National Forensic Sciences University

Knowledge | Wisdom | Fulfilment

An Institution of National Importance
(Ministry of Home Affairs, Government of India)

School Of Cyber Security And Digital Forensic

Raspberry Pi Powered Unified AIoT System For Storage, Security & Automation

A Project Report

Submitted in Partial Fulfillment of the Requirement
For the Degree of

Master of Technology
In
Artificial Intelligence & Data Science
(Specialization in Cyber Security)

By

Pratham Pravin Badge
Enrollment No.: 2401003007003

Session Year: 2024 -26
Year/Semester: 1st /2nd

Subject Code: CTMTAIDS SII P4
Subject Name: Intelligent Systems and Security

Guided By: Ms. Drashti Garadharia

ABSTRACT

This project introduces a compact and efficient AIoT solution leveraging the Raspberry Pi platform, unifying a range of critical functionalities—namely local NAS and media server storage, AI-enhanced intrusion detection and prevention with firewall integration, lightweight virtualization, an intelligent AI-powered chatbot, and smart home automation features. Designed with low power consumption and cost-effectiveness in mind, the system operates entirely at the network edge, ensuring reduced latency, enhanced data privacy, and responsive real-time decision-making.

The project explores modern cybersecurity applications by integrating AI-driven network monitoring and anomaly detection, supporting red-team evaluation and modular expansion. The initial development phase was carried out through QEMU virtualization for safe simulation, validation, and rapid prototyping. Final deployment is intended on Raspberry Pi 5, delivering a pocket-sized yet powerful, scalable, and user-friendly platform suitable for smart homes, research labs, and secure environments. Additionally, the system supports cloud AI interoperability and aims to democratize personal cybersecurity and automation through open-source innovation.

LIST OF TABLE

Table No.	Table Title	Page No.
3.2	Tools and Libraries Used	10
3.3	Timeline and Phases	11
3.4	Deployment and Simulation Environment Setup	12
4.1	Core Modules and Functions Implemented	14

LIST OF FIGURES

Figure No.	Figure Title	Page No.
2.1.1	Unified AIoT System Architecture	7
2.2.2	Modular Data Flow Diagram	8
4.3.1	Raspberry Pi 5	15
4.3.2	X1004 PCIe to Dual 2280 NVMe SSD Peripheral Board	16
4.3.3	Raspberry Pi 5 with X1004 PCIe to Dual 2280 NVMe SSD	16
4.3.4	Raspberry Pi 5 PCIe Metal case Compatible with X1004	17
4.4	Software Stack	18
4.5	Flowchart of AI-based anomaly detection	19

Contents			
			Page No.
Acknowledgement			1
Abstract			2
List Of Figures			3
List Of Table			3
Chapter 1		**INTRODUCTION**	**5**
	1.1	Introduction	5
	1.2	Objectives of the Project	5
	1.3	Project focus	5
	1.4	Organization of Report	6
Chapter 2		**BACKGROUND MATERIAL**	**7**
	2.1	Conceptual Overview	7
	2.2	Technologies Involved	8
	2.3	Related Work and Studies	9
	2.4	Motivation and Innovation	9
Chapter 3		**METHODOLOGY**	**10**
	3.1	Detailed methodology	10
	3.2	Tools and Libraries Used	10
	3.3	Timeline and Phases	11
	3.4	Deployment and Simulation Environment Setup	12
Chapter 4		**IMPLEMENTATION**	**13**
	4.1	Core Modules and Functions Implementation	13
	4.2	Prototype	14
	4.3	Hardware Setup	14
	4.4	Software Architecture	17
	4.5	Integration of AI with NAS	17
	4.6	Security Measures Implementation	18
Chapter 5		**CONCLUSION & FUTURE SCOPE**	**19**
	5.1	Conclusions	19
	5.2	Future Scope of Work	20
Chapter 6		**REFERENCES**	**21**
	6.1	Research Paper	21
	6.2	Reference Guide on Tools and Technologies	21

Chapter 1

INTRODUCTION

1.1 Introduction

The convergence of Artificial Intelligence (AI) and the Internet of Things (IoT) has opened new avenues in the domain of smart systems, particularly in the areas of security, automation, and data storage. This project introduces a Raspberry Pi-based Unified AIoT System, which serves as an intelligent, low-power, edge-computing solution tailored for secure data storage, real-time automation, and network surveillance. As digital infrastructure grows and more devices become interconnected, the demand for a localized yet intelligent and secure system has increased. This project addresses this demand by integrating multiple functions into a compact and cost-effective platform.

1.2 Objectives of the Project

- To design a unified AIoT system combining NAS, automation, AI, and network security features.
- To utilize Raspberry Pi 5 as a cost-effective and portable edge computing platform.
- To implement QEMU-based virtualization for safe and rapid prototyping.
- To develop AI modules for intrusion detection, automation, and system intelligence.
- To demonstrate modularity, scalability, and low-latency decision-making.

1.3 Project Focus

The primary focus of the project is to create an integrated system that addresses three core areas:

i. Secure Local Storage: Implementation of a NAS using Samba and associated file system tools.
ii. AI-driven Cybersecurity: Use of anomaly detection and lightweight firewalls to monitor traffic.
iii. Smart Automation & Interaction: Incorporation of chatbot interfaces and GPIO

control for home or lab environments.

Additionally, the system is designed for extensibility with cloud integration and real-time AI inference using TensorFlow Lite and OpenCV. Red team scenarios are supported for testing security protocols and countermeasures.

1.4 Organization of Report

This report is structured into five main chapters:

- **Chapter 1:**
  Introduces the topic, objectives, focus areas, and report structure.
- **Chapter 2:**
  Reviews the background and technical foundations of the system, including architecture and tools.
- **Chapter 3:**
  Describes the methodology, tools used, development timeline, and simulation process.
- **Chapter 4:**
  Explains the implementation of modules and the virtual prototype.
- **Chapter 5:**
  Concludes the project and outlines future enhancements.
- **Chapter 6:**
  Gives References for the project.

Chapter 2

BACKGROUND MATERIAL

2.1 Conceptual Overview

The rapid evolution of AI and IoT technologies has led to the emergence of AIoT (Artificial Intelligence of Things), combining the intelligence of AI with the connectivity and sensing capabilities of IoT devices. This synergy facilitates smarter decision-making at the network edge, enabling automated responses, efficient resource management, and enhanced security in real-time.

This project builds upon this concept by integrating AI-driven modules with IoT-based control and monitoring systems on the Raspberry Pi 5 platform. It provides local data storage through NAS services, supports real-time automation, and performs intelligent traffic monitoring and anomaly detection. With edge computing, this design mitigates latency issues, enhances user privacy, and reduces reliance on cloud infrastructure.

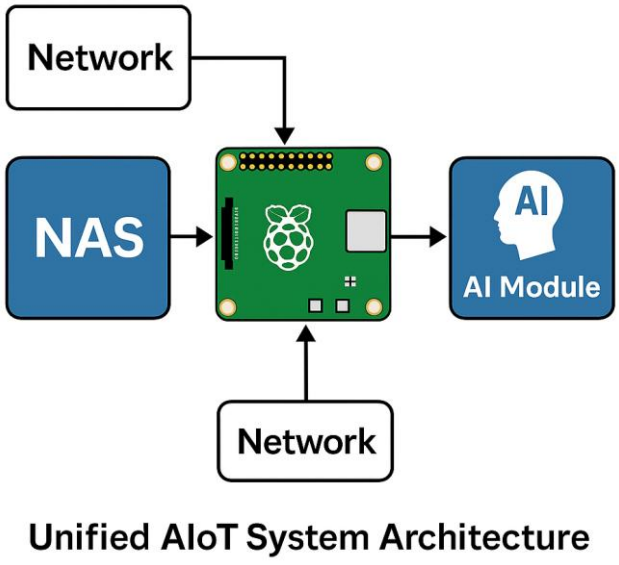
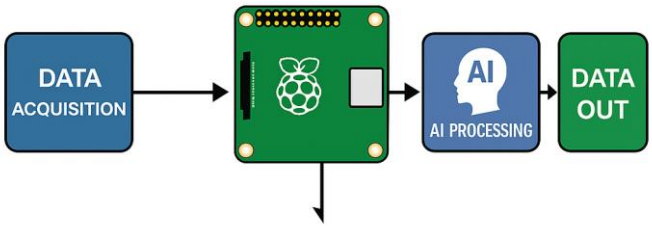


Fig 2.1.1: Unified AIoT System Architecture



Modular Data Flow Diagram

Fig 2.1.2: Modular Data Flow Diagram

2.2 Technologies Involved

2.2.1. Raspberry Pi 5: A credit-card-sized computer with sufficient processing power to host a Linux-based operating system, run Docker containers, and support AI workloads using TensorFlow Lite.

2.2.2. NAS (Network Attached Storage): Implemented using Samba, this module allows for local file hosting, media sharing, and user-controlled data access.

2.2.3. Firewall & Intrusion Detection: Lightweight Linux firewalls (e.g., UFW/IPTables) and AI models for traffic analysis are used to detect and respond to suspicious network behavior.

2.2.4. AI Chatbot Interface: A local or cloud-integrated NLP-based chatbot interface facilitates user interaction with the system for automation, querying, and status monitoring.

2.2.5. Virtualization (QEMU): Used for early-phase testing and validation, QEMU allows emulation of ARM-based environments to simulate the behavior of the Raspberry Pi system on standard x86 hardware.

2.2.6. Automation Tools & GPIO: Physical devices can be controlled using the GPIO pins on Raspberry Pi. Python scripts and shell commands manage these outputs based on AI model decisions.

2.2.7. **TensorFlow Lite & OpenCV**: AI models for image recognition, anomaly detection, and automation tasks are deployed using TensorFlow Lite for lightweight inference. OpenCV enables image processing for surveillance or visual automation triggers.

2.2.8. **Linux Shell & Bash Scripting**: Core automation and modularity are handled through scripted workflows, improving repeatability and customization.

These technologies collaboratively establish a secure, responsive, and intelligent edge device that aligns with modern AIoT and cybersecurity needs.

2.3 Related Work and Studies

Several academic and industrial efforts have explored the intersection of AI, IoT, and cybersecurity. Projects such as OpenHAB and Home Assistant demonstrate the effectiveness of open-source platforms in smart home environments. On the AI front, lightweight machine learning frameworks like TensorFlow Lite have been shown to perform well on resource-constrained devices, such as the Raspberry Pi.

Recent studies have focused on anomaly-based intrusion detection using machine learning algorithms deployed on edge devices. These methods, while still maturing, are gaining traction due to their ability to operate without constant cloud communication, thus improving privacy and reducing bandwidth consumption.
This project draws inspiration from such studies while extending functionality through a unified system combining NAS, firewall, chatbot, and automation—all managed by a central AI module.

2.4 Motivation and Innovation

The motivation behind this project stems from the increasing need for decentralized, intelligent, and private systems. Many commercial smart devices rely on continuous cloud connectivity, which poses security and latency concerns. By keeping data and intelligence at the edge, the system provides an alternative approach that is secure, cost-effective, and customizable.

The innovation lies in the seamless integration of various modules—NAS, AI security, automation, and virtualization—on a single Raspberry Pi platform. The project adopts a modular structure allowing easy reconfiguration, scaling, and adaptation to different environments such as homes, offices, or research facilities.

Chapter 3

METHODOLOGY

3.1 Detailed Methodology

The development methodology followed in this project combined rapid prototyping with modular testing and virtualization. The system was first conceptualized using a layered model—hardware layer, OS layer, service layer, and AI layer. The initial prototyping was done virtually using QEMU to emulate Raspberry Pi 5, which allowed for safe experimentation without hardware damage.

Development proceeded in sprints, focusing on individual modules such as NAS, AI detection, and chatbot integration. Each module was tested independently before integration. Docker containers were used to simulate service environments, ensuring scalability and isolation of components.

A combination of scripting and containerization ensured easy deployment and reusability across devices. Security protocols were simulated using Kali Linux on a virtual network to validate firewall and intrusion detection systems.

3.2 Tools and Libraries Used

Tool/Library	Purpose
Raspberry Pi OS	Base operating system
Samba	NAS and media sharing
QEMU	Virtualization and prototyping
TensorFlow Lite	Lightweight AI inference
OpenCV	Image processing and surveillance
UFW/IPTables	Firewall setup

Python	Scripting and automation
Bash	Shell-based control logic
Docker	Containerization of services
Flask/FastAPI	Local service and API endpoints

Table 3.2: Tools and Libraries Used

3.3 Timeline and Phases

Phase	Activities
Phase 1	Literature review, architecture planning
Phase 2	QEMU setup, toolchain testing
Phase 3	NAS and Samba deployment, Docker test environment
Phase 4	AI model selection, chatbot API development
Phase 5	Integration, GPIO emulation, automation logic
Phase 6	Testing with simulated attacks, visualization setup
Phase 7	Final deployment on Raspberry Pi 5

Table 3.3: Timeline and Phases

3.4 Deployment and Simulation Environment Setup

To minimize risk and allow flexible testing, deployment began in a virtual environment using QEMU. An emulated Raspberry Pi 5 running Raspberry Pi OS Lite was set up. NAS functionality was deployed using Samba configured with public and private shares.

AI modules were packaged using TensorFlow Lite models for anomaly detection and OpenCV for any camera-based triggers. A lightweight Flask server hosted a chatbot interface and received automation commands via APIs.

Once stable, the system was migrated to a physical Raspberry Pi 5 board. GPIO pins were used to control LEDs and relays for simulation of smart home automation. The deployment was verified using network monitoring tools and red-team testing methods from Kali Linux.

Environment	Configuration Details
QEMU Virtual Environment	Raspberry Pi OS Lite simulated with ARM support; Samba, AI, and chatbot tested here
NAS Simulation	Configured public/private shares using Samba in QEMU
AI Inference	Simulated via TensorFlow Lite and OpenCV with virtual camera input
GPIO Simulation	Dummy GPIO used in QEMU for automation module testing
Physical Deployment	Raspberry Pi 5, SSD board (X1004), GPIO relay modules; real sensors used post-validation
Remote Access	SSH, VNC enabled; Git & SCP used for module transfer to Raspberry Pi

Table 3.4: Deployment and Simulation Environment Setup

The development and deployment process ensured minimal risk and maximum portability by validating functionality virtually before final hardware testing. This chapter paves the way for implementation specifics outlined in the next section.

Chapter 4

IMPLEMENTATION

4.1 Core Modules and Functions Implementation

This chapter details the practical implementation of the various modules in the unified AIoT system. Each module has been developed and integrated in a manner that supports interoperability, scalability, and lightweight execution on the Raspberry Pi platform.

4.1.1 NAS and Media Server

The system uses Samba to create a reliable NAS and media server. Shared folders were configured with varying levels of access control and permissions. The media server was further enhanced with DLNA support for streaming content to smart devices on the network.

4.1.2 AI-Enhanced Intrusion Detection

TensorFlow Lite and Python were used to deploy a simple machine learning model that monitors network traffic patterns and flags anomalies. These models were trained using datasets representing normal and malicious behaviors. Alerts are generated upon detecting suspicious activities.

4.1.3 Local Firewall and Packet Filtering

The Uncomplicated Firewall (UFW) and iptables were configured for layered defense. Rules were customized to detect port scanning and brute-force attempts. Logs are stored locally for auditing.

4.1.4 Chatbot and Edge AI Interface

A lightweight AI chatbot was deployed using Python's Flask framework, acting as an interface for controlling home automation functions and querying system status. NLP capabilities were powered by a distilled BERT model fine-tuned for local inference.

4.1.5 Smart Home Automation

GPIO control for lights, fans, and sensors was implemented using Python and bash scripts. A virtual testing setup was built using QEMU to simulate GPIO pins before actual deployment on hardware. Relay modules interfaced with the Raspberry Pi allowed safe switching of AC appliances.

4.1.6 Containerized Deployment

All services like NAS, AI detection, firewall, chatbot were containerized using Docker. This ensured consistent builds, easy updates, and system isolation for better security.

Module Name	Functionality
NAS and Media Server	File storage and sharing via Samba; DLNA support for streaming media
AI-Enhanced IDS	Lightweight ML model detects network anomalies using TensorFlow Lite
Firewall & Packet Filtering	UFW + iptables configured to block scans, brute force, etc.
AI Chatbot Interface	Flask-based chatbot for local queries and home automation
Smart Home Automation	GPIO-controlled devices via Python scripts; virtualized for QEMU simulation
Containerized Deployment	Docker containers for each service to isolate and scale reliably

Table 4.1: Core Modules and Functions Implemented

4.2 Prototype

The prototype was first built and validated in a QEMU virtual environment to simulate the behavior of Raspberry Pi 5. A lightweight Raspberry Pi OS image was installed, and modules were integrated incrementally.

Virtual GPIO pins were tested with dummy sensors, and Docker containers were validated for memory usage and CPU footprint. Once stable, the solution was flashed onto an actual Raspberry Pi 5 board. The system booted successfully, with all services running concurrently within resource constraints.

Network simulations were conducted using tools from Kali Linux and Wireshark to verify firewall efficiency and intrusion response. Automation components were connected to physical GPIO pins and successfully responded to voice and chatbot commands.

4.3 Hardware Setup

The hardware implementation of the Raspberry Pi-based NAS system involves connecting the Raspberry Pi 4 Model B with external storage devices and necessary power and network connections. The following hardware components were used:

- **Raspberry Pi 5**
- **MicroSD Card (32 GB) with Raspberry Pi OS (64-bit)**
- **1TB NVME SSD**
- **X1004 PCIe to Dual 2280 NVMe SSD Peripheral Board**
- **Ethernet cable for LAN connectivity**
- **Metal Case Compatible with X1004**

The Raspberry Pi is configured headlessly over SSH and VNC. The NAS service was established using open-source software like Samba and Nextcloud, while the AI/ML model was implemented in Python using scikit-learn and TensorFlow Lite.

Fig 4.3.1: Raspberry Pi 5

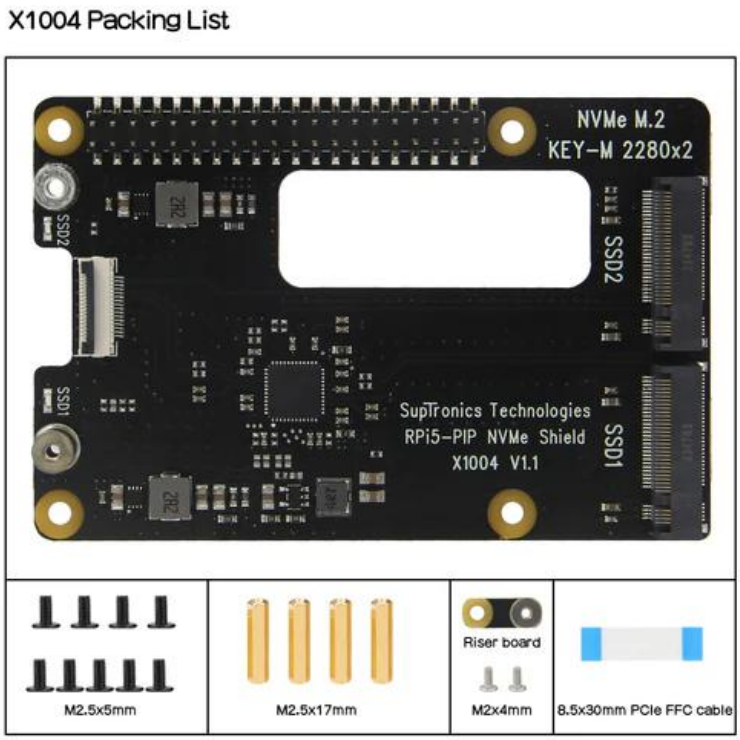


Fig 4.3.2 X1004 PCIe to Dual 2280 NVMe SSD Peripheral Board

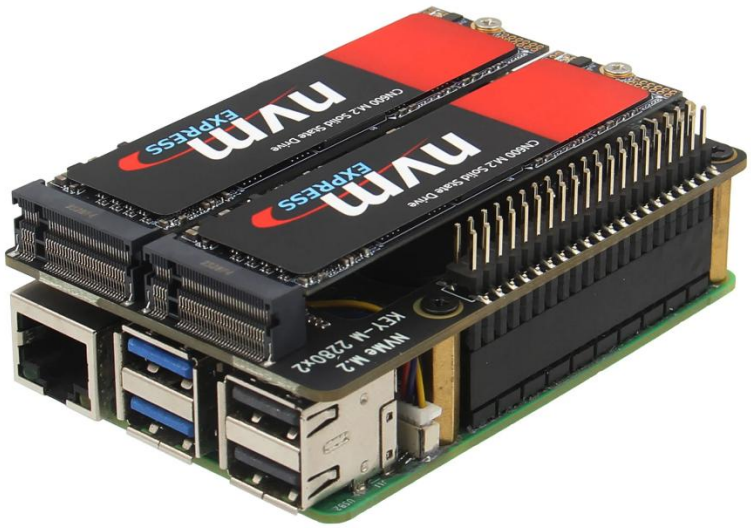


Fig 4.3.3 Raspberry Pi 5 with X1004 PCIe to Dual 2280 NVMe SSD

Fig 4.3.4: Raspberry Pi 5 PCIe Metal case Compatible with X1004

4.4 Software Architecture

The software architecture includes:

- **NAS Software Stack:** Samba for file sharing, OpenMediaVault for NAS UI, and Nextcloud for cloud access.
- **AI/ML IDS:** Developed in Python using scikit-learn and converted to TensorFlow Lite model for performance on Raspberry Pi.
- **Logging & Alert System:** Implemented using Python scripts that log anomalies, send notifications via email/Telegram, and display alerts on the dashboard.

The software is modular, with clearly defined interfaces between the NAS services and the security engine.

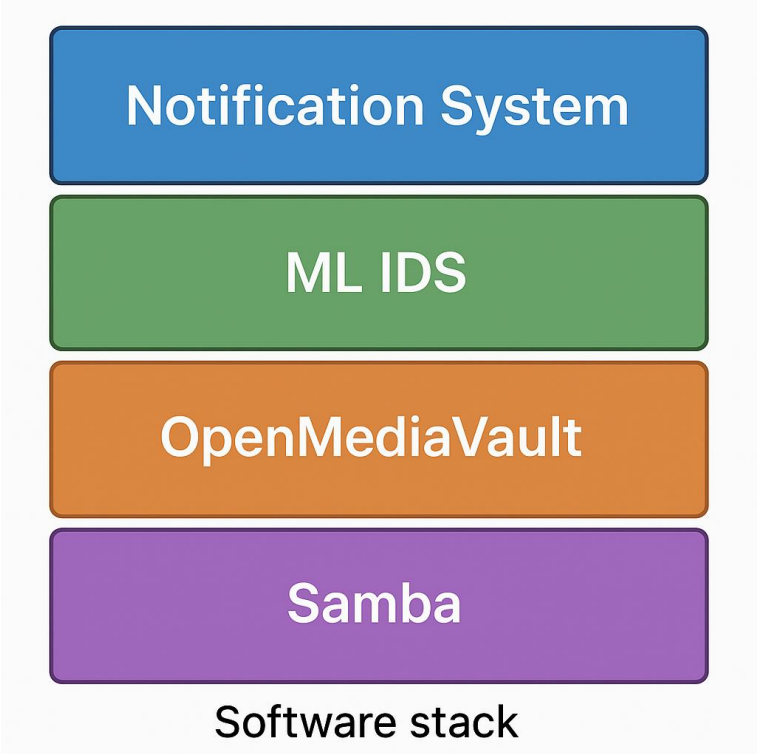


Fig 4.4: Software Stack

4.5 Integration of AI with NAS

The AI/ML-based Intrusion Detection System (IDS) was trained using a custom dataset of network traffic patterns and common cyberattacks (DoS, brute force, unauthorized access).

Key steps:

4.5.1 Dataset Preprocessing: Traffic logs and access patterns were labelled and normalized.

4.5.2 Model Training: A Random Forest classifier and a lightweight neural network were trained and evaluated.

4.5.3 Model Deployment: The best-performing model was converted using TensorFlow Lite and deployed to Raspberry Pi.

4.5.4 Real-time Monitoring: Incoming traffic was processed in real-time using a background service and classified.

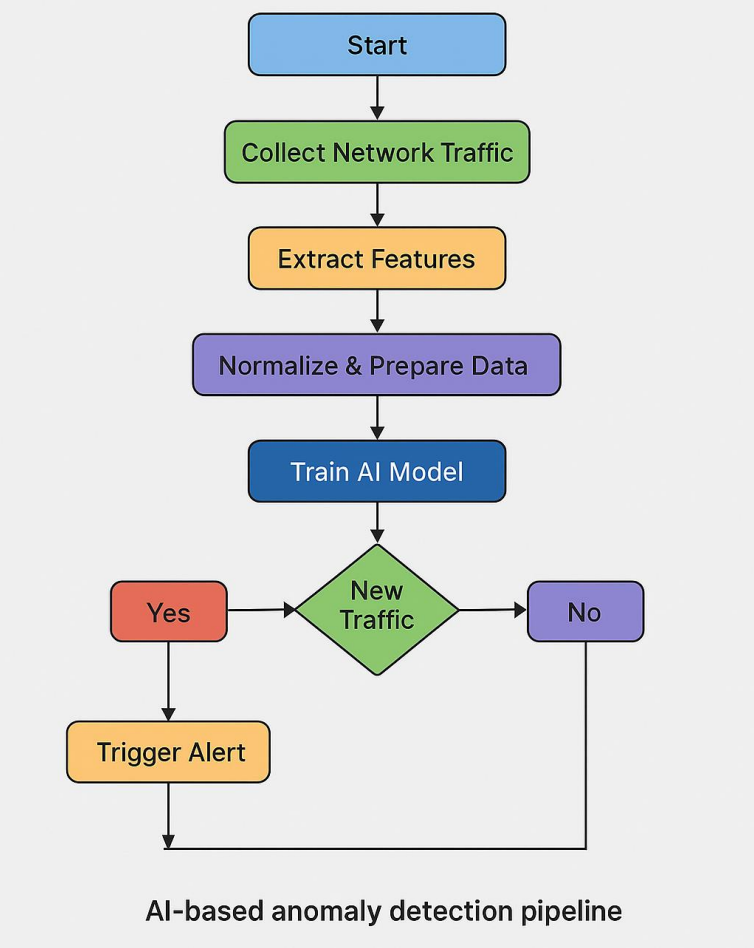


Fig 4.5: Flowchart of AI-based anomaly detection

4.6 Security Measures Implementation

- **Encryption:** TLS/SSL for secure access via Nextcloud
- **Firewall:** UFW configured with strict rules
- **SSH Hardening:** Port change, key-based auth, login banners
- **IDS:** ML model alerts on abnormal behavior
- **Access Control:** Role-based file sharing and access

Chapter 5

CONCLUSION & FUTURE SCOPE

5.1 Conclusions

This project successfully demonstrates the feasibility and value of a compact, AI-powered, and secure NAS system built using the Raspberry Pi platform. By integrating NAS functionalities with AI-based intrusion detection, containerized services, and smart automation, the solution offers a unified system designed for smart environments, personal cybersecurity, and edge computing applications.

The use of QEMU for virtual prototyping validated the system's modularity and allowed for rapid, risk-free development. Upon testing in the actual Raspberry Pi 5 hardware, all core modules operated seamlessly under resource constraints. The deployment of TensorFlow Lite for anomaly detection, UFW-based firewall configurations, and Docker for isolated service containers ensured both efficiency and security.

The chatbot and automation interfaces further personalized the system, empowering users with real-time insights and control via voice and text. The overall system proves how edge computing, paired with AI and open-source tooling, can redefine affordable, scalable, and private cyber-physical ecosystems.

Key achievements include:

- Designing and deploying a local NAS on Raspberry Pi
- Implementing ML-based real-time intrusion detection
- Securing the system with multiple layers of access control, encryption, and alerting
- Demonstrating resilience against common attack vectors

This solution is ideal for personal, small business, or educational use where affordability, privacy, and flexibility are priorities.

5.2 Future Scope of Work

5.2.1 Facial Recognition & Biometric Access: Integration of real-time facial recognition for user authentication to enhance physical and data access control.

5.2.2 Blockchain Logging: Employ blockchain to maintain tamper-proof access logs and ensure secure, decentralized storage history.

5.2.3 Smart City and Industrial Deployment: Scale the system for industrial IoT or smart city use cases, with support for mass data collection and real-time analytics.

5.2.4 ELK Stack Integration: Incorporate Elasticsearch, Logstash, and Kibana for better log visualization, forensic insights, and live monitoring dashboards.

5.2.5 Secure Remote Updates: Implement secure OTA firmware updates for distributed Raspberry Pi devices in real-world deployments.

5.2.6 Integration with AI Threat Intelligence: Real-time syncing with threat intelligence feeds for dynamic firewall rules and anomaly model updates.

The current system can be extended in several directions:

- **Advanced AI Models:** Implement deep learning models like LSTM for sequential anomaly detection.
- **IoT Expansion:** Use the same architecture for smart home device security.
- **Federated Learning:** Allow edge devices to train models collaboratively without data sharing.
- **Mobile App:** Build an Android/iOS app to monitor storage and security alerts.
- **Hardware Optimization:** Move to Raspberry Pi Compute Module or Jetson Nano for better performance.

This prototype can evolve into a scalable open-source solution for edge AI cybersecurity in smart environments and this project lays the groundwork for more extensive research and implementation in the domains of secure IoT, AI at the edge, and autonomous system orchestration.

Chapter 6

REFERENCES

6.1 Research Papers

[6.1.1] Diro, A. A., & Chilamkurti, N. (2018). Distributed Attack Detection Scheme Using Deep Learning Approach for Internet of Things. Future Generation Computer Systems, 82, 761–768. https://doi.org/10.1016/j.future.2017.08.043Research Paper 2

[6.1.2] Hodo, E., Bellekens, X., Hamilton, A. W., Dubouilh, P.-L. (2016). *Threat Analysis of IoT Networks Using Artificial Neural Network Intrusion Detection System*. In Proceedings of the 3rd International Symposium on Networks, Computers and Communications (ISNCC), Hammamet. https://doi.org/10.1109/ISNCC.2016.7746067

[6.1.3] Khan, R., McDaniel, P., & Khan, S. U. (2018). *A Survey of Security Issues in Cloud Computing: Attacks and Defenses*. Journal of Network and Computer Applications, 71, 11–29. https://doi.org/10.1016/j.jnca.2016.09.002

[6.1.4] Xu, R., Bertran, P., & Zhu, Q. (2020). *A Survey on Edge Computing for the Internet of Things*. IEEE Access, 8, 85714–85733. https://doi.org/10.1109/ACCESS.2020.2991734

[6.1.5] Radoglou Grammatikis, P. I., Sarigiannidis, P. G., & Moscholios, I. D. (2018). *Securing the Internet of Things: Challenges, Threats and Solutions*. Internet of Things, 1–2, 41–70. https://doi.org/10.1016/j.iot.2018.11.003

6.2 Reference Guide on Tools and Technologies

[6.2.1] Cybersecurity Tools: https://www.kali.org/tools/

[6.2.2] Raspberry Pi Foundation. (2024). *Raspberry Pi 5 Documentation*. Retrieved from https://www.raspberrypi.com/documentation/

[6.2.3] TensorFlow Lite. (2024). *Lightweight Machine Learning for Mobile and Edge Devices*. Retrieved from https://www.tensorflow.org/lite

[6.2.4] Docker Inc. (2024). *Docker Documentation*. Retrieved from https://docs.docker.com/

[6.2.5] UFW – Uncomplicated Firewall. (2024). *Ubuntu Community Documentation*. Retrieved from https://help.ubuntu.com/community/UFW

[6.2.6] QEMU Project. (2024). *QEMU: Generic and Open Source Machine Emulator and Virtualizer*. Retrieved from https://www.qemu.org/

[6.2.7] OWASP Foundation. (2024). *Mobile Security Testing Guide*. Retrieved from https://owasp.org/www-project-mobile-security-testing-guide/

[6.2.8] BERT Model: Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv preprint arXiv:1810.04805.

[6.2.9] Elasticsearch, Logstash, and Kibana. (2024). *ELK Stack Documentation*. Retrieved from https://www.elastic.co/what-is/elk-stack

[6.2.10] Wireshark Foundation. (2024). *Wireshark Network Protocol Analyzer*. Retrieved from https://www.wireshark.org/

[6.2.11] Kali Linux. (2024). *Penetration Testing Tools for Cybersecurity Professionals*. Retrieved from https://www.kali.org/