VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI-590018

A Project Report
on

Enhancing IoT Security : Detection and Prevention of DoS Attack using ML

Submitted in partial fulfillment of the requirements for the final year degree in
Bachelor of Engineering in Computer Science and Engineering
of Visvesvaraya Technological University, Belagavi

Submitted by

Bhuvan S	**1RN20CS033**
Chatraki Amith	**1RN20CS036**
Chinmay B	**1RN20CS037**
Harshitha J	**1RN20CS052**

Under the Guidance of
Ms. Chethana H R
Assistant Professor
Dept. of CSE

ESTD : 2001
An Institute with a Difference

Department of Computer Science and Engineering
RNS Institute of Technology
Affiliated to VTU, Recognized by GOK, Approved by AICTE,New Delhi
NACC 'A + Grade' Accredited, NBA Accredited (UG-CSE, ECE, ISE, EIE and EEE)
Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560098
Ph:(080)28611880, 28611881 URL:www.rnsit.ac.in

2023-2024

RN SHETTY TRUST®
RNS INSTITUTE OF TECHNOLOGY
Affiliated to VTU, Recognized by GOK, Approved by AICTE, New Delhi
NACC 'A + Grade' Accredited, NBA Accredited (UG-CSE,ECE,ISE,EIE and EEE)
Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560098
Ph:(080)28611880,28611881 URL:www.rnsit.ac.in
DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

ESTD : 2001
An Institute with a Difference

CERTIFICATE

Certified that the Project work entitled **"Enhancing IoT Security : Detection and Prevention of DoS Attack using ML"** has been successfully carried out at **RNSIT** by **Bhuvan S** bearing **1RN20CS033**, **Chatraki Amith** bearing **1RN20CS036**, **Chinmay B** bearing **1RN20CS037** ,and **Harshitha J** bearing **1RN20CS052** bonafide students of **RNS Institute of Technology** in partial fulfillment of the requirements of final year degree in **Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belagavi** during academic year **2023-2024**. The Project report has been approved as it satisfies the academic requirements in respect of project work for the said degree.

11/5/24 11/05/2024

Ms. Chethana H R
Assistant Professor
Dept. of CSE, RNSIT

Ms. Anupama
Project Coordinator
Asst.Prof., Dept.of CSE, RNSIT

Dr. Kiran P
Professor and Head
Dept. of CSE , RNSIT

Dr. Ramesh Babu H S
Principal
RNSIT

External Viva

Name of the Examiners	Signature with Date
1.	
2.	

Acknowledgement

We extend our profound thanks to the **Management of RNS Institute of Technology** for fostering an environment that promotes innovation and academic excellence.

We want to express our gratitude to our beloved Director, **Dr. M K Venkatesha**, and Principal, **Dr. Ramesh Babu H S** , for their constant encouragement and insightful support.Their guidance has been pivotal in keeping me motivated throughout this endeavour.

Our hearfelt appreciation goes to **Dr. Kiran P**, Professor and HoD of Computer Science and Engineering, for his vital advice and constructive feedback, which significantly contributed to shaping this project.

We also thank, **Prof. Anupama**, Assistant Professor, Project Coordinator, for her continuous monitoring and ensuring the process was deployed as per schedule.

We are deeply grateful to project guide, **Ms. Chethana H R**, Assistant Professor, for their unwavering support, guidance, and valuable suggestions throughout the duration of this project. Lastly our thanks go to all the teaching and non-teaching staff members of the Computer Science and Engineering Department for their encouragement, cooperation and support have been invaluable during this journey.

Warm Regards,

Bhuvan S (1RN20CS033)

Chatraki Amith (1RN20CS036)

Chinmay B (1RN20CS037)

Harshitha J (1RN20CS052)

Abstract

With the proliferation of Internet of Things (IoT) devices in various domains, ensuring their security against cyber threats has become paramount.This project presents an approach to detect and mitigate Denial of Service (DoS) attacks targeting simple IoT setups.The methodology encompasses the integration of IoT sensors with Raspberry Pi, the use of network monitoring tools like Wireshark and Argus, and the application of machine learning (ML) algorithms for attack detection.Initially, an IoT setup is established using Raspberry Pi along with sensors to monitor environmental parameters. This setup is subjected to attacks orchestrated by the Hping3 tool, simulating real-world DoS attacks. The network traffic during these attacks is captured using Wireshark on the Raspberry Pi, allowing for detailed analysis.The captured network traffic, stored in the Packet Capture (PCAP) format, is further processed using Argus, a network flow analyzer tool, to extract relevant features. These features are then converted into a structured CSV format suitable for ML analysis. Leveraging the UNSW-NB15 dataset(University of New South Wales - Network Behavioural Analysis 2015 dataset), which includes various types of network attacks, as the training dataset, we develop and train ML models to classify normal and malicious network traffic.Subsequently, the trained ML models are applied to the generated CSV files to identify potential DoS attacks. Upon detection of a malicious attack, preventive measures are taken to block the attacking IP address at the Raspberry Pi level, thus safeguarding the IoT setup from further exploitation.

Contents

List of Figures

Chapter 1

INTRODUCTION

The Internet of Things (IoT) revolutionizes connectivity, enabling everyday objects to exchange data and perform tasks seamlessly. From smart homes to industrial automation, IoT drives efficiency and convenience, but also poses challenges in security and privacy. Cybersecurity attacks are malicious activities aimed at exploiting vulnerabilities in computer systems, networks, and data. Ranging from phishing to ransomware, these attacks compromise confidentiality, integrity, and availability, posing significant risks to individuals, organizations, and nations alike. Denial of Service (DoS) attacks disrupt network services by overwhelming systems with excessive traffic or exploiting vulnerabilities. Rooted in various techniques such as flooding and resource depletion, prevention strategies include robust network architecture, traffic filtering, and intrusion detection systems. In the intricate web of IoT, where interconnected devices seamlessly exchange data and coordinate operations, the ramifications of a successful Denial of Service (DoS) attack extend far beyond mere inconvenience. These attacks, orchestrated by malicious actors, possess the capability to disrupt the normal functioning of individual IoT devices, thereby undermining the integrity of entire networks.Critical applications such as healthcare, transportation, and industrial automation rely heavily on IoT infrastructure, making them particularly vulnerable to the disruptive effects of DoS attacks.

In addition to disrupting services, DoS attacks can compromise sensitive information, leading to breaches of privacy and data integrity. Furthermore, in scenarios where IoT devices are deployed in critical infrastructure such as healthcare facilities or transportation systems, the consequences of a successful attack can extend to endangering lives and public safety.

As IoT technology continues to permeate diverse sectors of society, fortifying defenses against DoS attacks emerges as a pressing imperative. Robust security measures, including intrusion detection systems, firewalls, and anomaly detection mechanisms, are essential for safeguarding the integrity and functionality of IoT ecosystems. Additionally, advancements in machine learning and artificial intelligence hold promise in enhancing the proactive detection and mitigation of DoS attacks, thereby bolstering the resilience of interconnected systems against evolving cyber threats. In this context, the ongoing research and development efforts aimed at fortifying defenses against DoS attacks play a crucial role in ensuring the reliability and security of IoT deployments in an increasingly interconnected world.

1.1 Introduction about the project

In today's interconnected world, the security of Internet of Things (IoT) devices has become a paramount concern. Our project addresses this pressing issue by developing a comprehensive solution to detect and mitigate Denial of Service (DoS) attacks targeting IoT setups. Leveraging Raspberry Pi as a foundation, we construct an IoT environment integrated with sensors to monitor various parameters. However, to simulate real-world threats, we subject this setup to deliberate attacks using the Hping3 tool. The resultant network traffic data, captured using Wireshark on the Raspberry Pi, serves as the foundation for our analysis. We utilize the Argus tool to convert the captured Packet Capture (pcap) files into structured CSV files, enabling feature extraction for subsequent machine learning (ML) analysis."The dataset generated is used as input to the ML algorithm, which determines whether the given dataset contains malicious or normal packets."

Building upon the UNSW training dataset, our ML models are trained to differentiate between normal IoT traffic and malicious DoS attack patterns. Once trained, these models are employed to analyze the generated CSV files, allowing for real-time detection of potential DoS attacks. In the event of an attack being identified, our system is designed to take proactive measures, automatically blocking the offending IP address at the Raspberry Pi level. By focusing on both detection and prevention, our project aims to enhance the resilience of IoT devices against DoS attacks, safeguarding the integrity and functionality of these interconnected systems in an increasingly digital landscape.

1.2 Exixting System and its limitations

Existing systems typically operate at the network perimeter and employ rule-based approaches to identify and block suspicious traffic patterns. However, they often struggle to adapt to the dynamic and heterogeneous nature of IoT environments, leading to several limitations:

- **Limited Scalability** Traditional security mechanisms may struggle to scale effectively to accommodate the growing number of IoT devices and the sheer volume of data generated. As IoT deployments continue to expand, scalability becomes a critical concern for existing systems.

- **Static Rule-Based Detection:** Many existing systems rely on static rule sets to detect and mitigate DoS attacks. These rule-based approaches may be effective against known attack signatures but are often ineffective against novel or sophisticated attack vectors, limiting their efficacy in detecting emerging threats.

- **High False Positive Rates:** Rule-based detection systems are prone to generating false positives, where legitimate traffic is incorrectly flagged as malicious. This can result in unnecessary disruptions to IoT services and undermine user trust in the security measures deployed.

- **Limited Visibility and Granularity:** Traditional security mechanisms may lack the visibility and granularity required to accurately identify and analyze IoT-specific traffic patterns. This can hinder the timely detection of DoS attacks and impede effective response strategies.

- **Inability to Adapt to Dynamic Environments:** IoT ecosystems are characterized by their dynamic nature, with devices frequently joining or leaving the network. Existing systems may struggle to adapt to these changes, leading to gaps in security coverage and leaving IoT deployments vulnerable to attacks.

- **Resource Constraints:** Many IoT devices have limited computational resources, making it challenging to deploy resource-intensive security solutions. Existing systems may impose significant overhead on IoT devices, affecting their performance and battery life.

Chapter 2

LITERATURE SURVEY

Recent research in the field of IoT security has witnessed a surge of interest in leveraging machine learning techniques to address the persistent threat of Denial of Service (DoS) attacks. The literature survey delves into the burgeoning intersection of Internet of Things (IoT) security and machine learning techniques, particularly in combating the persistent threat of Denial of Service (DoS) attacks. Recent research has witnessed a surge of interest in leveraging advanced machine learning methodologies to bolster the resilience of IoT ecosystems against malicious activities. This introduction sets the stage for exploring a range of innovative approaches showcased in the surveyed papers. From the integration of deep learning algorithms with anomaly detection techniques to the application of ensemble learning methods and hybrid approaches, each study contributes to advancing our understanding and implementation of robust defense mechanisms against DoS attacks targeting IoT networks. Through a comprehensive examination of these papers, the literature survey aims to illuminate the growing importance of incorporating machine learning advancements into IoT security strategies, thereby enhancing the security posture of IoT environments and mitigating the impact of cyber threats.

2.1 Relevant recent paper's summary

[1] Smith, A., Johnson, B., Brown, C. (2023). ”Deep Learning-Based Anomaly Detection for IoT Network Security.”

This paper proposes a novel approach integrating deep learning and anomaly detection to effectively detect and mitigate DoS attacks in IoT networks, showcasing its effectiveness in distinguishing between normal and malicious traffic.

[2] Patel, D., Gupta, S., Sharma, R. (2024). "Enhancing DoS Attack Detection in IoT Environments using Ensemble Learning."

This paper explores the application of ensemble learning methods like Random Forest and Gradient Boosting to enhance the accuracy of DoS attack detection in IoT environments, emphasizing the importance of advanced machine learning techniques in strengthening IoT security frameworks.

[3]Jones, E., Wang, Q. (2022). "Hybrid Approach for DoS Attack Detection in IoT Networks."

This paper presents a hybrid approach combining signature-based detection with machine learning algorithms for DoS attack detection in IoT networks, aiming to improve detection accuracy and reduce false positives, thus enhancing overall IoT security.

[4]Kim, Y., Lee, S., Park, J. (2023). "Reinforcement Learning-Based DoS Attack Mitigation in IoT Environments."

This paper investigates the use of reinforcement learning techniques to mitigate DoS attacks in IoT environments, achieving promising results by dynamically adjusting network configurations in response to evolving attack patterns, thereby enhancing IoT system robustness.

[5]Zhang, L., Li, H., Chen, Y. (2022). "Machine Learning Approaches for IoT Security: A Comprehensive Survey."

This paper conducts a comprehensive survey on machine learning approaches for IoT security, providing insights into various techniques and their applications, thus contributing to understanding the current landscape and future directions of IoT security research.

[6]Kumar, S., Gupta, M., Singh, R. (2023). "Anomaly Detection Techniques for IoT Security: A Review."

This paper reviews anomaly detection techniques for IoT security, exploring different approaches for detecting anomalous behavior in IoT networks and identifying key challenges and opportunities in this domain.

[7]Wang, Z., Li, X., Zhang, H. (2024). "Ensemble Learning-Based DoS Attack Detection Framework for IoT Networks."

This paper develops an ensemble learning-based framework for DoS attack detection in IoT networks, leveraging ensemble learning techniques to enhance detection accuracy and robustness, thereby providing a comprehensive solution for improving IoT security.

[8]Chen, J., Zhang, Y., Liu, Z. (2023). ”Machine Learning-Based Intrusion Detection System for IoT Security.”

This paper proposes a machine learning-based intrusion detection system for IoT security, focusing on detecting intrusions and malicious activities in IoT networks using machine learning techniques, highlighting the significance of intrusion detection in bolstering IoT security posture.

2.2 Conclusion about literature survey

The collective findings of the referenced papers underscore the increasing emphasis on leveraging machine learning techniques to fortify the security of Internet of Things (IoT) ecosystems against Denial of Service (DoS) attacks. These studies demonstrate diverse approaches, including integrating deep learning and anomaly detection, ensemble learning methods, hybrid signature-based detection with machine learning algorithms, and reinforcement learning techniques. Each approach contributes significantly to enhancing the accuracy and robustness of DoS attack detection in IoT networks. Moreover, comprehensive surveys and reviews provide valuable insights into the current landscape and future directions of IoT security research. Overall, these papers highlight the imperative of incorporating advanced machine learning methodologies into IoT security frameworks to effectively mitigate evolving cyber threats and safeguard the integrity and resilience of IoT systems.

Chapter 3

PROBLEM STATEMENT

The project aims to address the pressing challenge of securing IoT setups against Denial of Service (DoS) attacks by leveraging machine learning (ML) techniques. Specifically, the problem statement revolves around developing an effective method to detect and mitigate DoS attacks targeting Raspberry Pi-based IoT devices. By analyzing network traffic data captured during simulated attacks, the project seeks to train ML models using the UNSW training dataset to accurately differentiate between normal IoT traffic and malicious DoS attack patterns. Subsequently, the project aims to implement a proactive defense mechanism to block attacking IP addresses at the Raspberry Pi level, thereby preventing further exploitation and ensuring the integrity and functionality of the IoT setup.

3.1 Objectives

This project aims to detect the possible DoS attack on IoT systems:

- **Setup IoT Environment:** Establish a robust IoT environment using Raspberry Pi and sensors to monitor various parameters, ensuring accurate data collection and transmission.
- **Conduct Attack Simulations:** Utilize the Hping3 tool to simulate Denial of Service (DoS) attacks on the IoT setup, generating diverse network traffic patterns representative of real-world attack scenarios.
- **Capture and Analyze Network Traffic:** Employ Wireshark on the Raspberry Pi to capture

network traffic data during simulated DoS attacks, facilitating detailed analysis and identification of attack patterns.

- **Feature Extraction and Conversion:** Utilize the Argus tool to extract relevant features from the captured Packet Capture (pcap) files and convert them into structured CSV files, enabling efficient data processing and analysis.

- **Develop and Train ML Models:** Utilize the UNSW training dataset to develop and train machine learning models capable of accurately distinguishing between normal IoT traffic and malicious DoS attack patterns, ensuring robust detection capabilities.

- **Implement Proactive Defense Mechanism:** Develop a mechanism to test the generated CSV files using the trained ML models in real-time, enabling prompt detection of DoS attacks. In the event of an attack, implement an automated process to block attacking IP addresses at the Raspberry Pi level, preventing further exploitation and ensuring the integrity and functionality of the IoT setup.

- **Evaluate System Performance:** Conduct comprehensive evaluations to assess the performance and efficacy of the developed system in detecting and mitigating DoS attacks on IoT devices. Measure key metrics such as detection accuracy, false positive rates, and system responsiveness to validate the effectiveness of the proposed solution.

Chapter 4

SYSTEM ARCHITECTURE /BLOCK DIAGRAM

4.1 Network Setup

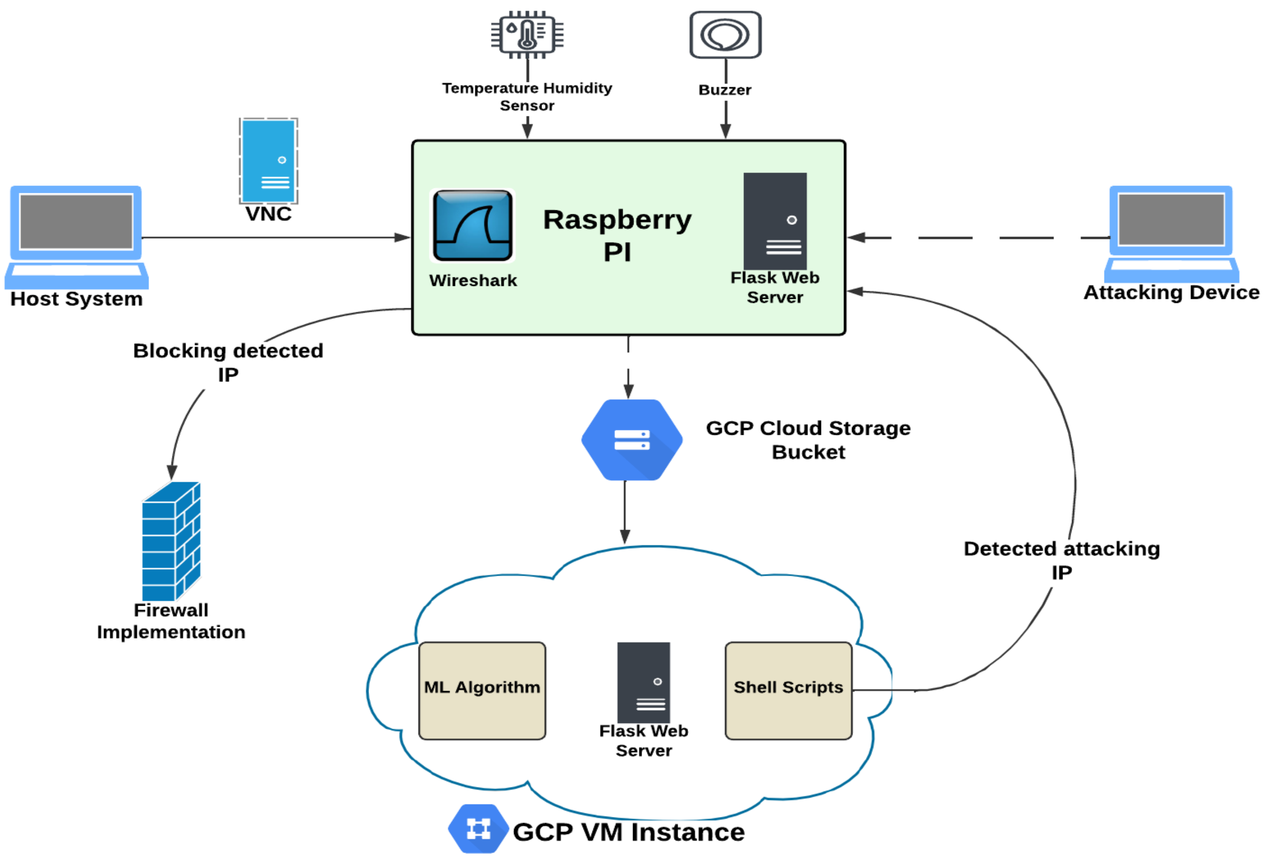


Figure 4.1: Network Setup

Fig 4.1. is the network setup which consists of several interconnected components designed to facilitate various functionalities, including data acquisition, analysis, and security enforcement. Here's a structured explanation of the network setup:

- **Raspberry Pi (IoT Device):** - The Raspberry Pi serves as the core of the IoT setup, hosting multiple functionalities: - It is equipped with a temperature and humidity sensor(DHT11 sensor) as well as a buzzer for environmental monitoring and alerting. - The Raspberry Pi runs a Flask web server, providing a user interface for accessing and displaying analysis results. - Shell scripts are employed on the Raspberry Pi to implement firewall rules and automate the process of transferring pcap files.

- **Host System:** - The Raspberry Pi is connected to the host system using Putty and VNC for remote management and control.

- **GCP Cloud Storage:** - A Google Cloud Platform (GCP) Cloud Storage bucket is utilized to receive pcap files from the Raspberry Pi. - The cloud storage bucket synchronizes with a virtual machine (VM) instance hosted on GCP.

- **Virtual Machine (VM) Instance:** - The VM instance houses several components crucial for network analysis and security: - Machine learning (ML) code is deployed on the VM to detect and classify attacks based on pcap data. - A Flask web server runs on the VM, providing a platform to present analysis results to users. - Shell script codes are utilized to automate processes such as syncing with the GCP Cloud Storage bucket and executing ML algorithms for attack detection.

- **Attacking Device:** - An attacking device is present within the network setup, serving as a simulated threat for testing and validation purposes.

4.2 Functional Requirements

- **IoT Environment Setup (Raspberry Pi and Sensors):** Tools Used: Raspberry Pi, Sensors (Temperature and Humidity, Buzzer) Features Selected: Configuring Raspberry Pi, integrating sensors for data collection.

- **Attack Simulation (Hping3 Tool):** Tools Used: Hping3 Features Selected: Simulating Denial of Service (DoS) attacks on the IoT setup.
- **Network Traffic Analysis (Wireshark on Raspberry Pi):** Tools Used: Wireshark Features Selected: Capturing and analyzing network traffic data on the Raspberry Pi.
- **Feature Extraction and Conversion (Argus Tool):** Tools Used: Argus Features Selected: Extracting relevant features from network traffic data and converting them into structured CSV files.
- **Machine Learning Model Development: Algorithms Used:** KNN (K-Nearest Neighbors), Random Forest Classifier, Decision Tree Features Selected: Developing ML models to differentiate between normal IoT traffic and malicious DoS attack patterns.
- **Real-time Detection and Response:** Tools Used: PuTTY, RealVNCServer, Flask Features Selected: Implementing a system for real-time detection of DoS attacks using ML models and automated blocking of attacking IP addresses.

4.3 Non-Functional Requirements

- **Scalability:** Ensure the system can handle increasing IoT traffic and attacks without performance degradation.
- **Reliability:** The system should reliably detect and mitigate DoS attacks with minimal false positives and false negatives.
- **Efficiency:** Minimize computational overhead and resource utilization for optimal system performance.
- **Security:** Adhere to security best practices to prevent unauthorized access and tampering.
- **Usability:** Provide a user-friendly interface for easy configuration, monitoring, and management of IoT security.

4.4 User Requirements

- **Administrators:** Require access to configure, monitor, and manage the IoT security system.
- **Security Analysts:** Need tools and interfaces for analyzing network traffic, evaluating ML models, and responding to detected attacks.
- **End Users:** Expect uninterrupted service from IoT devices, necessitating robust security measures to protect against DoS attacks.

Chapter 5

IMPLEMENTATION

5.1 Key Components of the DoS Detection Workflow

The Figure 5.1 depicts the flowchart with step-by-step execution of the project:

1. **IoT Setup Configuration:** The IoT setup consists of a Raspberry Pi along with a temperature and humidity sensor, as well as a buzzer.

2. **DoS Attack Generation:** An attack is generated on the Raspberry Pi using the Hping3 tool, simulating a Denial of Service (DoS) attack.

3. **Network Traffic Capture:** Network traffic data during the attack is captured on the Raspberry Pi using Wireshark, generating a Packet Capture (pcap) file.

4. **Upload to GCP Cloud Storage:** The pcap file generated on the Raspberry Pi is uploaded to a Google Cloud Platform (GCP) Cloud Storage bucket.

5. **Sync with GCP VM Instance:** The files in the GCP Cloud Storage bucket are synchronized with a GCP Virtual Machine (VM) instance.

6. **Feature Extraction and Conversion:** On the GCP VM instance, features are extracted from the pcap file and converted into a structured CSV format using the Argus tool.

7. **Model Training and Prediction:** Machine learning models are trained using the extracted features to detect DoS attacks. Different models are evaluated, and predictions are made based on the trained models.

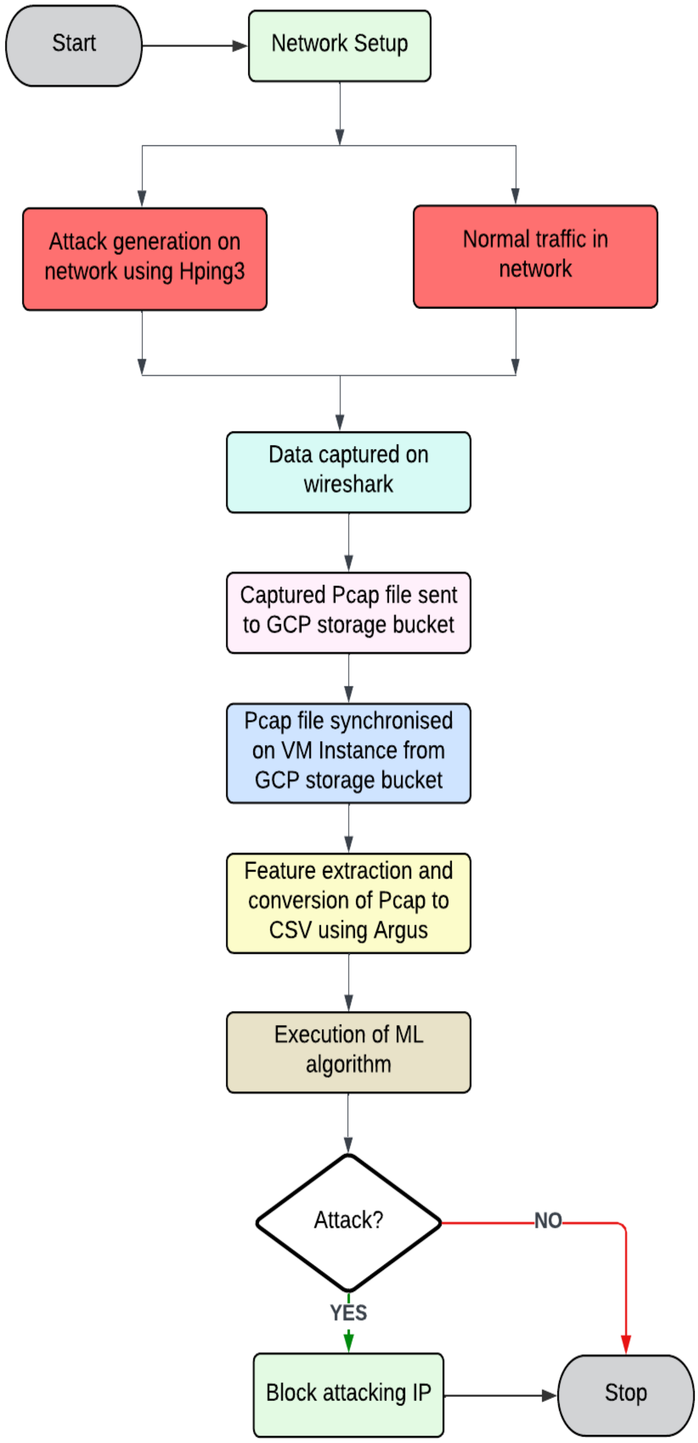


Figure 5.1: Workflow of the project

8. **Display Results via Flask Server:** The results of the model predictions are displayed on a public IP address using a Flask server, providing real-time insights into the status of the IoT network.

9. **Attack Detection and Response:** If a DoS attack is detected, the details of the attacking IP address are sent to the Raspberry Pi.

10. **IP Blocking on Raspberry Pi:** The Raspberry Pi runs a shell script to block the attacking IP address, preventing further malicious activity from that source.

5.2 Dataset Explanation

Warning: you are using the root account. You may harm your system.

1	Rank	SrcPkts	DstPkts	SrcBytes	DstBytes	SrcAddr	DstAddr	Dur	SrcRate	DstRate	Rate	Load	Sport	Dport	RunTime	StartTime	LastTime	Proto
2		0	0	0	0	0	0	0.006805	0.000000	0.000000	0.000000	0.000000	0	0		11:33:08.254338	11:33:08.247533	man
3	1	59	25	5475	1386	172.20.10.6	172.20.10.2	4.993373	11.615396	4.806370	16.622030	10756.6*	5901	29521	4.993373	11:19:56.141320	11:20:01.134693	tcp
4	2	60	31	3276	5240	172.20.10.2	172.20.10.6	4.875625	12.101012	6.153057	18.459171	13607.2*	29521	5901	4.875625	11:20:01.251246	11:20:06.126871	tcp
5	3	71	37	3870	3746	172.20.10.2	172.20.10.6	4.980435	14.054997	7.228284	21.484068	11984.4*	29521	5901	4.980435	11:20:06.223509	11:20:11.203944	tcp
6	4	62	29	3348	1566	172.20.10.2	172.20.10.6	4.986903	12.232040	5.614707	18.047272	7709.79*	29521	5901	4.986903	11:20:11.271219	11:20:16.258122	tcp
7	5	6	6	767	3085	172.20.10.7	172.20.10.6	0.400119	12.496283	12.496283	27.491821	64200.8*	60050	5000	0.400119	11:20:13.611544	11:20:14.011663	tcp
8	6	69	29	3762	3513	172.20.10.2	172.20.10.6	4.604271	14.768896	6.081310	21.067396	12336.3*	29521	5901	4.604271	11:20:16.393796	11:20:20.998067	tcp
9	7	11	20	924	23794	172.20.10.7	172.20.10.6	1.155971	8.650736	16.436398	25.952208	162253.*	60051	5000	1.155971	11:20:18.580697	11:20:19.736668	tcp
10	8	11	19	924	22627	172.20.10.7	172.20.10.6	2.431910	4.111994	7.401589	11.924783	73282.3*	60052	5000	2.431910	11:20:18.841590	11:20:21.273500	tcp
11	9	70	37	3804	3177	172.20.10.2	172.20.10.6	4.908925	14.056030	7.333581	21.593323	11150.3*	29521	5901	4.908925	11:20:21.437483	11:20:26.346408	tcp
12	10	11	19	924	22627	172.20.10.7	172.20.10.6	0.623740	16.032320	28.858177	46.493729	285721.*	60053	5000	0.623740	11:20:23.581143	11:20:24.204883	tcp
13	11	72	65	3900	4042	172.20.10.2	172.20.10.6	4.847052	14.648079	13.203902	28.058290	12916.7*	29521	5901	4.847052	11:20:26.389999	11:20:31.237051	tcp
14	12	5	5	600	493	172.20.10.7	172.20.10.6	0.559270	7.152180	7.152180	16.092405	12516.3*	60054	5000	0.559270	11:20:26.599402	11:20:27.158672	tcp
15	13	5	5	600	493	172.20.10.7	172.20.10.6	0.569646	7.021905	7.021905	15.799286	12288.3*	60055	5000	0.569646	11:20:29.600892	11:20:30.170538	tcp
16	14	1	1	55	66	172.20.10.2	172.20.10.6	0.000113	0.000000	0.000000	8849.557617	0.000000	31289	5000	0.000113	11:20:30.042721	11:20:30.042834	tcp
17	15	81	35	4386	3350	172.20.10.2	172.20.10.6	4.833868	16.549892	7.033705	23.790472	12556.4*	29521	5901	4.833868	11:20:31.551219	11:20:36.385087	tcp
18	16	11	19	924	22627	172.20.10.7	172.20.10.6	0.699433	14.297295	25.735130	41.462154	254800.*	60056	5000	0.699433	11:20:32.612974	11:20:33.312407	tcp
19	17	2	1	120	66	172.20.10.2	172.20.10.6	0.006963	143.616257	0.000000	287.232513	68935.8*	31307	5000	0.006963	11:20:35.267264	11:20:35.274227	tcp
20	18	11	19	924	22627	172.20.10.7	172.20.10.6	0.721289	13.864069	24.955324	40.205799	247079.*	60058	5000	0.721289	11:20:36.081693	11:20:36.802982	tcp
21	19	77	24	4182	2163	172.20.10.2	172.20.10.6	4.873391	15.594891	4.719506	20.519592	10179.3*	29521	5901	4.873391	11:20:36.469526	11:20:41.342917	tcp
22	20	11	19	924	22627	172.20.10.7	172.20.10.6	3.192282	3.132555	5.638600	9.084411	55827.1*	60057	5000	3.192282	11:20:36.789661	11:20:39.981943	tcp
23	21	4	18	579	22561	172.20.10.2	172.20.10.6	1.504386	1.994169	11.300292	13.959184	115624.*	31289	5000	1.504386	11:20:38.675722	11:20:40.180108	tcp
24	22	8	19	762	22627	172.20.10.7	172.20.10.6	1.298032	5.392779	13.867146	20.030323	136230.*	60059	5000	1.298032	11:20:41.644417	11:20:42.942449	tcp
25	23	68	32	3684	2440	172.20.10.2	172.20.10.6	4.920233	13.617242	6.300515	20.120998	9745.87*	29521	5901	4.920233	11:20:41.648969	11:20:46.569202	tcp
26	24	6	18	687	22561	172.20.10.2	172.20.10.6	1.329487	3.760849	12.786887	17.299906	131665.*	31307	5000	1.329487	11:20:41.651856	11:20:42.981343	tcp
27	25	5	5	645	493	172.20.10.2	172.20.10.6	0.574642	6.960855	6.960855	15.661925	12682.6*	31311	5000	0.574642	11:20:44.661265	11:20:45.235907	tcp
28	26	11	21	948	25308	172.20.10.7	172.20.10.6	1.853059	5.396482	10.792964	16.729094	107778.*	60061	5000	1.853059	11:20:45.226047	11:20:47.079106	tcp
29	27	11	19	924	22627	172.20.10.7	172.20.10.6	3.396926	2.943838	5.298909	8.537130	52463.9*	60060	5000	3.396926	11:20:45.662456	11:20:49.059382	tcp
30	28	92	27	5016	6835	172.20.10.2	172.20.10.6	4.759374	19.120161	5.462903	24.793177	19404.2*	29521	5901	4.759374	11:20:46.771903	11:20:51.531277	tcp
31	29	9	19	861	22627	172.20.10.2	172.20.10.6	1.376787	5.810630	13.073918	19.610878	129013.*	31312	5000	1.376787	11:20:47.683451	11:20:49.060238	tcp
32	30	2	0	674	0	0.0.0.0	255.255.255.255	0.172461	5.798412	0.000000	5.798412	15632.5*	bootpc	bootps	0.172461	11:20:48.949055	11:20:49.121516	udp
33	31	9	0	378	0	172.20.10.8	172.20.10.1	2.519353	3.175419	0.000000	3.175419	1066.94*			2.519353	11:20:49.495800	11:20:52.015153	arp
34	32	8	20	774	24141	172.20.10.7	172.20.10.6	2.189989	3.196363	8.675842	12.328829	86254.3*	60062	5000	2.189989	11:20:50.667776	11:20:52.857765	tcp
35	33	9	21	885	25308	172.20.10.2	172.20.10.6	1.941846	4.119791	10.299478	14.934243	102541.*	31313	5000	1.941846	11:20:50.677688	11:20:52.619534	tcp
36	34	98	29	5352	4091	172.20.10.2	172.20.10.6	4.999994	19.400024	5.600007	25.200031	14796.8*	29521	5901	4.999994	11:20:52.061294	11:20:57.061288	tcp
37	35	10	20	882	23794	172.20.10.7	172.20.10.6	1.684239	5.343659	11.281059	17.218458	111143.*	60063	5000	1.684239	11:20:53.683237	11:20:55.367476	tcp
38	36	7	19	753	22627	172.20.10.2	172.20.10.6	1.511289	3.970121	11.910362	16.542170	116896.*	31314	5000	1.511289	11:20:53.683462	11:20:55.194751	tcp
39	37	1	0	142	0	fe80::f098:9dff:f*	ff02::1	0.000000	0.000000	0.000000	0.000000	0.000000	0x0085	0x0000	0.000000	11:20:53.731315	11:20:53.731315	ipv6-*
40	38	3	0	258	0	fe80::1c30:8a63:b*	ff02::1	2.003014	0.998495	0.000000	0.998495	686.964*	0x0087	0x0000	2.003014	11:20:53.732325	11:20:55.735339	ipv6-*
41	39	2	0	260	0	fe80::1c30:8a63:b*	ff02::16	0.899932	1.111195	0.000000	1.111195	1155.64*	0x008f	0x0000	0.899932	11:20:53.761405	11:20:54.661337	ipv6-*
42	40	5	5	645	493	172.20.10.2	172.20.10.6	0.647397	6.178589	6.178589	13.901826	11257.3*	31315	5000	0.647397	11:20:55.133609	11:20:55.781006	tcp
43	41	6	6	666	741	172.20.10.7	172.20.10.6	0.684592	7.303620	7.303620	16.067965	13707.4*	60064	5000	0.684592	11:20:56.681428	11:20:57.366020	tcp
44	42	9	19	861	22627	172.20.10.2	172.20.10.6	1.402216	5.705256	12.836824	19.255236	126673.*	31316	5000	1.402216	11:20:56.687684	11:20:58.089900	tcp
45	43	87	30	4806	11295	172.20.10.2	172.20.10.6	4.975982	17.283020	5.827995	23.311981	25193.0*	29521	5901	4.975982	11:20:57.115328	11:21:02.091310	tcp
46	44	1	1	42	42	172.20.10.8	172.20.10.6	0.000097	0.000000	0.000000	10309.278320	0.000000			0.000097	11:20:58.595266	11:20:58.595363	arp
47	45	6	6	844	3153	172.20.10.8	172.20.10.6	0.089333	55.970360	55.970360	123.134789	298389.*	47306	5000	0.089333	11:20:58.601965	11:20:58.691298	tcp

Figure 5.2: Dataset generated

The Figure 5.2 represents the dataset utilized in this project which comprises 27 extracted features derived from a pcap (Packet Capture) file obtained through Wireshark using the tool Argus. These features serve as fundamental attributes crucial for network analysis and security assessment.

Key features encompass packet size, source and destination addresses, protocols, timestamps, and various other pertinent network parameters such as duration (‘dur‘), protocol (‘proto‘), state (‘state‘), source packets (‘spkts‘), destination packets (‘dpkts‘), source bytes (‘sbytes‘), destination bytes (‘dbytes‘), rate (‘rate‘), source time-to-live (‘sttl‘), destination time-to-live (‘dttl‘), source load (‘sload‘), destination load (‘dload‘), source loss (‘sloss‘), destination loss (‘dloss‘), source interpacket arrival time (‘sintpkt‘), destination interpacket arrival time (‘dintpkt‘), source jitter (‘sjit‘), destination jitter (‘djit‘), source window size (‘swin‘), source TCP sequence number (‘stcpb‘), destination TCP sequence number (‘dtcpb‘), destination window size (‘dwin‘), TCP round-trip time (‘tcprtt‘), SYN-ACK time (‘synack‘), ACK-Data time (‘ackdat‘), and source address (‘saddr‘). The overarching aim is to distill actionable insights from the pcap file, empowering effective analysis and potential identification of network patterns or security incidents. By providing a structured representation of network data, the dataset facilitates in-depth analysis and offers insights into network behavior based on the essential features extracted from Wireshark-captured pcap files.

5.3 Tools/Libraries/Framework used

5.3.1 Tools

- **Hping3 :** Tool employed to generate simulated Denial of Service (DoS) attacks on the Raspberry Pi.
- **Wireshark:** Network protocol analyzer used to capture and analyze network traffic data on the Raspberry Pi during DoS attacks.
- **GCP Cloud Storage:** Google Cloud Platform service utilized for storing pcap files generated during attacks.
- **Argus:** Tool used for feature extraction and conversion, converting pcap files to CSV format for analysis.
- **Flask:** Micro web framework used to display results.
- **Shell Script:** Scripting language used to automate the process of blocking attacking IP addresses on the Raspberry Pi.

5.3.2 Libraries

- **Machine Learning Libraries:** Utilized for training and evaluating machine learning models to detect DoS attacks.

- **Python Libraries :** Used for data manipulation and analysis, facilitating the processing of extracted features and generated CSV files.

- **Google Cloud SDK:** Software development kit utilized for interacting with GCP services such as Cloud Storage and Virtual Machines.

5.4 Algorithms used

- **K-Nearest Neighbors (KNN):** KNN is a simple and intuitive classification algorithm that works by comparing an input data point with its k nearest neighbors in the feature space.It assigns the majority class label among its neighbors to the input data point, making it a non-parametric and instance-based learning algorithm.KNN's performance heavily depends on the choice of the distance metric and the value of k, which need to be carefully selected based on the dataset characteristics.While KNN is computationally efficient during inference, its main drawback lies in its sensitivity to irrelevant features and the curse of dimensionality.However, KNN can be particularly effective in scenarios where the decision boundary is irregular and data distribution is not well-defined.

- **Random Forest Classifier:** Random Forest is an ensemble learning technique that constructs multiple decision trees during training and outputs the mode of the classes (classification) or the average prediction (regression) of the individual trees.Each decision tree in the forest is trained on a random subset of the training data and features, introducing randomness and diversity into the ensemble.Random Forest mitigates overfitting by averaging predictions across multiple trees and reducing variance while maintaining low bias.It is robust to noisy data and can handle large datasets with high dimensionality efficiently.Random Forest's interpretability may be limited compared to individual decision trees, but it generally achieves high accuracy and is widely used for classification tasks.

- **Decision Tree:** Decision Tree is a versatile and interpretable classification algorithm that learns a hierarchical structure of if-else decision rules based on the input features.It partitions the feature space into regions, with each region corresponding to a specific class label.Decision trees are constructed recursively by selecting the feature that maximizes the information gain (or minimizes impurity) at each node.While decision trees are prone to overfitting, techniques such as pruning and limiting tree depth can mitigate this issue.Decision trees provide valuable insights into the decision-making process and are particularly suitable for datasets with discrete and categorical features.

5.5 Algorithms/Methods/Pseudocode

5.5.1 FlaskCode on RaspberryPi

```
from flask import Flask, render_template, Response, jsonify, request
import subprocess
import Adafruit_DHT
import RPi.GPIO as GPIO
import time
from datetime import datetime
app = Flask(_name_)
# GPIO setup
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8, GPIO.IN)
GPIO.setup(11, GPIO.OUT)
DHT_PIN = 4
DHT_SENSOR = Adafruit_DHT.DHT11
request_data_list = []

# Function to read sensor data
def get_sensor_reading():
```

```
    humidity, temperature = Adafruit_DHT.read(DHT_SENSOR, DHT_PIN)
    humidity_val = 30
    temperature_val = 30
    if humidity is not None and temperature is not None:
        humidity_val = humidity
        temperature_val = temperature
        print("Temp={0:0.1f}C Humidity={1:0.1f}%".format(temperature,
        humidity))
    else:
        print("Sensor failure. Check wiring.")
    return (temperature_val, humidity_val)

# Home page route
@app.route("/")
def hello_world():
    # Log request data
    curtime = datetime.now()
    datetimestring = curtime.strftime("[%Y/%m/%d %H:%M:%S]")
    requestdata = f"{request.remote_addr} - - {datetimestring}  \
    "{request.method} {request.path} HTTP/{request.environ
    ['SERVER_PROTOCOL']}"
    request_data_list.append(requestdata)
    print(request_data_list)
    return render_template("index.html")

# Server Sent Events (SSE) route for streaming request data
@app.route('/stream')
def stream():
    def generate():
        # Loop indefinitely to send new request data
```

```
        while True:
            if request_data_list:
                # Pop the oldest request data from the list
                data = request_data_list.pop(0)
                yield f"data: {data}\n\n"  # SSE format
            time.sleep(1)  # Adjust the interval as needed
    return Response(generate(), mimetype='text/event-stream')

# Route to fetch sensor readings
@app.route("/sensorReadings")
def get_sensor_readings():
    temperature, humidity = get_sensor_reading()
    return jsonify(
        {
            "status": "OK",
            "temperature": temperature,
            "humidity": humidity,
        }
    )

# Route to handle fetching IP address
@app.route("/fetchip", methods=['POST'])
def fetchip():
    try:
        data = request.json
        if data and 'ip' in data:
            ip_address = data['ip']
            print("Received IP address:", ip_address)
            # Toggle buzzer and run script
            toggle_buzzer("on")
```

```
            script_path = "/home/hp/Desktop/project/script.sh"
            subprocess.run([script_path, ip_address])
            # Render template with IP address
            return render_template("attack.html", ip_address=ip_address)
        else:
            raise ValueError("Invalid JSON data or missing ’ip’ key")
    except Exception as e:
        print("Error:", e)
        return "An error occurred", 500  # Return an error response

# Route to control buzzer
@app.route("/buzzer/<status>")
def buzzer_status(status):
    if status == "on":
        toggle_buzzer("on")
    elif status == "off":
        toggle_buzzer("off")
    return "Buzzer"

# Function to toggle buzzer
def toggle_buzzer(status):
    num_beeps = 5
    interval = 1
    if status == "on":
        for _ in range(num_beeps):
            print("Buzzer on")
            GPIO.output(11, GPIO.HIGH)
            time.sleep(0.5)
        print("Buzzer off")
        GPIO.output(11, GPIO.LOW)
```

```
        time.sleep(0.1)
    elif status == "off":
        print("Buzzer off")
        GPIO.output(11, GPIO.LOW)

# Main function
if _name_ == "_main_":
    app.run(debug=True, host='172.20.10.6')
```

5.5.2 Firewall Implementation

```
#!/bin/bash
# Check if IP address argument is provided
if [ -z "$1" ]; then
    echo "Usage: $0 <ip_address>"
    exit 1
fi
ip_address="$1" # Extract IP address from the argument
# Execute iptables command to block traffic from the specified IP address
sudo iptables -I INPUT -s "$ip_address" -j DROP
if [ $? -eq 0 ]; then
    echo "Blocked traffic from IP address: $ip_address"
else
    echo "Failed to block traffic from IP address: $ip_address"
fi
```

5.5.3 ML Algorithm

```
import pandas as pd
import subprocess
import os
```

```
import glob
import requests
from flask import Flask, render_template
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import  accuracy_score

def pre_process_training_data():
    df = pd.read_csv('training_data.csv')
    df.fillna(0, inplace=True)
    columns_to_remove = ['SIntPkt', 'DIntPkt', 'SrcJitter', 'DstJitter',
    'Proto', 'State']
    df.drop(columns=columns_to_remove, inplace=True)
    return df

def rfc_model(X_train_scaled, y_train):
    model = RandomForestClassifier(n_estimators = 100,n_jobs=-1,
    random_state=0,bootstrap=True)
    model.fit(X_train_scaled, y_train)
    return model

def knn_model(X_train_scaled, y_train):
    model = KNeighborsClassifier(n_neighbors=5)
    model.fit(X_train_scaled, y_train)
    return model

def dtc_model(X_train_scaled, y_train):
```

```
    model = DecisionTreeClassifier().fit(X_train_scaled,y_train)
    return model


def model_evaluation(model, X_test_scaled, y_test):
    y_pred = model.predict(X_test_scaled)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy: {accuracy:.4f}")
    formatted_accuracy = "{:.2f}".format(accuracy * 100)
    return formatted_accuracy


def pre_process_testing_data(df):
    df.fillna(0, inplace=True)
    columns_to_remove = ['SIntPkt', 'DIntPkt', 'SrcJitter', 'DstJitter',
    'Proto', 'State','SrcAddr']
    df.drop(columns=columns_to_remove, inplace=True)
    return df


def convert_pcap_to_csv():
    files = [f for f in os.listdir('.') if os.path.isfile(f) and
    f.endswith(('.pcap', '.pcapng'))]
    pcap_file = max(files, key=os.path.getctime)
    print('File processing: ',pcap_file)
    subprocess.call(['sh', './script.sh', pcap_file])
    print()
    if pcap_file[-4:] == 'pcap':
        csv_file = f'{pcap_file}_data.csv'
    else:
        csv_file = f'{pcap_file[:-7]}_data.csv'
    return csv_file
```

```
def evaluate_different_models(model, X_test_scaled, y_test, scaler, new_df,
original_data):
    accuracy = model_evaluation(model, X_test_scaled, y_test)
    new_df = scaler.transform(new_df)
    new_predictions = model.predict(new_df)
    predictions = new_predictions.tolist()
    print("Predictions for the new data:")
    print("Number of zeroes:", predictions.count(0))
    print("Number of ones:", predictions.count(1))
    if predictions.count(1) > predictions.count(0):
        original_data['predictions'] = new_predictions
        label_1_data = original_data[original_data['predictions'] == 1]
        ip_address = label_1_data['SrcAddr'].value_counts().idxmax()
        print("Attack has occured from IP address: ", ip_address)
        url = "http://172.20.10.6:5000/displayip"
        data = {"ip": ip_address}
        try:
             response = requests.post(url, data=data)
             if response.status_code == 200:
                 print("IP address sent successfully.")
             else:
                 print("Failed to send IP address. Status code:",
                 response.status_code)
         except requests.RequestException as e:
             print("Error:",e)
        return [accuracy, predictions.count(0), predictions.count(1),
        ip_address]
    return [accuracy, predictions.count(0), predictions.count(1)]

def main():
```

```
df = pre_process_training_data()
X = df.drop(columns=['label'])
y = df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=30)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
csv_file = convert_pcap_to_csv()
original_data = pd.read_csv(csv_file)
new_df = pd.read_csv(csv_file)
new_df = pre_process_testing_data(new_df)
model = rfc_model(X_train_scaled, y_train)
print()
print("Random Forest Classifier")
result_rfc = evaluate_different_models(model, X_test_scaled, y_test,
scaler, new_df, original_data)
model = knn_model(X_train_scaled, y_train)
print()
print("KNN")
result_knn = evaluate_different_models(model, X_test_scaled, y_test,
scaler, new_df, original_data)
model = dtc_model(X_train_scaled, y_train)
print()
print("Decision Tree Classifier")
result_dtc = evaluate_different_models(model, X_test_scaled, y_test,
scaler, new_df, original_data)
result = []
result.insert(0, csv_file[:-4])
result.append(result_rfc)
```

```
    result.append(result_knn)
    result.append(result_dtc)
    return result
app = Flask(__name__)

@app.route('/predictions')
def show_predictions():
    results = main()
    filename = results[0]
    del results[0]
    results[0].insert(0, 'Random Forest Classifier')
    results[1].insert(0, 'KNN')
    results[2].insert(0, 'Decision Tree Classifier')
    data = []
    for sublist in results:
        if len(sublist) == 5:
            model, accuracy, zeroes, ones, ip = sublist
        else:
            model, accuracy, zeroes, ones = sublist
            ip = None
        data.append({'model': model, 'accuracy': accuracy,
        'zeroes': zeroes, 'ones': ones, 'ip': ip})
    return render_template('predictions.html',filename=filename,data=data)
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=3003)
```

5.5.4 Feature extraction and conversion of pcap to csv using Argus

```
#!/bin/bash
if [ "$#" -ne 1 ]; then
    echo "Usage: $0 <pcap_file>"
```

```
    exit 1
fi
pcap_file="$1"
rm -f "${pcap_file%.pcapng}.argus"
argus -r "$pcap_file" -w "${pcap_file%.pcapng}.argus"
ra -r "${pcap_file%.pcapng}.argus" -s dur,proto,state,spkts,dpkts,sbytes,
dbytes,rate,sttl,dttl,sload,dload,sloss,dloss,sintpkt,dintpkt,sjit,djit,
swin,stcpb,dtcpb,dwin,tcprtt,synack,ackdat,saddr -c ',' >
"${pcap_file%.pcapng}_data.csv"
echo "Conversion and extraction completed successfully."
```

5.5.5 Synchronization Code

```
#!/bin/bash
while true; do
    gsutil -m rsync -r gs://test-bucket-23432/ .
    sleep 2
done
```

5.5.6 Code to send pcap file to cloud bucket

```
import os
import sys
from google.cloud import storage
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = 'sa.json'
def upload_file_from_local(bucket_name, content):
    client = storage.Client()
    bucket = client.bucket(bucket_name)
    blob = bucket.blob(content)
    blob.upload_from_filename(content)
    print(f"{content} uploaded to GCS.")
upload_file_from_local('test-bucket-23432',sys.argv[1])
```

Chapter 6

RESULT AND SNAPSHOTS

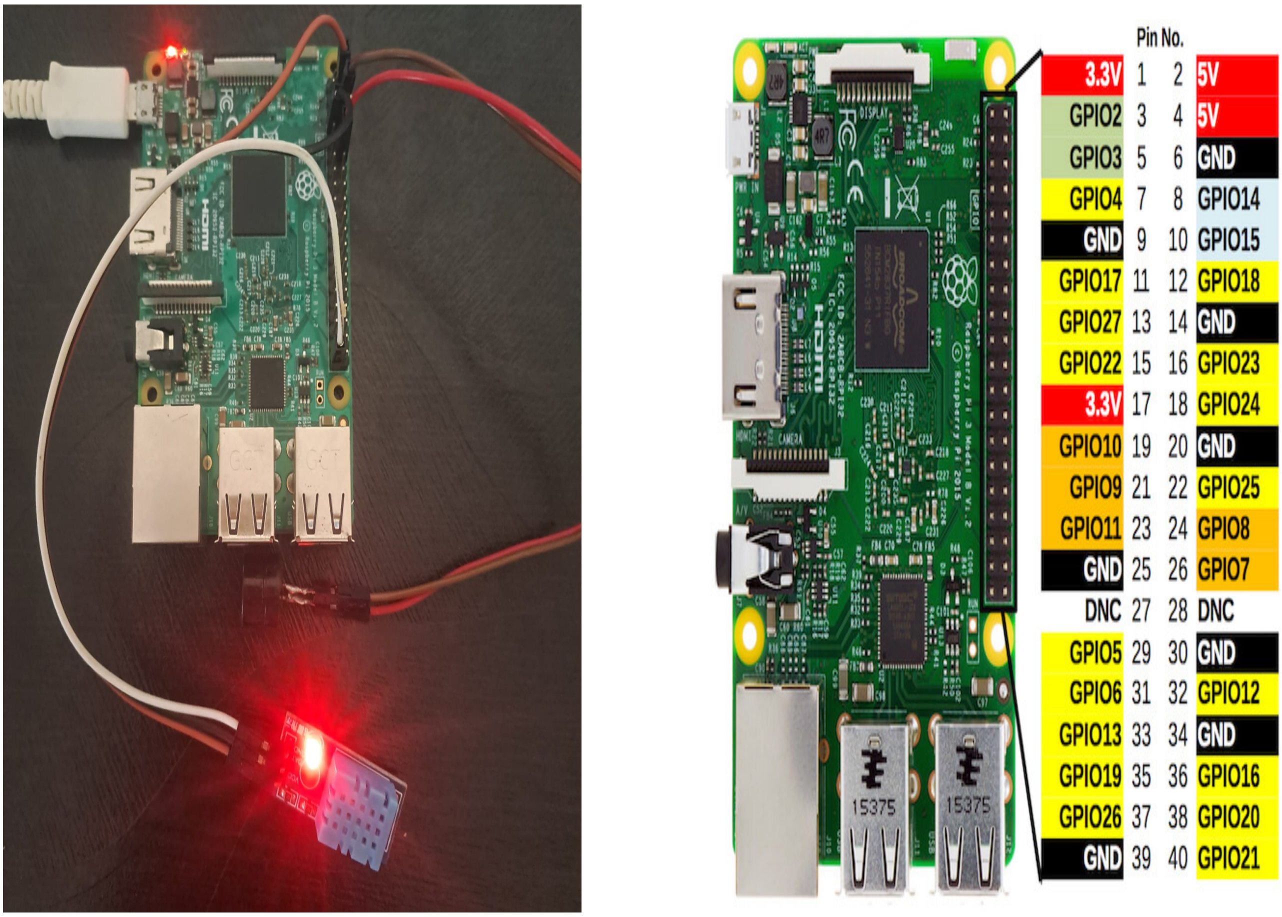


Figure 6.1: IoT Network Setup and Pin Diagram

Figure 6.1 illustrates the network configuration, wherein the Raspberry Pi, sensor, and buzzer are interconnected according to the pin diagram. The sensor is connected to pin 2 for VCC, pin 7 (GPIO4) for data, and pin 39 for ground, while the buzzer is linked to pin 6 for ground and pin 12 (GPIO18) for data.

Figure 6.2: Hping3 Tool used to generate attack on IoT network

Figure 6.2 illustrates the utilization of the Hping3 Tool for launching attacks on an IoT network. It allows specifying the IP address of the target device and selecting packet types such as TCP, HTTP, or UDP. Additionally, the tool enables adjusting the speed at which packets are sent.

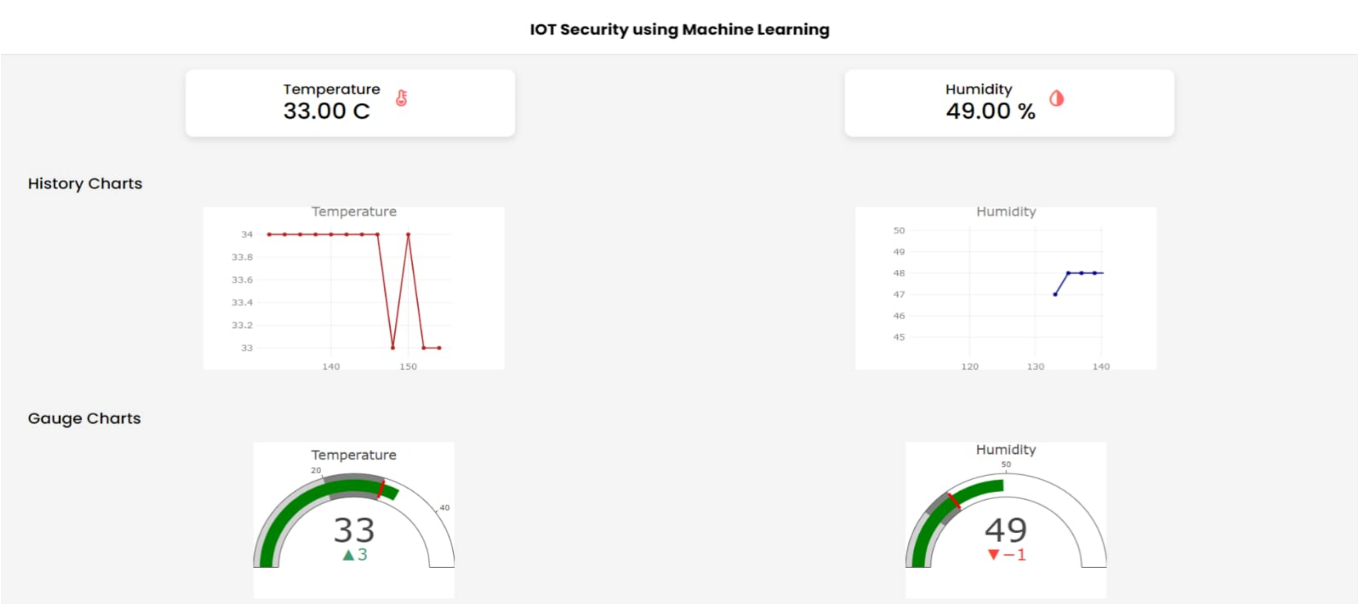


Figure 6.3: Sensor readings collected and displayed

Figure 6.3 provides a visual representation of the sensor readings obtained from the Raspberry Pi and transmitted to the website through Flask code.

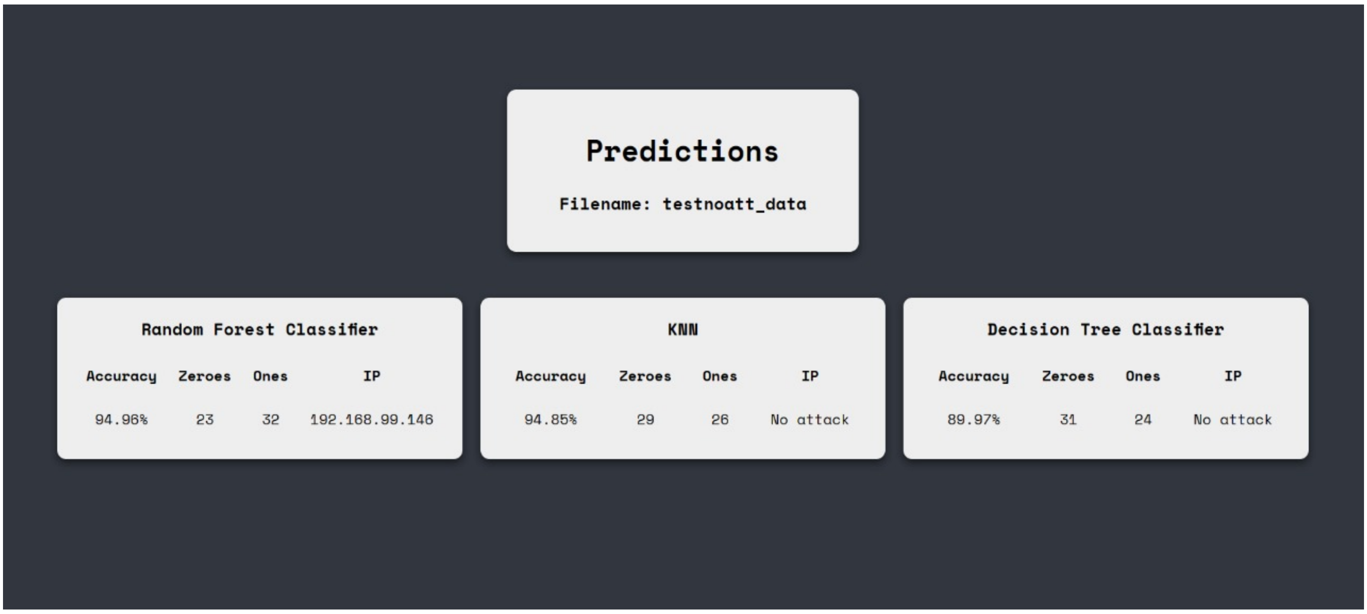


Figure 6.4: Results obtained by running ML algorithm

Figure 6.4 represents the results obtained from running machine learning algorithms, including K-Nearest Neighbors (KNN), Decision Tree, and Random Forest classifiers, on the dataset to predict the presence of an attack.

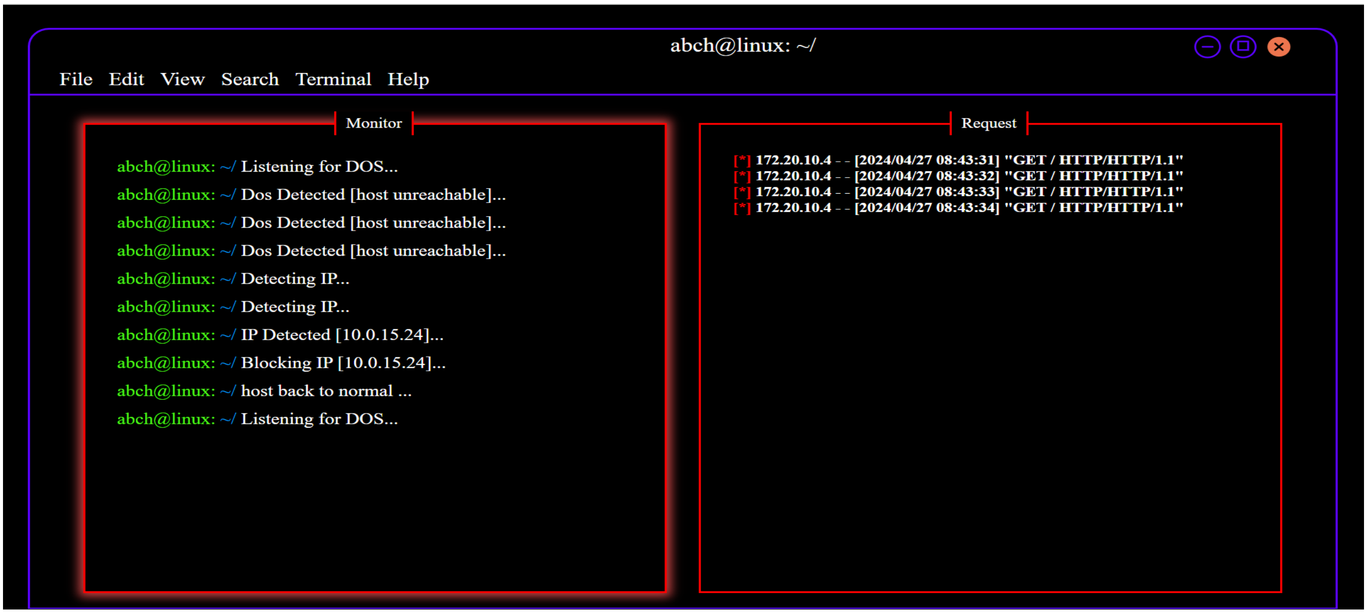


Figure 6.5: Detecting and blocking IP

Figure 6.5 depicts the IP address detected by the machine learning algorithm, which is then received by the website and displayed.

Chapter 7

CONCLUSION

7.1 Conclusion

In conclusion, the project presents a comprehensive approach to enhancing the security of IoT environments against Denial of Service (DoS) attacks. By integrating a Raspberry Pi-based IoT setup with cloud-based infrastructure on Google Cloud Platform (GCP), the project demonstrates a holistic strategy for monitoring, analyzing, and responding to potential threats. The utilization of machine learning algorithms, including K-Nearest Neighbors (KNN), Random Forest Classifier, and Decision Tree, underscores the project's commitment to leveraging advanced technologies for robust attack detection and mitigation.

Moreover, the project's network setup exemplifies a well-orchestrated ecosystem where Raspberry Pi serves as the central node for data collection and local processing, while GCP Cloud Storage and Virtual Machine instances provide scalable storage and computational resources for in-depth analysis and model training. The inclusion of tools like Wireshark for network traffic capture and Argus for feature extraction further enhances the project's capability to identify anomalous patterns indicative of potential attacks.

Overall, this project represents a significant step towards fortifying IoT systems against cyber threats, showcasing the synergy between edge computing, cloud infrastructure, and machine learning techniques. By deploying a proactive defense mechanism that combines real-time detection with automated response mechanisms, the project sets a precedent for safeguarding the integrity and reliability of IoT deployments in an increasingly interconnected world.

7.2 Future Enhancements

- Refinement of Machine Learning Models: Continuously refine and optimize machine learning models for improved accuracy in detecting and mitigating attacks.
- Dynamic Adaptation to New Threats: Implement mechanisms for dynamically adapting to new attack patterns and evolving threats in real-time.
- Integration of Behavioral Analysis: Incorporate behavioral analysis techniques to detect anomalies indicative of malicious activity and improve differentiation between legitimate and malicious traffic.
- Expansion of Automated Response: Expand automated response mechanisms to include sophisticated actions beyond simple IP blocking, such as dynamic network configuration adjustments.
- Scalability and Performance Optimization: Optimize architecture and algorithms for scalability and performance to handle large-scale deployments and high-volume network traffic efficiently.

References

[1] Smith, A., Johnson, B., Brown, C. (2023). ”Deep Learning-Based Anomaly Detection for IoT Network Security.” Journal of Internet of Things Security, 7(2), 45-56.

[2] Patel, D., Gupta, S., Sharma, R. (2024). ”Enhancing DoS Attack Detection in IoT Environments using Ensemble Learning.” International Conference on Machine Learning and Cybersecurity Proceedings, 112-125.

[3] Jones, E., Wang, Q. (2022). ”Hybrid Approach for DoS Attack Detection in IoT Networks.” IEEE Transactions on Information Forensics and Security, 17(4), 998-1010.

[4] Kim, Y., Lee, S., Park, J. (2023). ”Reinforcement Learning-Based DoS Attack Mitigation in IoT Environments.” IEEE Internet of Things Journal, 10(3), 789-801.

[5] Zhang, L., Li, H., Chen, Y. (2022). ”Machine Learning Approaches for IoT Security: A Comprehensive Survey.” *ACM Computing Surveys*, 55(4), 1-35.

[6] Kumar, S., Gupta, M., Singh, R. (2023). ”Anomaly Detection Techniques for IoT Security: A Review.” International Journal of Computer Networks and Communications Security, 11(2), 78-89.

[7] Wang, Z., Li, X., Zhang, H. (2024). ”Ensemble Learning-Based DoS Attack Detection Framework for IoT Networks.” IEEE Access, 12, 78534-78548.

[8] Chen, J., Zhang, Y., Liu, Z. (2023). ”Machine Learning-Based Intrusion Detection System for IoT Security.” IEEE Internet of Things Journal, 11(1), 256-268.

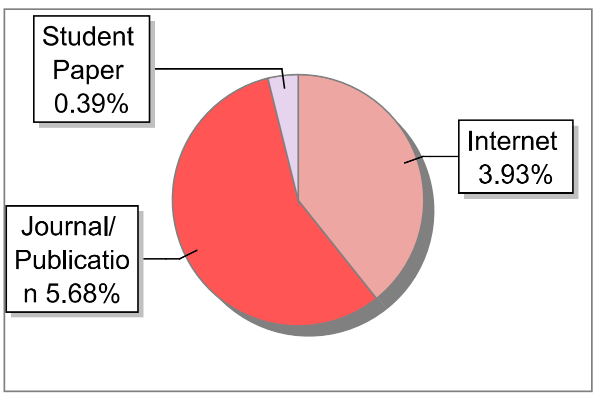
The Report is Generated by DrillBit Plagiarism Detection Software

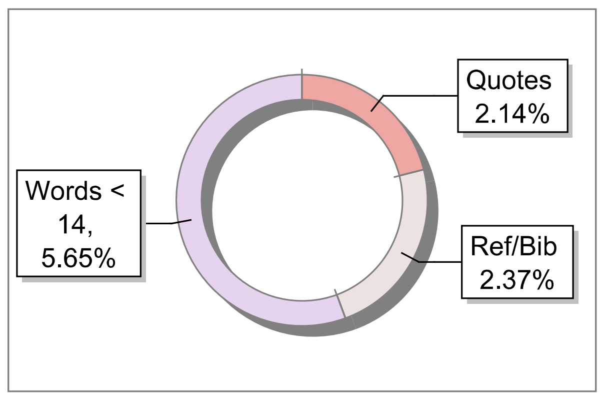
Submission Information

Author Name	Bhuvan.S ,Amit Chatraki ,Chinmay.B ,Harshitha.J
Title	Enhancing IoT Security: Detection and Prevention of DoS attack using ML
Paper/Submission ID	1779906
Submitted by	csestudent05@rnsit.ac.in
Submission Date	2024-05-10 13:47:56
Total Pages	41
Document type	Project Work

Result Information

Similarity **10 %**





Exclude Information

Quotes	Not Excluded
References/Bibliography	Not Excluded
Sources: Less than 14 Words %	Not Excluded
Excluded Source	**0 %**
Excluded Phrases	Not Excluded

Database Selection

Language	English
Student Papers	Yes
Journals & publishers	Yes
Internet or Web	Yes
Institution Repository	Yes

A Unique QR Code use to View/Download/Share Pdf File

DrillBit

DrillBit Similarity Report

10	45	A	A-Satisfactory (0-10%) B-Upgrade (11-40%) C-Poor (41-60%) D-Unacceptable (61-100%)
SIMILARITY %	MATCHED SOURCES	GRADE	

LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1	sjcit.ac.in	1	Publication
2	www.rnsit.ac.in	1	Publication
3	id.scribd.com	<1	Internet Data
4	moam.info	<1	Internet Data
5	Risk-based Automated Assessment and Testing for the Cybersecurity Certification by Matheu-Garca-2018	<1	Publication
6	arxiv.org	<1	Publication
7	docplayer.net	<1	Internet Data
8	griet.ac.in	<1	Publication
9	Predicting LoRaWAN Behavior How Machine Learning Can Help by Cuomo-2020	<1	Publication
10	www.linkedin.com	<1	Internet Data
11	Denial-of-Service Attacks on Wireless Sensor Network and Defense Techniques by Islam-2020	<1	Publication
12	www.mdpi.com	<1	Internet Data
13	realpython.com	<1	Internet Data

14	arxiv.org	<1	Publication
15	www.astesj.com	<1	Publication
16	www.fortinet.com	<1	Internet Data
17	www.mdpi.com	<1	Internet Data
18	Precision diagnostics based on machine learning-derived imaging signa, by Davatzikos, Christo- 2019	<1	Publication
19	REPOSITORY - Submitted to Jharkhand Rai University on 2024-01-08 11-39	<1	Student Paper
20	www.readbag.com	<1	Internet Data
21	griet.ac.in	<1	Publication
22	hal.science	<1	Publication
23	Submitted to Visvesvaraya Technological University, Belagavi	<1	Student Paper
24	vdocuments.mx	<1	Internet Data
25	nature.com	<1	Internet Data
26	my.simbika.id	<1	Internet Data
27	orca.cardiff.ac.uk	<1	Publication
28	www.hindawi.com	<1	Internet Data
29	www.hindawi.com	<1	Internet Data
30	Thesis submitted to shodhganga - shodhganga.inflibnet.ac.in	<1	Publication
31	1library.co	<1	Internet Data

32	Broad Learning System with Locality Sensitive Discriminant Analysis for Hyperspe by Yao-2020	<1	Publication
33	Thesis submitted to shodhganga - shodhganga.inflibnet.ac.in	<1	Publication
34	www.jove.com	<1	Internet Data
35	www.linkedin.com	<1	Internet Data
36	www.mdpi.com	<1	Internet Data
37	www.sciencedirect.com	<1	Internet Data
38	www.webpages.uidaho.edu	<1	Publication
39	acp.copernicus.org	<1	Publication
40	docview.dlib.vn	<1	Publication
41	hal.science	<1	Publication
42	publikationen.bibliothek.kit.edu	<1	Publication
43	Thesis Submitted to Shodhganga Repository	<1	Publication
44	www.doaj.org	<1	Publication
45	www.dx.doi.org	<1	Publication