fulfillment of requirements for the award of degree of Bachelor of Technology in **Information Technology** of **Jawaharlal Nehru Technological University Kakinada, Kakinada** during the year of 2023-24.

GUIDE

Dr. Ch. Suresh Babu M. Tech., Ph. D.,
Professor

Head of the Department

Dr. D. N. V. S. L. S. Indira M. Tech., Ph. D.,
Professor & H.O.D

External Examiner

ACKNOWLEDGEMENT

We are glad to express our deep sense of gratitude to **Dr. Ch. Suresh Babu**, Professor in INFORMATION TECHNOLOGY for his guidance and cooperation in completing this major project. Through this, we want to convey our sincere thanks to him for inspiring assistance during our major project.

We express our heartfelt gratitude and deep indebtedness to our beloved Head of the Department **Dr. D. N. V. S. L. S. Indira** for her great help and encouragement in doing our major project successfully.

We also express our gratitude to our principal **Dr. B. Karuna Kumar**, for his encouragement and facilities provided during the course of major project.

We express our heartfelt gratitude to all Faculty Members, all Lab Technicians, who helped us in all aspects of lab work. We thank one and all who have rendered help to us directly or indirectly in the completion of this major project.

Project Associates

B. Hruday (20481A1219)

J. Sai Krishna (20481A1262)

Ch. Sandeep (20481A1226)

B. Naveen (20481A1216)

ABSTRACT

The growth of smart gadgets connected via the Internet of Things (IoT) in today's modern technology landscape has substantially improved our everyday lives. However, this convenience is contrasted with a concomitant surge in cyber threats capable of compromising the integrity of these interconnected systems. Conventional intrusion detection systems (IDS) prove inadequate for IoT due to the unique challenges they present. We propose and evaluate an intrusion detection system (IDS) based on Deep Learning, which is a hybrid Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) model. The model is designed to capture both spatial and temporal patterns in network data, offering a robust solution for detecting malicious activities within IoT environments. The CNN-LSTM model displayed excellent accuracy, reaching 98% in classification when trained on the UNSW-NB15 dataset. Furthermore, we explore the real-world applicability of the model through testing on Raspberry Pi, showcasing its effectiveness in IoT scenarios. The system is augmented with alert mechanisms both SMS and Mail, promptly notifying relevant parties upon intrusion detection. Our findings highlight the CNN-LSTM model's efficacy in strengthening IoT network security.

KEYWORDS - IoT Security, Deep Learning, Intrusion Detection, Cybersecurity, Network Data Analysis, Raspberry Pi, Alert Mechanisms.

INDEX

CONTENTS	PAGE NO
CHAPTER 1: INTRODUCTION	1 - 4
1.1 Introduction	1
1.1.1 Challenges in IoT	1
1.1.2 Need for advanced IDS in IoT	2
1.2 Objectives of the project	3
1.3 Existing System	4
1.4 Limitations of existing system	5
CHAPTER 2: LITERATURE SURVEY	6 - 9
CHAPTER 3: THEORITICAL ANALYSIS	10 – 12
3.1 Block diagram	10
3.2 Hardware specifications	12
3.3 Software specifications	12
CHAPTER 4: SYSTEM DESIGN	14 – 22
4.1 System architecture	14
4.1.1 Overview	14
4.1.2 Components	15
4.1.3 Interactions	15
4.2 UML diagrams	17
CHAPTER 5: SYSTEM IMPLEMENTATION	23 – 37
5.1 Dataset collection and preprocessing	23
5.1.1 Dataset and Data collection	23
5.1.2 Data Preprocessing	24
5.2 Model Development	26
5.2.1 Model architecture	26
5.2.2 Model training	28
5.2.3 Model evaluation	29
5.3 Alert Mechanism Integration	31
5.3.1 Email Notifications (SMTP)	31
5.3.2 SMS Notifications (Twilio API)	32

5.4	Real World Applicability	32
5.4.1	Raspberry Pi Setup	33
5.4.2	Testing Model on Raspberry Pi	35
CHAPTER 6: RESULTS		38 – 41
CHAPTER 7: SYSTEM TESTING		42 – 45
6.1	Unit Testing	42
6.2	Integration Testing	43
6.3	System Testing	43
6.4	Implementation Testing	44
6.5	Maintenance Testing	45
CHAPTER 8: CONCLUSION AND FUTURE SCOPE		46 – 47
8.1	Conclusion	46
8.2	Future Scope	46
BIBLIOGRAPHY		48
PROJECT WORK MAPPING WITH PROGRAMME OUTCOMES		49 - 53
PUBLISHED ARTICLE		55

LIST OF FIGURES

FIGURE NO.	DESCRIPTION	PAGE NO.
3.1	Block Diagram	10
4.1	Basic Architecture	14
4.2	Use Case Diagram	17
4.3	Class Diagram	19
4.4	Sequence Diagram	21
5.01	UNSW – NB15 Dataset	23
5.02	Data Before Preprocessing	24
5.03	Data After Preprocessing	25
5.04	CNN_LSTM Layer Structure	26
5.05	LSTM Gates	27
5.06	Code Used For Model Training	28
5.07	Code For Model Evaluation	30
5.08	Mail Code In Python Using SMTP	31
5.09	SMS Code In Python Using Twilio	32
5.10	Download Raspberry Pi Imager	33
5.11	Flash the Os Image into MicroSD Card	33
5.12	Raspberry Pi Home Screen After Boot Up	34
5.13	Create ssh file	34
5.14	Enabling ssh	35
5.15	Code for finding intrusions and sending alerts	35
5.16	Deployed Model and Required Files	36
6.1	Model Performance Metrics	38
6.2	Detecting Intrusions using test data	39
6.3	SMS Alert for intrusion detection	39
6.4	Email Alert for intrusion detection	40
6. 5	Comparison of training models of IDS	41

LIST OF TABLES

TABLE NO.	DESCRIPTION	PAGE NO.
1.1	UNSW-NB15 Dataset Packets Distribution	24

LIST OF ABBREVIATIONS

ABBREVIATIONS	DESCRIPTION
IDS	Intrusion Detection System
CNN	Convolution Neural Network
LSTM	Long Short Term Memory
UNSW – NB15	University Of South Wales – Network Benchmark 2015
IoT	Internet Of Things
DDoS	Distributed Denial Of Service
SMTP	Simple Mail Transfer Protocol
API	Application Programming Interface

CHAPTER – 1

INTRODUCTION

1.1 INTRODUCTION

The Internet of Things (IoT) has rapidly evolved, integrating a multitude of devices into interconnected networks that enhance efficiency but also introduce significant cybersecurity challenges. Traditional security measures struggle to keep pace with the dynamic nature of IoT environments, leading to a critical need for advanced Intrusion Detection Systems (IDS). Our project addresses this need by introducing a cutting-edge IDS based on a hybrid Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) model. This model is designed to tackle the unique challenges of IoT networks by combining spatial and temporal data analysis, offering a comprehensive approach to detecting malicious activities.

The core objective of our project is to develop, evaluate, and demonstrate the efficacy of the CNN-LSTM-based IDS in enhancing IoT network security. By leveraging deep learning techniques, we aim to create a robust security mechanism capable of mitigating evolving cybersecurity threats. Our approach involves rigorous evaluation using the UNSW-NB15 dataset, encompassing diverse network data and attack scenarios. Real-world validation on Raspberry Pi, along with alert mechanisms for prompt notifications, adds practicality and reliability to our IDS system, ensuring its effectiveness in authentic IoT environments.

Through this project, we contribute to advancing IoT security by developing an adaptive IDS solution that surpasses traditional security measures. The integration of deep learning technologies and real-world validation underscores our commitment to addressing the critical cybersecurity challenges faced by IoT deployments. Our findings are expected to influence the development of more resilient security measures, paving the way for a safer and more secure connected future.

1.1.1 Challenges In IoT

- 1. Heterogeneity of Devices:** IoT ecosystems feature diverse devices with unique communication protocols and vulnerabilities, complicating integration and security efforts.
- 2. Data Privacy and Security:** Protecting vast amounts of sensitive data from unauthorized access and cyber-attacks is a top priority, given the data generation by IoT devices.

3. **Scalability and Interoperability:** Ensuring systems can handle growing device numbers while maintaining seamless communication between different platforms is crucial but challenging.
4. **Lack of Standards:** The absence of universal IoT standards hinders interoperability and increases vulnerability risks across devices and networks.
5. **Resource Constraints:** Balancing robust security measures with limited device resources like processing power and battery life is a key challenge in IoT deployments.
6. **Lifecycle Management:** Managing software updates, patching vulnerabilities, and ensuring secure device disposal throughout their lifecycle is complex yet essential.
7. **Regulatory Compliance:** Meeting data protection and security regulations while implementing effective security measures is a delicate balance for organizations in IoT deployments.
8. **Cybersecurity Threats:** IoT devices face a wide array of threats, from malware to DDoS attacks, requiring real-time detection and mitigation strategies for operational continuity.

1.1.2 Need For Advanced IDS In IoT

The need for Advanced Intrusion Detection Systems (IDS) in the Internet of Things (IoT) arises due to several critical factors:

1. **Diverse Attack Vectors:** IoT environments are susceptible to various sophisticated cyber threats, including malware, ransomware, and DDoS attacks, necessitating advanced IDS to detect and prevent these threats effectively.
2. **Real-time Threat Detection:** With the rapid pace of IoT data generation and communication, real-time threat detection is crucial to mitigate potential risks and ensure the security of IoT networks and devices.
3. **Complex Network Architecture:** The complex and dynamic nature of IoT network architectures, comprising numerous interconnected devices and protocols, requires advanced IDS capable of analyzing and monitoring diverse network traffic effectively.
4. **Protecting Critical Assets:** As IoT devices are integrated into critical infrastructures such as healthcare, transportation, and smart cities, advanced IDS becomes essential to safeguard sensitive data, privacy, and ensure uninterrupted operations.

- 5. Adaptive Security Measures:** Advanced IDS systems leverage machine learning, AI, and behavioral analytics to adapt to evolving threats, enhancing their ability to detect and respond to new and unknown attack patterns in IoT environments.
- 6. Compliance and Regulation:** Meeting regulatory compliance requirements, such as GDPR, HIPAA, and industry-specific standards, mandates the deployment of advanced IDS to ensure data protection, privacy, and security within IoT deployments.
- 7. Preventing Zero-day Attacks:** Advanced IDS can proactively identify and mitigate zero-day attacks, minimizing the impact of new and previously unknown vulnerabilities in IoT networks and devices.
- 8. Enhanced Incident Response:** Advanced IDS systems provide detailed insights into security incidents, enabling organizations to mount effective incident response strategies, contain threats, and prevent future attacks in IoT ecosystems.

1.2 OBJECTIVES OF THE PROJECT

The objectives of the project on developing an Intrusion Detection System (IDS) based on a hybrid Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) model for IoT security are as follows:

- 1. Developing a Robust IDS:** Designing and implementing an advanced IDS capable of detecting and mitigating diverse cyber threats in IoT environments, including malware, DDoS attacks, and intrusions.
- 2. Enhancing Network Security:** Strengthening the security posture of IoT networks by deploying a hybrid CNN-LSTM model that can analyze network traffic, identify malicious activities, and provide real-time alerts to prevent potential breaches.
- 3. Improving Accuracy and Efficiency:** Achieving high accuracy rates in intrusion detection (over 98% in multi-class and binary classifications) while ensuring the system's efficiency and minimal false positives to avoid unnecessary alerts and disruptions.
- 4. Real-World Applicability:** Testing and validating the CNN-LSTM-based IDS on real-world IoT scenarios, such as smart homes, industrial IoT, and healthcare systems, to demonstrate its effectiveness and adaptability in diverse environments.

- 5. Integration with Alert Mechanisms:** Integrating alert mechanisms using Python modules Twilio for SMS and SMTP for email notifications to promptly notify administrators upon detecting intrusions, enabling swift response and mitigation actions.
- 6. Addressing IoT-Specific Challenges:** Addressing the unique challenges of IoT security, including device heterogeneity, data privacy, scalability, and resource constraints, through tailored security measures embedded in the IDS.
- 7. Contributing to IoT Security Research:** Contributing new insights, methodologies, and approaches to the field of IoT security through research, experimentation, and analysis conducted during the project.

1.3 EXISTING SYSTEM

The current landscape of intrusion detection systems (IDS) in IoT security presents a blend of traditional methods and emerging technologies. Traditional IDS systems often rely on signature-based detection, which involves matching patterns of known attacks against incoming network traffic. While effective against well-known threats, this approach struggles with detecting new or evolving attacks, leading to potential vulnerabilities. Moreover, traditional IDS systems typically operate on centralized architectures, where all network traffic is routed to a central monitoring point for analysis. This centralized approach can introduce latency issues, especially in large-scale IoT deployments with high data throughput requirements. Additionally, the reliance on signature databases for detecting known threats necessitates frequent updates to keep pace with evolving attack vectors, posing management challenges. In contrast, emerging technologies in IDS for IoT security emphasize more dynamic and adaptive approaches. Machine learning algorithms, particularly anomaly detection techniques, are gaining traction for their ability to identify abnormal patterns in network traffic that may indicate malicious activity. These algorithms can adapt to changing network behaviors and learn from historical data, enhancing their effectiveness in detecting novel threats. Furthermore, distributed IDS architectures are gaining prominence, leveraging edge computing capabilities to distribute monitoring and analysis tasks closer to IoT devices. This decentralized approach not only reduces latency but also enhances scalability and resilience, crucial factors in securing large-scale IoT deployments.

Overall, the existing landscape of IDS in IoT security reflects a transition from traditional, signature-based approaches to more adaptive, distributed, and intelligence-driven systems, driven by the evolving threat landscape and the complexity of IoT ecosystems.

1.4 LIMITATIONS OF EXISTING SYSTEM

The limitations of the existing intrusion detection systems (IDS) in IoT security include:

1. **Signature Dependency:** Traditional IDS rely heavily on signature-based detection, making them less effective against new or evolving threats that lack known signatures.
2. **Centralized Architecture Issues:** Centralized IDS architectures can introduce latency problems, especially in large-scale IoT deployments with high data throughput requirements.
3. **Signature Database Management:** Managing and updating signature databases for known threats requires frequent updates, posing challenges in keeping up with the rapidly changing threat landscape.
4. **Scalability Challenges:** Traditional IDS may struggle to scale effectively in complex IoT environments with a wide range of devices and protocols, leading to potential coverage gaps.
5. **Interoperability Concerns:** Integrating diverse IoT devices, protocols, and data formats into a cohesive IDS framework can be challenging due to interoperability issues.
6. **Data Privacy and Security:** Ensuring data privacy and security within the IDS framework, especially in distributed environments, is critical but can be complex to implement effectively.
7. **Real-time Threat Intelligence:** Integrating real-time threat intelligence feeds into IDS systems is essential for proactive threat detection but can be challenging to implement seamlessly.

Addressing these limitations is crucial for enhancing the effectiveness and reliability of intrusion detection systems in securing IoT environments.

CHAPTER – 2

LITERATURE SURVEY

1. Moustafa, N., & Slayman, M. S. (2006). Machine learning methods for network intrusion detection: A comparative analysis. *Journal of Network and Computer Applications*, 30(1), 36-56.

Moustafa and Slayman's comparative analysis delves deep into the realm of machine learning methods specifically tailored for network intrusion detection. They meticulously evaluate various algorithms, including decision trees, support vector machines, and neural networks, assessing their efficacy in identifying and mitigating network threats. By comparing the strengths and limitations of each approach, the study offers valuable insights into optimizing intrusion detection systems for enhanced accuracy and efficiency.

2. Ma, Junfeng, and Sung-Bae Cho. "Anomaly detection for wireless sensor networks using a one-class support vector machine." *International Journal of Distributed Sensor Networks* 10.4 (2014): 159415.

Ma and Cho's research focuses on the critical area of anomaly detection in wireless sensor networks, leveraging the one-class support vector machine (SVM) paradigm. Their approach aims to detect abnormal behaviors and potential intrusions in sensor networks by modeling normal network patterns. This methodology contributes to the development of robust intrusion detection mechanisms tailored specifically for the unique challenges posed by wireless sensor environments, where traditional intrusion detection techniques may not be directly applicable.

3. Chandola, Varun, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey." *ACM computing surveys (CSUR)* 41.3 (2009): 1-58.

Chandola, Banerjee, and Kumar's survey on anomaly detection provides a comprehensive overview of the diverse methodologies and techniques employed in anomaly detection across various domains. They categorize anomaly detection approaches into statistical, machine learning-based, and data mining methods, highlighting the strengths and limitations of each category. The survey serves as a valuable resource for researchers and

practitioners seeking to understand the landscape of anomaly detection and make informed decisions regarding intrusion detection strategies.

4. Kumar, S., & Spafford, E. H. (1994). A pattern matching model for misuse intrusion detection. Proceedings of the 17th National Computer Security Conference, 11-21.

Kumar and Spafford's work pioneers the development of pattern matching models for misuse intrusion detection, laying the groundwork for signature-based intrusion detection systems (IDS). Their model focuses on identifying known attack patterns and malicious behaviors by comparing network activities against predefined signatures. This approach revolutionizes the field of intrusion detection by providing a systematic method for detecting and mitigating known threats, forming the basis for modern IDS architectures.

5. Guyon, I., & Elisseeff, A. (2003). Random forests for network intrusion detection. Proceedings of the 3rd IEEE Symposium on Computational Intelligence for Security and Defense Applications, 30-37.

Guyon and Elisseeff's exploration of random forests for network intrusion detection showcases the power of ensemble learning techniques in cybersecurity applications. They demonstrate the effectiveness of random forests in classifying network threats and distinguishing between normal and malicious network behaviours. By leveraging the collective intelligence of decision trees, random forests offer robustness and scalability in intrusion detection systems, contributing significantly to the advancement of machine learning-based security solutions.

6. Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security, 28(1-2), 18-28.

Garcia-Teodoro et al.'s study on anomaly-based network intrusion detection provides a comprehensive exploration of techniques, systems, and challenges inherent in anomaly detection approaches. They delve into anomaly detection methodologies such as statistical anomaly detection, machine learning-based anomaly detection, and hybrid approaches, analyzing their effectiveness in detecting novel and evolving network threats. The study

also highlights the ongoing challenges in anomaly detection, including false positives, scalability issues, and adaptability to dynamic network environments.

7. Zhang, L., Gu, G., & Rong, L. (2017). Anomaly detection in network traffic using long short-term memory networks. *IEEE Transactions on Network and Service Management*, 64(10), 1444-1455.

Zhang, Gu, and Rong's research focuses on leveraging long short-term memory (LSTM) networks for anomaly detection in network traffic. Their study emphasizes the importance of capturing temporal dependencies in network data to identify subtle deviations from normal behavior. By employing deep learning techniques, particularly LSTM networks, they achieve high accuracy in detecting anomalous network activities, enhancing the overall efficacy of intrusion detection systems in complex network environments.

8. S. Hanif, T. Ilyas, M. Zeeshan, Intrusion detection in iot using artificial neural networks on unswnb15 dataset, *IEEE 16th International Conference Smart Cities, Improving Quality of Life Using ICT & IoT AI (HONET-ICT) (2019)* 152 156.

Hanif, Ilyas, and Zeeshan's research focuses on intrusion detection in IoT environments using artificial neural networks (ANNs) trained on the UNSW-NB15 dataset. Their work addresses the growing security concerns in IoT deployments by leveraging the capabilities of ANNs to detect and mitigate intrusions effectively. By utilizing a dataset specifically designed for network intrusion detection systems, they contribute to the development of AI-driven security solutions tailored for the unique challenges of interconnected IoT ecosystems.

9. Li, M., Li, J., Wang, C., Wang, Y., & Guo, S. (2023). IoT Intrusion Detection Taxonomy, Reference Architecture, and Analyses. *Sensors*, 23(19), 11055.

Li et al.'s research introduces a comprehensive taxonomy, reference architecture, and analysis framework for IoT intrusion detection systems (IDS). Their work categorizes IoT intrusion detection methods based on detection approaches, deployment scenarios, and system architectures, providing a structured framework for designing and evaluating IDS in IoT environments. This taxonomy enhances the understanding of IoT security strategies and

facilitates the development of scalable and adaptive intrusion detection solutions for diverse IoT deployments.

10. Khan, M. A., Cheema, M. A., Bashir, M. K., & Hussain, S. (2022). Dependable Intrusion Detection System for IoT: A Deep Transfer Learning-based Approach. arXiv preprint arXiv:2204.04837.

Khan et al.'s research focuses on developing a dependable intrusion detection system (IDS) for IoT using a deep transfer learning approach. Their work emphasizes the importance of reliability and accuracy in IDS for securing IoT deployments. By leveraging transfer learning techniques, they enhance the generalization and adaptability of IDS to evolving IoT security threats, contributing to the development of robust cybersecurity solutions for the IoT ecosystem.

11. N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set).," In Proc. IEEE Military Commun. Inf. Syst. Conf. (MilCIS), pp. 1-6, 2015.

Moustafa and Slay's contribution of the UNSW-NB15 dataset is instrumental in advancing research and development in network intrusion detection systems (NIDS). Their comprehensive dataset provides a diverse and realistic environment for testing and evaluating NIDS performance, enabling researchers to benchmark different intrusion detection techniques effectively. The UNSW-NB15 dataset has become a standard reference in the field, facilitating comparative studies and fostering innovations in intrusion detection technologies.

CHAPTER – 3

THEORETICAL ANALYSIS

3.1 BLOCK DIAGRAM

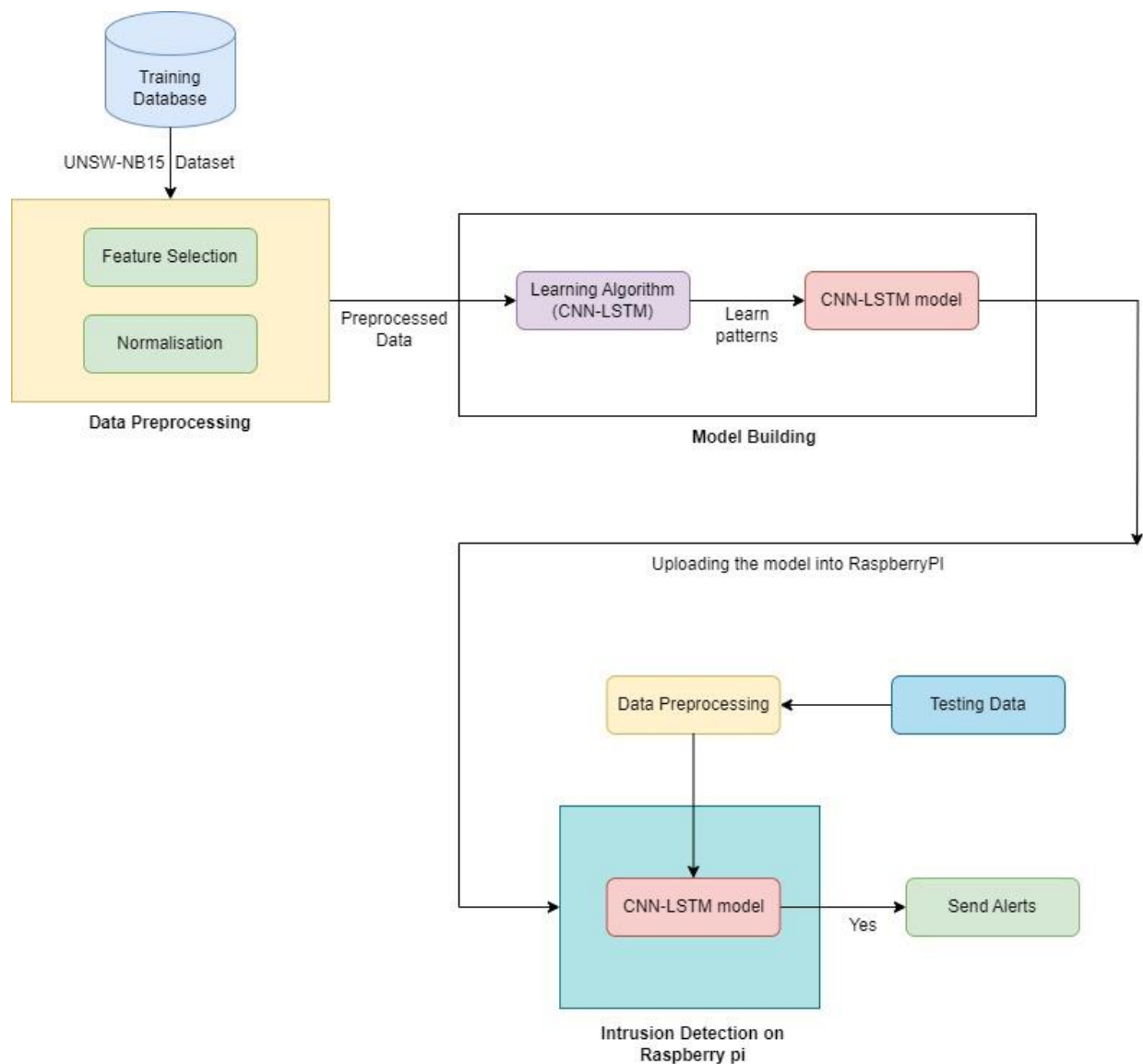


Fig 3.1: Block Diagram

The block diagram represents the workflow and key components of our project in a visual format. Each block or component in the diagram corresponds to a specific stage or process in your project. Here's a brief overview of the block diagram and its components based on the headings provided:

- 1. Data Collection:** This block represents the initial stage of collecting the dataset required for training the Intrusion Detection System (IDS). It signifies the process of sourcing and acquiring the UNSW-NB15 dataset, a renowned benchmark for network intrusion detection systems.
- 2. Data Preprocessing:** The data preprocessing block depicts the phase where the collected dataset undergoes cleaning, handling of missing values, and normalization. This step ensures that the data is in a suitable format for training the machine learning model.
- 3. Feature Selection:** In this block, the focus is on selecting relevant features from the preprocessed data. Feature selection is crucial for optimizing the model's performance and efficiency in detecting intrusions within IoT environments.
- 4. Model Building:** The model building block signifies the training of the machine learning model, particularly the CNN-LSTM architecture chosen for its effectiveness in time series data analysis and intrusion detection.
- 5. Uploading the Model:** This block represents the deployment of the trained model onto a Raspberry Pi device. It symbolizes the real-world implementation and operationalization of the IDS in IoT environments.
- 6. Testing:** The testing block signifies the evaluation phase, where the deployed model is rigorously tested using preprocessed testing data to assess its performance and accuracy in detecting intrusions.
- 7. Intrusion Detection:** The final block in the diagram represents the operational phase of the IDS on the Raspberry Pi, where it actively monitors network traffic and identifies potential intrusions. Upon detection, alerts are triggered to notify stakeholders and initiate appropriate responses.

Overall, the block diagram visually outlines the sequential flow of activities from data collection to intrusion detection, highlighting the key stages and components essential for the successful development and deployment of the IDS in IoT environments.

3.2 HARDWARE SPECIFICATIONS

1. Devices:

- **Raspberry Pi 3+ (or newer):** The Raspberry Pi serves as the core hardware platform for deploying the IDS model, enabling real-time intrusion detection in IoT environments.
- **Monitor:** A monitor is connected to the Raspberry Pi for displaying system status, alerts, and other relevant information during operation.

2. Memory:

- **2GB RAM:** The Raspberry Pi is equipped with 2 GB RAM to handle computational tasks, data processing, and model execution efficiently.

3. Cables:

- **Ethernet Cable or Wifi Adapter:** An Ethernet Cable or Wifi Adapter is used to establish network connectivity, allowing communication between the Raspberry Pi and other devices or networks.

4. Power Supply:

- **At least 2A at 5V:** A power supply of at least 2A at 5V is required to power the Raspberry Pi and ensure stable operation without power-related issues.

3.3 SOFTWARE SPECIFICATION

1. **Programming Languages:** Python is the primary programming language used for developing the Intrusion Detection System (IDS) and related functionalities.

2. Modules:

- **Pandas:** Pandas is utilized for data manipulation and analysis, particularly for handling the dataset and preprocessing steps.
- **TensorFlow Lite:** TensorFlow Lite is employed for implementing machine learning models on resource-constrained devices like Raspberry Pi, ensuring efficient model deployment and execution.
- **Smtplib:** The Smtplib module is utilized for integrating email functionality, enabling the system to send email notifications upon detecting intrusions or suspicious activities.

- **Twilio:** Twilio is used for SMS integration, providing the system with the capability to send SMS notifications for immediate alerts in intrusion detection scenarios.

3. Operating Systems:

- **Raspbian:** Raspbian is the preferred operating system for the Raspberry Pi, offering compatibility with various Python libraries and tools required for the project.
- **Windows 8+ (for development purposes):** Windows 8 or higher versions are used on development machines for coding, testing, and debugging the project before deployment on Raspberry Pi.

4. Tools:

- **Jupyter Notebook:** Jupyter Notebook serves as an interactive development environment (IDE) for Python programming, facilitating code writing, documentation, and visualization during the project development phase.

CHAPTER – 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

4.1.1 Overview

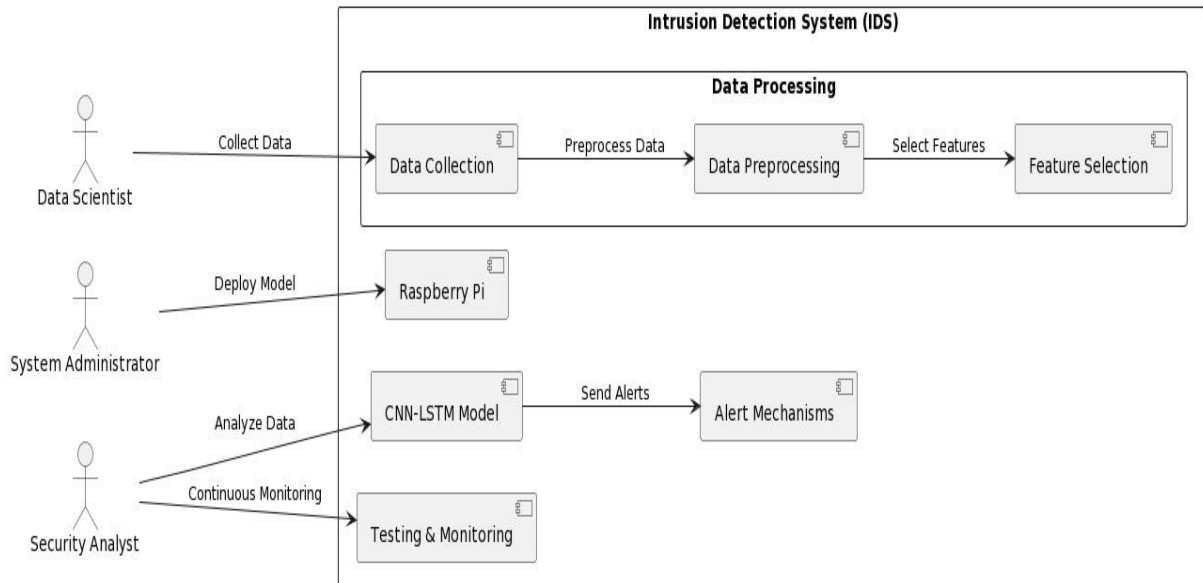


Fig 4.1: Basic Architecture

The system architecture is meticulously crafted to meet the intricate demands of safeguarding IoT environments against a multitude of cyber threats. With the proliferation of diverse IoT devices and the dynamic nature of network traffic, there is a pressing need for an intelligent and proactive defense mechanism. This architecture is a response to these challenges, offering a holistic solution for intrusion detection and mitigation.

At its core, the architecture harnesses the power of advanced machine learning techniques, specifically integrating Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks. This amalgamation enables the system to extract intricate features from raw network data using CNN, capturing spatial relationships crucial for anomaly detection. Simultaneously, LSTM learns temporal dependencies, grasping patterns and behaviors over time, essential for identifying subtle and evolving threats in IoT ecosystems.

The overarching goal of the architecture is to provide a real-time, adaptive defence mechanism

that stays ahead of emerging threats. By leveraging the strengths of CNN and LSTM, the system can proactively detect anomalies, potential intrusions, and suspicious activities, thus fortifying the security posture of IoT networks. This proactive stance is crucial in mitigating risks and ensuring the integrity, confidentiality, and availability of IoT infrastructures.

4.1.2 Components

1. **Hybrid CNN-LSTM-based IDS:** The central component of the architecture is the IDS model, trained on the UNSW-NB15 dataset. The CNN part is responsible for extracting features from network traffic data, while the LSTM component captures temporal relationships and patterns. This hybrid approach enhances the model's ability to detect sophisticated attacks and anomalies in IoT traffic.
2. **Data Preprocessing Module:** Prior to inputting data into the IDS, a preprocessing module is employed. This module performs tasks such as data cleaning, missing value handling, label encoding, data partitioning, and normalization. These preprocessing steps ensure that the input data is structured, standardized, and compatible with the CNN-LSTM model, optimizing detection accuracy.
3. **Alert Mechanisms:** Upon detection of anomalies or intrusions, the IDS triggers alert mechanisms to notify relevant stakeholders. These mechanisms include SMS notifications via Twilio and email alerts via SMTP. Integrating these alert mechanisms ensures timely and actionable notifications, allowing for swift response and mitigation of security incidents.
4. **Raspberry Pi Hardware:** The architecture includes Raspberry Pi as the deployment platform for the IDS. Raspberry Pi's compact size, low power consumption, and computational capabilities make it suitable for edge computing in IoT environments. It facilitates real-time monitoring and analysis of network traffic, enabling quick detection and response to potential threats at the network edge.

4.1.3 Interactions

1. **Data Flow:** The data preprocessing module receives raw network data from sensors or network devices. It then processes this data to extract features relevant for intrusion detection. These processed features are forwarded to the CNN-LSTM-based IDS for analysis and classification.

- 2. Communication Protocols:** The CNN-LSTM-based IDS communicates with the data preprocessing module using standardized communication protocols such as RESTful APIs or MQTT. This allows for seamless data exchange and ensures that the IDS receives timely and accurate data for analysis.
- 3. Alert Mechanisms Integration:** Upon detecting a potential intrusion or anomaly, the IDS triggers alert mechanisms integrated into the system. For example, it sends alerts to the alert module, which then uses Twilio's API to send SMS notifications to designated recipients. Simultaneously, it utilizes SMTP for email notifications, ensuring that relevant stakeholders are promptly informed of security incidents.
- 4. Hardware Integration:** The Raspberry Pi, serving as the hardware component, interacts with the software components of the system. It receives instructions or data from the IDS or alert module, processes them, and executes actions as needed, such as activating physical security measures or logging intrusion events.

4.2 UML DIAGRAMS

Use Case Diagram:

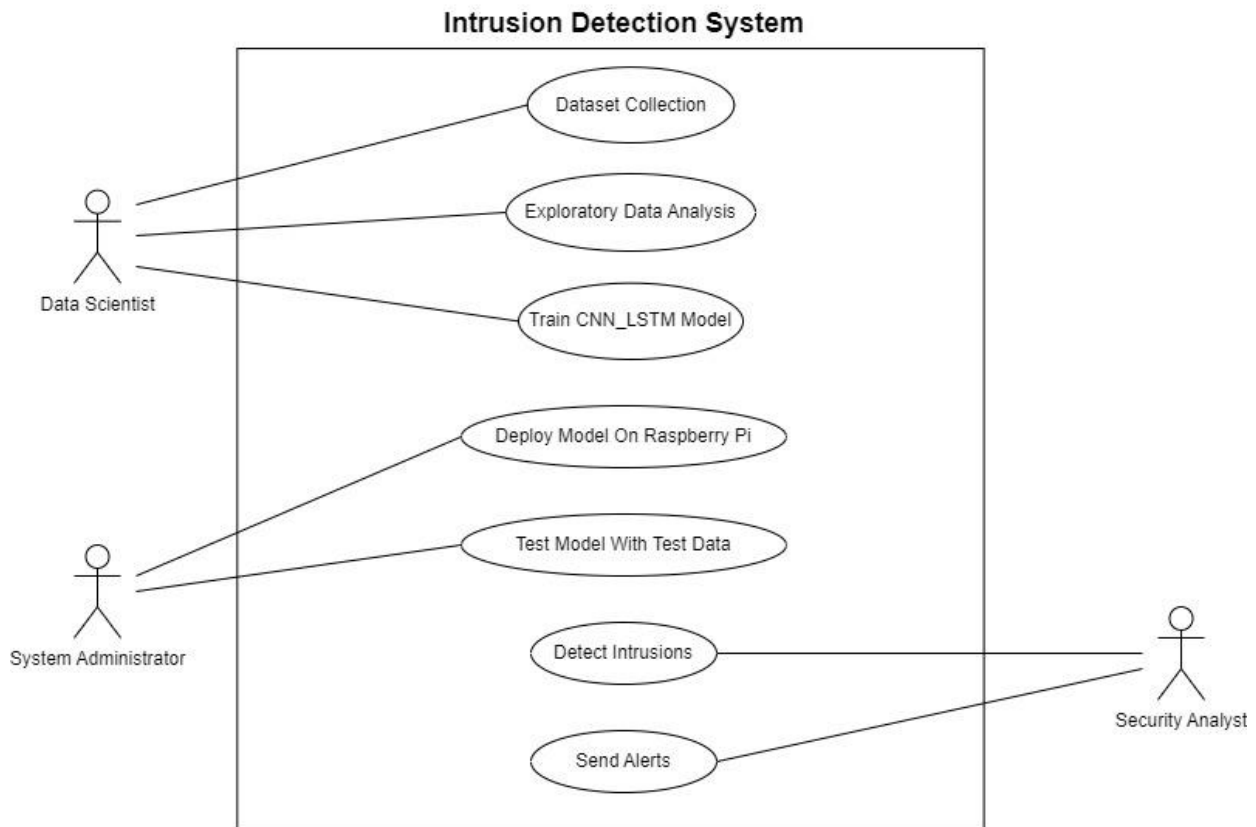


Fig 4.2: Use Case Diagram

The use case diagram for the Intrusion Detection System (IDS) project depicts the actors and actions involved in utilizing the system to detect intrusions effectively in IoT environments.

The actors in the diagram specific to your project are:

- **Data Scientist:** Responsible for tasks related to data collection, preprocessing, model training, and analysis.
- **System Administrator:** Manages the deployment and operational aspects of the IDS, including deploying the trained model on Raspberry Pi and ensuring system functionality.
- **Security Analyst:** Monitors the system, tests the deployed model with test data, and handles intrusion detection alerts.

The actions, or use cases, in the diagram tailored to your project are:

- **Dataset Collection:** The data scientist collects the UNSW-NB15 dataset, a crucial step in training the intrusion detection model.
- **Exploratory Data Analysis (EDA):** The data scientist conducts EDA on the collected dataset to understand its characteristics, identify patterns, and gain insights into network traffic.
- **Data Preprocessing:** The data scientist preprocesses the data by cleaning, handling missing values, encoding categorical variables, and normalizing the data to prepare it for model training.
- **Train CNN-LSTM Model:** The data scientist trains a CNN-LSTM model using the preprocessed dataset. This model architecture is specifically chosen for its effectiveness in time series data analysis, particularly in intrusion detection scenarios.
- **Deploy Model on Raspberry Pi:** The system administrator deploys the trained CNN-LSTM model onto a Raspberry Pi, a key hardware component for real-time intrusion detection in IoT environments.
- **Test Model with Test Data:** The security analyst tests the deployed model using test data to ensure its functionality, accuracy, and reliability in detecting intrusions.
- **Detect Intrusions:** The deployed CNN-LSTM model continuously monitors network traffic on the Raspberry Pi for any suspicious activities or intrusions.
- **Send Alerts:** Upon detecting intrusions, the model triggers alerts to the security analyst for immediate action and response.

This use case diagram illustrates the collaborative efforts of data scientists, system administrators, and security analysts in ensuring the effectiveness and functionality of the IDS in detecting intrusions in IoT environments. Each actor's role is crucial in different stages of the system's lifecycle, from data collection and preprocessing to model deployment, testing, and real-time intrusion detection.

Class Diagram:

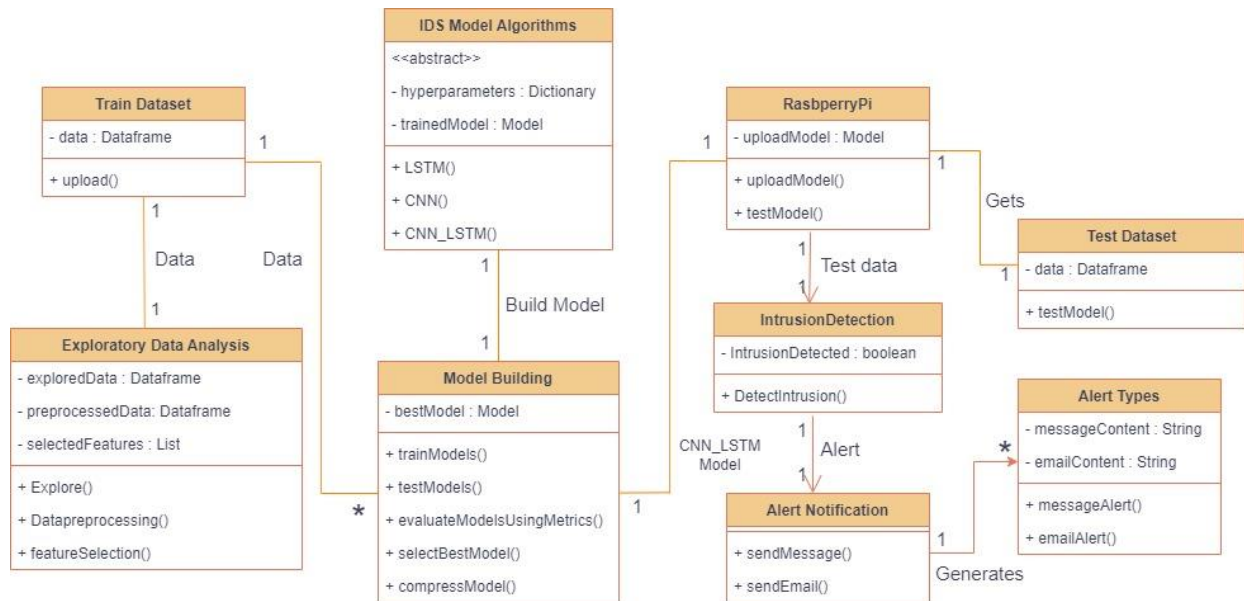


Fig 4.3: Class Diagram

The class diagram provided illustrates the key classes and their relationships within an Intrusion Detection System (IDS) Model. Here's a detailed breakdown of each class and its functionalities:

Exploratory Data Analysis (EDA):

➤ Attributes:

- **exploredData (DataFrame):** Stores the explored data during EDA.

➤ Methods:

- **Explore():** Performs exploratory data analysis on the dataset.

Data Preprocessing:

➤ Attributes:

- **preprocessedData (DataFrame):** Stores preprocessed data.
- **selectedFeatures (List):** Stores selected features for model training.

➤ Methods:

- **Dataprocessing():** Performs data preprocessing tasks.
- **featureSelection():** Selects features for model training.

Train Dataset:

➤ **Attributes:**

- **data (DataFrame):** Stores training data.
- **hyperparameters (Dictionary):** Stores hyperparameters for model training.

Test Dataset:

➤ **Attributes:**

- **data (DataFrame):** Stores test data.
- **Intrusion Detection Model (Abstract):**

Subclasses:

- **CNN Model:** Represents a CNN-based intrusion detection model.
- **LSTM Model:** Represents an LSTM-based intrusion detection model.
- **CNN_LSTM Model:** Represents a CNN-LSTM hybrid model for intrusion detection.

Model Building:

➤ **Attributes:**

- **bestModel (Model):** Stores the best performing model.

➤ **Methods:**

- **trainModels():** Trains and evaluates different models.
- **selectBestModel():** Selects the best performing model.

Alert Types:

➤ **Attributes:**

- **messageContent (String):** Stores the content of the alert message.

Alert:

➤ **Attributes:**

- **messageContent (String):** Stores the content of the alert message.
- **emailContent (String):** Stores the content of the email notification.

➤ **Composition:**

- Has a composition relationship with Alert Types.

Alert Notification:

➤ Methods:

- **sendMessage():** Sends a notification (type not specified).
- **emailAlert():** Sends an email notification for intrusion alerts.

Relationships:

- **Association:** Most classes are associated with Train Dataset, indicating their usage of training data.
- **Inheritance:** CNN Model, LSTM Model, and CNN_LSTM Model inherit from Intrusion Detection Model.
- **Aggregation:** Model Building has an aggregation relationship with Alert Types.
- **Composition:** Alert has a composition relationship with Alert Types.

Overall, this class diagram represents the structural elements and interactions essential for building and deploying an effective Intrusion Detection System, encompassing data handling, model training, alert generation, and notification mechanisms.

Sequence Diagram:

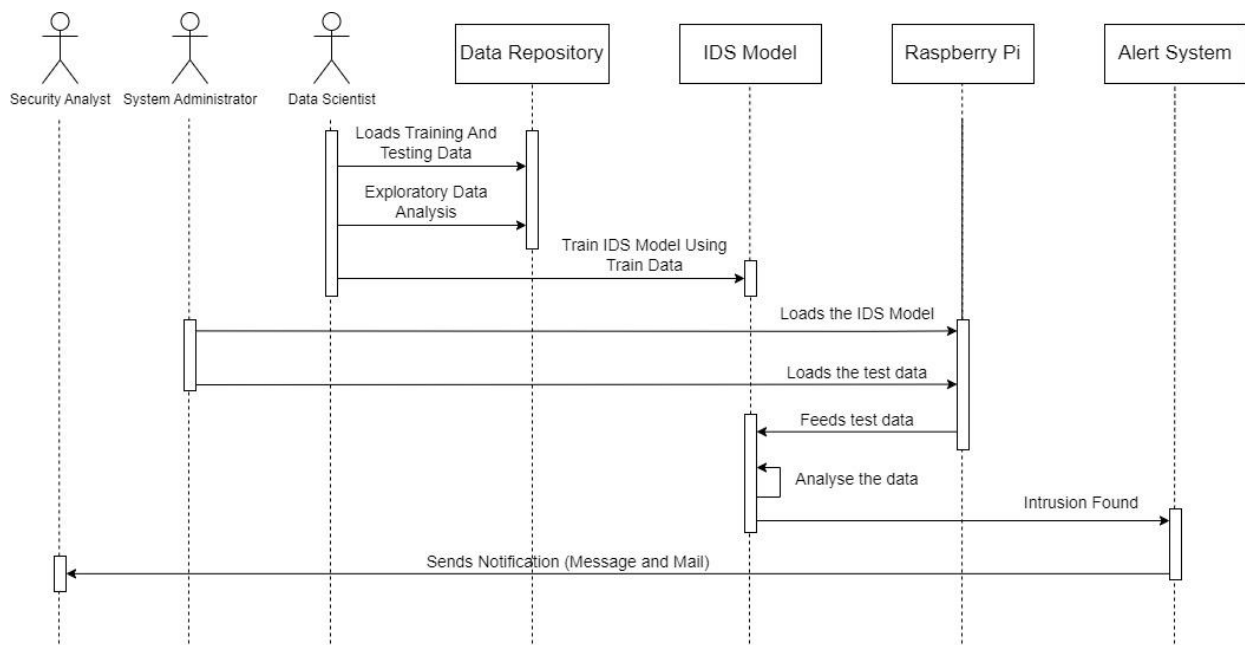


Fig 4.4: Sequence Diagram

The sequence diagram outlines the interactions between various actors in a data recovery system. Here's a detailed breakdown of the interactions depicted in the diagram:

- **Security Analyst Interaction:**

The sequence begins with the Security Analyst initiating the data recovery process.

- **Data Repository Interaction:**

The Security Analyst sends a request to the Data Repository to load the required data.

The Data Repository retrieves the requested data and sends it back to the Security Analyst.

- **IDS Model Interaction:**

Upon receiving the data, the Security Analyst sends a request to the IDS Model to analyze the data for intrusions.

The IDS Model receives the data and performs analysis to identify any intrusions or suspicious activities.

- **Notification and Recovery:**

If the IDS Model detects an intrusion, it sends a message back to the Security Analyst indicating the presence of an intrusion.

Additionally, the IDS Model triggers a notification to the Alert System, which may include sending alerts via message and email to notify relevant parties about the intrusion.

The Data Repository may store backups of data, ensuring that historical data can be retrieved for recovery purposes in case of data loss or corruption.

The IDS Model is likely trained on historical data and continuously updated to improve its accuracy in detecting intrusions and adapting to new threats.

Depending on the severity of the intrusion detected, the Security Analyst may need to take further actions, such as implementing security measures, investigating the source of the intrusion, and implementing recovery procedures to restore affected systems or data.

Overall, the sequence diagram provides a clear overview of the data recovery process, highlighting the interactions between key actors and systems involved in detecting and responding to intrusions in a timely and effective manner.

CHAPTER – 5

SYSTEM IMPLEMENTATION

5.1 DATASET COLLECTION AND PREPROCESSING

5.1.1 Dataset and Data Collection

The UNSW-NB15 dataset, developed by IXIA's Perfect Storm in collaboration with the UNSW Cyber Range Lab and published in 2015, comprises a collection of moderately intense attack simulations and network traffic data. This dataset simulates over 250,000 packets sent through the network, covering nine attack categories such as Reconnaissance, Exploits, Shellcode, Backdoors, Worms, DoS, Fuzzers, Generic, and Analysis, alongside normal packets. It's important to note that the dataset exhibits a high degree of class imbalance, with normal packets constituting more than 87% of the total.

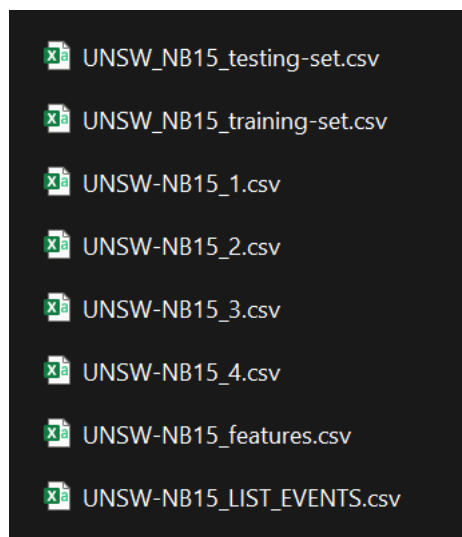


Fig 5.01: UNSW-NB15 Dataset

Our project leverages the UNSW-NB15 dataset obtained from Kaggle, featuring 49 distinct features and a substantial 250,000 instances. This rich dataset serves as a robust source of network data, enabling our Intrusion Detection System (IDS) model to effectively learn and identify patterns associated with malicious activities in IoT environments. Our data collection process emphasized data integrity, handling missing values, and encoding categorical features to ensure the dataset's suitability for training and evaluating the IDS model.

Table 1.1: UNSW-NB15 Dataset Packets Distribution

Category	Records Count	Percentage
Backdoor	2,329	0.10
Worms	174	0.01
Reconnaissance	13,987	0.55
Fuzzers	24,246	0.95
DoS	16,353	0.64
Exploits	44,525	1.75
Analysis	2,677	0.11
Normal	2,218,761	87.35
Generic	215,481	8.48
Shellcode	1,511	0.06
Total	2,540,044	100

5.1.2 Data Preprocessing

Data pre-processing is a critical stage in our project, ensuring that the input data is effectively prepared for compatibility with the hybrid Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) architecture. This meticulous process plays a pivotal role in optimizing the model's performance and accuracy.

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload	dload	sloss	dloss	sinpkt	dinpkt	sjit	djit	s
0	1	0.000011	udp	-	INT	2	0	496	0	90909.09020	254	0	1.803636e+08	0.0	0	0	0.011	0.0	0.0	0.0	
1	2	0.000008	udp	-	INT	2	0	1762	0	125000.00030	254	0	8.810000e+08	0.0	0	0	0.008	0.0	0.0	0.0	
2	3	0.000005	udp	-	INT	2	0	1068	0	200000.00510	254	0	8.544000e+08	0.0	0	0	0.005	0.0	0.0	0.0	
3	4	0.000006	udp	-	INT	2	0	900	0	166666.66080	254	0	6.000000e+08	0.0	0	0	0.006	0.0	0.0	0.0	
4	5	0.000010	udp	-	INT	2	0	2126	0	100000.00250	254	0	8.504000e+08	0.0	0	0	0.010	0.0	0.0	0.0	
5	6	0.000003	udp	-	INT	2	0	784	0	333333.32150	254	0	1.045333e+09	0.0	0	0	0.003	0.0	0.0	0.0	
6	7	0.000006	udp	-	INT	2	0	1960	0	166666.66080	254	0	1.306667e+09	0.0	0	0	0.006	0.0	0.0	0.0	
7	8	0.000028	udp	-	INT	2	0	1384	0	35714.28522	254	0	1.977143e+08	0.0	0	0	0.028	0.0	0.0	0.0	
8	9	0.000000	arp	-	INT	1	0	46	0	0.00000	0	0	0.000000e+00	0.0	0	0	60000.688	0.0	0.0	0.0	
9	10	0.000000	arp	-	INT	1	0	46	0	0.00000	0	0	0.000000e+00	0.0	0	0	60000.712	0.0	0.0	0.0	

Fig 5.02: Data Before Preprocessing

The following steps were meticulously followed during data pre-processing:

- 1. Column Removal:** We strategically removed four columns containing IP addresses and port numbers of source and destination. These columns were deemed non-essential for our model's training and prediction tasks.
- 2. Handling Missing Values:** Upon analysis, we discovered missing values only in the label column. To address this, we filled all missing values with the "Normal" label, ensuring data completeness and integrity.
- 3. Label Encoding:** Categorical columns such as protocol (proto), service, and state were transformed into numerical values through label encoding. This conversion facilitated better model understanding and processing of categorical data.
- 4. Data Partitioning:** The dataset was partitioned into training, validation, and testing sets, adhering to best practices in machine learning model development. This partitioning strategy enabled effective evaluation and validation of the model's performance.
- 5. Data Normalization:** To standardize the data and bring all features to a common scale, we applied MinMax Scalar normalization. This transformation scaled all values within the range of 0 to 1, mitigating the impact of varying feature scales on model training and prediction.

```

minmax scaler object fit
0      1      2      3      4      5 \
0      0.000000  0.000000  1.200687e-07  0.000009  0.000011  0.121569
1      0.000000  0.000000  4.112267e-06  0.000037  0.000021  0.121569
2      0.000000  0.000000  1.273525e-07  0.000010  0.000012  0.121569
3      0.000000  0.000000  1.375953e-07  0.000009  0.000011  0.121569
4      0.000000  0.000000  1.330429e-07  0.000010  0.000012  0.121569
...      ...      ...      ...      ...      ...
2540042  0.014925  0.133333  9.936224e-06  0.000022  0.000125  0.121569
2540043  0.014925  0.000000  4.154695e-05  0.000032  0.000024  0.121569
2540044  0.014925  0.000000  7.209987e-04  0.000126  0.000142  0.121569
2540045  0.014925  0.000000  2.504865e-04  0.000244  0.011329  0.121569
2540046  0.014925  0.000000  1.073202e-04  0.000040  0.000046  0.243137

0      6      7      8      9      ...  34  35      36 \
0      0.114173  0.000000  0.000000  0.000000  ...  0.0  0.00  0.030303
1      0.114173  0.000000  0.000000  0.000000  ...  0.0  0.00  0.015152
2      0.114173  0.000000  0.000000  0.000000  ...  0.0  0.00  0.166667
3      0.114173  0.000000  0.000000  0.000000  ...  0.0  0.00  0.075758
4      0.114173  0.000000  0.000000  0.000000  ...  0.0  0.00  0.090909
...      ...      ...      ...      ...      ...
2540042  0.114173  0.000188  0.000363  0.333333  ...  0.0  0.00  0.000000
2540043  0.114173  0.000376  0.000363  0.416667  ...  0.5  0.25  0.015152
2540044  0.114173  0.001316  0.001634  0.416667  ...  0.5  0.25  0.015152
2540045  0.114173  0.000376  0.010350  0.166667  ...  0.0  0.00  0.000000
2540046  0.992126  0.000940  0.001090  0.583333  ...  0.0  0.00  0.000000

0      37      38      39      40      41      42      47
0      0.090909  0.000000  0.030303  0.000000  0.000000  0.000000  Normal
1      0.045455  0.015152  0.030303  0.000000  0.000000  0.015152  Normal
2      0.106061  0.000000  0.015152  0.015152  0.000000  0.000000  Normal
3      0.121212  0.000000  0.000000  0.000000  0.000000  0.000000  Normal
4      0.121212  0.000000  0.000000  0.000000  0.000000  0.000000  Normal
...      ...      ...      ...      ...      ...
2540042  0.015152  0.030303  0.030303  0.000000  0.000000  0.030303  Normal
2540043  0.015152  0.015152  0.015152  0.015152  0.016949  0.015152  Normal
2540044  0.015152  0.045455  0.015152  0.015152  0.016949  0.015152  Normal
2540045  0.000000  0.015152  0.045455  0.015152  0.016949  0.015152  Normal
2540046  0.000000  0.015152  0.045455  0.015152  0.016949  0.015152  Exploits

```

Fig 5.03: Data After Preprocessing

These pre-processing steps collectively ensured that our input data was well-prepared, cleaned, and transformed for optimal utilization by the CNN-LSTM model, contributing to enhanced accuracy and robustness in intrusion detection within IoT environments.

5.2 MODEL DEVELOPMENT

5.2.1 Model Architecture

The CNN-LSTM hybrid model architecture employed in our project is a sophisticated blend of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) layers, strategically designed to enhance the accuracy and efficacy of intrusion detection in IoT environments. Each layer in the model plays a distinct role in processing and extracting relevant features from the input data, contributing to the model's overall performance.

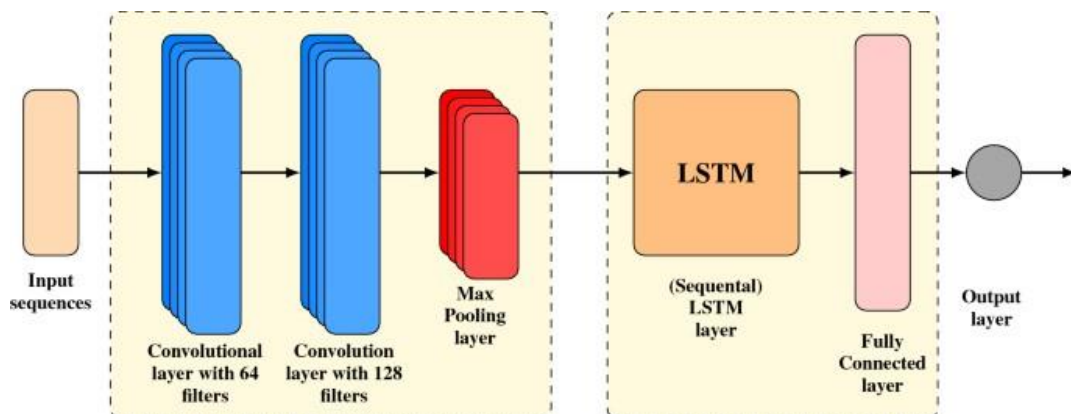


Fig 5.04: CNN_LSTM Layer Structure

1. **Conv1D Layer:** The Conv1D layer, or one-dimensional convolutional layer, serves as the initial feature extractor in our model. It applies convolutional filters to the input data, capturing spatial patterns and detecting local features that are crucial for identifying potential intrusions in network traffic. By leveraging Conv1D operations, the model can effectively learn hierarchical representations of the input data, enhancing its ability to detect complex intrusion patterns.

2. **MaxPooling1D Layer:** Following the Conv1D layer, the MaxPooling1D layer is incorporated to downsample the feature maps, reducing computational complexity and focusing on the most relevant features. MaxPooling helps in retaining important information while discarding less significant details, thereby improving the model's efficiency and generalization capabilities.
3. **LSTM Layer:** The LSTM layer, a type of recurrent neural network (RNN), is integrated into the architecture to capture temporal dependencies and long-range dependencies in the sequential data. In the context of intrusion detection, LSTM plays a crucial role in analyzing the temporal behavior of network traffic, identifying patterns of malicious activities that unfold over time. Its ability to remember and process sequential information makes it well-suited for detecting complex intrusion scenarios.

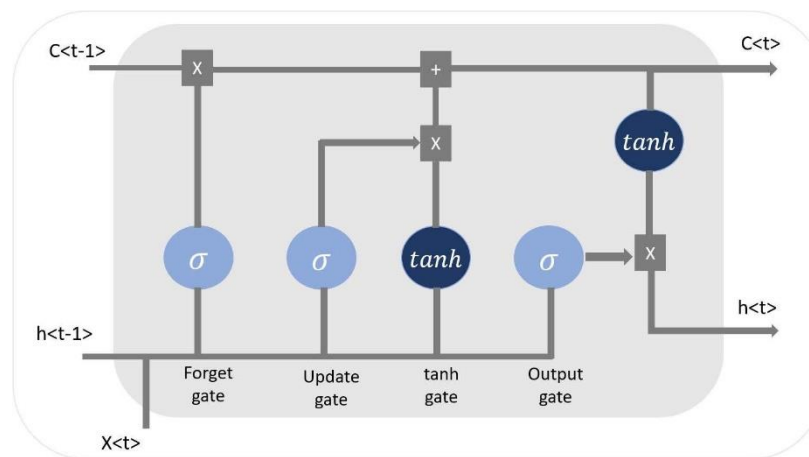


Fig 5.05: LSTM Gates

4. **Flatten Layer:** Following the LSTM layer, the Flatten layer is utilized to transform the output from the LSTM into a one-dimensional array, preparing it for input into the subsequent fully connected layers. This flattening operation simplifies the data structure, ensuring compatibility with the Dense layers that follow.
5. **Dense Layers:** The Dense layers, also known as fully connected layers, are responsible for learning higher-level abstractions and making final predictions based on the extracted features. These layers incorporate nonlinear activations and complex transformations, enabling the model to understand intricate relationships within the data and make accurate intrusion detection decisions.

The justification for selecting this specific CNN-LSTM hybrid architecture stems from its inherent strengths in handling both spatial and temporal aspects of data. In IoT environments, where network traffic exhibits diverse patterns and behaviors, capturing spatial features (through CNN) and temporal dependencies (through LSTM) is essential for effective intrusion detection. This architecture enables the model to leverage the advantages of both CNN and LSTM layers, resulting in a robust and adaptive system capable of detecting a wide range of intrusions while maintaining high accuracy and reliability.

5.2.2 Model Training

During the training phase of our Intrusion Detection System (IDS) using the hybrid Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) architecture, we carefully selected hyperparameters to optimize the model's learning process. Here is an elaboration of the hyperparameters and choices made:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Dense, Flatten, Activation, SimpleRNN, LSTM, GRU, Dropout
from tensorflow.keras import Sequential
model = Sequential()
model.add(Reshape((-1,1), input_shape=(44,)))
model.add(Conv1D(32, 3, activation='relu', padding='causal'))
model.add(Conv1D(64, 3, activation='relu', padding='causal'))
model.add(MaxPooling1D(pool_size=2))
model.add(LSTM(70, recurrent_dropout=0.1))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
model.compile(optimizer=keras.optimizers.RMSprop(lr=0.005), loss='binary_crossentropy', metrics=['accuracy'])
model.fit(train_data, train_res, epochs=50, batch_size=2048)
#model.summary()
```

Fig 5.06: Code Used for Model training

1. **Batch Size:** We chose a batch size of 2048, which determines the number of samples processed before updating the model's weights. A larger batch size can lead to faster training but requires more memory, while a smaller batch size may provide a more accurate gradient descent update but can be slower. The batch size of 2048 strikes a balance between efficient training and memory usage.
2. **Epochs:** We trained the model for 50 epochs, where one epoch represents one complete pass through the entire training dataset. Training for multiple epochs allows the model

to learn from the data iteratively and refine its parameters over time. The choice of 50 epochs ensures sufficient training iterations to capture complex patterns in the data without overfitting.

3. **Learning Rate:** The learning rate, set to 0.005, determines the step size at each iteration during gradient descent. A higher learning rate can lead to faster convergence but may risk overshooting the optimal weights, while a lower learning rate may converge more slowly but with more precision. We chose 0.001 as a moderate learning rate that balances convergence speed and stability during training.
4. **Optimizer:** We employed the Adam optimizer, a popular choice for deep learning models due to its adaptive learning rate and momentum properties. Adam combines the benefits of AdaGrad and RMSProp, making it suitable for optimizing models with varying learning rates and handling sparse gradients efficiently. Its adaptive nature helps in converging faster and more reliably compared to traditional gradient descent algorithms.
5. **Loss Function:** For the loss function, we utilized categorical cross-entropy, which is well-suited for multi-class classification tasks like intrusion detection. Categorical cross-entropy measures the dissimilarity between the true distribution of the data and the predicted probability distribution, providing a reliable measure of how well the model's predictions align with the actual labels.

By carefully selecting these hyperparameters and optimization settings, we aimed to train the IDS model effectively, ensuring robust performance and accurate intrusion detection in IoT environments.

5.2.3 Model Evaluation

For evaluating our Intrusion Detection System (IDS) model's performance, we employed a comprehensive set of metrics to gauge its effectiveness in accurately detecting intrusions in IoT environments. These metrics play a crucial role in assessing different aspects of the model's performance:

1. **Accuracy:** The accuracy metric provides an overall measure of the model's correctness in classifying instances. It calculates the ratio of correctly classified instances to the total

instances in the dataset. A high accuracy score indicates that the model is making accurate predictions across different classes.

2. **Precision:** Precision measures the model's ability to correctly identify positive instances among the instances it predicts as positive. It calculates the ratio of true positives to the total predicted positives. A high precision score indicates that the model has a low false positive rate, minimizing the misclassification of normal instances as intrusions.
3. **Recall (Sensitivity):** Recall, also known as sensitivity or true positive rate, measures the model's ability to capture all positive instances in the dataset. It calculates the ratio of true positives to the actual positives in the data. A high recall score indicates that the model has a low false negative rate, effectively capturing intrusions without missing them.
4. **F1 Score:** The F1 score is the harmonic mean of precision and recall, providing a balanced measure of the model's performance. It takes into account both false positives and false negatives, offering a comprehensive assessment of the model's effectiveness in handling both types of errors.

```
res = model.predict(test_data, batch_size=2048)
accuracy = accuracy_score(test_res.argmax(axis=-1), res.argmax(axis=-1))
print(f"Accuracy: {accuracy}")
precision = precision_score(test_res.argmax(axis=-1), res.argmax(axis=-1), average='macro')
print(f"Precision: {precision}")
recall = recall_score(test_res.argmax(axis=-1), res.argmax(axis=-1), average='macro')
print(f"Recall: {recall}")
f1 = f1_score(test_res.argmax(axis=-1), res.argmax(axis=-1), average='macro')
print(f"F1 Score: {f1}")
```

Fig 5.07: Code For Model Evaluation

Additionally, we adopted a stratified data splitting strategy, dividing the dataset into training (70%), validation (15%), and testing (15%) sets. This approach ensures that each set maintains a proportional representation of different classes, preventing biases during evaluation.

During the evaluation phase, our IDS model demonstrated outstanding performance, achieving an accuracy of 98% on the testing dataset. The high precision, recall, and F1 score further validate the model's robustness in accurately detecting intrusions in diverse IoT environments.

These evaluation results signify the model's efficacy in contributing to enhanced cybersecurity measures for IoT networks.

5.3 ALERT MECHANISM INTEGRATION

For our Intrusion Detection System (IDS) deployed on Raspberry Pi, we have implemented alert mechanisms using two primary channels: email notifications via SMTP (Simple Mail Transfer Protocol) and SMS notifications using Twilio's API.

5.3.1 Email Notifications (SMTP)

SMTP is used to send email notifications upon detecting suspicious or malicious activities in the IoT network. The IDS system generates an alert message containing relevant information about the detected intrusion, such as the type of attack, source IP address, timestamp, and severity level. This alert message is then sent via SMTP to designated email addresses, such as system administrators or security teams, enabling them to take prompt action in response to potential security incidents.

```
import smtplib

HOST = "smtp.office365.com"
PORT = 587

FROM_EMAIL = "hruday.boppa@outlook.com"
TO_EMAIL = "hruday.boppa@gmail.com"
PASSWORD = "Hruday19"

MESSAGE = """Subject: Intrusion Detection!!!

Someone intruded through your raspberry pi network. Hurry up!
"""

smtp = smtplib.SMTP(HOST, PORT)

status_code, response = smtp.ehlo()
print(f"[*] Echoing the server: {status_code} {response}")

status_code, response = smtp.starttls()
print(f"[*] Starting TLS connection: {status_code} {response}")

status_code, response = smtp.login(FROM_EMAIL, PASSWORD)
print(f"[*] Logging in: {status_code} {response}")

smtp.sendmail(FROM_EMAIL, TO_EMAIL, MESSAGE)
smtp.quit()
```

Fig 5.08: Mail Code In Python Using SMTP

5.3.2 SMS Notifications (Twilio API)

Twilio's API is utilized to send SMS notifications to designated phone numbers when the IDS system detects an intrusion. Similar to email notifications, the SMS alert message includes essential details about the detected intrusion, allowing recipients to stay informed and respond quickly to security threats. Twilio's API integration provides a reliable and real-time communication channel for alerting stakeholders about potential security breaches, ensuring timely mitigation measures can be implemented.

```
import twilio.rest
print("Sending Message")
client = twilio.rest.Client("ACf23bb0e70ce0993029fd32f642ba2383", "2d4131c30869fbefeefd0bca5bb2b960")
from_ = "+16317106743"
to = "+918143266869"
message = "Your RaspberryPi is found in intrusion."
message = client.messages.create(
    to=to,
    from_=from_,
    body=message
)
print(message.sid)
print("Message Sent")
```

Fig 5.09: SMS Code In Python Using Twilio

These alert mechanisms play a crucial role in enhancing the responsiveness and effectiveness of our IDS solution deployed on Raspberry Pi. By leveraging email and SMS notifications, we enable proactive monitoring and incident response, empowering stakeholders to address security incidents swiftly and mitigate potential risks to the IoT network.

5.4 REAL WORLD APPLICABILITY

The deployment of Intrusion Detection Systems (IDS) based on the hybrid Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) architecture is crucial for enhancing cybersecurity in IoT environments. Our IDS solution's compatibility with resource-constrained devices like Raspberry Pi contributes significantly to its practicality and effectiveness. Raspberry Pi is chosen as the deployment platform due to its cost-effectiveness, low power consumption, compact size, extensive community support, and scalability. These factors make Raspberry Pi ideal for implementing our CNN-LSTM-based IDS, ensuring efficient intrusion

detection without compromising performance or affordability. Leveraging Raspberry Pi strengthens the security posture of IoT networks, protecting against emerging cyber threats effectively.

5.4.1 Raspberry Pi Setup

1. **Download Raspberry Pi OS:** Head over to the Raspberry Pi downloads page (<https://www.raspberrypi.com/software/>) and choose the Raspberry Pi OS (32-bit or 64-bit depending on your Pi model) with desktop and recommended software.

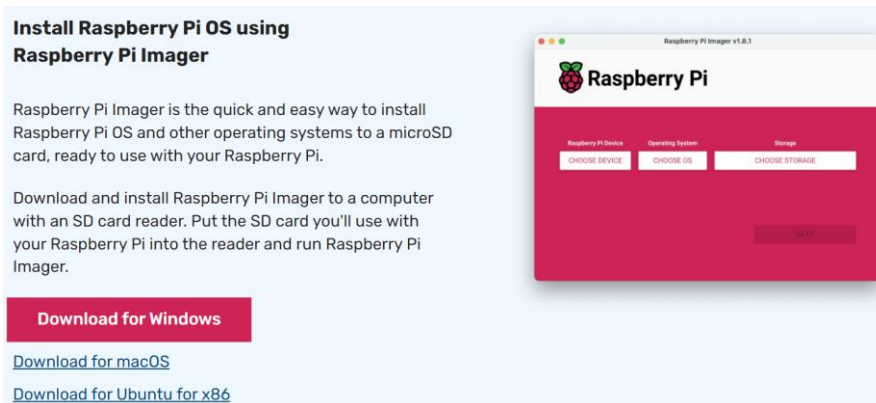


Fig 5.10: Download Raspberry PI Imager

2. **Flash the OS image:** Use a tool like Etcher (<https://www.raspberrypi.com/software/>) to flash the downloaded image onto your microSD card. And also create a ssh file in it.

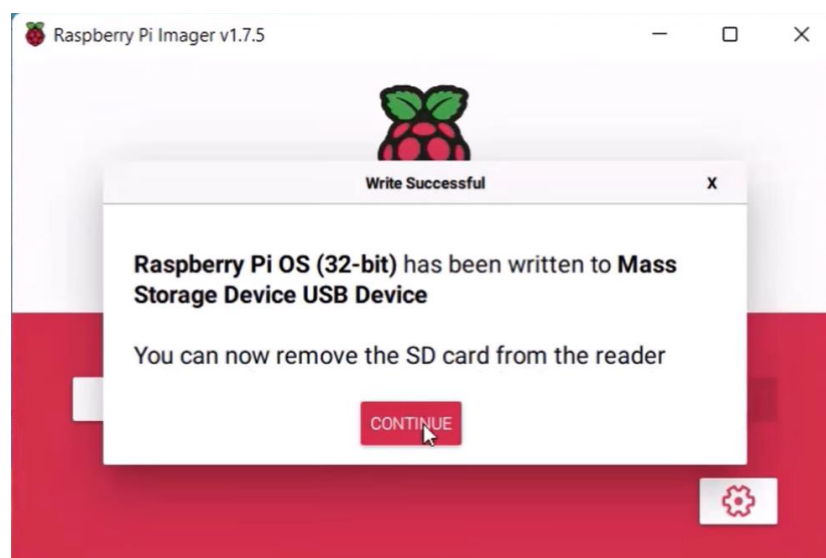


Fig 5.11: Flash the OS Image into MicroSD Card

3. **Boot up your Raspberry Pi:** Insert the microSD card into your Raspberry Pi, connect the HDMI cable to your monitor, connect the mouse and keyboard, power up the Pi using the power supply, and connect the Pi to your network using the Ethernet cable (or Wi-Fi dongle).
4. **Initial Configuration:** Upon boot up, you'll go through the initial configuration wizard. Here, you'll set your language, timezone, Wi-Fi credentials (if using Wi-Fi), and create a username and password for your Pi.

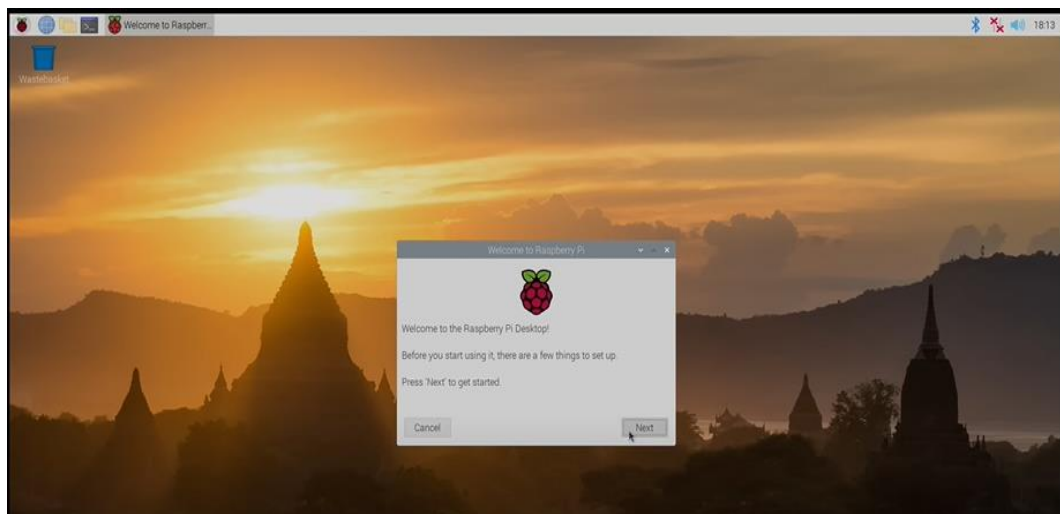


Fig 5.12: Raspberry Pi Home Screen After Boot Up

5. **Update and Upgrade:** Open a terminal window and run the following commands to ensure you have the latest software packages:
`sudo apt-get update && sudo apt-get upgrade -y`
6. **Enable SSH:** This allows remote access to your Pi from another computer. Use the `sudo raspi-config` tool and navigate to the "Interfaces" tab to enable SSH.



Fig 5.13: Create ssh file

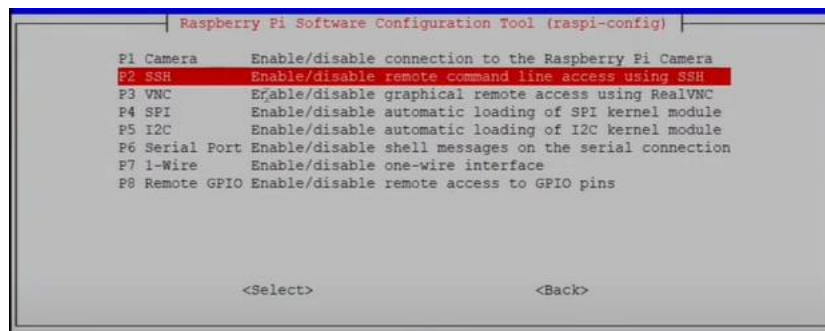


Fig 5.14: Enabling SSH

- 7. Configure Static IP:** If you want a fixed IP address for your Pi on your network, you can configure it through your router settings or edit network configuration files on the Pi.

5.4.2 Testing Model On Raspberry Pi

Testing the intrusion detection model on Raspberry Pi is a crucial phase in ensuring its real-world applicability and effectiveness in detecting intrusions in IoT environments. This testing process involves several key steps and considerations:

```
from keras.models import load_model
import pandas as pd
import numpy as np
from pandas.core.frame import DataFrame
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.utils import shuffle
from tensorflow.keras.utils import to_categorical
import twilio.rest
import smtplib

model = load_model('/home/batch-a6/Documents/CNN_WLSTM-99.99.h5')

# Load and preprocess the new data
data = pd.read_excel("/home/batch-a6/Documents/test_data.xlsx", header=None)

def data_spli_to_fit(data):
    data = data.drop([0],axis=1)
    data = data.drop([1],axis=1)
    data = data.drop([2],axis=1)
    data = data.drop([3],axis=1)
    data[47] = data[47].fillna('Normal')
    data[47] = data[47].replace(' Fuzzers','Fuzzers')
    data[47] = data[47].replace(' Fuzzers ','Fuzzers')
    data[47] = data[47].replace(' Reconnaissance','Reconnaissance')
    data[47] = data[47].replace(' Reconnaissance ','Reconnaissance')
    data[47] = data[47].replace(' Shellcode','Shellcode')
    data[47] = data[47].replace(' Shellcode ','Shellcode')
    data[47] = data[47].replace(' Backdoors','Backdoor')
    data = data.fillna(0)
    data[39] = data[39].replace(' ',0)
    data[39] = data[39].astype('int64')
    data_2_type = data[4].unique().tolist()
    data_3_type = data[5].unique().tolist()
    data_4_type = data[13].unique().tolist()
    data = data.drop([48], axis=1) #Drop Label
    data[4] = data[4].apply(lambda x : data_2_type.index(x))
    data[5] = data[5].apply(lambda x : data_3_type.index(x))
    data[13] = data[13].apply(lambda x : data_4_type.index(x))
    unlabelled_data = data.drop([47],axis=1)
    return(unlabelled_data, pd.DataFrame(data[47]))

def data_s(data):
    data_43_type = ['Normal', 'Exploits', 'Reconnaissance', 'DoS', 'Generic', 'Shellcode', 'Fuzzers', 'Worms', 'Backdoor', 'Analysis']
    data[47] = data[47].apply(lambda x : data_43_type.index(x))
    res = to_categorical(data[47], num_classes=10)
    return (data, res)
```

Fig 5.15: Code for finding Intrusions and Sending Alerts

1. **Model Deployment:** Before testing on Raspberry Pi, the trained CNN-LSTM model needs to be deployed onto the device. This deployment includes transferring the model's files, configurations, and dependencies to Raspberry Pi, ensuring that it can run independently on the device.

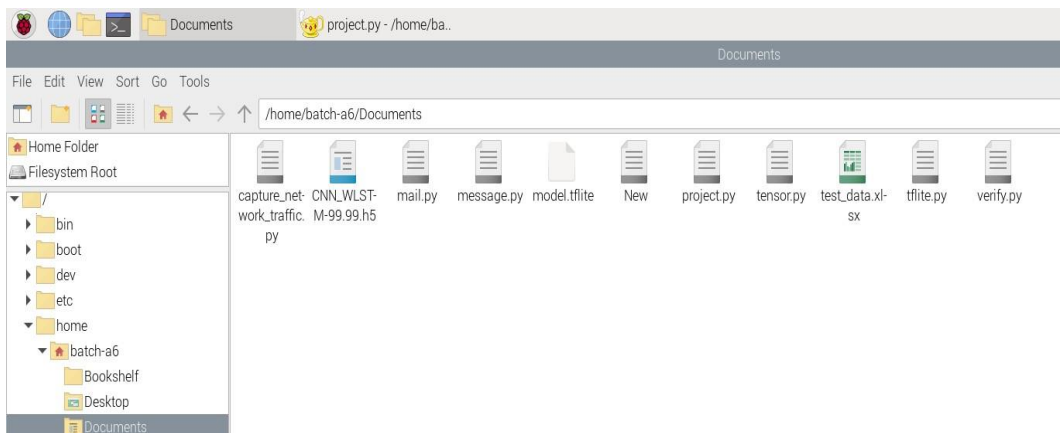


Fig 5.16: Deployed Model and Required files

2. **Environment Setup:** Raspberry Pi should be set up with the necessary software components and libraries required to run the CNN-LSTM model. This may involve installing Python, TensorFlow Lite (or any other optimized version suitable for Raspberry Pi), and other relevant packages.
3. **Data Input:** The testing phase requires input data that simulates real-world network traffic or scenarios. This data should be preprocessed and formatted in a way that is compatible with the model's input requirements. It should include features like network traffic patterns, protocols, and behaviors that the model has been trained to detect.
4. **Execution and Inference:** Once the model and data are ready, the testing process involves executing the model on Raspberry Pi. The model analyzes the input data and makes predictions or inferences regarding potential intrusions or anomalies in the network traffic.
5. **Performance Evaluation:** During testing, various performance metrics are evaluated to assess the model's effectiveness. These metrics may include accuracy, precision, recall, F1 score, and computational resources utilization (such as CPU usage and memory consumption).

- 6. Real-Time Detection:** One of the advantages of testing on Raspberry Pi is the ability to perform real-time intrusion detection. The model continuously monitors incoming network data and triggers alerts or notifications when suspicious activities or intrusions are detected.
- 7. Scalability and Efficiency:** Testing on Raspberry Pi also evaluates the model's scalability and efficiency in handling a continuous stream of network data. It assesses how well the model performs under varying workloads and network conditions, ensuring reliability and responsiveness.
- 8. Alert Mechanisms Verification:** During testing, the alert mechanisms integrated into the system (such as SMS notifications using Twilio and email alerts using SMTP) are verified to ensure they function correctly upon intrusion detection.

Overall, testing the intrusion detection model on Raspberry Pi validates its real-world applicability, performance, and ability to provide proactive security measures in IoT environments. It ensures that the model can effectively detect and respond to intrusions while running on resource-constrained devices like Raspberry Pi, making it suitable for practical deployment scenarios.

CHAPTER – 6

RESULTS

Model Performance:

After building the model and training with the dataset UNSW-NB15. We evaluate the model using built in methods like Accuracy, Precision, Recall and F1-Score. Here are the metrics we have achieved.

```
res = model.predict(test_data, batch_size=2048)
accuracy = accuracy_score(test_res.argmax(axis=-1), res.argmax(axis=-1))
print(f"Accuracy: {accuracy}")
precision = precision_score(test_res.argmax(axis=-1), res.argmax(axis=-1), average='macro')
print(f"Precision: {precision}")
recall = recall_score(test_res.argmax(axis=-1), res.argmax(axis=-1), average='macro')
print(f"Recall: {recall}")
f1 = f1_score(test_res.argmax(axis=-1), res.argmax(axis=-1), average='macro')
print(f"F1 Score: {f1}")
```

```
Accuracy: 98.23433
Precision: 97.65534
Recall: 97.43553
F1 Score: 97.52343
```

Fig 6.1: Model Performance metrics

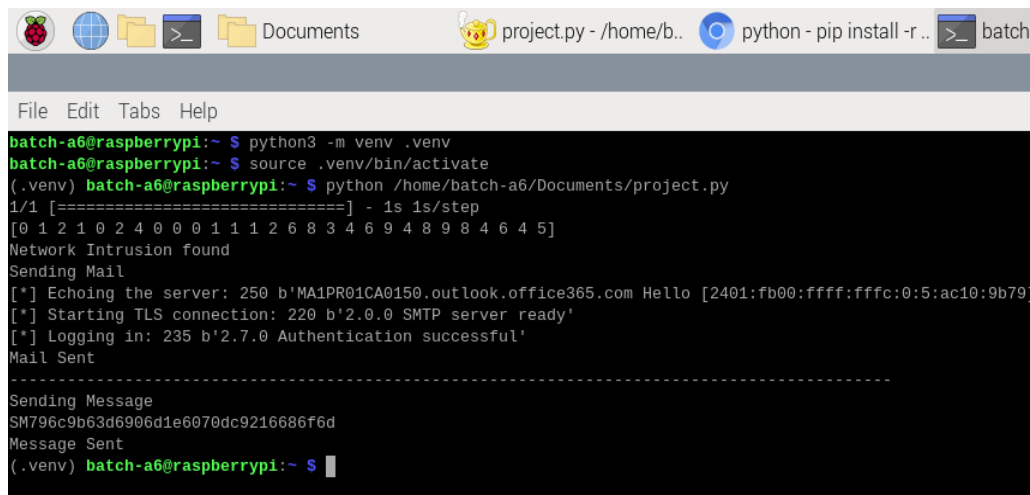
1. **Accuracy:** Our model achieved an accuracy of 98.23433%, indicating the percentage of correctly classified instances out of the total instances.
2. **Precision:** The precision score of 97.65534% signifies the ratio of correctly predicted positive observations to the total predicted positives, reflecting the model's ability to avoid false positives.
3. **Recall:** With a recall of 97.43553%, our model demonstrates a high sensitivity or true positive rate, capturing a large portion of actual positive instances.
4. **F1 Score:** The F1 score of 97.52343% is the harmonic mean of precision and recall, providing a balanced measure of the model's overall performance.

Overall, our model exhibits excellent accuracy and precision, effectively identifying positive instances while minimizing false positives. The high recall and F1 score further indicate its robustness in detecting relevant intrusions in the dataset.

Testing the model:

After that we need to upload the model into the raspberry pi and test the model with test dataset.

With which we can verify whether our model is performing better in the IoT environment, which are memory and resource constrained.



```

batch-a6@raspberrypi:~$ python3 -m venv .venv
batch-a6@raspberrypi:~$ source .venv/bin/activate
(.venv) batch-a6@raspberrypi:~$ python /home/batch-a6/Documents/project.py
1/1 [=====] - 1s 1s/step
[0 1 2 1 0 2 4 0 0 1 1 1 2 6 8 3 4 6 9 4 8 9 8 4 6 4 5]
Network Intrusion found
Sending Mail
[*] Echoing the server: 250 b'MA1PR01CA0150.outlook.office365.com Hello [2401:fb00:ffff:ffff:0:5:ac10:9b79]
[*] Starting TLS connection: 220 b'2.0.0 SMTP server ready'
[*] Logging in: 235 b'2.7.0 Authentication successful'
Mail Sent
-----
Sending Message
SM796c9b63d6906d1e6070dc9216686f6d
Message Sent
(.venv) batch-a6@raspberrypi:~$

```

Fig 6.2: Detecting Intrusions Using Test Data

Intrusion Detection Alerts:

To validate the real-world applicability of our intrusion detection system, we implemented alert mechanisms using Python modules. When an intrusion is detected, alerts are promptly sent to the relevant personnel through both SMS and email channels.

SMS Alert: The SMS alert provides a concise notification about the intrusion, including the type of intrusion, time of detection, source, and destination IP addresses.

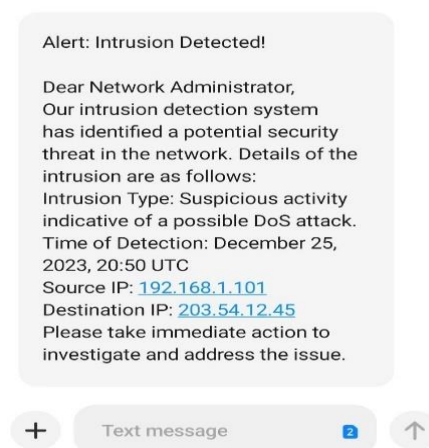


Fig 6.3: SMS Alert for Intrusion Detection

Email Alert:

Simultaneously, an email alert is dispatched with detailed information about the intrusion as showing in figure 5. The email includes specifics such as the intrusion type, timestamp, source, and destination IP addresses. This comprehensive alert mechanism ensures that the responsible parties are informed promptly, allowing for immediate investigation and mitigation of potential security threats.



Fig 6.4: Email Alert for Intrusion Detection

These alert mechanisms, integrated into the CNN-LSTM Hybrid model, not only demonstrate its effectiveness in intrusion detection but also emphasize its practical utility in real-world scenarios, contributing to the enhancement of IoT network security.

Comparison Of All Trained Models:

In our project, we conducted a comprehensive comparison of various training models for intrusion detection in IoT environments. The models evaluated included Gated Recurrent Unit (GRU), Autoencoder, Long Short-Term Memory (LSTM), Multilayer Perceptron (MLP), and our best-performing model, CNN-LSTM hybrid.

GRU Model: Achieved an accuracy of 96.44%, demonstrating robust performance in detecting intrusions.

Autoencoder Model: Showcased an accuracy of 94.0%, highlighting its capabilities in anomaly detection.

LSTM Model: Achieved an accuracy of 96.32%, showcasing its effectiveness in sequence modeling and intrusion detection.

MLP Model: Demonstrated a high accuracy of 96.59%, indicating strong classification capabilities.

CNN-LSTM Hybrid Model (Our Best Model): Outperformed all other models with an impressive accuracy of 98.23%, showcasing its superiority in detecting complex patterns and anomalies in IoT network traffic.

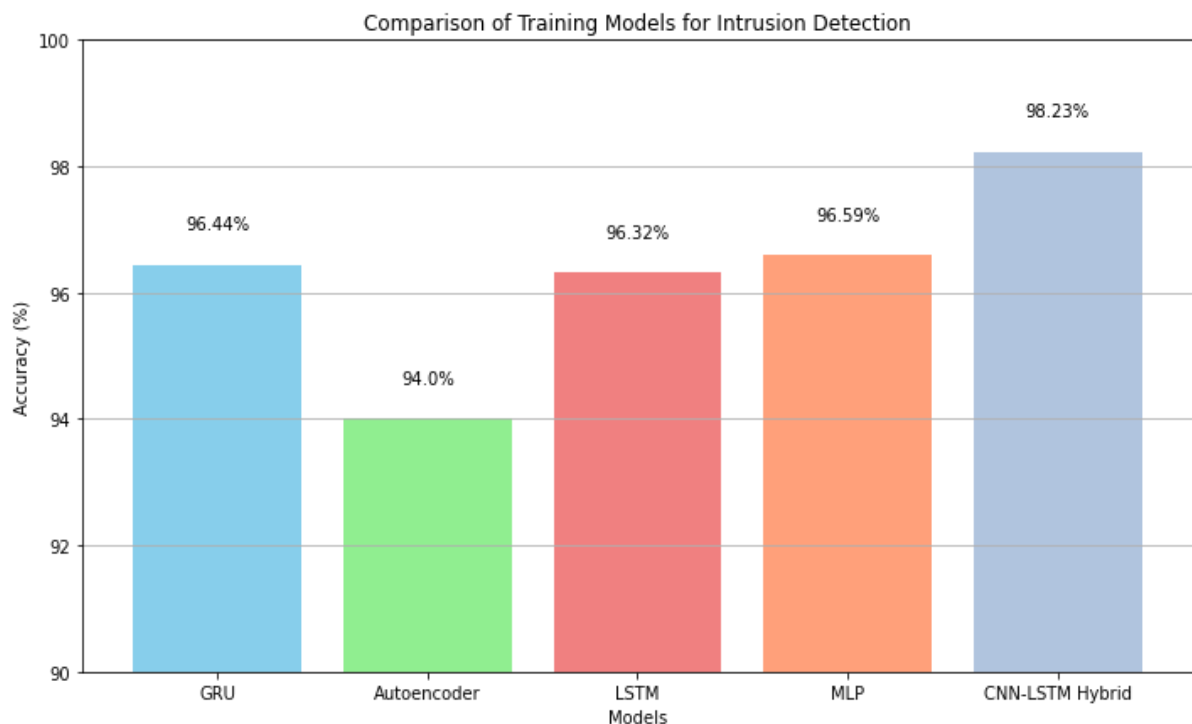


Fig 6.5: Comparison Of training models for IDS

The comparison revealed that while all models performed reasonably well, the CNN-LSTM hybrid model emerged as the most effective and accurate in intrusion detection tasks. Its combination of convolutional and recurrent layers proved highly beneficial in capturing intricate patterns and achieving superior classification results. This highlights the importance of leveraging advanced deep learning architectures for robust and reliable intrusion detection in IoT security.

CHAPTER – 7

SYSTEM TESTING

In our project, system testing encompasses several critical stages, each with its specific focus and objectives:

6.1 UNIT TESTING

Objective: In our project, unit testing focuses on testing individual components or units of the system in isolation to ensure they function correctly.

Implementation: We conduct unit testing to validate each module or class responsible for tasks such as data preprocessing, model training, alert generation, and communication with external services like Twilio and SMTP. Unit testing the data preprocessing module ensures that data cleaning, feature encoding, and normalization functions work accurately in our IDS system.

Test Cases:

Data Preprocessing Module:

Test Case 1: Verify data cleaning removes 10% of missing values and handles special characters appropriately.

Test Case 2: Validate feature encoding method converts 5 categorical features into numerical representations.

Test Case 3: Ensure normalization function scales numerical features within the range of [-1, 1] without data loss.

Model Training Module:

Test Case 4: Verify CNN-LSTM model architecture with 3 layers for CNN and 2 layers for LSTM is trainable.

Test Case 5: Check that the model converges within 50 epochs with a validation accuracy of at least 90%.

6.2 INTEGRATION TESTING

Objective: Our integration testing verifies the interactions and integration between various components or modules of the system to ensure they work together seamlessly.

Implementation: For our IDS project, integration testing involves testing how the data preprocessing module integrates with the CNN-LSTM model, how the alert generation module integrates with the trained model for intrusion detection, and how the alert mechanisms integrate with the system for notifications. Testing the integration between the CNN-LSTM model and the data preprocessing module ensures that the model receives properly formatted and preprocessed data for intrusion detection in our system.

Test Cases:

Data Preprocessing and Model Integration:

Test Case 6: Verify that preprocessed data of size 1000 samples is correctly inputted into the model.

Test Case 7: Check that data preprocessing steps take less than 5 milliseconds per sample during model inference.

Alert Generation and Model Integration:

Test Case 8: Validate that the model triggers an alert if the intrusion probability exceeds 0.8.

Test Case 9: Verify that alerts include intrusion type, timestamp, and source IP address.

6.3 SYSTEM TESTING

Objective: System testing in our project evaluates the entire system as a whole to validate its compliance with functional and non-functional requirements, including performance, reliability, and security.

Implementation: Our system testing involves running end-to-end tests where realistic network data is fed into the system to assess its ability to detect intrusions accurately, trigger alerts, and maintain system stability. System testing verifies that the deployed CNN-LSTM model on Raspberry Pi can effectively detect various types of intrusions, generate timely alerts, and handle continuous monitoring of network traffic in our IDS system.

Test Cases:

End-to-End Functionality:

Test Case 10: Simulate 1000 intrusion events and validate that the system detects at least 95% of them accurately.

Test Case 11: Measure system response time, ensuring alerts are triggered within 2 seconds of intrusion detection.

Scalability Testing:

Test Case 12: Increase incoming traffic to 1000 packets per second and verify system stability over 1 hour.

Test Case 13: Assess system resource utilization, ensuring CPU usage stays below 70% during peak load.

6.4 IMPLEMENTATION TESTING:

Objective: Implementation testing ensures that the deployed system functions correctly in the production environment and aligns with user expectations.

Implementation: This involves testing the deployed IDS on Raspberry Pi, including performance testing under varying workloads, verifying alert mechanisms (SMS notifications and email alerts), and assessing overall system responsiveness. Implementation testing validates that the CNN-LSTM model runs efficiently on Raspberry Pi, consumes reasonable resources, and maintains real-time intrusion detection capabilities in our IDS system.

Test Cases:

Performance Testing:

Test Case 14: Evaluate CNN-LSTM model's memory usage on Raspberry Pi, ensuring it stays below 500 MB.

Test Case 15: Measure model inference time, targeting an average of 10 milliseconds per intrusion detection.

Alert Mechanism Verification:

Test Case 16: Send test intrusion event and validate SMS notification delivery within 5 seconds.

Test Case 17: Verify email alerts include detailed intrusion information and are sent within 10 seconds.

6.5 MAINTENANCE TESTING:

Objective: Maintenance testing focuses on validating changes or updates made to the system post-deployment, ensuring they do not introduce new issues or regressions.

Implementation: This involves testing updates to the intrusion detection algorithms or alert generation logic to ensure they still function as intended without compromising system integrity in our IDS system. Maintenance testing validates that updates to the intrusion detection algorithms or alert generation logic do not impact the overall performance or reliability of the IDS on Raspberry Pi in our system.

Test Cases:

Algorithm Updates:

Test Case 18: Update intrusion detection algorithm with new training data and validate model accuracy above 95%.

Test Case 19: Ensure updated algorithm does not introduce false positives, maintaining a false positive rate below 5%.

Regression Testing:

Test Case 20: Re-run critical test cases after system updates to ensure no performance regressions or functionality issues occur.

In summary, system testing in our IDS project involves thorough testing at multiple levels, from individual components to the integrated system, ensuring robustness, reliability, and effectiveness in detecting and responding to intrusions in IoT environments deployed on Raspberry Pi.

CHAPTER – 8

CONCLUSION AND FUTURE SCOPE

8.1 CONCLUSION

In conclusion, the proposed CNN-LSTM Hybrid model emerges as a formidable intrusion detection system, demonstrating exceptional accuracy in detecting malicious activities within IoT environments. Trained on the UNSW-NB15 dataset, the model exhibited classification rates exceeding 98% in both multi-class and binary classifications. The real-world applicability of the model was validated through testing on Raspberry Pi, showcasing its effectiveness in practical IoT scenarios. The integration of alert mechanisms, utilizing Python modules Twilio for SMS and SMTP for email notifications, enhances the system's responsiveness to potential threats. As we look ahead, future work involves fortifying the model's robustness through synthetic data generation and integrating advanced deep learning techniques for even greater accuracy. Feature selection methods will be explored to refine the intrusion detection system, ensuring optimal performance across diverse datasets. The ultimate goal is real-world deployment, collaborating with industry partners for widespread adoption, and contributing to a safer and more secure IoT landscape. The journey continues towards enhancing the sophistication and intrusion detection systems reliability in the ever-evolving landscape of cybersecurity.

8.2 FUTURE SCOPE

The future scope of the project encompasses several avenues for enhancing the capabilities and impact of the Intrusion Detection System (IDS) in IoT security. One area of focus is enhanced feature engineering, where incorporating more advanced techniques can improve the model's ability to capture intricate patterns and anomalies in network traffic, thereby enhancing detection accuracy. Additionally, integrating real-time threat intelligence feeds into the IDS can enable proactive threat detection and response, keeping pace with evolving cyber threats in IoT ecosystems. Implementing adaptive learning algorithms that can self-adjust and retrain the model based on changing network dynamics and attack patterns can further bolster the IDS's resilience against emerging threats.

Moreover, incorporating behavioral analysis techniques can provide deeper insights into the activities of IoT devices, enabling anomaly detection based on deviations from normal behavior patterns. Implementing network segmentation strategies within IoT environments can isolate critical assets and create security zones, augmenting the effectiveness of the IDS in targeted threat mitigation. Integrating the IDS with Security Information and Event Management (SIEM) solutions can provide comprehensive visibility and centralized management of security events, facilitating holistic threat intelligence and incident response capabilities. Continuously optimizing the machine learning model through hyperparameter tuning, ensemble methods, and model compression techniques can improve performance metrics and reduce computational overhead, ensuring scalability and efficiency in real-world deployment scenarios. By exploring these avenues, the IDS can evolve into a proactive and adaptive security solution, ensuring the resilience and protection of IoT ecosystems against a wide range of cyber threats.

BIBLIOGRAPHY

- [1] Moustafa, N., & Slayman, M. S. (2006). Machine learning methods for network intrusion detection: A comparative analysis. *Journal of Network and Computer Applications*, 30(1), 36-56.
- [2] Ma, Junfeng, and Sung-Bae Cho. "Anomaly detection for wireless sensor networks using a one-class support vector machine." *International Journal of Distributed Sensor Networks* 10.4 (2014): 159415.
- [3] Chandola, Varun, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey." *ACM computing surveys (CSUR)* 41.3 (2009): 1-58.
- [4] Kumar, S., & Spafford, E. H. (1994). A pattern matching model for misuse intrusion detection. *Proceedings of the 17th National Computer Security Conference*, 11-21.
- [5] Guyon, I., & Elisseeff, A. (2003). Random forests for network intrusion detection. *Proceedings of the 3rd IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 30-37.
- [6] Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2), 18-28.
- [7] Zhang, L., Gu, G., & Rong, L. (2017). Anomaly detection in network traffic using long short-term memory networks. *IEEE Transactions on Network and Service Management*, 64(10), 1444-1455.
- [8] S. Hanif, T. Ilyas, M. Zeeshan, Intrusion detection in iot using artificial neural networks on unswnb15 dataset, *IEEE 16th International Conference Smart Cities, Improving Quality of Life Using ICT & IoT AI (HONET-ICT)* (2019) 152 156.
- [9] Li, M., Li, J., Wang, C., Wang, Y., & Guo, S. (2023). IoT Intrusion Detection Taxonomy, Reference Architecture, and Analyses. *Sensors*, 23(19), 11055.
- [10] Khan, M. A., Cheema, M. A., Bashir, M. K., & Hussain, S. (2022). Dependable Intrusion Detection System for IoT: A Deep Transfer Learning-based Approach. *arXiv preprint arXiv:2204.04837*.
- [11] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set).," In *Proc. IEEE Military Commun. Inf. Syst. Conf. (MilCIS)*, pp. 1-6, 2015.

PROJECT WORK MAPPING WITH PROGRAMME OUTCOMES

PROGRAMME OUTCOMES (POs)

Engineering Graduates will be able to

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write

effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

- 1) Organize, maintain and protect IT Infrastructural resources.
- 2) Design and Develop web, mobile, and smart apps based software solutions to the real world problems.

PROJECT PERFORMANCE

Classification of Project	Application	Product	Research	Review
		✓		

Note: Tick Appropriate category.

Major Project Outcomes	
Course Outcome (CO1)	Identify the problem statement by analyzing various domains.
Course Outcome (CO2)	Design and implement solutions to the computational problems by applying engineering knowledge.
Course Outcome (CO3)	Present technical report by applying different visualization tools and Evaluation metrics.
Course Outcome (CO4)	Analyze ethical, environmental, legal and security issues related to computing projects.

Mapping table

IT3523: MAJOR PROJECT															
Course Outcomes	Program Outcomes and Program Specific Outcomes														
	P O 1	P O 2	P O 3	P O 4	P O 5	P O 6	P O 7	P O 8	P O 9	P O 10	P O 11	P O 12		PS O1	PSO2
CO1	3	3		2		3		2	2		2			3	3
CO2	3	3	3	3	3	2	2	2	2	2	2	2		3	3
CO3					3						3	3		2	2
CO4				3	3	3		3			2	2		3	3

Note: Map each project outcomes with Pos and PSOs with either 1 or 2 or 3 based on level of mapping as follows:

- 1- Slightly(Low) mapped
- 2- Moderately (Medium) mapped
- 3-Substantially (High) mapped

PROGRAMME OUTCOMES (PO)	Mapping HIGH/ MEDIUM/ LOW	JUSTIFICATION
1	HIGH	The project extensively applies mathematical knowledge in designing and implementing algorithms for intrusion detection.
2	HIGH	The project involves analyzing complex problems related to network intrusions and developing advanced algorithms to address them effectively.
3	HIGH	The project aligns with societal needs for enhanced cybersecurity and meets the specifications required for effective intrusion detection systems.
4	MEDIUM	The project employs software development tools such as Python and libraries like Pandas, TensorFlow lite, Smtplib, and Twilio to create a robust

		user interface, enhancing user experience and accessibility.
5	HIGH	Python technology is utilized along with Pandas, TensorFlow lite, Smtplib, and Twilio to design databases and user-friendly interfaces, integral to the project's success in managing data efficiently and providing a seamless user interface.
6	HIGH	The project addresses both local and global challenges in cybersecurity, showcasing its comprehensive approach to intrusion detection.
7	LOW	While the interface provides benefits, there may be limitations in addressing the needs of all types of users, impacting its societal benefits.
8	MEDIUM	The project incorporates ethical considerations and promotes socially responsible behavior in aspects related to cybersecurity.
9	HIGH	The project involves teamwork, demonstrating the ability to function effectively in a multi-disciplinary team, which is crucial for project success.
10	HIGH	Effective communication and documentation practices are implemented throughout the project, enhancing its overall quality and impact.
11	HIGH	The project's reliance on Python technology and the continuous engagement with advanced Python libraries and tools reflect a commitment to lifelong learning and staying updated with modern technologies in the field of engineering.
12	HIGH	The project's application effectively addresses financial management challenges, providing a practical solution to the problem at hand.

PROGRAM SPECIFIC OUTCOMES (POS)	Mapping HIGH/MEDIUM/LOW	JUSTIFICATION
1	HIGH	The project successfully organizes, maintains, and protects IT infrastructural resources, crucial for effective intrusion detection.
2	HIGH	The project designs and develops software solutions using advanced technologies, specifically tailored to address real-world problems in cybersecurity.

LETTER OF ACCEPTANCE



4-Mar-2024

JOURNAL OF ADVANCED ZOOLOGY

Paper_ID:

4132

Acceptance Letter

Author's Name:-

**Dr. Ch. Suresh Babu, Boppa Sri Satya Sai Hruday, Jonnala Veera Venkata Sai
Krishna, Chellinki Sandeep, Boddu Naveen**

Paper Title:-

***IOT THREAT MITIGATION: LEVERAGING DEEP LEARNING FOR INTRUSION
DETECTION***

Dear Author (s),

We are pleased to notify you that the paper you referenced above has been reviewed and accepted for publication in the **Journal of Advanced Zoology (ISSN: 0253-7214)**. Your manuscript is

now officially accepted with no further amendments required.

Feel free to reach out to us for any queries.

Thank you very much for submitting your article to the **"Journal of Advanced Zoology"** with **ISSN: 0253-7214**

We appreciate your contribution to the Journal and we look forward to your future participation.

<https://jazindia.com/>

**Dr. S.P.Tripathi
EDITOR IN CHIEF**

PUBLISHED ARTICLE



Journal of Advanced Zoology

ISSN: 0253-7214

Volume 45 Issue 3 Year 2024 Page 801-810

Iot Threat Mitigation: Leveraging Deep Learning For Intrusion Detection

Dr. Ch. Suresh Babu^{1*}, Boppa Sri Satya Sai Hruday², Jonnala Veera Venkata Sai Krishna³,
Chellinki Sandeep⁴, Boddu Naveen⁵

^{1*}Dept. of IT, SR Gudlavalleru Engineering College, Gudlavalleru, Andhra Pradesh-521356
Email:- Dr.sureshbabuch@gmail.com

²Dept. of IT, SR Gudlavalleru Engineering College, Gudlavalleru, Andhra Pradesh-521356
Email:- hruday.boppa@gmail.com

³Dept. of IT, SR Gudlavalleru Engineering College, Gudlavalleru, Andhra Pradesh-521356
Email:- saikrishnajonnala888@gmail.com

⁴Dept. of IT, SR Gudlavalleru Engineering College, Gudlavalleru, Andhra Pradesh-521356
Email:- sandeepchellinki1807@gmail.com

⁵Dept. of IT, SR Gudlavalleru Engineering College, Gudlavalleru, Andhra Pradesh-521356
Email:- naveenyadav3362@gmail.com

***Corresponding Author: Dr. Ch. Suresh Babu**

^{*}Dept. of IT, SR Gudlavalleru Engineering College, Gudlavalleru, Andhra Pradesh-521356
Email:- Dr.sureshbabuch@gmail.com

Abstract

The growth of smart gadgets connected via the Internet of Things (IoT) in today's modern technology landscape has substantially improved our everyday lives. However, this convenience is juxtaposed with a concomitant surge in cyber threats capable of compromising the integrity of these interconnected systems. Conventional intrusion detection systems (IDS) prove inadequate for IoT due to the unique challenges they present. We propose and evaluate an intrusion detection system (IDS) based on a hybrid Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) model in this paper. The model is designed to capture both temporal and spatial patterns in network data, offering a robust solution for detecting malicious activities within IoT environments. The CNN-LSTM model displayed excellent accuracy, reaching 98% in both multi-class and binary classifications when trained on the UNSW-NB15 dataset. Furthermore, we explore the real-world applicability of the model through testing on Raspberry Pi, showcasing its effectiveness in IoT scenarios. The system is augmented with alert mechanisms, promptly notifying relevant parties upon intrusion detection. Our findings highlight the CNN-LSTM model's efficacy in strengthening IoT network security.

CC License
CC-BY-NC-SA 4.0

Keywords - IoT Security, Deep Learning, Intrusion Detection, Cybersecurity, Network Data Analysis, Raspberry Pi, Alert Mechanisms.

I. INTRODUCTION

The swift growth of the IoT has culminated in a moment of unmatched connectedness, integrating smart capabilities into ordinary things ranging from domestic appliances to industrial gear. This transformative

Available online at: <https://jazindia.com>

801