SKILL INCUBATION INNOVATION ENTREPRENEURSHIP DEVELOPMENT CENTRE, UNIVERSITY OF JAMMU

DESIGN YOUR DEGREE

TECHNOLOGIES OF THE FUTURE

COURSE CODE: UFDDPC305

PROJECT TITLE:

"DESIGN AND DEVELOPMENT OF AN INTRUSION DETECTION SYSTEM (IDS) FOR IOT DEVICES USING BLOCKCHAIN TECHNOLOGY"

Submitted by - Cybersecurity

Roll No	Name
DYD-23-01	Adil Mahajan
DYD-23-08	Ishaan Uppal
DYD-23-13	Neamat Kour
DYD-23-18	Sarnish Kour
DYD-23-20	Suhani Behl
DYD-23-21	Tavishi Amla

Submitted to:

Dr. Jatinder Manhas
Associate Professor
Computer Science & IT (SIIEDC)
University of Jammu

CERTIFICATE

The report titled Blockchain-Integrated Intrusion Detection System for IoT: A Multi-Algorithm Approach has been completed by the group comprised of Adil Mahajan, Ishaan Uppal, Neamat Kour, Sarnish Kour, Suhani Behl and Tavishi Amla as a major project for Semester III. It was conducted under the guidance of Dr. Jatinder Manhas for the partial fulfilment of the Design Your Degree, Four-Year Undergraduate Programme at the University of Jammu, Jammu. This project report is original and has not been submitted anywhere else.

Signature of Students

Adil Mahajan

Ishaan Uppal

Neamat Kour

Sarnish Kour

Suhani Behl

Tavishi Amla

Signature of the Mentors

Dr. Jatinder Manhas

Prof. Alka Sharma

Director, SIIEDC, University of Jammu

ACKNOWLEDGEMENT

We are deeply grateful to all those who have been instrumental in the successful completion of our work. Our foremost thanks go to the Almighty for granting us the passion and spiritual strength to overcome this challenge.

We extend our heartfelt gratitude to Prof. Umesh Rai, Vice Chancellor of the University of Jammu, for his unwavering support and encouragement. His leadership and vision have been a constant source of inspiration for us.

We are also profoundly thankful to Prof. Alka Sharma from SIIEDC at the University of Jammu. Her generous assistance, valuable advice, and provision of necessary facilities significantly contributed to our project.

We are deeply indebted to our mentors: Dr. Jatinder Manhas from the Department of Computer Science at the University of Jammu. Their inspiring guidance, generous encouragement, and expert advice have been a beacon of light throughout our work. Despite their busy schedules, they provided unwavering support and motivation, for which we feel extremely honored and fortunate.

Our sincere thanks extend to all the respected professors and mentors who played a pivotal role during the execution of this major project. Their unwavering support and encouragement fueled our progress, and we cherish the cooperation and affection that accompanied our work.

Furthermore, we recognize the support of our team members, whose fellowship and teamwork made our research journey enjoyable and rewarding.

INDEX

CHAPTER	CHAPTER NAME	PAGE No.
1	Introduction	6 – 9
2	Literature Review	10 – 15
3	Project Objective and Aim	16
4	Scope and Methodology	17 – 21
5	Tools and Technology	22 – 25
6	Data Collection and Data Structure	26 – 29
7	Experimental Setup	30 – 67
8	Conclusion	68 – 69
	References	70 – 73

ABSTRACT

The proliferation of the Internet of Things (IoT) has revolutionized industries, homes, and critical infrastructures by interconnecting devices for enhanced efficiency and convenience. However, this interconnectedness brings significant cybersecurity challenges, as traditional security mechanisms struggle to address the resource constraints and diverse operational environments of IoT networks. To tackle these vulnerabilities, this project integrates blockchain technology with Intrusion Detection Systems (IDS), creating a robust security framework tailored for IoT ecosystems. This study proposes a blockchain-based IDS leveraging decentralized, immutable, and transparent blockchain architecture. Blockchain ensures tamper-proof logging of intrusion events, eliminates single points of failure, and improves data integrity. The system employs smart contracts to automate real-time responses to detected threats, minimizing impact and improving response efficiency. Additionally, machine learning models such as Isolation Forest and Random Forest are integrated to enhance real-time anomaly detection, adapting to evolving security threats while maintaining resource efficiency. The project's objectives include creating a scalable, lightweight IDS capable of operating on resource-constrained IoT devices, ensuring tamper-proof data storage, and providing an intuitive user interface for real-time monitoring and management. The system's scope encompasses anomaly detection, automated threat response, and a web-based dashboard for user-friendly interactions. Key technological components include TensorFlow Lite, Ethereum blockchain, Flask APIs, and IPFS for off-chain storage. Methodologically, the project progresses through stages of data collection, anomaly detection modeling, blockchain integration, real-time alerting, system development, and testing. Experimental setups feature IoT devices like Raspberry Pi and sensors, combined with blockchain frameworks to log and mitigate security events in real-time. Challenges such as scalability, resource constraints, and data privacy are addressed through solutions like lightweight consensus mechanisms, cloud-based task offloading, and encrypted logging. This research significantly contributes to IoT cybersecurity by integrating advanced machine learning with blockchain technology. It provides a transformative solution to address the vulnerabilities of centralized IoT systems while ensuring user trust and system reliability. The findings pave the way for future applications in critical sectors like healthcare, smart cities, and industrial automation, marking a pivotal step toward building secure, scalable IoT ecosystems.

CHAPTER 1
INTRODUCTION

In the era of rapid technological advancement, the Internet of Things (IoT) has emerged as a transformative force, interconnecting a myriad of devices and systems to enhance efficiency and convenience across various sectors. However, this interconnectedness also introduces significant vulnerabilities, making IoT devices prime targets for cyber intrusions. As these devices proliferate in homes, industries, and critical infrastructures, ensuring their security has become paramount. Traditional security mechanisms often fall short in addressing the unique challenges posed by IoT environments, which are characterized by resource constraints and diverse operational contexts. To combat these security challenges, innovative solutions are required. One promising approach is the integration of blockchain technology into the design and development of Intrusion Detection Systems (IDS) tailored for IoT devices. Blockchain offers a decentralized and tamper-resistant framework that can enhance data integrity and provide transparent audit trails for security events. By leveraging the inherent properties of blockchain—such as immutability, transparency, and distributed consensus—this project aims to develop a robust IDS that not only detects intrusions but also facilitates secure communication among IoT devices. The proposed system will utilize smart contracts to automate responses to detected threats, thereby minimizing response times and reducing the potential impact of intrusions. Additionally, the use of blockchain will empower users with greater control over their data, allowing them to verify the authenticity of communications and interactions within their IoT ecosystem. This dual focus on intrusion detection and data integrity is crucial for fostering trust in IoT applications, particularly in sensitive domains such as healthcare, finance, and smart cities. Through this research, we aim to contribute to the growing body of knowledge surrounding cybersecurity in IoT environments while addressing the pressing need for secure and resilient systems. By combining advanced intrusion detection techniques with blockchain technology, this project seeks to pave the way for a more secure future where IoT devices can operate safely and efficiently without compromising user privacy or system integrity.

This research explores recent advancements, challenges, and future directions in implementing IDS for IoT devices using blockchain technology.

A. *IoT:* The Internet of Things (IoT) refers to the network of physical objects “things” that are embedded with sensors, software, and other technologies to connect and exchange data with other devices and systems over the internet. These “things” can

range from ordinary household items like refrigerators and thermostats to sophisticated industrial tools and machines. The main idea is to create smarter environments that improve efficiency, convenience, and decision-making.

B. *Blockchain:* Blockchain is a database that maintains a continuously growing set of data records. It is distributed in nature, meaning that there is no master computer holding the entire chain. Rather, the participating nodes have a copy of the chain. It's also ever-growing - data records are only added to the chain.

C. *Machine Learning:* Machine learning (ML) is a sub-field of Artificial Intelligence (AI) and can be defined as "the field of computer science that gives computers the ability to learn without being explicitly programmed". Several ML algorithms are already available i.e., Linear Regression, Support Vector Machine (SVM), Decision Tree, Naïve Bayes, Artificial Neural Network (ANN), Random Forest (RF), Deep Learning (DL), K-Nearest Neighbour (KNN) etc.

1.1 INTRUSION DETECTION SYSTEM (IDS)

An Intrusion Detection System (IDS) is a security mechanism designed to monitor network traffic and identify suspicious activities or policy violations. IDS can be categorized into two main types:

- **Network-based IDS (NIDS)**: Monitors network traffic for all devices on the network.
- **Host-based IDS (HIDS)**: Monitors individual devices or hosts for suspicious activity.

1.1.1 Uses of IDS:

- **Threat Detection**: IDS helps in detecting unauthorized access attempts, malware infections, and other malicious activities in real-time.
- **Incident Response**: By identifying potential threats, an IDS allows organizations to respond promptly to security incidents, minimizing damage and data loss.
- **Compliance Monitoring**: Many industries are required to comply with regulations that mandate monitoring and reporting of security incidents. IDS assists organizations in meeting these compliance requirements.
- **Forensic Analysis**: In the event of a security breach, an IDS can provide valuable data for forensic analysis, helping to understand how the breach occurred and what vulnerabilities were exploited.

1.2 BLOCKCHAIN TECHNOLOGY

Blockchain is a decentralized digital ledger technology that records transactions across many computers in such a way that the registered transactions cannot be altered retroactively. This ensures transparency, security, and integrity of data.

1.2.1. USES OF BLOCKCHAIN:

- **Cryptocurrency Transactions**: The most well-known application of blockchain is in cryptocurrencies like Bitcoin, where it facilitates secure peer-to-peer transactions without a central authority.
- **Smart Contracts**: Blockchain enables the creation of smart contracts—self-executing contracts with the terms directly written into code—which automatically enforce and execute agreements when conditions are met.
- **Supply Chain Management**: Blockchain can enhance traceability and transparency in supply chains by providing an immutable record of product movements from origin to consumer.
- **Identity Verification**: It can be used for secure identity management systems, allowing
  individuals to control their own digital identities without relying on central authorities.

1.3 ROLE OF BLOCKCHAIN IN CYBERSECURITY

Integrating blockchain technology into cybersecurity strategies can significantly enhance the effectiveness of Intrusion Detection Systems (IDS). Here's how:

- **Data Integrity**: Blockchain's immutable nature ensures that once data is recorded, it cannot be altered. This feature can be utilized by IDS to maintain a secure log of all network activities, making it difficult for attackers to manipulate logs or erase traces of their actions.
- **Decentralization**: By decentralizing the storage of IDS data across multiple nodes, blockchain reduces the risk of a single point of failure. This makes it more resilient against attacks targeting centralized systems.
- **Enhanced Authentication**: Blockchain can improve authentication processes through decentralized identity verification methods. This reduces the likelihood of unauthorized access, which is critical for maintaining the integrity of IoT devices.

- **Automated Responses**: Smart contracts can automate responses to detected threats by executing predefined actions when certain conditions are met, such as isolating compromised devices or alerting administrators instantly.

  By leveraging these capabilities, organizations can develop more robust intrusion detection systems that not only identify threats but also enhance overall cybersecurity posture against evolving cyber threats.

Figure 1.1: Working of blockchain

CHAPTER 2
LITERATURE REVIEW

A collaborative threat intelligence framework was proposed by Nazir et al. (2024) [1] to enhance the security of IoT systems by integrating machine learning (ML) models with blockchain technology. The framework leverages an iOS application as a central control center to detect and report threats in real time. Threat data is securely stored on a blockchain network, enabling ML models, such as Random Forest and LSTM, to access diverse threat scenarios for improved detection accuracy. Experimental evaluations using the IoT23 dataset demonstrated that the system significantly reduced false negatives and improved detection rates, achieving high performance in identifying threats while maintaining data integrity.

A federated learning and blockchain-based model was introduced by Govindaram and Jegatheesan (2024) [2] to enhance IoT security. The proposed FLBC-IDS system integrated Horizontal Federated Learning (HFL) with Hyperledger Blockchain and EfficientNet, offering tamper-proof, decentralized intrusion detection. The system used HFL to preserve data privacy and blockchain to ensure transparency and model integrity. It achieved high metrics, such as 98.89% accuracy and an F1-score of 98.29%, when tested on datasets like CICIDS-2018 and CICIoT-2023. The study demonstrated the system's effectiveness in detecting cyber threats while addressing challenges of latency, model poisoning, and resource limitations in IoT networks.

Samaniego et al. (2016) [3] proposed a Blockchain as a Service (BaaS) model for IoT, comparing the suitability of cloud and fog computing environments. The study highlighted blockchain's tamper-proof storage capability for managing IoT device configurations, sensor data, and micro-payments. The experiments revealed that fog computing outperformed cloud computing in latency-sensitive scenarios, making it a preferable host for blockchain-enabled IoT systems. The evaluation emphasized that network latency was the primary factor affecting system performance. The study concluded that fog-based blockchain offered significant advantages, including reduced latency and enhanced scalability for IoT applications.

A blockchain-based approach was implemented by Huh et al. [4] to manage and synchronize IoT devices. The proposed system utilized Ethereum's smart contracts to enable secure key management and fine-grained control over IoT configurations. The study highlighted limitations of the traditional client-server model for large-scale IoT networks and proposed blockchain as a solution to enhance data integrity and prevent attacks such as denial-of-service. A proof of concept was developed using Raspberry Pi and smartphones,

demonstrating secure synchronization and authentication. Although tested on a small scale, the system laid the groundwork for managing thousands of IoT devices using a decentralized ledger.

Tanzir et al. [5] proposed a lightweight blockchain-based framework for managing IoT security and scalability. The study integrated Ethereum blockchain with terminal devices, offering decentralized storage and enhanced privacy. Simulation results demonstrated secure data exchange using smart contracts. The framework addressed IoT limitations like memory overhead and centralized systems, ensuring tamper-proof transactions and high availability. The proposed model emphasized scalability and reduced computational costs for IoT systems.

A private blockchain-based model was developed by Calastry et al. [6] to secure IoT data management. Using Ethereum Swarm and IPFS, the system provided decentralized storage for IoT-generated data, ensuring privacy and authenticity. A testbed employing Raspberry Pi demonstrated a transaction throughput of 8–9 transactions per second with encryption, suitable for small industrial applications. The study also employed Trusted Platform Module (TPM) for secure key storage and highlighted the feasibility of low-cost, secure IoT blockchain platforms.

Raj et al. [7] implemented a private Ethereum blockchain for secure communication between IoT devices. The proposed system used smart contracts to authenticate devices, ensuring that only registered devices could interact. A smart home network was demonstrated using Raspberry Pi, smartphones, and LED devices. The system improved privacy by keeping data within a private network and eliminated the need for intermediaries. The results showcased the application of Ethereum in securing IoT devices in home automation and commercial settings.

Ibrahim et al. [8] proposed a decentralized system to prevent DDoS attacks on IoT devices using Ethereum. The study employed smart contracts to authenticate IoT devices and track malicious IP addresses. The blockchain-based system replaced traditional centralized solutions and maintained a tamper-proof record of device credentials. Experimental evaluation demonstrated the system’s efficiency in handling authentication requests with minimal latency, providing a robust solution against DDoS and identity spoofing attacks.

Alsharif et al. [9] integrated machine learning and blockchain to develop an intrusion detection system (IDS) for IoT networks. Random Forest achieved a detection accuracy of 99.9% on simulated attack data. Blockchain was utilized to encrypt interactions and provide tamper-proof storage for detection results. The study highlighted blockchain's decentralization and immutability, addressing IoT vulnerabilities like data tampering and unauthorized access. The

approach demonstrated the effectiveness of combining machine learning with blockchain for IoT security.

Tukur et al. (2021) [10] proposed an Edge-based Blockchain-enabled anomaly detection framework for preventing insider attacks in IoT systems. The framework integrates edge computing to reduce latency and bandwidth constraints and employs Ethereum smart contracts for anomaly detection. Experimental evaluations using real IoT system datasets demonstrated that the system accurately detected and corrected anomalies while ensuring data integrity. However, the reliance on fog computing posed challenges related to cloud security risks.

Ahamad et al. (2021) [11] explored the role of blockchain and IoT in securing financial transactions in cryptocurrency markets. The study highlighted blockchain's immutability, decentralization, and cryptographic security as essential for preventing fraud. The key challenge identified was the scalability of IoT networks, as centralized architectures create single points of failure and privacy concerns. Blockchain was proposed as a solution to enhance data integrity and traceability. However, energy consumption and regulatory uncertainty remain critical concerns.

Kumar et al. (2022) [12] introduced a distributed Intrusion Detection System (IDS) for detecting DDoS attacks in blockchain-enabled IoT networks. The proposed system integrates fog computing with AI-based detection mechanisms, utilizing Random Forest (RF) and XGBoost algorithms for attack detection. The IDS was tested using the BoT-IoT dataset, revealing that XGBoost outperformed RF in binary attack detection, while RF was more efficient in multi-attack detection. Despite improved accuracy, the model suffered from computational complexity and resource constraints on distributed fog nodes.

Singh et al. (2021) [13] analyzed various blockchain security attacks and countermeasures. Key threats include transaction privacy leakage, selfish mining, DAO attacks, and BGP hijacking. Solutions such as Oyente (Ethereum contract bug detection), Hawk (privacy-preserving smart contracts), and Town Crier (secure data retrieval) were identified as effective security mechanisms. The study concluded that blockchain-based IoT systems require enhanced privacy models to prevent network-layer attacks and data manipulation.

Mishra et al. (2019) [14] developed a Blockchain-based Intrusion Detection System (IDS) for securing IoT-based healthcare applications. The IDS employed Network Intrusion Detection (NIDS) and Host-based Intrusion Detection (HIDS) techniques for detecting unauthorized access. The study demonstrated improved data privacy and access control, but scalability and real-time performance were identified as limitations due to resource-constrained IoT devices.

Li et al. (2018) [15] proposed a Blockchain-based Authentication and Security Mechanism

for IoT. The system used Hyperledger Fabric to manage device identity and employed asymmetric cryptography for peer-to-peer authentication. The study addressed insider threats and single-point failures but noted that blockchain overhead could delay authentication in large-scale IoT deployments.

The paper by Singh, S., et al. (2020) [16] presents a novel approach to enhancing IoT security through the integration of blockchain technology and machine learning models. The framework uses a collaborative threat intelligence system where threat data is shared and stored securely on a blockchain. This allows machine learning models to learn from a diverse range of threat scenarios, improving their accuracy and reducing false negatives. The research demonstrates the effectiveness of this system using various machine learning models on the IoT23 dataset.

The paper by Porkodi, S., et al. (2020) [17] presents a chapter "Integration of Blockchain and Internet of Things" from the *Handbook of Research on Blockchain Technology* explores how Blockchain and IoT can be integrated to address security, data integrity, and decentralization challenges in IoT networks. It highlights blockchain's benefits, including tamper-resistant data storage, improved trust, and peer-to-peer communication, which help mitigate cyber threats such as data tampering, denial-of-service (DoS) attacks, and sensor data manipulation. Additionally, the chapter examines various blockchain protocols, such as Proof of Work (PoW) and Byzantine Fault Tolerance (BFT), and their role in enhancing IoT security and scalability. However, challenges remain, including scalability issues, transaction speed limitations, and the need for optimized consensus mechanisms tailored for IoT applications. The chapter concludes by suggesting future research directions, such as editable blockchain technology and simplified payment verification, to further enhance the security and efficiency of IoT systems.

The paper by Jiang, Y., et al. (2019) [18] discusses how the convergence of Blockchain and Artificial Intelligence (AI) is transforming the architecture of smart city networks to create sustainable and efficient ecosystems. It highlights the opportunities and challenges of integrating these technologies to achieve the goals of sustainable smart cities. The use of blockchain technology ensures secure data sharing and transaction verification, while AI enhances data analysis and decision-making processes. The article also explores various blockchain security enhancement solutions, including decentralized access control and secure data storage, and their applications in developing intelligent transportation systems. Additionally, the article delves into the potential benefits of these technologies in other smart city domains, such as energy management, healthcare, and public safety.

The paper by Chen, Y. H., et al. (2018) [19] discusses a blockchain-based smart contract system for E-auction to address the issues of high transaction costs and trust in intermediaries. It

proposes the integration of blockchain technology into the bidding process to ensure secure, private, and tamper-proof transactions. The smart contract developed on the Ethereum platform includes essential information such as the auctioneer's address, auction start time, deadline, current highest bidder, and the highest bid amount. The proposed system removes the need for a centralized intermediary, thereby reducing transaction costs and enhancing security. The experiments demonstrate the effectiveness of the system in ensuring confidentiality, non-repudiation, and immutability of bids in both public and sealed bidding processes.

The paper by Christidis, K., et al. (2016) [20], published in IEEE Access, explores the integration of blockchains and smart contracts within the Internet of Things (IoT). The authors highlight how blockchains, with their distributed peer-to-peer networks, enable interactions between non-trusting members without intermediaries, offering verifiable transactions. The paper details the functionality of blockchains and smart contracts, emphasizing their potential to automate multi-step processes. They demonstrate how combining blockchain and IoT facilitates the sharing of services and resources, creating a decentralized marketplace and automating workflows in a cryptographically verifiable manner. Additionally, the authors address important considerations like transactional privacy and the value of digitized assets before deploying blockchain networks in IoT settings. They conclude that the synergy between blockchain and IoT holds significant transformative potential for various industries, paving the way for new business models and innovative applications.

The paper by Dika, A., et al. (2018) [21] explores the security vulnerabilities present in Ethereum smart contracts. Ethereum, being a global computing platform, uses smart contracts to facilitate various applications. The paper reviews known security issues and categorizes them into a comprehensive taxonomy. It also evaluates the effectiveness of several security analysis tools such as Oyente, Securify, Remix, and SmartCheck, using a representative sample of vulnerable contracts. The findings show inconsistencies among the tools, with SmartCheck being the most effective and Oyente the most accurate. The authors propose improvements to user interfaces, interpretation of results, and an enhanced list of vulnerability checks.

The paper "A Blockchain-Based Contractual Routing Protocol for the Internet of Things Using Smart Contracts" by Ramezan, G., et al. (2018) [22] Leung proposes a novel blockchain-based contractual routing (BCR) protocol for a network of untrusted IoT devices. Unlike conventional secure routing protocols that require a central authority (CA) for device identification and authentication, the BCR protocol operates in a decentralized manner using smart contracts. The protocol allows intermediary devices to establish a route from a source IoT device to a destination device or gateway. The performance of BCR is compared with the Ad-hoc On-

Demand Distance Vector (AODV) routing protocol, showing that BCR has significantly lower routing overhead but a slightly lower packet delivery ratio. The paper also demonstrates the protocol's resilience to Blackhole and Greyhole attacks.

The paper by Pal et al. (2019) [23] discusses the integration of blockchain technology with the Internet of Things (IoT) to enhance access right delegation. The authors propose a dual blockchain architecture where attributes are stored on a secure private blockchain, while the public blockchain handles access tokens and delegation control. The model leverages the identity-less, asynchronous, and distributed nature of blockchain to provide a scalable and secure solution for IoT systems. The paper also presents a detailed system design and implementation, demonstrating the effectiveness of the proposed approach through performance evaluation on the Ethereum blockchain network.

The paper by Pan et al. (2019) [24] introduces "EdgeChain," an edge-IoT framework that integrates blockchain and smart contracts to tackle the scalability and security challenges of IoT. EdgeChain leverages edge computing to provide resources closer to IoT devices, reducing latency and enhancing performance. The framework uses a permissioned blockchain and a credit-based resource management system to securely manage IoT device behavior and resource allocation. Smart contracts enforce predefined rules and policies, ensuring secure data logging and auditing. The prototype implementation and experiments demonstrate that EdgeChain effectively integrates blockchain and smart contracts into edge computing without incurring excessive costs.

The paper by Shahbazi, Z., et al. (2021) [25] investigates the integration of blockchain, IoT, and machine learning to enhance multistage quality control and security in smart manufacturing. The authors propose a system that combines these technologies to secure transactions and handle datasets, mitigating the risks of fake data. The implementation uses the Hyperledger Fabric blockchain platform and applies hybrid prediction techniques for fault diagnosis. The quality control system employs non-linear machine learning methods to improve accuracy and reliability. Real-time monitoring is achieved through IoT sensors, and big data techniques are used for data management and analysis.

CHAPTER 3
PROJECT OBJECTIVE AND AIM

3.1 PROJECT OBJECTIVE

1. **Decentralized Security**: Implement a blockchain-based framework to eliminate single points of failure in Intrusion Detection Systems (IDS).
2. **Tamper-Proof Logging**: Ensure that all intrusion detection logs are securely stored on an immutable blockchain ledger, preventing unauthorized alterations.
3. **Real-Time Anomaly Detection**: Integrate machine learning models for real-time detection of security threats and anomalous behavior in IoT networks.
4. **Automated Threat Response**: Utilize smart contracts to trigger automated responses, such as isolating compromised devices or alerting administrators when a threat is detected.
5. **Resource Efficiency**: Design a lightweight system that operates effectively on resource- constrained IoT devices without compromising performance.
6. **Scalability**: Ensure the system is scalable to accommodate large IoT networks and can be adapted for broader applications, including industrial and enterprise environments.
7. **User-Friendly Monitoring**: Develop a user-friendly interface for real-time monitoring, reporting, and management of intrusion detection events.

3.2 AIM

The aim of the project is to create an innovative and scalable solution for IoT security that leverages blockchain's decentralized architecture to provide transparency, reliability, and trustworthiness in intrusion detection. The system will incorporate machine learning for real-time anomaly detection and blockchain technology for immutable and decentralized recording of intrusion events, ensuring robust and adaptable security for IoT environments.

CHAPTER 4
SCOPE AND METHODOLOGY

4.1 PROJECT SCOPE

This project aims to develop a **blockchain-based Intrusion Detection System (IDS)** for IoT networks, focusing on detecting, logging, and mitigating security threats. By leveraging blockchain's immutability and decentralization, alongside machine learning for real-time anomaly detection, the scope includes:

1. **Intrusion Detection Mechanism**: Designing anomaly detection and signature-based models to identify unauthorized access and abnormal IoT behavior in real-time.
2. **Tamper-Proof Logging**: Using blockchain technology to ensure immutable and decentralized recording of intrusion events.
3. **Automated Response**: Implementing smart contracts to trigger automated actions, such as isolating compromised devices or notifying administrators.
4. **User Interaction**: Providing a web-based interface for administrators to monitor intrusion logs, manage devices, and respond to alerts.
5. **Scalability**: Developing a system adaptable to various IoT deployments, including smart homes, industrial setups, and healthcare systems.

4.2 PROPOSED METHODOLOGY

The proposed methodology for this project outlines a structured for developing the IDS, from anomaly detection and blockchain integration to performance testing and optimization.

1. **Data Collection**

Choose Data Sources:

- **Tools**: IoT devices like Raspberry Pi or ESP32, network monitoring tools.
- **Technology**: Packet sniffers (e.g., Wireshark), CPU usage monitors.

This phase involves collecting network data from IoT devices, focusing on:

- **Traffic patterns** (e.g., volume, source/destination).
- **Device behaviour** (e.g., CPU/memory spikes, login attempts).

Malicious activity is simulated to create labelled datasets for training detection models.

2. **Anomaly Detection and Modelling**

Develop Detection Models:

- **Tools**: TensorFlow Lite, Scikit-learn.
- **Technology**: Anomaly-based and signature-based detection.

This step includes training machine learning models on the collected data:

1. **Anomaly Detection**: Use Isolation Forest and Random Forest algorithms for identifying deviations from normal patterns.
2. **Signature Detection**: Match activity against known attack signatures.

Models are validated for accuracy, precision, and recall to minimize false positives.

3. **Blockchain integration**

Implement Blockchain for Logging:

- **Tools**: Ethereum (local test net with Hardhat).
- **Technology**: Solidity for smart contracts, Web3.py for Python integration.

Blockchain ensures tamper-proof logging of detected intrusions. Steps include:

1. Deploying smart contracts to log intrusion metadata (e.g., device ID, timestamp, anomaly type).
2. Storing hashes of detailed intrusion logs on-chain, with raw data kept off-chain (e.g., in IPFS).

4. **Real Time Alerting**

Automate Responses:

- **Tools**: Smart contract triggers, Flask APIs.
- **Technology**: Real-time notification systems.

This phase automates actions upon threat detection, such as:

- Blocking compromised devices.
- Sending alerts to administrators via email or dashboard notifications.

5. **System Development and Deployment**

Develop the System:

- **Tools**: Unity for interface design, Flask for backend services.
- **Technology**: Interactive dashboards, real-time monitoring.

The system integrates detection, logging, and alerting mechanisms into a cohesive dashboard accessible to administrators.

Hardware Deployment:

- Raspberry Pi setups for physical IoT environments.
- Simulated environments using GPIO simulators.

Figure 4.1: Shows division of work

6. Testing and Optimization

Performance and Usability Testing:

- **Tools**: Benchmarking software, feedback platforms.
- **Technology**: Frame rate analysis, synchronization testing.

Testing ensures:

1. High detection accuracy with minimal latency.
2. Synchronization between detection, logging, and alerting components.
3. Seamless user interaction with dashboards and notifications.

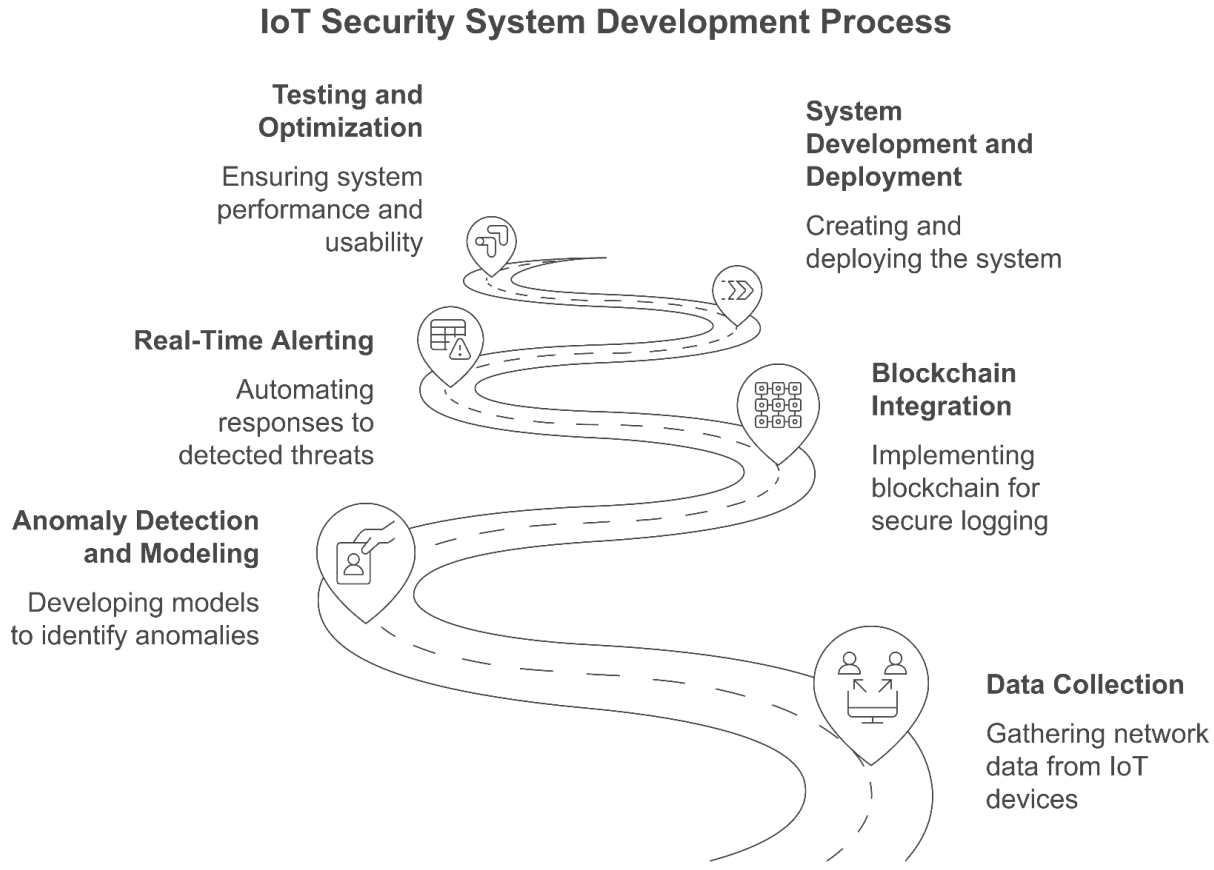


Figure 4.2: IoT Security Development Process

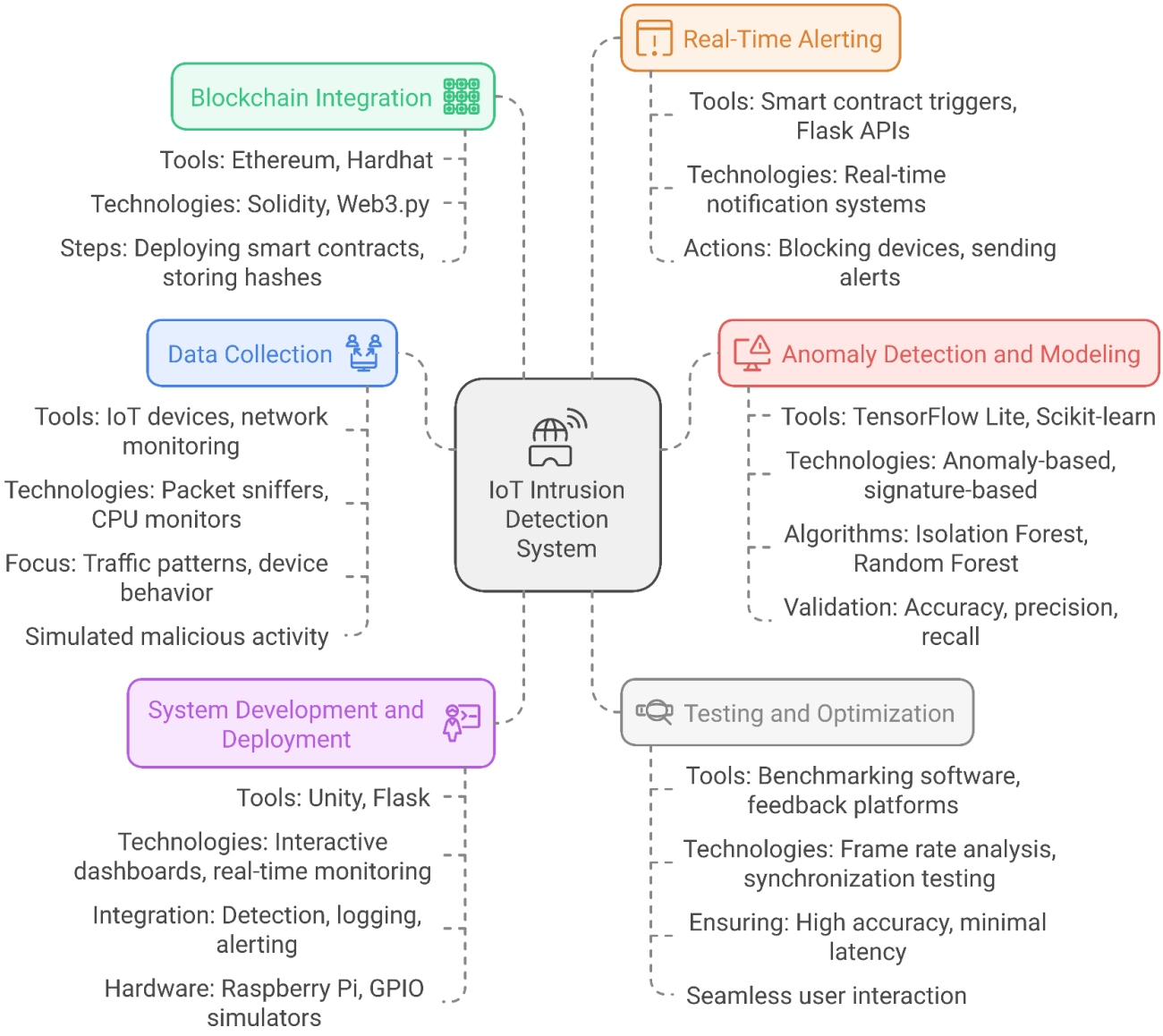


Figure 4.3: Tools and Technologies used in each step

CHAPTER 5
TOOLS AND TECHNOLOGY

The project integrates machine learning (ML), blockchain, cloud computing, and decentralized storage to enhance security in IoT environments.

1. Programming Languages

- **Python** – Used for implementing machine learning models and blockchain interaction (Web3.py).
- **Solidity** – Smart contract programming language for Ethereum blockchain deployment.
- **JavaScript (Node.js)** – Backend scripting and server-side functionalities.

Machine Learning & Data Processing Frameworks

- **TensorFlow Lite** – A lightweight ML framework for real-time anomaly detection in IoT devices.
- **Scikit-learn** – Used for developing and testing anomaly detection models (Isolation Forest, Random Forest).
- **Google Colab** – Cloud-based Python environment used for ML model training and testing.

Blockchain and Decentralized Storage

- **Ethereum (Testnet: Hardhat, Ganache)** – Used for deploying and testing smart contracts.
- **Web3.py** – Python library for blockchain interaction and smart contract execution.

Smart Contracts & Security

- **Solidity** – Used to write smart contracts that automate security responses.
- **MetaMask & Hardhat** – Tools for interacting with Ethereum Testnet and managing transactions.

Decentralized Storage

- **IPFS (InterPlanetary File System)** – A distributed storage system used to store detailed intrusion logs securely.

2. Intrusion Detection & Security Components

IDS & Anomaly Detection Models

- **Isolation Forest** – Detects unusual activity in network traffic by identifying outliers.
- **Random Forest** – Used for classifying network behaviors as normal or malicious.
- **Signature-based Detection** – Compares network patterns to a database of known cyber threats.

Real-Time Alerting & Response

- **Flask APIs** – Backend framework used for real-time alerts and notifications.
- **Smart Contracts** – Automate responses (e.g., isolating a compromised device) when a threat is detected.

3. IoT Hardware & Connectivity

IoT Devices

- **Raspberry Pi 5 / ESP32** – Small computing devices that collect network data and run lightweight ML models.
- **PIR Motion Sensors & Environmental Sensors (BME680)** – Used for real-time threat detection.

Communication Protocols

- **Wi-Fi (USB Wi-Fi Adapter for Raspberry Pi)** – Used for IoT connectivity.
- **MQTT (Message Queuing Telemetry Transport)** – Lightweight communication protocol for IoT networks.

4. Web & Dashboard Development

Frontend Technologies

- **React.js** – JavaScript framework for building the web-based dashboard.

- **Chart.js & D3.js** – Used for real-time data visualization of security events.

Backend Technologies

- **Flask (Python Web Framework)** – Manages API requests for intrusion logging and alerts.
- **MongoDB** – Database for logging intrusion events (if off-chain storage is required).

5. Testing & Deployment

Virtualization & Testing Environments

- **Docker** – Used to containerize and test the IDS system across different environments.
- **Hardhat (Ethereum Development Environment)** – Simulates blockchain transactions before deployment.

6. **Performance Monitoring Tools**

- **Wireshark** – Used to analyze network traffic for intrusion detection validation.
- **Multimeter (Fluke 101)** – Tests electrical connections in IoT hardware deployments.

5.1 TECHNOLOGICAL IMPLEMENTATION

The project leverages cutting-edge technologies and tools:

1. **Machine Learning**: TensorFlow Lite and Scikit-learn for lightweight, efficient detection models.
2. **Blockchain**: Ethereum (Hardhat for test net), Solidity for smart contracts.
3. **Frontend and Backend**: Flask for APIs, React.js for dashboards.
4. **Storage**: IPFS for off-chain raw log storage, blockchain for tamper-proof hashing.
5. **Hardware**: Raspberry Pi for physical deployment, GPIO simulators for development testing.

5.2 SIGNIFICANCE OF THE PROJECT

This project provides a transformative solution to IoT security by addressing challenges like centralized vulnerabilities and tampered intrusion logs. Key significance includes:

1. **Enhanced Security**: Blockchain ensures decentralized, tamper-proof intrusion detection and logging.
2. **Accessibility**: Scalable for various IoT applications, including smart homes, industrial setups, and healthcare systems.
3. **Resilience to Evolving Threats**: Machine learning models can adapt to detect emerging security threats.
4. **Global Impact**: By ensuring secure IoT environments, the system enhances the reliability of critical IoT applications in healthcare, transportation, and smart cities.

CHAPTER 6
DATA COLLECTION AND DATA STRUCTURE

Data Collection and Dataset Structure

The dataset, extracted from Zeek logs, captures network traffic details, including timestamps, IP addresses, port numbers, protocols, and connection states. The conn.log.labeled file was analyzed to understand network behavior and identify potential security threats.

Dataset Structure

The conn.log.labeled file was parsed using the header information provided by the Zeek log. The columns (fields) in the dataset include:

1. **ts**
    - *Type:* Timestamp (floating point number representing seconds since the epoch)
    - *Description:* When the connection started.
2. **uid**
    - *Type:* Unique identifier (string)
    - *Description:* A unique ID assigned to the connection flow.
3. **id.orig_h**
    - *Type:* IP address (string)
    - *Description:* Originating (source) host IP address.
4. **id.orig_p**
    - *Type:* Port (number)
    - *Description:* Source port number.
5. **id.resp_h**
    - *Type:* IP address (string)
    - *Description:* Responding (destination) host IP address.
6. **id.resp_p**
    - *Type:* Port (number)
    - *Description:* Destination port number.
7. **proto**
    - *Type:* Protocol (string)
    - *Description:* Protocol used (e.g., TCP, UDP).
8. **service**
    - *Type:* Service (string)

- *Description:* Application-level service (if detected), sometimes empty or a dash (-).

9. **duration**
    - *Type:* Duration (numeric)
    - *Description:* How long the connection lasted.
10. **orig_bytes**
    - *Type:* Numeric
    - *Description:* Number of bytes sent by the originator.
11. **resp_bytes**
    - *Type:* Numeric
    - *Description:* Number of bytes sent by the responder.
12. **conn_state**
    - *Type:* String
    - *Description:* Connection state (for example, “SF”, “S0”, etc., as defined by Zeek).
13. **local_orig**
    - *Type:* String (often a flag or indicator)
    - *Description:* Indicates if the originator is local.
14. **local_resp**
    - *Type:* String
    - *Description:* Indicates if the responder is local.
15. **missed_bytes**
    - *Type:* Numeric
    - *Description:* Number of bytes missed during capture.
16. **history**
    - *Type:* String
    - *Description:* A summary of the connection’s behavior (e.g., a sequence of letters indicating flag events).
17. **orig_pkts**
    - *Type:* Numeric
    - *Description:* Number of packets sent by the originator.
18. **orig_ip_bytes**
    - *Type:* Numeric
    - *Description:* Number of IP-level bytes sent by the originator.

19. **resp_pkts**
    - *Type:* Numeric
    - *Description:* Number of packets sent by the responder.
20. **resp_ip_bytes**
    - *Type:* Numeric
    - *Description:* Number of IP-level bytes sent by the responder.
21. **tunnel_parents**
    - *Type:* String
    - *Description:* Information on any tunneling parents, if applicable (often a dash -).
22. **label**
    - *Type:* String
    - *Description:* A label describing the connection (commonly "benign" or indicating a type of attack). In your current data, you mostly see the value "benign".
23. **detailed-label**
    - *Type:* May include additional details about the label (often empty or represented as a dash).

In addition to these columns, your preprocessing step added a new column:

24. **binary_label**
    - *Type:* Numeric (0 or 1)
    - *Description:* A binary representation of the label (e.g., 0 for benign and 1 for attack). In your current data, it appears that almost all (or all) entries are marked as benign (0).

Dataset Size (for 1 file we have 18 of these)

- **Number of Rows (Observations):**

452 This means the log file contained 452 individual connection flows.

- **Number of Columns:**

23 (as loaded from the file) Plus, after preprocessing, you have an additional column (binary_label), bringing the total to 24 columns in your final DataFrame.

Summary

- **Data Composition:**

The dataset contains detailed network flow information including timestamps, source

and destination IPs/ports, protocol, service, duration, byte counts, packet counts, and connection states. These fields are typical for intrusion detection as they capture both the behavior of the connection and network traffic characteristics.

- **Size:**

  In this particular scenario (CTU-Honeypot-Capture-4-1), the dataset has 452 rows and 23 fields (plus the added binary_label).

CHAPTER 7

EXPERIMENTAL SETUP:

SIMULATION AND METHODOLOGY

Real-time Network Security Monitoring System

1. System Architecture

Frontend

- **Technology:** React-based dashboard for real-time monitoring.
- **Features:** Provides a user-friendly interface to visualize and interact with network data.

Real-time Communication

- **Protocol:** MQTT (Message Queuing Telemetry Transport).
- **Purpose:** Enables lightweight, efficient messaging for IoT device data.

Data Visualization

- **Tool:** Chart.js.
- **Functionality:** Visualizes network activity and anomalies.

State Management

- **Technology:** React hooks.
- **Role:** Manages local state efficiently for real-time updates.

2. Attack Detection System

Real-time Monitoring

- Continuously tracks network traffic.

Detection Mechanisms

- **Threshold-based Detection:** Identifies when traffic exceeds predefined levels.
- **Pattern Recognition:** Detects anomalies using behavioural patterns.

Feedback

- Provides immediate visual feedback to the user for detected anomalies.

3. Tech Stack Breakdown

Frontend Framework

- **React 18.3.1:** Modern library for building responsive UI.

- **TypeScript:** Ensures type safety and better code maintainability.

UI Components

- **Lucide React:** Modern icon library for intuitive UI.
- **Tailwind CSS:** Utility-first framework for styling.

Data Visualization

- **Chart.js:** Advanced charting capabilities.
- **react-chartjs-2:** React wrapper for Chart.js.

Real-time Communication

- **Protocol:** MQTT.
- **Library:** mqtt.js for efficient, lightweight messaging.

4. Implementation Steps

Project Setup

```
# Initial setup
npm create vite@latest

# Install dependencies
npm install react-chartjs-2 chart.js mqtt lucide-react
```

Component Structure

```
src/
  components/
    DataChart.tsx      # Network activity visualization
    SecurityStats.tsx  # Security metrics display
    AnomalyList.tsx    # Attack notifications
  hooks/
    useMQTT.ts         # MQTT connection management
  types/
    mqtt.d.ts          # TypeScript definitions
```

5. Key Concepts Explained

Rate and Rate Limiting

- **Rate:** Frequency of requests from a device.
- **Rate Limiting:** Restricts the frequency of requests to prevent misuse.

6. Attack Types and Detection

Threshold Breach

- Detected when value > 100 (indicating abnormally high traffic).

DDoS (Distributed Denial of Service)

- Sustained high request volume where value > 70.

Unauthorized Access

- Identifies suspicious patterns when value > 50.

7. Prevention Measures Status Types

```
type Status =
  | 'blocked'      // Complete access denial
  | 'rate-limited' // Restrict request frequency
  | 'monitoring'   // Enhanced observation
  | 'normal';      // No threats detected
```

Severity Levels

```
type Severity =
  | 'low'    // Minor anomalies
  | 'medium' // Suspicious patterns
  | 'high';  // Critical threats
```

8. System Features

- **Real-time Attack Detection:** Continuous monitoring of network traffic.
- **Visual Attack Patterns:** Displays anomalies through graphs and charts.
- **Immediate Threat Response:** Provides visual and actionable feedback for anomalies.
- **Historical Data Analysis:** Tracks past attacks for trend analysis.
- **Preventive Measures:** Implements and tracks countermeasures to reduce threats.

Conclusion

This real-time network security monitoring system leverages modern technologies to provide robust protection against cyber threats. Its modular architecture, efficient communication protocol, and comprehensive visualization capabilities ensure a seamless and secure user experience.

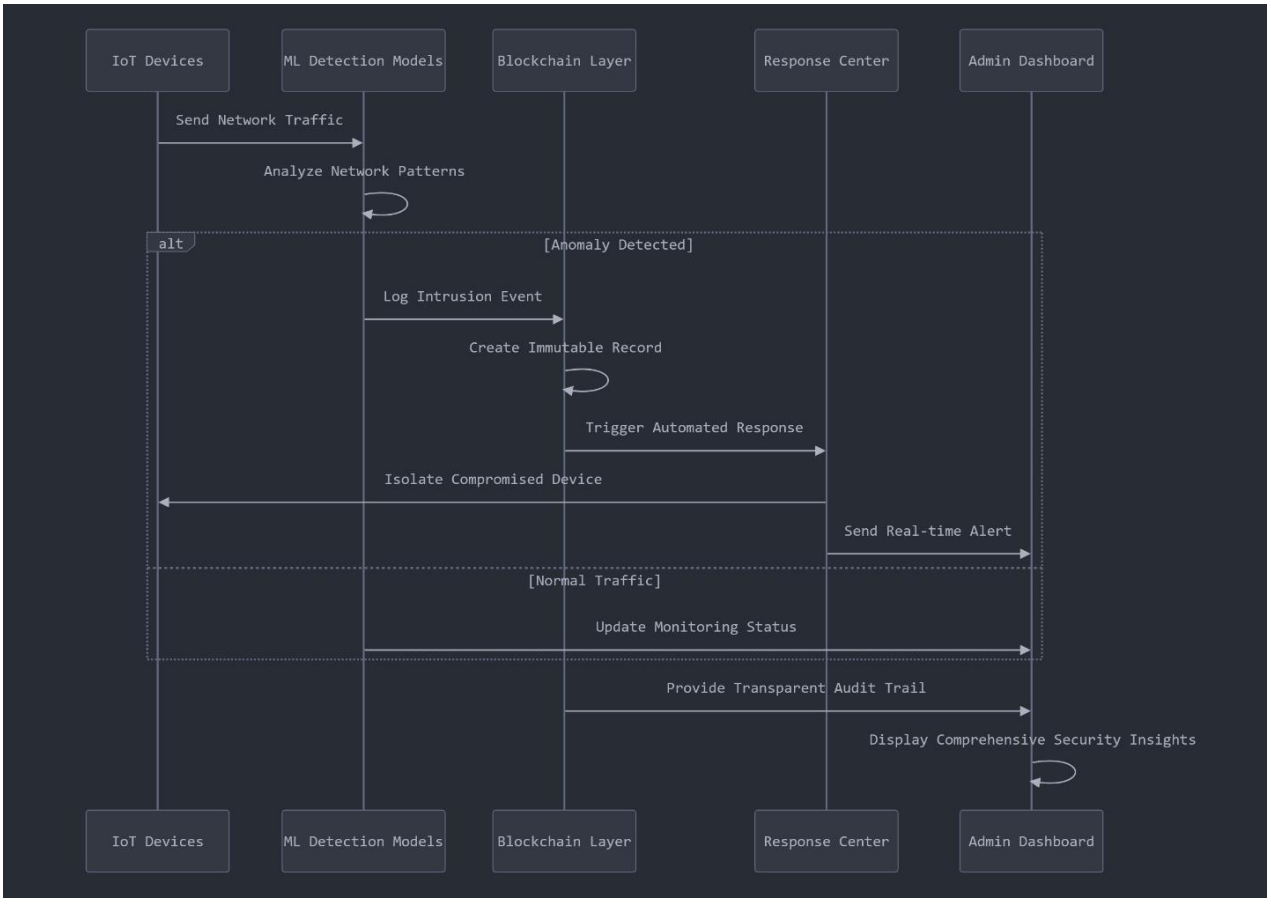


Figure 7.1 Flowchart of the Framework

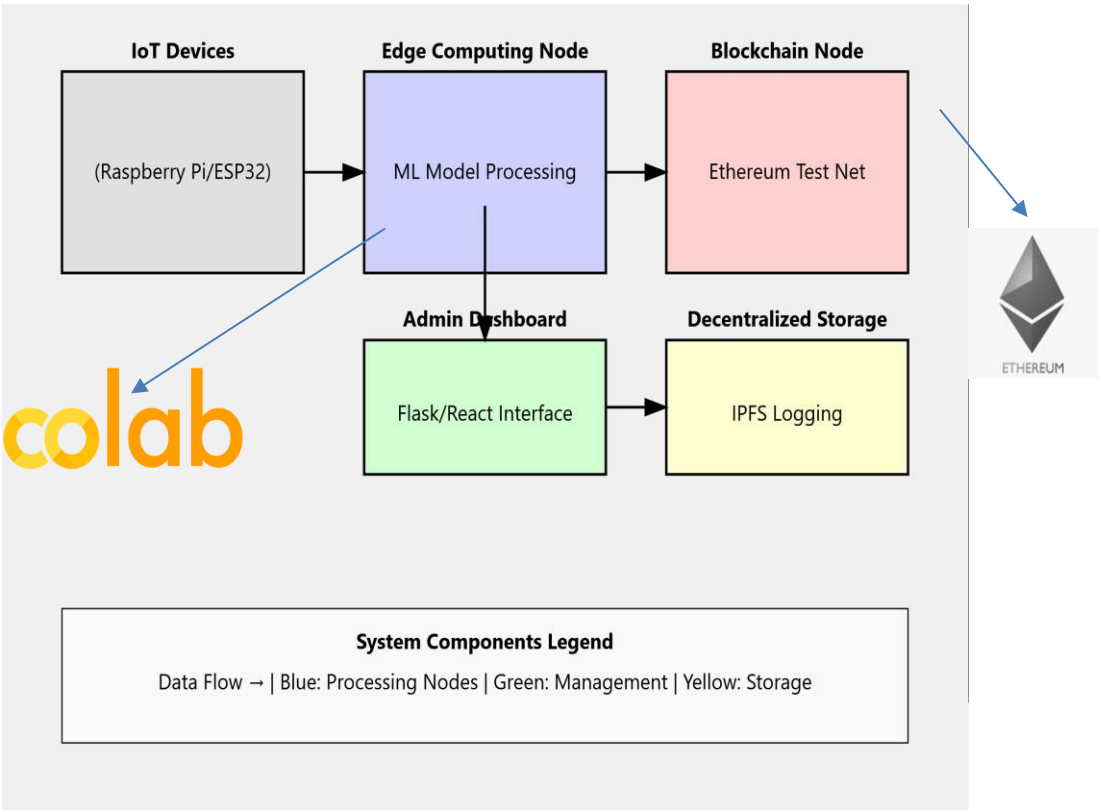


Figure 7.2 Working of the Experimental Setup

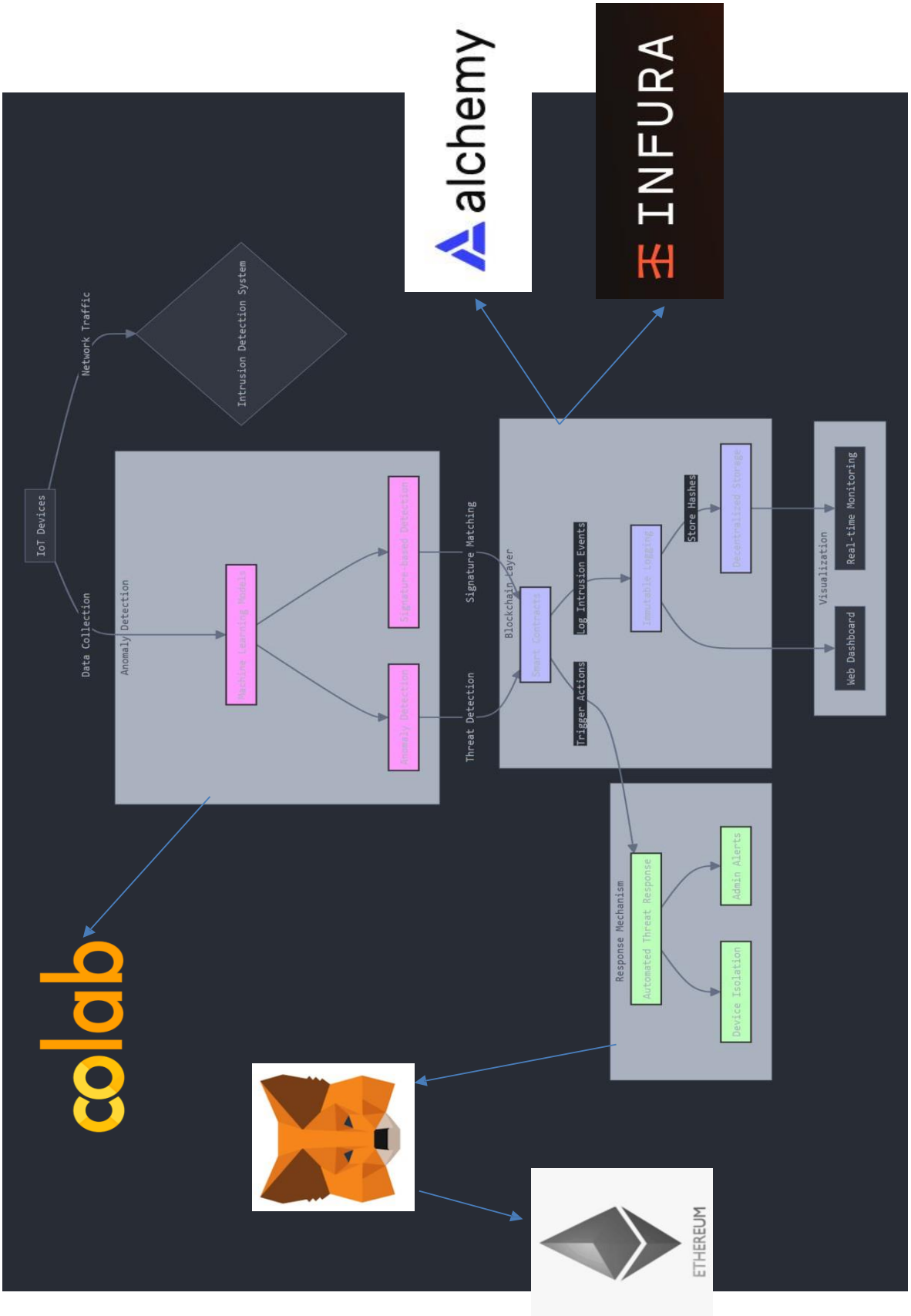


Figure 7.3: Experimental Setup and Process involved

Figure 7.4: Blockchain and Smart Contracts

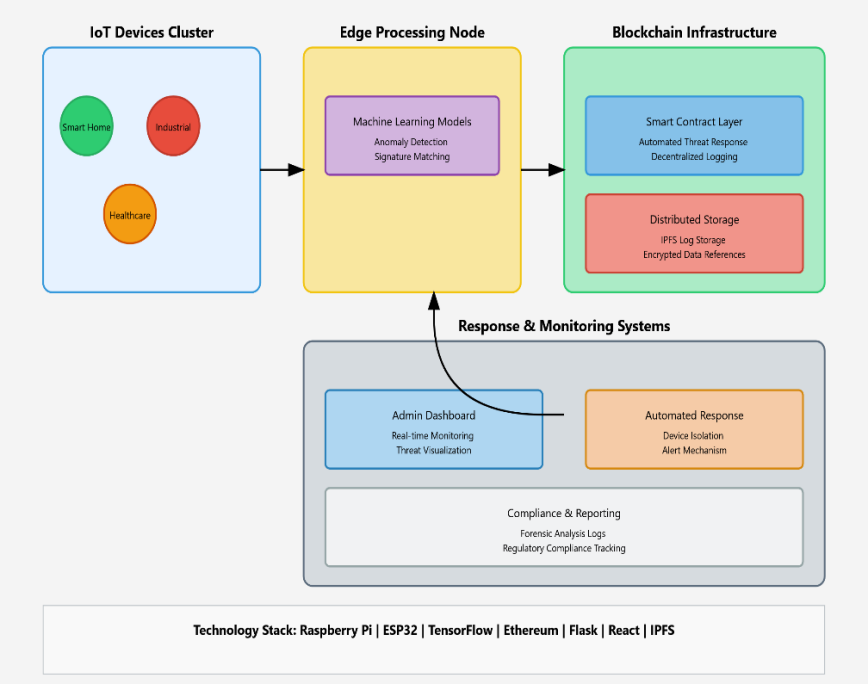


Figure 7.5: Design of the Framework

Procedure:

1. **Data Collection:** Deploy sensors to capture real-time data.
2. **Machine Learning:** Train and run intrusion detection models on the Raspberry Pi.
3. **Blockchain Integration:** Record detected anomalies with smart contracts on the blockchain.
4. **Automation:** Trigger responses like alerts via Flask APIs when an anomaly is detected.
5. **Testing:** Optimize system performance with simulated attacks and debugging tools like multimeters.

BACKEND CODE AND ALGORITHMS :

```
# %% [markdown]
# # IoT Intrusion Detection System with Blockchain Logging Across Multiple Scenarios
#
# This notebook demonstrates:
# 1. Extracting the IoT-23 dataset from a tar.gz file.
# 2. Looping over multiple scenario directories (e.g., CTU-Honeypot and CTU-Iot-Malware ca
#    and loading each Zeek connection log file (`conn.log.labeled`).
# 3. Concatenating all logs into one DataFrame.
# 4. Preprocessing the dataset.
# 5. Training and testing a Random Forest classifier and an Isolation Forest for anomaly d
# 6. Logging a sample prediction as a new block in a simple blockchain ledger.
#
# **Note:**
# Adjust file paths, scenario directory names, and feature names based on your actual data

# %% [code]
import tarfile
import os
import pandas as pd
import numpy as np
```

```
from datetime import datetime
import hashlib
import json

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier, IsolationForest
from sklearn.metrics import classification_report, roc_auc_score
import joblib

# %% [markdown]
# ## Step 1: Extract the Dataset from the tar.gz Archive
#
# Update the file path to your tar.gz archive.

# %% [code]
# Path to your tar.gz file (update this path as needed)
tar_path = r"C:\Users\mahaj\Downloads\iot_23_datasets_full.tar.gz"
extract_dir = "iot23_extracted"
```

```
if not os.path.exists(extract_dir):
    os.makedirs(extract_dir)

with tarfile.open(tar_path, "r:gz") as tar:
    tar.extractall(path=extract_dir)

print(f"Dataset extracted to: {extract_dir}")

# %% [markdown]
# ## Step 2: Define the Scenario Directories and Load All conn.log.labeled Files
#
# We define a list of scenario directory names that contain the `conn.log.labeled` file.
# Each file is located at:
#
#    iot23_extracted/opt/Malware-Project/BigDataset/IotScenarios/<SCENARIO_DIR>/bro/conn.l
#
# We then loop over the list, load each file (using its header information from the "#fiel
# and concatenate all data into one DataFrame.

# %% [code]
```

```
# List of scenario directories to process
scenario_dirs = [
    "CTU-Honeypot-Capture-4-1",
    "CTU-Honeypot-Capture-5-1",
    "CTU-Honeypot-Capture-7-1",
    "CTU-Iot-Malware-Capture-1-1",
    "CTU-Iot-Malware-Capture-3-1",
    "CTU-Iot-Malware-Capture-7-1",
    "CTU-Iot-Malware-Capture-8-1",
    "CTU-Iot-Malware-Capture-9-1",
    "CTU-Iot-Malware-Capture-17-1",
    "CTU-Iot-Malware-Capture-20-1",
    "CTU-Iot-Malware-Capture-21-1",
    "CTU-Iot-Malware-Capture-33-1",
    "CTU-Iot-Malware-Capture-34-1",
    "CTU-Iot-Malware-Capture-35-1",
    "CTU-Iot-Malware-Capture-36-1",
    "CTU-Iot-Malware-Capture-39-1",
    "CTU-Iot-Malware-Capture-42-1",
    "CTU-Iot-Malware-Capture-43-1",
```

```
    "CTU-Iot-Malware-Capture-44-1",
    "CTU-Iot-Malware-Capture-48-1",
    "CTU-Iot-Malware-Capture-49-1",
    "CTU-Iot-Malware-Capture-52-1",
    "CTU-Iot-Malware-Capture-60-1"
]

# Base directory for scenarios
base_dir = os.path.join(extract_dir, "opt", "Malware-Project", "BigDataset", "IotScenarios

# Container for DataFrames
df_list = []

for scenario in scenario_dirs:
    conn_log_path = os.path.join(base_dir, scenario, "bro", "conn.log.labeled")
    if os.path.exists(conn_log_path):
        print(f"Processing scenario: {scenario}")
        # Read header from the file (using the "#fields" line)
        header_cols = None
```

```
        with open(conn_log_path, 'r') as f:
            for line in f:
                if line.startswith("#fields"):
                    header_cols = line.strip().split()[1:]
                    break

        if header_cols is None:
            print(f"Warning: Could not find header in {scenario}. Skipping.")
            continue

        # Read the file with the header columns
        try:
            df_temp = pd.read_csv(conn_log_path, sep='\t', comment='#', names=header_cols)
            # Optionally, add a column for scenario identification:
            df_temp["scenario"] = scenario
            df_list.append(df_temp)
        except Exception as e:
            print(f"Error reading {scenario}: {e}")
    else:
        print(f"File not found for scenario: {scenario}")
```

```
if not df_list:
    raise ValueError("No data files were loaded. Check your scenario directory list and f

# Combine all DataFrames
df = pd.concat(df_list, ignore_index=True)
print("Combined connection log shape:", df.shape)
print(df.head())

# %% [markdown]
# ## Step 3: Preprocess the Dataset
#
# In this step, we:
# - Process the label column safely, handling missing values.
# - Convert selected feature columns to numeric values.
# - Select a subset of numeric features for demonstration.
#
# **Note:** Adjust the feature list based on the actual header of your logs.

# %% [code]
```

```
# Process the label column safely, handling missing values.
if 'label' in df.columns:
    df['binary_label'] = df['label'].apply(
        lambda x: 0 if pd.isna(x) or (isinstance(x, str) and x.strip().lower() == 'benign'
    )
else:
    # For demonstration, randomly assign binary labels.
    np.random.seed(42)
    df['binary_label'] = np.random.choice([0, 1], size=df.shape[0], p=[0.8, 0.2])
    print("No 'label' column found. Random labels assigned for demonstration.")

# Define a list of features; update these names as needed.
selected_features = ['duration', 'orig_bytes', 'resp_bytes', 'orig_pkts', 'resp_pkts']

# Check if these features exist
print("Available columns:", df.columns.tolist())
for feat in selected_features:
    if feat not in df.columns:
        print(f"Feature '{feat}' not found; please adjust the feature list accordingly.")
```

```
# Convert selected feature columns to numeric values (coerce errors to NaN)
for feat in selected_features:
    df[feat] = pd.to_numeric(df[feat], errors='coerce')

# Filter DataFrame to include the selected features and fill missing values with the mean.
try:
    df_features = df[selected_features].copy().fillna(df[selected_features].mean())
except KeyError:
    raise KeyError("One or more selected features are not present in the DataFrame. Update

df_labels = df['binary_label']

# %% [markdown]
# ## Step 4: Split Data and Scale Features

# %% [code]
X_train, X_test, y_train, y_test = train_test_split(df_features, df_labels, test_size=0.2,
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# %% [markdown]
# ## Step 5: Train the Models
#
# - **Random Forest (Supervised):** Trained on all data with binary labels.
# - **Isolation Forest (Unsupervised):** Trained only on benign flows (label 0) to model n

# %% [code]
# Train Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_scaled, y_train)
y_pred_rf = rf_model.predict(X_test_scaled)
print("=== Random Forest Classification Report ===")
print(classification_report(y_test, y_pred_rf))

# Check if the model has more than one class to use predict_proba
if len(rf_model.classes_) > 1:
    y_proba_rf = rf_model.predict_proba(X_test_scaled)[:, 1]
else:
    y_proba_rf = np.zeros(len(y_test))
```

```
    print("Only one class present in training; setting predicted probabilities to 0.")

roc_auc_rf = roc_auc_score(y_test, y_proba_rf)
print("Random Forest ROC AUC: {:.3f}".format(roc_auc_rf))

# Train Isolation Forest on benign samples only (label 0)
X_train_normal = X_train_scaled[y_train == 0]
iso_model = IsolationForest(n_estimators=100, contamination=0.05, random_state=42)
iso_model.fit(X_train_normal)
iso_pred = iso_model.predict(X_test_scaled)
# In Isolation Forest, predictions are 1 for inliers and -1 for anomalies.
iso_pred_binary = np.where(iso_pred == 1, 0, 1)
print("=== Isolation Forest Classification Report ===")
print(classification_report(y_test, iso_pred_binary))

# Save models, scaler, and feature list for later use
joblib.dump(rf_model, 'rf_model.pkl')
joblib.dump(iso_model, 'iso_model.pkl')
joblib.dump(scaler, 'scaler.pkl')
joblib.dump(selected_features, 'feature_cols.pkl')
```

```
print("Training complete and models saved.")

# %% [markdown]
# ## Step 6: Implement a Simple Blockchain for Logging Predictions
#
# Define classes for `Block` and `Blockchain` to create an immutable ledger of prediction

# %% [code]
class Block:
    def __init__(self, index, timestamp, data, previous_hash):
        self.index = index
        self.timestamp = timestamp
        self.data = data
        self.previous_hash = previous_hash
        self.hash = self.compute_hash()

    def compute_hash(self):
        block_string = json.dumps(self.__dict__, sort_keys=True, default=str)
        return hashlib.sha256(block_string.encode()).hexdigest()
```

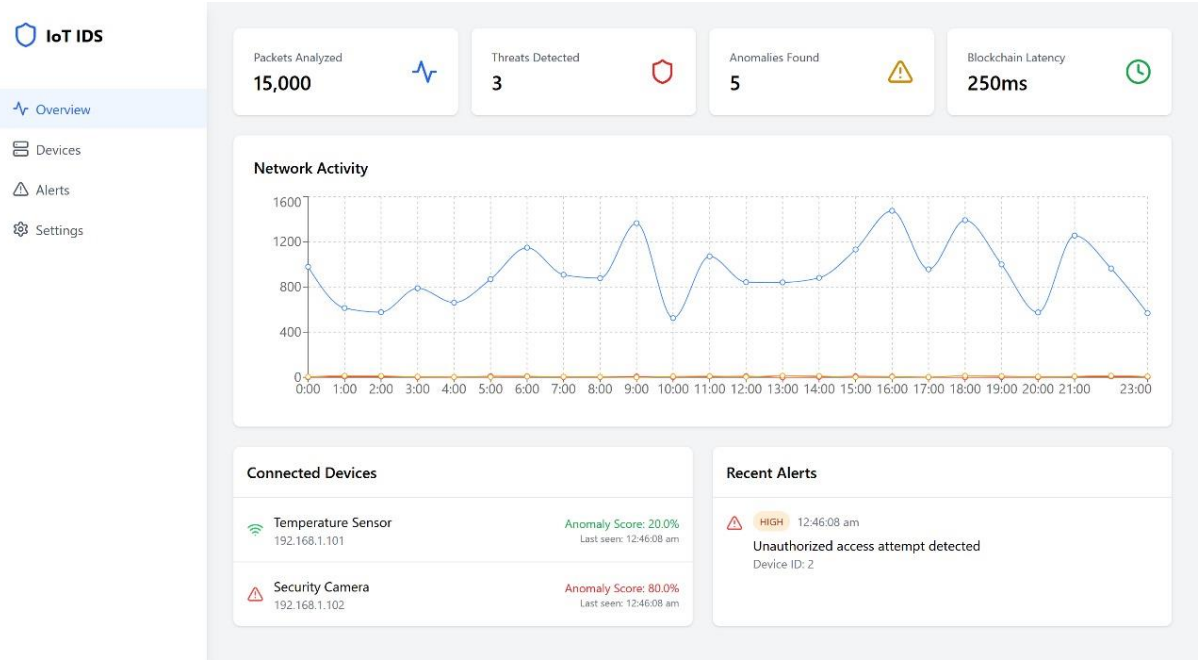
```
class Blockchain:
    def __init__(self):
        self.chain = []
        self.create_genesis_block()

    def create_genesis_block(self):
        genesis_block = Block(0, str(datetime.utcnow()), {"info": "Genesis Block"}, "0")
        self.chain.append(genesis_block)

    def get_last_block(self):
        return self.chain[-1]

    def add_block(self, data):
        last_block = self.get_last_block()
        new_block = Block(last_block.index + 1, str(datetime.utcnow()), data, last_block.h
        self.chain.append(new_block)
        return new_block

    def get_chain(self):
        return [block.__dict__ for block in self.chain]
```

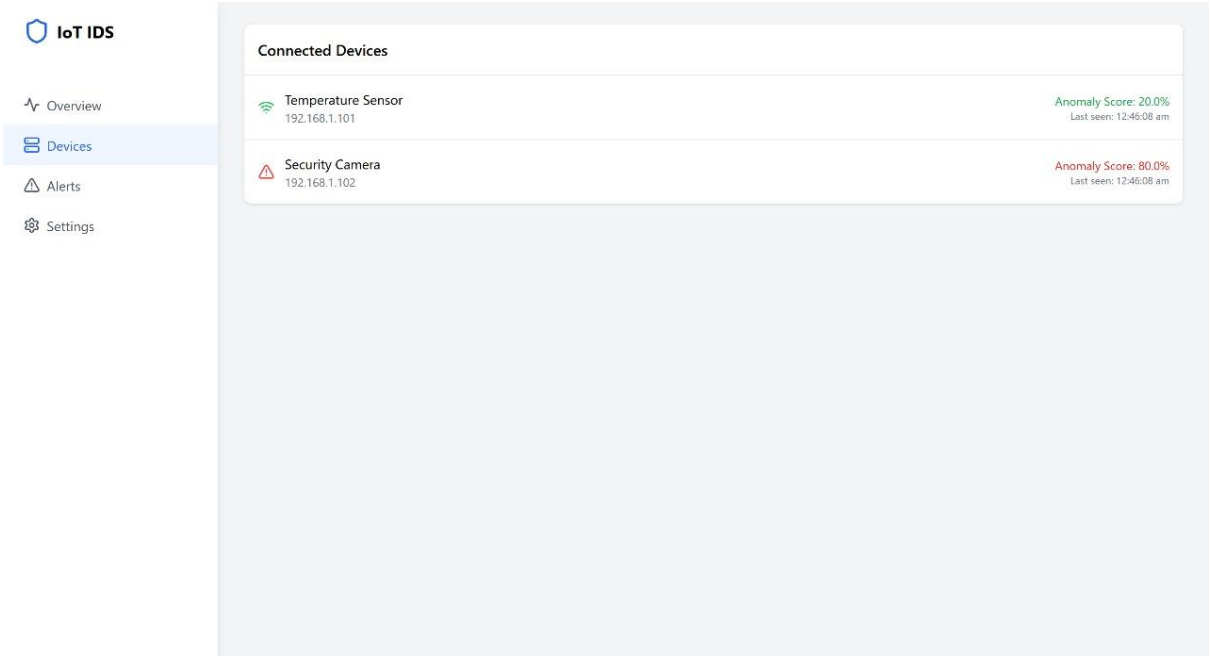
```
# Initialize blockchain
blockchain = Blockchain()

# %% [markdown]
# ## Step 7: Simulate a Prediction and Log It to the Blockchain
#
# Simulate new network traffic data (in production, real-time observations would be used),

# %% [code]
new_observation = {
    "duration": 0.5,         # seconds
    "orig_bytes": 500,       # bytes
    "resp_bytes": 300,       # bytes
    "orig_pkts": 10,         # packet count
    "resp_pkts": 8           # packet count
}
```

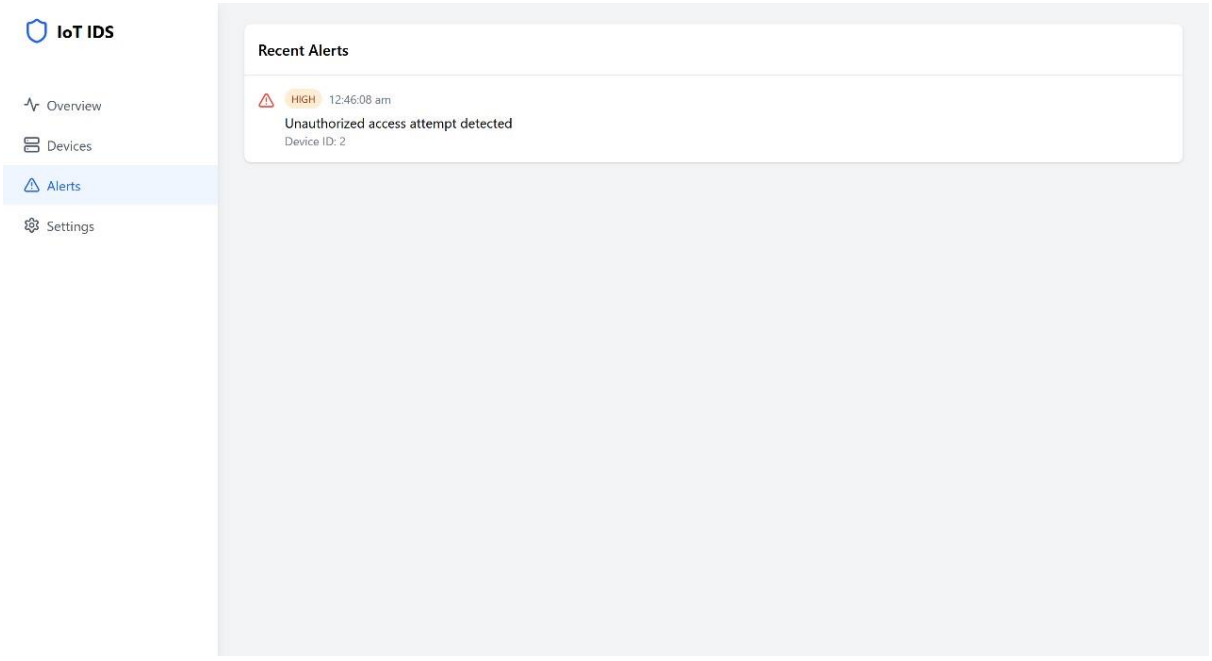
```
new_data = pd.DataFrame([new_observation])
new_data = new_data[selected_features].fillna(new_data[selected_features].mean())
new_data_scaled = scaler.transform(new_data)

# Get predictions
rf_prediction = rf_model.predict(new_data_scaled)
if len(rf_model.classes_) > 1:
    rf_probability = rf_model.predict_proba(new_data_scaled)[:, 1]
else:
    rf_probability = [0]
iso_pred = iso_model.predict(new_data_scaled)
iso_prediction = np.where(iso_pred == 1, 0, 1)

prediction_result = {
    "rf_prediction": rf_prediction.tolist(),
    "rf_probability": rf_probability if isinstance(rf_probability, list) else rf_probabili
    "iso_prediction": iso_prediction.tolist()
}
```

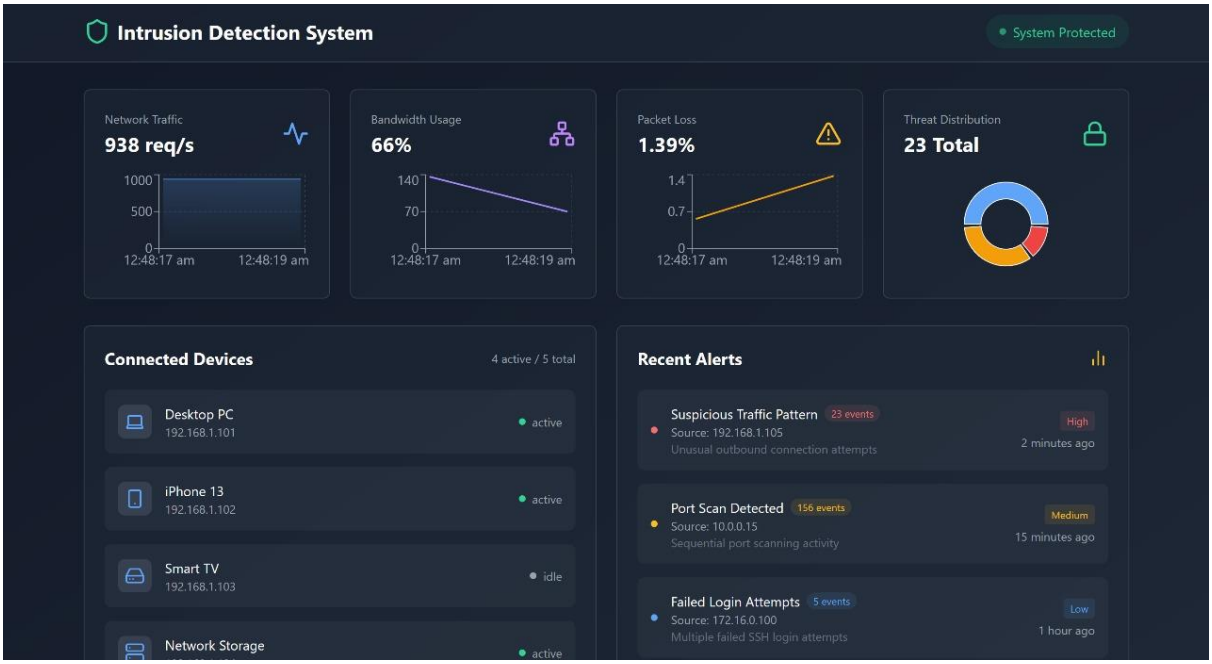
```
# Log prediction in blockchain
block_data = {
    "input": new_data.to_dict(orient="records"),
    "predictions": prediction_result
}
new_block = blockchain.add_block(block_data)
print("New block added to blockchain:")
print(new_block.__dict__)

# %% [markdown]
# ## Step 8: (Optional) View the Blockchain Ledger
#
# Print out the full blockchain ledger.

# %% [code]
chain = blockchain.get_chain()
print("Blockchain Ledger:")
for block in chain:
    print(block)
```

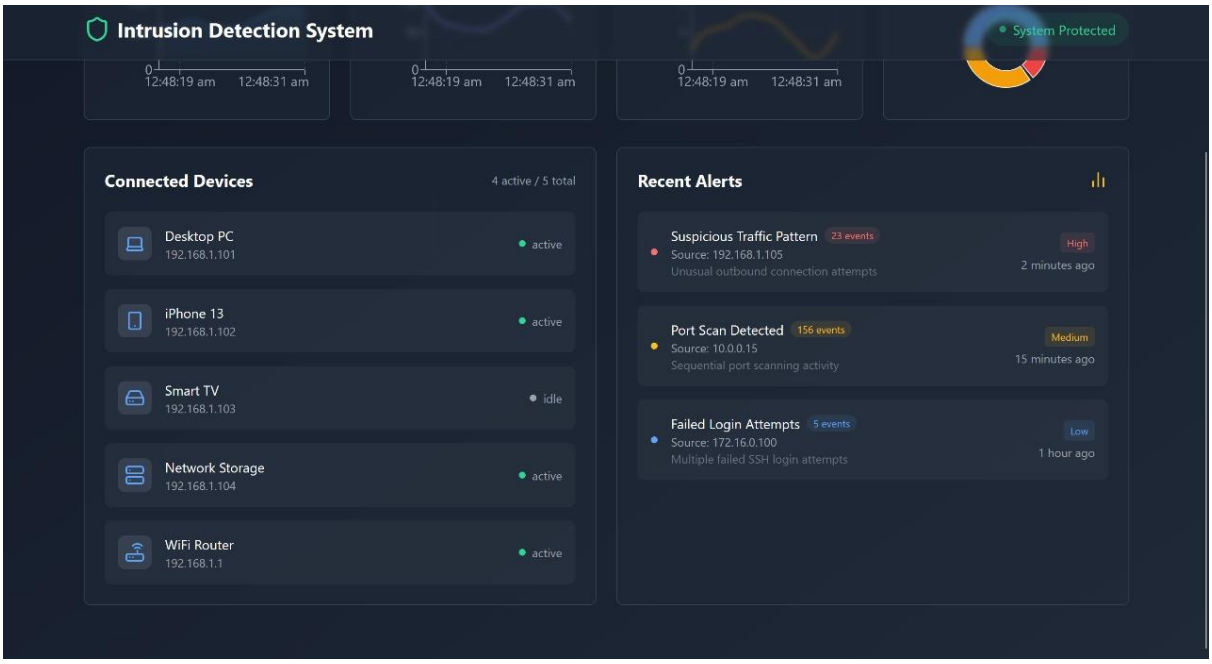
Dashboards:

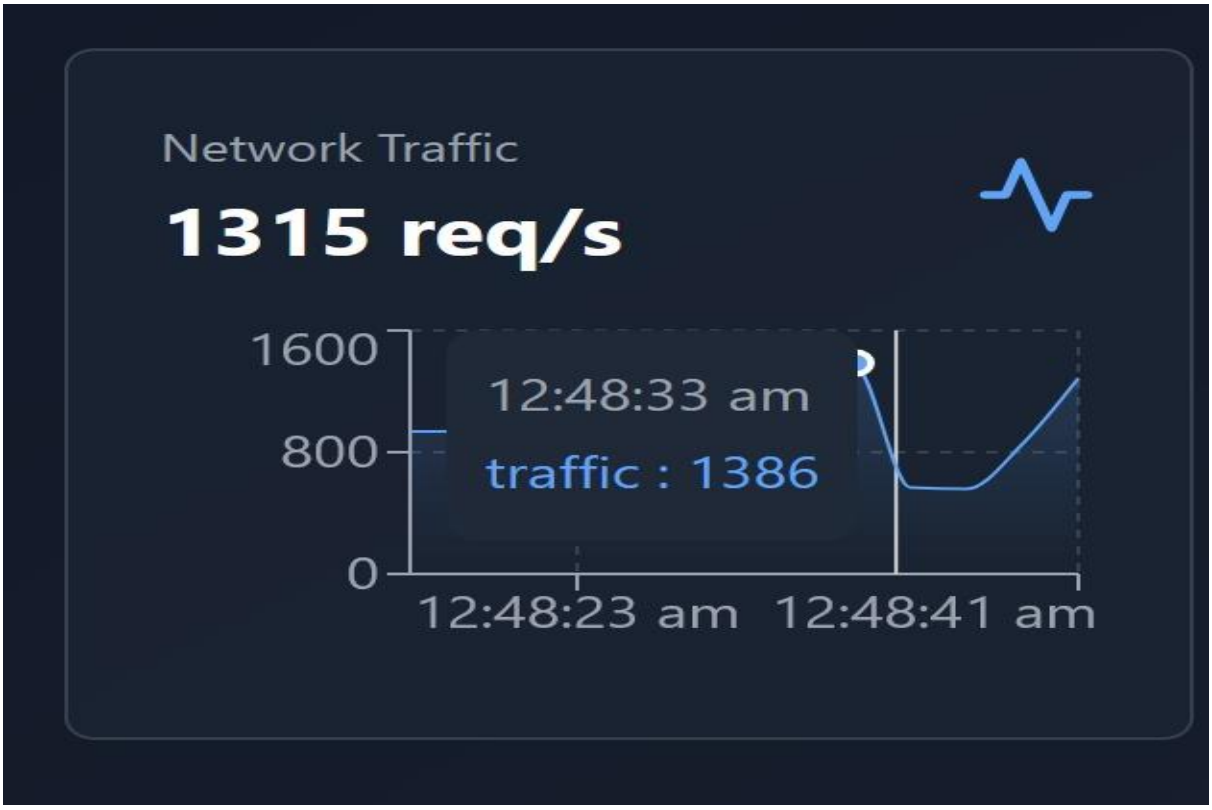


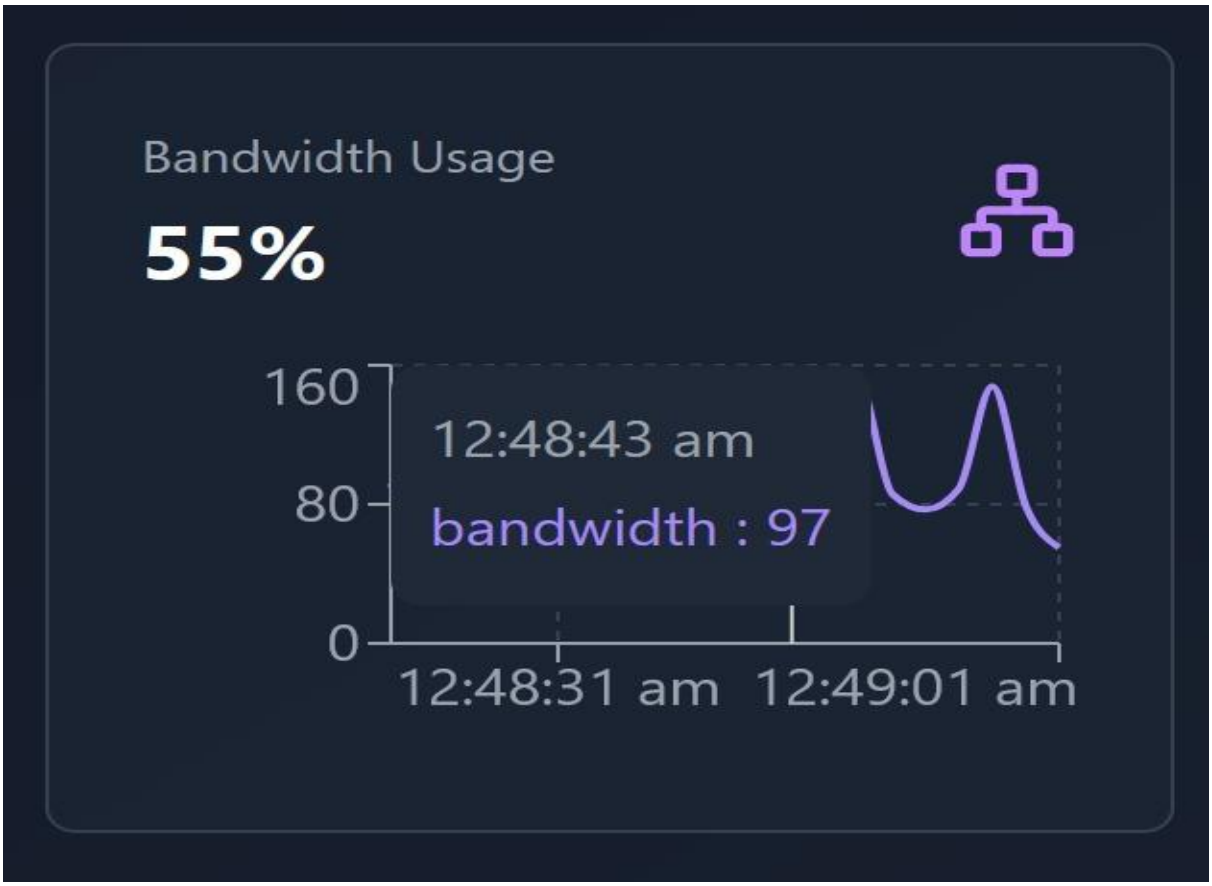

IoT IDS
Overview
Devices
Alerts
Settings
Connected Devices
Temperature Sensor
192.168.1.101
Anomaly Score: 20.0%
Last seen: 12:46:08 am
Security Camera
192.168.1.102
Anomaly Score: 80.0%
Last seen: 12:46:08 am

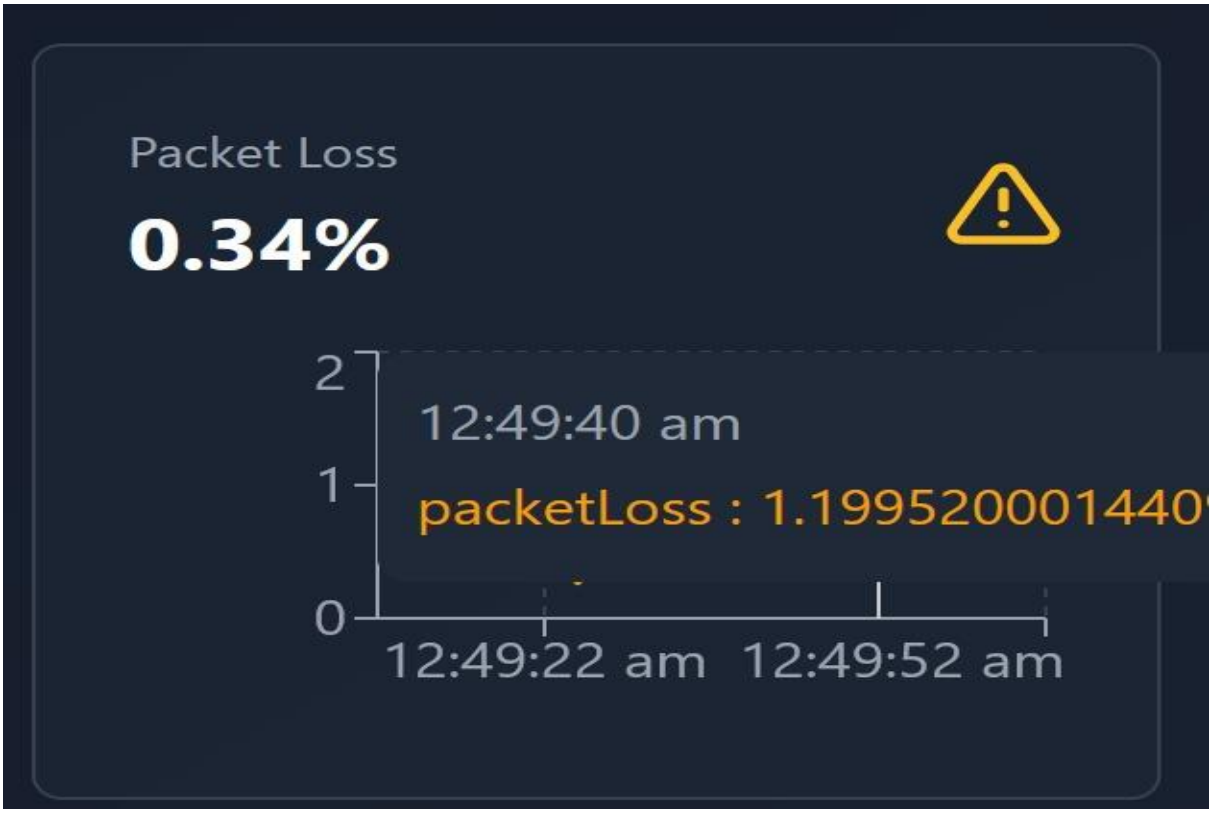

IoT IDS
Overview
Devices
Alerts
Settings
Recent Alerts
HIGH
12:46:08 am
Unauthorized access attempt detected
Device ID: 2


Intrusion Detection System
System Protected
Network Traffic
938 req/s
1000
500
0
12:48:17 am
12:48:19 am
Bandwidth Usage
66%
140
70
0
12:48:17 am
12:48:19 am
Packet Loss
1.39%
1.4
0.7
0
12:48:17 am
12:48:19 am
Threat Distribution
23 Total
Connected Devices
4 active / 5 total
Desktop PC
192.168.1.101
active
iPhone 13
192.168.1.102
active
Smart TV
192.168.1.103
idle
Network Storage
active
Recent Alerts
Suspicious Traffic Pattern
23 events
Source: 192.168.1.105
Unusual outbound connection attempts
High
2 minutes ago
Port Scan Detected
156 events
Source: 10.0.0.15
Sequential port scanning activity
Medium
15 minutes ago
Failed Login Attempts
5 events
Source: 172.16.0.100
Multiple failed SSH login attempts
Low
1 hour ago

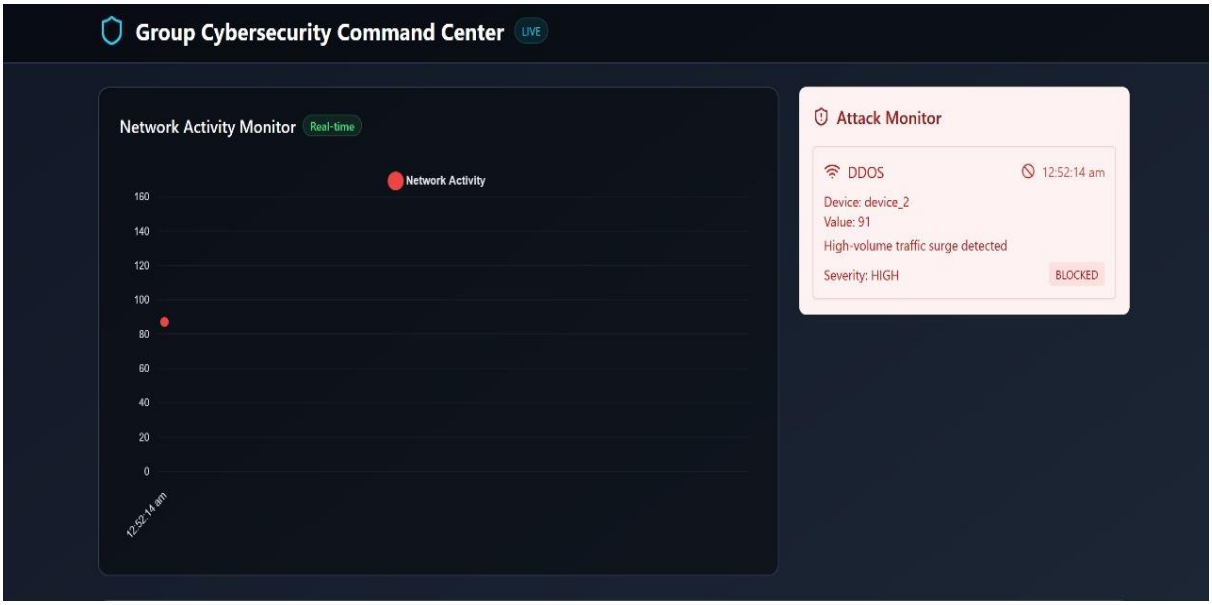

Intrusion Detection System
System Protected
0
12:48:19 am
12:48:31 am
Connected Devices
4 active / 5 total
Desktop PC
192.168.1.101
active
iPhone 13
192.168.1.102
active
Smart TV
192.168.1.103
idle
Network Storage
192.168.1.104
active
WiFi Router
192.168.1.1
active
Recent Alerts
Suspicious Traffic Pattern
23 events
Source: 192.168.1.105
Unusual outbound connection attempts
High
2 minutes ago
Port Scan Detected
156 events
Source: 10.0.0.15
Sequential port scanning activity
Medium
15 minutes ago
Failed Login Attempts
5 events
Source: 172.16.0.100
Multiple failed SSH login attempts
Low
1 hour ago

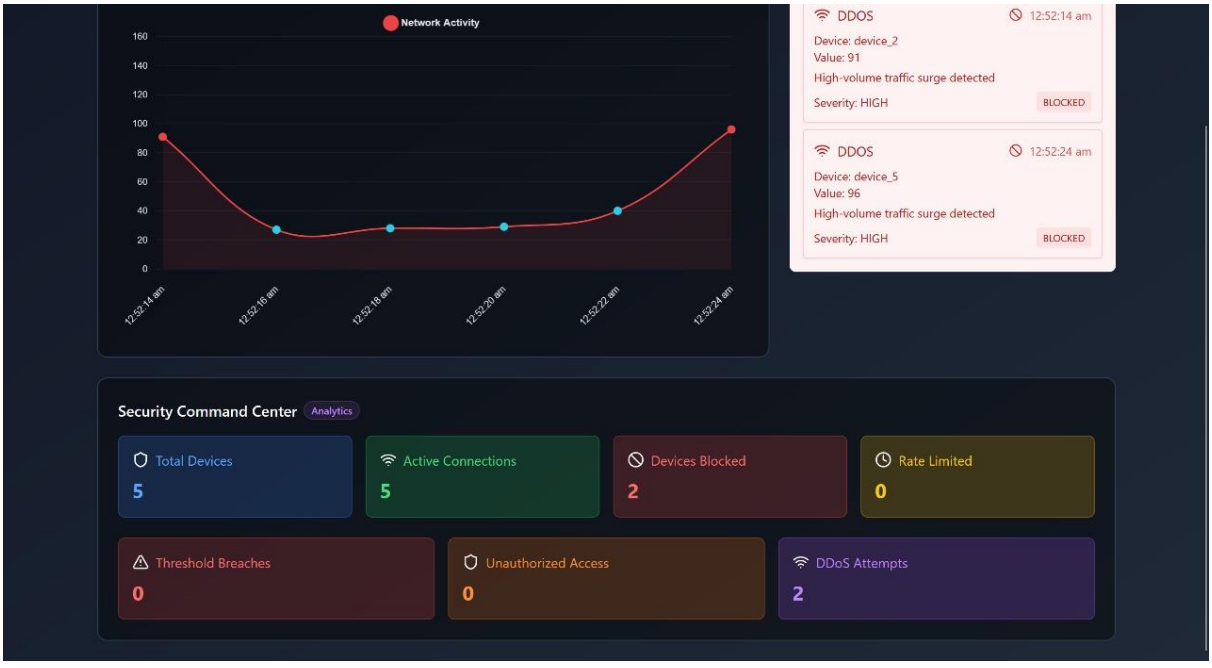

Network Traffic
1315 req/s
1600
800
0
12:48:33 am
traffic : 1386
12:48:23 am
12:48:41 am

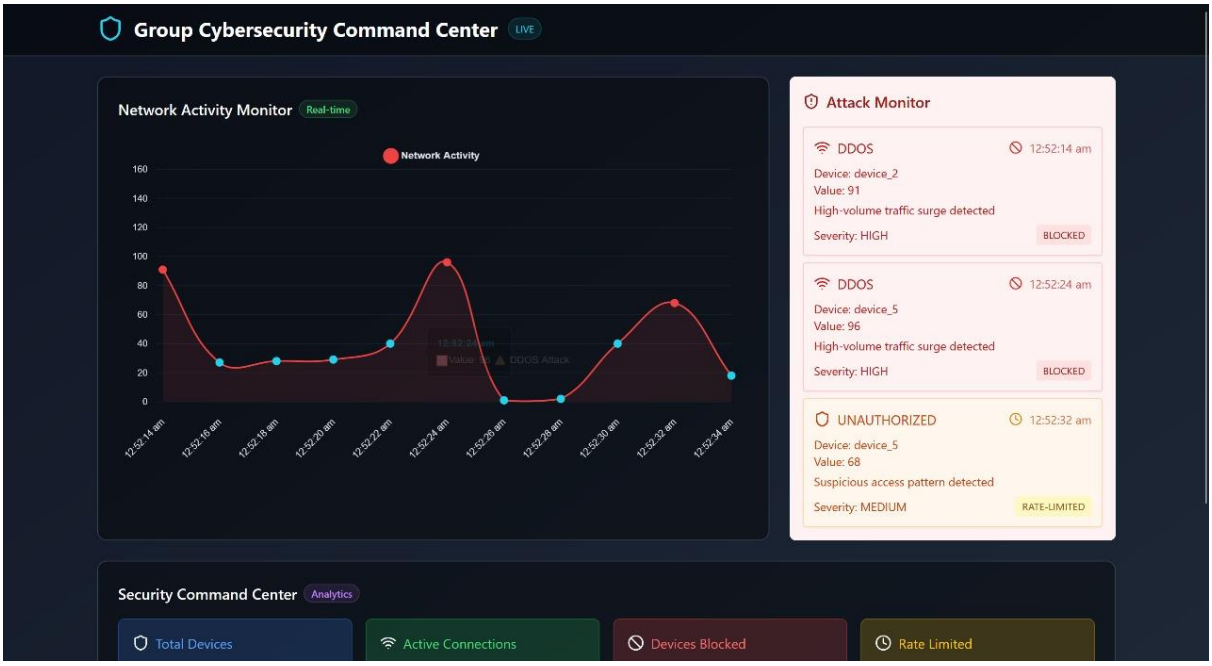

Bandwidth Usage
55%
160
80
0
12:48:43 am
bandwidth : 97
12:48:31 am
12:49:01 am


Packet Loss
0.34%
2
1
0
12:49:40 am
packetLoss : 1.199520001440
12:49:22 am
12:49:52 am


Threat Distribution
23 Total
Medium : 8


Group Cybersecurity Command Center
LIVE
Network Activity Monitor
Real-time
Network Activity
160
140
120
100
80
60
40
20
0
12:52:14 am
Attack Monitor
DDOS
12:52:14 am
Device: device_2
Value: 91
High-volume traffic surge detected
Severity: HIGH
BLOCKED


Network Activity
160
140
120
100
80
60
40
20
0
12:52:14 am
12:52:16 am
12:52:18 am
12:52:20 am
12:52:22 am
12:52:24 am
DDOS
12:52:14 am
Device: device_2
Value: 91
High-volume traffic surge detected
Severity: HIGH
BLOCKED
DDOS
12:52:24 am
Device: device_5
Value: 96
High-volume traffic surge detected
Severity: HIGH
BLOCKED
Security Command Center
Analytics
Total Devices
5
Active Connections
5
Devices Blocked
2
Rate Limited
0
Threshold Breaches
0
Unauthorized Access
0
DDoS Attempts
2


Group Cybersecurity Command Center
LIVE
Network Activity Monitor
Real-time
Network Activity
160
140
120
100
80
60
40
20
0
12:52:14 am
12:52:16 am
12:52:18 am
12:52:20 am
12:52:22 am
12:52:24 am
12:52:26 am
12:52:28 am
12:52:30 am
12:52:32 am
12:52:34 am
Attack Monitor
DDOS
12:52:14 am
Device: device_2
Value: 91
High-volume traffic surge detected
Severity: HIGH
BLOCKED
DDOS
12:52:24 am
Device: device_5
Value: 96
High-volume traffic surge detected
Severity: HIGH
BLOCKED
UNAUTHORIZED
12:52:32 am
Device: device_5
Value: 68
Suspicious access pattern detected
Severity: MEDIUM
RATE-LIMITED
Security Command Center
Analytics
Total Devices
Active Connections
Devices Blocked
Rate Limited


Network Activity Monitor
Real-time
Network Activity
160
140
120
100
80
60
40
20
0
12:52:32 am
Value: 68
UNAUTHORIZED Attack
12:52:14 am
12:52:16 am
12:52:18 am
12:52:20 am
12:52:22 am
12:52:24 am
12:52:26 am
12:52:28 am
12:52:30 am
12:52:32 am
12:52:34 am
12:52:36 am
12:52:38 am
12:52:40 am
12:52:42 am
12:52:44 am
12:52:46 am
12:52:48 am
12:52:50 am

Analysis of the Output

1. Dataset Extraction and Loading

- **Extraction Success:**

The tar archive (iot_23_datasets_full.tar.gz) was successfully extracted into the directory iot23_extracted.

Message: Dataset extracted to: iot23_extracted

- **File Verification:**

The script confirmed the presence of the file conn.log.labeled inside the expected directory.

Message: Files in the directory: ['conn.log.labeled']

- **Header Parsing:**

The script correctly parsed the header from the Zeek log file using the #fields line. The parsed headers include fields such as ts, uid, id.orig_h, duration, orig_bytes, resp_bytes, label, etc.

Message: Parsed header columns: [...]

- **Data Loading:**

The log file was loaded into a Pandas DataFrame with 452 rows and 23 columns. The printed sample rows confirm that the data corresponds to network connection flows.

Message: Connection log loaded. Shape: (452, 23)

2. Preprocessing and Label Handling

- **Label Processing:**

The script attempts to create a binary label (binary_label) by processing the original label column. It uses a lambda function that treats missing values or entries that aren't clearly marked as "benign" as attack (1).

- **Observation:** In the output, the printed classification reports later suggest that nearly all labels are benign (class 0). This indicates that your dataset (or at least this scenario) contains very few, if any, attack samples.

- **Feature Selection and Conversion:**

The selected features (e.g., duration, orig_bytes, resp_bytes, orig_pkts, and resp_pkts) are converted to numeric values. Any non-numeric values are coerced to NaN, and missing values are filled with the mean.

- **Consideration:** Ensure that these feature names match what you expect from your logs; otherwise, you might need to update the feature list.

3. Model Training and Evaluation

Random Forest Classifier:

- The model was trained using the selected features. The classification report shows a perfect score (precision, recall, f1-score all 1.00) for class 0.
- **Issue:** The training data only contains one class (benign), which results in the model not having any information about attack traffic.
- **Impact:** When calling predict_proba, the model only returns one column (because it only learned one class), so the code sets the predicted probabilities to 0. This also leads to an undefined ROC AUC score (shown as nan) and triggers a warning.

Isolation Forest:

- This unsupervised model is designed to detect anomalies by modeling only the benign traffic.
- **Result:** The classification report for the Isolation Forest shows that while it can detect benign flows with high precision, there is no proper metric for the attack class (since there are no true attack samples).
- **Warnings:** Undefined metrics (recall for class 1 is set to 0) indicate that the metric calculation is not meaningful when one of the classes has no true samples.
- **Overall Training Considerations:** Your current training data appears imbalanced, containing only benign samples. For a robust intrusion detection system, you'll need a dataset with both benign and attack flows.
- **Recommendation:** Consider reviewing your labeling logic or merging data from multiple scenarios (or manually adding known attack flows) to ensure that both classes are represented.

4. Blockchain Logging

Blockchain Implementation:

- A simple blockchain mechanism was implemented to log prediction events. The blockchain starts with a genesis block and adds subsequent blocks that store prediction data along with metadata (timestamp, previous hash, etc.).
- **Output:** The ledger shows two blocks:

1. The genesis block, which is the first block of the chain.
2. A new block that logs the simulated prediction input (a sample network flow with features) and the corresponding predictions from both the Random Forest and Isolation Forest models.

- **Utility:** This step simulates how predictions might be recorded immutably, which can be useful for audit trails or secure logging in production environments.

5. Warnings and Deprecations

Deprecation Warnings:

- The warning related to tar.extractall indicates that future versions of Python will change how tar archives are extracted. While it's not an immediate issue, consider updating your extraction code in the future.
- Similarly, warnings about datetime.utcnow() suggest that you might need to switch to timezone-aware datetime objects later on.

Undefined Metrics:

- The ROC AUC and recall metrics are undefined or zero for the missing class due to the dataset's imbalance. This is a clear indication that the dataset does not have a representative sample of both benign and attack traffic.

Summary

- **Dataset:** Successfully extracted and loaded, with headers correctly parsed.
- **Preprocessing:** Performed as expected; however, the label assignment indicates a one-class problem (likely all benign).

- **Model Training:**

- Random Forest shows perfect performance on benign samples but fails to evaluate on missing attack data.
- Isolation Forest is applied for anomaly detection but has metrics that are hard to interpret due to the lack of attack samples.

- **Blockchain Logging:** A new block was successfully added to the blockchain, logging a simulated prediction.

Next Steps:

- Revisit your dataset to ensure you have a balanced representation of both benign and attack traffic.
- Consider refining your feature selection and label assignment for more robust model training.
- Update code for future Python compatibility (e.g., tar.extractall and datetime.utcnow()).

1. Data Ingestion and Preprocessing

- **Log File Extraction and Parsing:**

  Your IDS starts by extracting and reading Zeek connection logs (i.e., conn.log.labeled) from multiple scenarios. The system automatically reads the header information (the #fields line) and uses that to structure the data. This ensures that all incoming log files are standardized into a DataFrame with the correct columns.

- **Feature Selection and Conversion:**

  You have defined key features (such as duration, orig_bytes, resp_bytes, orig_pkts, and resp_pkts) that are important for identifying network behavior. The system converts these fields to numeric values and fills any missing values with column averages. This preprocessing step is critical because it ensures that the models receive clean, consistent data.

2. Model Training

- **Supervised Model (Random Forest):**

  The Random Forest classifier is trained using the processed features and a binary label that you derive from the log data (e.g., marking flows as “benign” or “attack”).

However, the output indicates that the training data in your current scenario consists only of benign traffic.

- **Implication:** Since your training data lacks attack examples, the model is learning only what "benign" looks like. In practical terms, this means that the supervised part of your IDS will always predict benign when it sees similar traffic. In a real deployment, you would need a balanced dataset (with both benign and malicious samples) so that the model can learn to differentiate between normal and anomalous behaviors.

- **Unsupervised Model (Isolation Forest):**

  The Isolation Forest is used for anomaly detection by modeling only the benign traffic. It flags deviations from what is considered normal.

- **Implication:** This model is useful because even if you don't have many (or any) labeled attack samples, it can potentially catch unusual or outlier network flows that deviate from the norm. However, the metrics show that since there are no true attack labels, the evaluation (like recall for the attack class) isn't very meaningful yet.

3. Blockchain Logging

- **Immutable Audit Trail:**

  The IDS logs its prediction events (i.e., the inputs and model outputs) into a simple blockchain ledger. Each prediction is recorded in a new block that includes a timestamp, the prediction data, and a cryptographic hash linking it to the previous block.

- **Implication:** This design adds a layer of security and accountability by creating an immutable audit trail. In a production IDS, such logging can be invaluable for forensic analysis, traceability, and ensuring that prediction data is not tampered with after the fact.

4. Overall System Behavior and Readiness

Current Limitations:

- **Training Data Imbalance:** The fact that your models have been trained on a dataset that appears to contain only benign traffic means that, as it stands, your IDS is not

capable of detecting malicious activity because it hasn't learned what "attack" looks like.

- **Evaluation Metrics:** Metrics such as ROC AUC and recall for the attack class are not defined or meaningful in this situation. This highlights the need for incorporating attack samples into your training data.

Potential in Production:

- **Data Pipeline:** The system shows that you have a robust data ingestion and preprocessing pipeline that can handle multiple scenarios and standardize log data.
- **Modeling Approach:** With the supervised (Random Forest) and unsupervised (Isolation Forest) components in place, your IDS is architecturally sound. Once you provide a more balanced dataset that includes labeled attack traffic, the Random Forest will be able to distinguish malicious flows from benign ones, and the Isolation Forest will help flag anomalies in real time.
- **Blockchain Logging:** The blockchain component is a forward-looking feature that enhances the trustworthiness and auditability of your system's predictions.

Summary

How Your IDS Works:

- **Data Collection:** It collects and standardizes network flow data from Zeek logs.
- **Feature Engineering:** It extracts and cleans key numeric features from the logs.
- **Detection Models:** It applies a supervised model (Random Forest) and an unsupervised anomaly detector (Isolation Forest) to predict whether a network flow is benign or anomalous.
- **Audit Logging:** It logs each prediction into a blockchain ledger for secure, immutable record-keeping.

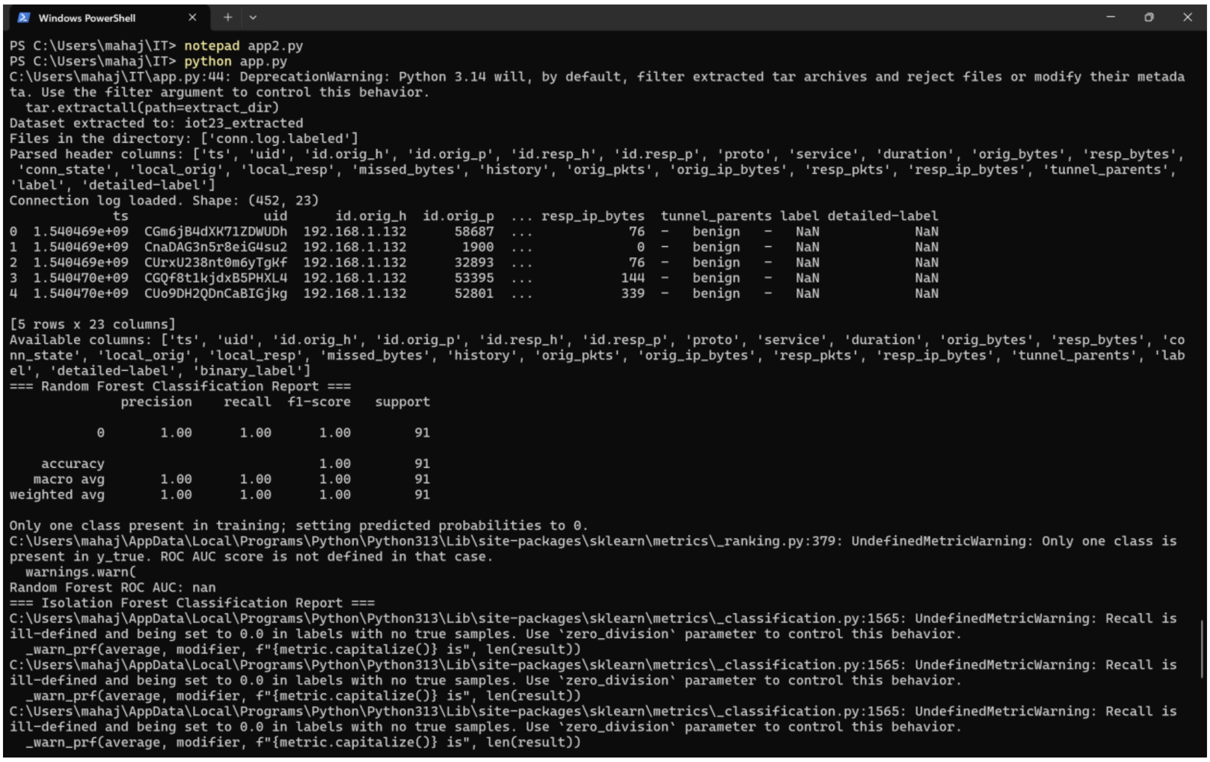
What It Tells You About Its Effectiveness:

- **Current State:** With only benign data, the IDS currently acts as a "dummy" classifier that always predicts benign. This isn't useful for detecting attacks yet.
- **Future Potential:** The system is well-architected and ready to be trained on a more balanced dataset. Once you include attack samples, the supervised model can

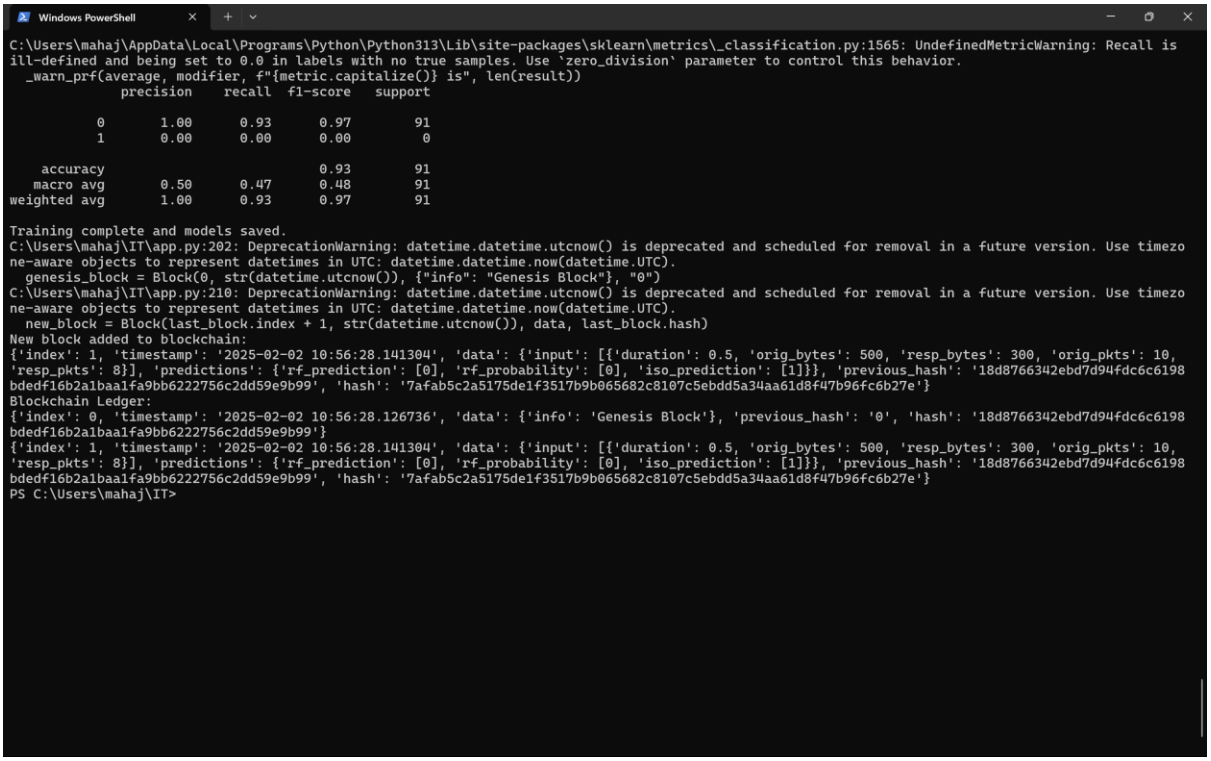
effectively learn the difference between benign and malicious traffic, while the unsupervised model can help identify unexpected anomalies.

- **Security and Auditing:** The blockchain logging adds a layer of security and accountability that is beneficial for a production IDS.

In conclusion, the output shows that your IDS is correctly set up in terms of data handling, modeling, and logging. However, its current predictive power is limited by the lack of attack data. For a fully operational IDS, you will need to augment your dataset with representative malicious samples so that the models can learn to detect and alert on true anomalies.



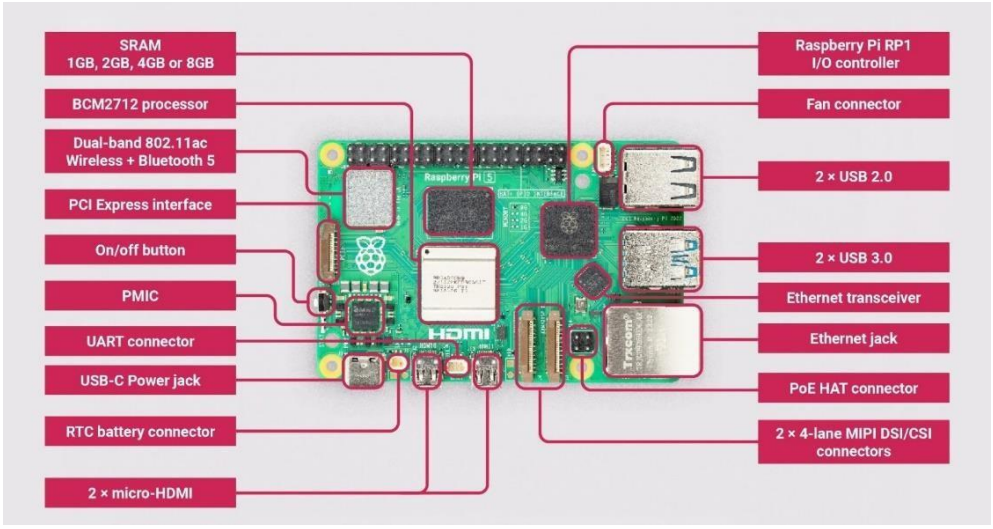
```
PS C:\Users\mahaj\IT> notepad app2.py
PS C:\Users\mahaj\IT> python app.py
C:\Users\mahaj\IT\app.py:44: DeprecationWarning: Python 3.14 will, by default, filter extracted tar archives and reject files or modify their metada
ta. Use the filter argument to control this behavior.
  tar.extractall(path=extract_dir)
Dataset extracted to: iot23_extracted
Files in the directory: ['conn.log.labeled']
Parsed header columns: ['ts', 'uid', 'id.orig_h', 'id.orig_p', 'id.resp_h', 'id.resp_p', 'proto', 'service', 'duration', 'orig_bytes', 'resp_bytes',
 'conn_state', 'local_orig', 'local_resp', 'missed_bytes', 'history', 'orig_pkts', 'orig_ip_bytes', 'resp_pkts', 'resp_ip_bytes', 'tunnel_parents',
'label', 'detailed-label']
Connection log loaded. Shape: (452, 23)
             ts                  uid      id.orig_h  id.orig_p  ... resp_ip_bytes  tunnel_parents label detailed-label
0  1.540469e+09  CGm6jB4dXK71ZDWUDh  192.168.1.132      58687  ...            76  -    benign   -   NaN            NaN
1  1.540469e+09  CnaDAG3n5r8eiG4su2  192.168.1.132       1900  ...             0  -    benign   -   NaN            NaN
2  1.540469e+09  CUrxU238nt0m6yTgKf  192.168.1.132      32893  ...            76  -    benign   -   NaN            NaN
3  1.540470e+09  CGQf8t1kjdxB5PHXL4  192.168.1.132      53395  ...           144  -    benign   -   NaN            NaN
4  1.540470e+09  CUo9DH2QDnCaBIGjkg  192.168.1.132      52801  ...           339  -    benign   -   NaN            NaN

[5 rows x 23 columns]
Available columns: ['ts', 'uid', 'id.orig_h', 'id.orig_p', 'id.resp_h', 'id.resp_p', 'proto', 'service', 'duration', 'orig_bytes', 'resp_bytes', 'co
nn_state', 'local_orig', 'local_resp', 'missed_bytes', 'history', 'orig_pkts', 'orig_ip_bytes', 'resp_pkts', 'resp_ip_bytes', 'tunnel_parents', 'lab
el', 'detailed-label', 'binary_label']
=== Random Forest Classification Report ===
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        91

    accuracy                           1.00        91
   macro avg       1.00      1.00      1.00        91
weighted avg       1.00      1.00      1.00        91

Only one class present in training; setting predicted probabilities to 0.
C:\Users\mahaj\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics\_ranking.py:379: UndefinedMetricWarning: Only one class is
present in y_true. ROC AUC score is not defined in that case.
  warnings.warn(
Random Forest ROC AUC: nan
=== Isolation Forest Classification Report ===
C:\Users\mahaj\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning: Recall is
ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\mahaj\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning: Recall is
ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\mahaj\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning: Recall is
ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

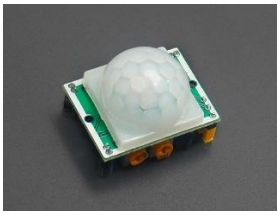


The experimental setup incorporates IoT devices like Raspberry Pi, sensors, and edge computing hardware for anomaly detection, connected to a blockchain framework for tamper-proof data logging. This architecture ensures secure and scalable real-time intrusion detection.

Components:

1. IoT Layer:





- **Raspberry Pi 5:** Runs machine learning models for anomaly detection.

- **Sensors:** PIR motion sensors for detecting physical anomalies, **BME680** for environmental data simulation.

- **ESP32:** Gathers data from sensors and transfers it to the Raspberry Pi.

2. Data Storage Layer:

Micro SD Cards (128 GB, UHS-1) : Store OS and local data logs.

External SSD (256 GB, USB 3.2): Backup logs and blockchain data.

3. Power Management:

Official Raspberry Pi power supplies ensure consistent performance.

4. Connectivity:

- **USB Wi-Fi adapters** and **GPIO kits** for stable communication between components.
- **HDMI Cable** 1. necessary connection between the Raspberry Pi 5 and the display monitor. For example: "Connect the Raspberry Pi 5 to a monitor using a high-speed HDMI cable for video output."

- **Multimeter (Fluke 101):** Highlight its use for checking connections and verifying voltage levels: A Fluke 101 multimeter was used to verify power supply outputs and ensure stable operation during the setup.
- **Case for Raspberry Pi 5 with Fan:** Stress the importance of thermal management:

The Raspberry Pi 5 was housed in a compatible case with an integrated fan to prevent overheating during extended use.

5. Blockchain Layer:

Smart contracts record intrusion metadata on Ethereum testnet (using Solidity).

IPFS stores detailed logs off-chain, with hash references on-chain for tamper-proof

logging.

6. User Interface Layer:

Dashboard: Built with Flask and React.js for real-time intrusion monitoring and management.

Steps for Experimental Setup:

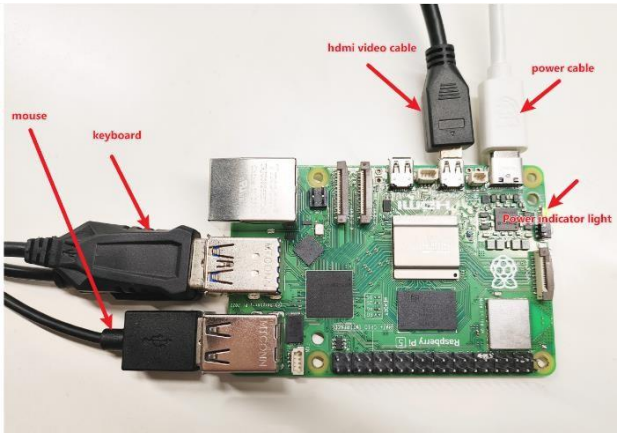
Step 1: Preparing Raspberry Pi 5 Devices

1. Insert Micro SD Cards:

- Load the Raspberry Pi OS onto the micro SD cards (128 GB) using tools like Balena Etcher.
- Insert the micro SD card into each Raspberry Pi.

2. Power Connection and Network Setup:

- Insert USB Wi-Fi adapters into the Raspberry Pi devices to enable internet connectivity.
- Connect the official Raspberry Pi power supplies to the devices to ensure stable power delivery.
- Connect each Raspberry Pi to a monitor, keyboard, and mouse for the initial setup.
- Use an HDMI cable to connect each Raspberry Pi to a monitor for the initial setup process.



3. Cooling Setup:

- Place each Raspberry Pi in its protective case with a fan to prevent overheating during prolonged operation.

4. Wi-Fi USB Adapter Setup:

- Insert the USB Wi-Fi adapter into each Raspberry Pi to enable internet connectivity for data synchronization, software installation, and remote access.

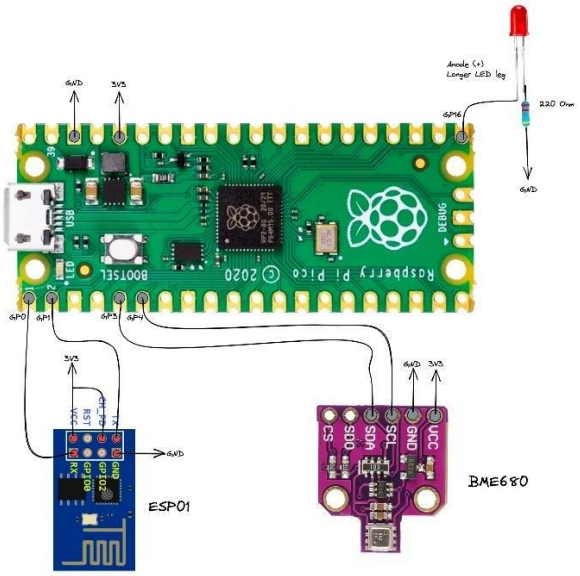
Step 2: Connecting Sensors to Raspberry Pi 5

1. BME680 Sensors (Environmental Monitoring):

Connect the BME680 sensor to one Raspberry Pi using the GPIO kit as follows:

- GND to ground pin.
- VCC to a 3.3V power pin.
- SDA to GPIO2.
- SCL to GPIO3.

This sensor will monitor environmental parameters like temperature, humidity, and air quality.

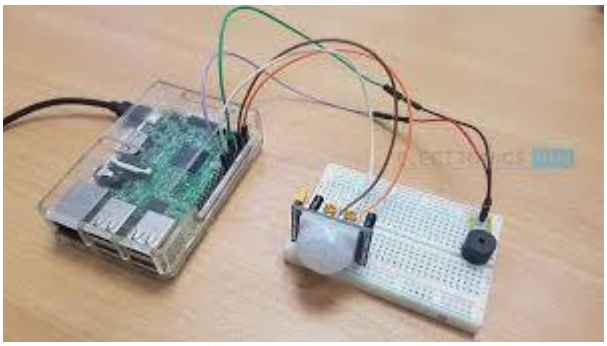


2. PIR Motion Sensors (Intrusion Detection):

Attach PIR motion sensors to the second Raspberry Pi using GPIO pins:

- OUT to GPIO17 (data input).
- VCC to 5V power pin.
- GND to ground pin.

Position the sensors near doors, windows, or areas requiring motion detection.



3. ESP32 Modules (Wireless Sensor Data):

- Connect the ESP32 modules to power using USB.
- Configure the ESP32 to wirelessly transmit data from additional sensors to the Raspberry Pi devices.

4. Multimeter (Fluke 101)

- Insert the black probe into the COM port and the red probe into the V/Ω/Hz port of the multimeter.

- Use it for:

  a. Measuring voltage across GPIO pins (e.g., 3.3V or 5V).

  b. Checking power supply stability from the Raspberry Pi power adapter.

  c. Verifying continuity of connections and wires.

  d. Testing sensor outputs (e.g., voltage changes from the PIR motion sensor during motion detection)

Step 3: Setting Up Storage and Blockchain

1. External SSD (256 GB):

- Connect the SSD to the Raspberry Pi designated for blockchain using the USB 3.2 port.
- This SSD will store the blockchain data, including anomaly logs.

- Connect the BME680 sensors to Raspberry Pi GPIO pins using jumper wires (pins: GND, VCC, SDA, and SCL).
- Attach the PIR motion sensors similarly to the GPIO pins for motion detection.
- For ESP32, connect it to a power source and configure it to send data wirelessly to the Raspberry Pi.

2. Installing Blockchain Software:

- Deploy the Ethereum testnet (e.g., Hardhat) on the Raspberry Pi hosting the blockchain node.

3. Data Backup and Logs:

- Configure the SSD to back up critical logs and data periodically.

CHAPTER 8

CHALLENGES

To develop a seamless and secure Blockchain-Based Intrusion Detection System (IDS) for IoT networks, several challenges need to be addressed, including scalability, resource constraints, and data privacy. By implementing innovative solutions, the project prioritizes efficiency, cost- effectiveness, and accessibility to ensure robust security across diverse IoT applications.

a. **Scalability**

Distributed Architecture:

- Challenge: Blockchain systems can face delays and high costs when scaling to large IoT networks.
- Solution: Implement lightweight consensus mechanisms like Proof of Authority (PoA) and layer-2 solutions to process transactions faster.

Dynamic Node Management:

- Challenge: Expanding blockchain nodes in diverse environments without performance drops.
- Solution: Introduce auto-scaling blockchain nodes to dynamically adjust to workload demands.

b. **Resource Constraints IoT Device Limitations:**

- Challenge: Limited computational power, memory, and bandwidth in IoT devices.
- Solution: Offload computationally intensive tasks (e.g., model training) to edge or cloud systems while keeping detection lightweight on devices

Energy Efficiency:

- Challenge: Prolonged use of IoT devices for security monitoring may drain battery life.
- Solution: Optimize data collection and transmission intervals to balance resource usage and effectiveness.

1. Data Privacy and Security Anonymizing Sensitive Data:

- Challenge: Preventing exposure of device-specific data during blockchain logging.
- Solution: Use encryption techniques and anonymized hashing before storing data on-chain.

Secure Off-Chain Storage:

- Challenge: Managing large intrusion logs without overloading the blockchain.

- Solution: Store detailed logs in decentralized off-chain systems like IPFS while recording hash references on the blockchain.

2. Usability for Diverse Networks Heterogeneous IoT Environments:

- Challenge: Integrating the IDS into various IoT systems (e.g., smart homes, industrial IoT).
- Solution: Develop modular components adaptable to different IoT setups and network protocols.

Admin Accessibility:

- Challenge: Simplifying dashboards for non-technical administrators.
- Solution: Provide an intuitive web-based interface with real-time monitoring and guided troubleshooting

CHAPTER 9
CONCLUSION

"Securing the Internet of Things" marks a groundbreaking advancement in leveraging blockchain technology to safeguard IoT networks from evolving cyber threats. This project exemplifies how cutting-edge innovations can enhance security, ensure data integrity, and build trust in systems critical to modern life. By integrating decentralized blockchain architecture with advanced intrusion detection mechanisms, this initiative addresses the vulnerabilities inherent in traditional IoT security approaches and sets a new standard for resilient and reliable IoT systems.

The system provides a robust and scalable solution to protect IoT devices across diverse applications, from smart homes and industrial automation to healthcare and transportation. Through real-time anomaly detection, tamper-proof logging, and automated response mechanisms, the system ensures that potential threats are identified and mitigated swiftly and effectively. Its lightweight design, optimized for resource-constrained IoT environments, makes it accessible and adaptable to a wide range of use cases.

This project also emphasizes the importance of protecting sensitive data in a world increasingly reliant on connected devices. By utilizing blockchain's immutability and encryption, it guarantees the confidentiality, integrity, and availability of IoT data, even in the face of sophisticated cyberattacks. The system not only builds a secure digital foundation but also fosters trust among users and stakeholders, enabling the broader adoption of IoT technologies.

Moreover, "Securing the Internet of Things" serves as a blueprint for how emerging technologies can be harnessed to address real-world challenges. It demonstrates the synergy between machine learning and blockchain, showcasing their combined potential to revolutionize cybersecurity. This initiative is a testament to the power of innovation to bridge the gap between existing vulnerabilities and future-proof solutions.

By creating a secure, scalable, and user-friendly intrusion detection system, this project paves the way for a safer IoT ecosystem. It inspires the application of similar technologies to other domains, ensuring that critical infrastructure and digital systems remain protected against evolving threats.

In essence, "Securing the Internet of Things" is more than a security solution; it is a step toward building a more secure, connected future. It reaffirms the role of technology in solving complex problems and highlights the potential for interdisciplinary approaches to reshape the landscape of cybersecurity in the IoT era.

References:

[1] Nazir, A., He, J., Zhu, N., Wajahat, A., Ullah, F., Qureshi, S., Ma, X., & Pathan, M. S. (2024). Collaborative threat intelligence: Enhancing IoT security through blockchain and machine learning integration. *Journal of King Saud University - Computer and Information Sciences*, *36*(2), 101939. https://doi.org/10.1016/j.jksuci.2024.101939

[2] Govindaram, A., & Jegatheesan, A. (2024). FLBC-IDS: A federated learning and blockchain-based intrusion detection system for secure IoT environments. Multimedia Tools and Applications. https://doi.org/10.1007/s11042-024-19777-6

[3] Samaniego, M., Jamsrandorj, U., & Deters, R. (2016). Blockchain as a Service for IoT. In *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)* (pp. 433-436). https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2016.102

[4] Huh, S., Cho, S., & Kim, S. (2017). Managing IoT devices using blockchain platform. In *2017 19th International Conference on Advanced Communication Technology (ICACT)* (pp. 464-467). https://doi.org/10.23919/ICACT.2017.7890132

[5] Mehedi, S. K. T., Shamim, A. A. M., & Miah, M. B. A. (2019). Blockchain-based security management of IoT infrastructure with Ethereum transactions. *Iran Journal of Computer Science*, *2*(3), 189-195. https://doi.org/10.1007/s42044-019-00044-z

[6] Ramesh, V. K. C., Kim, Y., & Jo, J.-Y. (2020). Secure IoT Data Management in a Private Ethereum Blockchain. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*. https://digitalscholarship.unlv.edu/compsci_fac_articles/251/

[7] Raj, A., Maji, K., & Shetty, S. D. (2021). Ethereum for Internet of Things security. *Multimedia Tools and Applications*, *80*(12), 18901-18915. https://doi.org/10.1007/s11042-021-10715-4

[8] Ibrahim, R. F., Abu Al-Haija, Q., & Ahmad, A. (2022). DDoS Attack Prevention for Internet of Thing Devices Using Ethereum Blockchain Technology. *Sensors*, *22*(18), 6806. https://doi.org/10.3390/s22186806

[9] Alsharif, N. A., Mishra, S., & Alshehri, M. (2023). IDS in IoT using Machine Learning and Blockchain. *Engineering, Technology & Applied Science Research*, *13*(4), 11197–11203. https://doi.org/10.48084/etasr.5992

[10] Tukur, Y. M., Thakker, D., & Awan, I. U. (2021). Edge-based blockchain enabled anomaly detection for insider attack prevention in Internet of Things. Transactions on Emerging Telecommunications Technologies. https://doi.org/10.1002/ett.4158

[11] Ahamad, S., Gupta, P., Acharjee, P. B., Kiran, K. P., Khan, Z., & Hasan, M. F., "The Role of Blockchain Technology and Internet of Things (IoT) to Protect Financial Transactions in Cryptocurrency Market"2021 https://www.sciencedirect.com/science/article/abs/pii/S2214785321074319

[12] Kumar, R., Kumar, P., Tripathi, R., Gupta, G. P., Garg, S., & Hassan, M. M. (2022). A distributed intrusion detection system to detect DDoS attacks in blockchain-enabled IoT network. Journal of Parallel and Distributed Computing, 164, 55–68. https://doi.org/10.1016/j.jpdc.2022.01.030

[13] Saurabh Singh, A. S. M. Sanwar Hosen, and Byungun Yoon, *"Blockchain Security Attacks, Challenges, and Solutions for the Future Distributed IoT Network,"* IEEE Access, vol. 9, pp. 13937–13955, 2021. https://ieeexplore.ieee.org/document/9323061

[14] Shashvi Mishra and Amit Kumar Tyagi, *"Intrusion Detection in Internet of Things (IoTs) Based Applications using Blockchain Technology,"* Proceedings of the Third International Conference on I-SMAC (IoT in Social, Mobile, Analytics, and Cloud), IEEE, 2019, pp. 123-128. https://ieeexplore.ieee.org/abstract/document/9032557

[15] Dongxing Li, Wenping Deng, Wei Peng, and Fangyu Gai, *"A Blockchain-Based Authentication and Security Mechanism for IoT,"* Proceedings of the 27th International Conference on Computer Communications and Networks (ICCCN), IEEE, 2018, pp. 1-7. https://ieeexplore.ieee.org/document/8487449.

[16] Singh, S., Sharma, P. K., Yoon, B., Shojafar, M., Cho, G. H., & Ra, I.-H. (2020). Convergence of Blockchain and Artificial Intelligence in IoT Network for the Sustainable Smart City. *Sustainable Cities and Society.* https://doi.org/10.1016/j.scs.2020.102364.

[17] Porkodi, S., & Kesavaraja, D. (2020). *Integration of Blockchain and Internet of Things*.

In *Handbook of Research on Blockchain Technology* (Chapter 3). Elsevier. DOI: 10.1016/B978-0-12-819816-2.00003-4

[18] Jiang, Y., Wang, C., Wang, Y., & Gao, L. (2019). *A Cross-Chain Solution to Integrating Multiple Blockchains for IoT Data Management*. *Sensors, 19*(9), 2042. DOI: 10.3390/s19092042.

[19] Chen, Y. H., Chen, S. H., & Lin, I. C. (2018). Blockchain based Smart Contract for Bidding System. In Proceedings of IEEE International Conference on Applied System Innovation 2018 (ICASI 2018) (pp. 208-211). IEEE. ISBN: 978-1-5386-4342-6.

[20] Christidis, K., & Devetsikiotis, M. (2016). Blockchains and smart contracts for the Internet of Things. *IEEE Access*, 4, 2292-2303. doi: 10.1109/ACCESS.2016.2566339.

[21] Ardit Dika and Mariusz Nowostawski, "Security Vulnerabilities in Ethereum Smart Contracts," 2018 IEEE Confs on Internet of Things, Green Computing and Communications, Cyber, Physical and Social Computing, Smart Data, Blockchain, Computer and Information Technology, Congress on Cybermatics, 2018, pp. 955-962. doi: 10.1109/Cybermatics_2018.2018.00182.

[22] G. R. Ramezan and C. Leung, "A Blockchain-Based Contractual Routing Protocol for the Internet of Things Using Smart Contracts," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 4029591, 14 pages, 2018. doi: 10.1155/2018/4029591.

[23] Shantanu Pal, Tahiry Rabehaja, Ambrose Hill, Michael Hitchens, and Vijay Varadharajan, "On the Integration of Blockchain to the Internet of Things for Enabling Access Right Delegation," *IEEE Internet of Things Journal*, vol. XX, no. XX, pp. 1-10, June 2019. doi: 10.1109/JIOT.2019.2952141.

[24] J. Pan, J. Wang, A. Hester, I. Alqerm, Y. Liu, and Y. Zhao, "EdgeChain: An Edge-IoT Framework and Prototype Based on Blockchain and Smart Contracts," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4719-4732, June 2019. doi: 10.1109/JIOT.2018.2878154.

[25] Z. Shahbazi and Y.-C. Byun, "Integration of Blockchain, IoT and Machine Learning for Multistage Quality Control and Enhancing Security in Smart Manufacturing," *Sensors*, vol. 21, no. 4, pp. 1467, Feb. 2021. doi: 10.3390/s21041467.