الجمهورية الجزائرية الديمقراطية الشعبية

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

وزارة التعليم العالي والبحث العلمي

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

جامعة 20اوت 1955 -سكيكدة

UNIVERSITY OF 20 August 1955- Skikda

Faculty of Sciences

Department of Computer Science

Master's Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master

Specialization: Advanced Software Engineering and Applications

Title:

A Collaborative IDS for IoT Environments Using Machine Learning and Blockchain

Prepared by:

NAHAL Amina

REHAIBI Raihana

Supervised by:

Dr.CHEIKH Mohamed

Dr. MAZOUZI Rabah

Examination Committee:

Title	**First Name/Last Name**
Committee Chair	Mr CHEIKH Remdane
Supervisors	Mr CHEIKH Mohamed Mr MAZOUZI Rabah
Examiner	Ms ALIGUECHI Farida

2024/2025

Acknowledgment

All praise is due to Allah, who granted me the strength and perseverance to complete this modest graduation project.

I would like to express my heartfelt gratitude to ***Dr. Mohamed Cheikh*** *for proposing this enriching topic and for his invaluable supervision, insightful guidance, and unwavering support throughout every stage of this work.*

My sincere appreciation also goes to ***Dr. Rabah Mazouzi*** *for his generous assistance, encouragement, and continuous support, which greatly contributed to the progress and completion of this project.*

I am also thankful to the honorable jury members for taking the time to evaluate my work and for their constructive feedback, which I deeply value

Dedication

We dedicate this final year project to our beloved parents, whose unconditional love, endless sacrifices, unwavering support, and constant prayers have been the foundation of our strength and perseverance throughout this academic journey. Without them, none of this would have been possible.

To our mothers, the infinite sources of love and encouragement.
To our fathers, the pillars of wisdom and resilience.

We also extend our heartfelt gratitude to our supervisors, ***Dr. Mohamed Cheikh*** *and* ***Dr. Rabah Mazouzi****, for their guidance, valuable advice, and continuous support, which greatly contributed to the realization of this work.*

To everyone who supported us, directly or indirectly, we express our sincere thanks and appreciation

Abstract

The rapid expansion of the Internet of Things (IoT) has revolutionized industrial environments, offering unprecedented connectivity and automation, yet it also introduces significant security challenges due to the heterogeneous nature and resource constraints of IoT devices. This research presents a decentralized, edge-centric Intrusion Detection System (IDS) tailored for Industry 4.0 settings, integrating machine learning-based anomaly detection with lightweight blockchain technology to ensure secure, real-time monitoring and alert propagation. The system comprises layered components, including strong devices for processing and detection, edge gateways for coordination and blockchain interaction, and weak devices with minimal computational capabilities. Various attack scenarios are simulated using the CICIDS2017 dataset, with machine learning algorithms such as Random Forest and XGBoost employed for classification, alongside hyperparameter optimization using the Grey Wolf Optimizer (GWO). Alerts are securely logged and shared via a custom blockchain implemented using SQLite, maintaining data integrity and traceability. Experimental results demonstrate high detection accuracy and efficient response coordination, validating the system's scalability, lightweight design, and effectiveness in securing IoT infrastructures within smart industrial environments.

Keywords: Internet of Things (IoT), Intrusion Detection System (IDS), Edge Computing, Industry 4.0, Machine Learning, Blockchain, Anomaly Detection, Random Forest, XGBoost, Grey Wolf Optimizer (GWO), Cybersecurity, Smart Industry.

الملخص

شهد التوسع السريع في تقنيات إنترنت الأشياء (IoT) تحولًا جذريًا في البيئات الصناعية، حيث وفر مستوى غير مسبوق من الاتصال والأتمتة، إلا أن هذا النمو صاحبته تحديات أمنية كبيرة نتيجة لتنوع الأجهزة وتفاوت قدراتها الحاسوبية. تقدم هذه الدراسة نظام كشف اختراق (IDS) لامركزي ومتمركز على الحافة، مصمم خصيصًا لبيئات الصناعة 4.0، ويعتمد على تقنيات الكشف عن الشذوذ باستخدام خوارزميات التعلم الآلي، إلى جانب دمج تقنية البلوكشين خفيفة الوزن لضمان المراقبة الآنية الآمنة ونشر التنبيهات. يتكون النظام من طبقات تشمل أجهزة قوية للمعالجة والكشف، وبوابات حافة للتنسيق والتفاعل مع البلوكشين، وأجهزة ضعيفة ذات قدرات محدودة. وقد تم محاكاة عدة سيناريوهات هجومية باستخدام مجموعة بياناتCICIDS2017 ، وتم توظيف خوارزميات مثل Random Forest و XGBoostفي التصنيف، مع تحسين المعاملات باستخدام خوارزمية الذئب الرمادي .(GWO) يتم تسجيل التنبيهات ومشاركتها بشكل آمن باستخدام بلوكشين مخصص تم تنفيذه باستخدامSQLite ، مما يضمن سلامة البيانات وإمكانية تتبعها. أظهرت النتائج التجريبية دقة عالية في الكشف عن الهجمات، وكفاءة في التنسيق بين الأجهزة، مما يؤكد على قابلية توسع النظام، وخفة وزنه، وفعاليته في تأمين بنى إنترنت الأشياء في البيئات الصناعية الذكية.

الكلمات المفتاحية: انترنات الاشياء .نظام كشف الاختراق . الحوسبة الطرفية . الصناعة 4.0 . التعلم الالي . البلوك شين . كشف الشذوذ . الغابة العشوائية . XGBoost . خوارزميات الذئاب الرمادية .الامن السبيراني . الصناعة الذكية

Résumé

L'expansion rapide de l'Internet des Objets (IoT) a profondément transformé les environnements industriels en offrant une connectivité et une automatisation sans précédent, tout en introduisant d'importants défis en matière de sécurité, notamment en raison de l'hétérogénéité et des contraintes de ressources des dispositifs IoT. Cette recherche propose un système de détection d'intrusion (IDS) décentralisé et centré sur l'edge, conçu pour les environnements de l'industrie 4.0. L'architecture intègre une détection d'anomalies basée sur l'apprentissage automatique avec une technologie blockchain légère, afin d'assurer une surveillance sécurisée en temps réel et une propagation fiable des alertes. Le système est structuré en plusieurs couches : des dispositifs puissants assurent le traitement et la détection, des passerelles edge coordonnent les communications et interagissent avec la blockchain, et des dispositifs faibles, à capacité limitée, assurent la collecte de données. Divers scénarios d'attaques sont simulés à l'aide du jeu de données CICIDS2017, en utilisant des algorithmes tels que Random Forest et XGBoost, avec une optimisation des hyperparamètres via l'algorithme Grey Wolf Optimizer (GWO). Les alertes sont enregistrées et partagées de manière sécurisée à l'aide d'une blockchain personnalisée implémentée avec SQLite, garantissant l'intégrité et la traçabilité des données. Les résultats expérimentaux montrent une grande précision de détection et une coordination efficace entre les dispositifs, validant ainsi la scalabilité, la légèreté et l'efficacité du système dans la sécurisation des infrastructures IoT dans des environnements industriels intelligents

Mots-clés : Internet des objets, Système de détection d'intrusion, Informatique en périphérie, Apprentissage automatique, Blockchain, Industrie 4.0, Détection d'anomalies, Forêt aléatoire (Random Forest), XGBoost, Optimiseur du Loup Gris (GWO), Cybersécurité, Industrie intelligente.

Table des matières

List of Figures:

Chapter 1 – Literature Review

Chapter 2 – Proposed Edge-Centric IDS Architecture

Chapter 3 – Machine Learning-Based Detection Model Design

Chapter 4 – Experimental Setup and Evaluation

List of Tables:

General Introduction

The advent of the Internet of Things (IoT) has ushered in a transformative era across various sectors such as healthcare, manufacturing, agriculture, transportation, and especially in smart cities. These environments are now populated with billions of interconnected smart devices capable of collecting, processing, and exchanging data in real-time. While this hyper-connectivity enhances automation, efficiency, and responsiveness, it also introduces a complex and often under-addressed set of security challenges. The heterogeneity, constrained resources, and distributed deployment of IoT devices create a vast attack surface for malicious actors. Traditional security approaches centered around cloud-based infrastructures are increasingly incapable of providing the required protection due to issues like latency, scalability limitations, centralized failure points, and privacy concerns.

To address these growing threats, this research proposes an edge-centric Intrusion Detection System (IDS) architecture tailored to the unique characteristics and limitations of IoT networks. The proposed system departs from conventional models by decentralizing the detection and response logic. It categorizes IoT devices into strong and weak nodes: strong devices are equipped with sufficient computational resources to run intelligent detection algorithms locally, while weak devices remain lightweight and react to incoming alerts. Two types of gateways manage alert flow, interacting with a lightweight blockchain built on SQLite to ensure that alerts are stored in a tamper-resistant and auditable format. This blockchain-backed propagation mechanism enhances trust and data integrity across all components.

Machine learning (ML) forms the foundation of the system's threat detection mechanism. Leveraging the CICIDS2017 dataset a well-established benchmark in the cybersecurity community this research trains and evaluates a hybrid ensemble model combining Random Forest and XGBoost classifiers. To ensure optimal model performance and computational efficiency, feature selection is performed using Grey Wolf Optimization (GWO), a metaheuristic inspired by natural predator behavior.

This work is structured into four primary chapters. The first chapter introduces the foundational theories and technologies in IoT security, examining existing IDS approaches and highlighting the limitations they face in Industry 4.0 environments. The second chapter presents the design and internal behavior of the proposed edge-centric IDS architecture, including device classification, gateway functionality, and blockchain integration. Chapter three details the construction of the ML-based detection model, including data preprocessing, feature selection, and model evaluation. Finally, the fourth chapter provides an experimental validation of the proposed system in a simulated smart city scenario, demonstrating its ability to detect and respond to cyber threats in real-time.

Ultimately, this thesis aims to contribute a practical, modular, and secure IDS solution tailored for the rapidly evolving IoT ecosystem, providing a foundation for future extensions such as smart contract-based automation and federated learning-based model training.

Chapter 1: Security in IoT Systems

1. Introduction:

The Internet of Things (IoT) has transformed industries by enabling the seamless connection of smart devices that communicate, collect, and exchange data. These interconnected systems have found applications across various fields such as healthcare, transportation, agriculture, and industrial automation, driving significant advancements in technology and efficiency. However, the rapid proliferation of IoT devices and the complexity of their networks have raised substantial concerns about security. The increasing sophistication of cyber threats, coupled with the limited resources of many IoT devices, makes it increasingly difficult to ensure the integrity, confidentiality, and availability of IoT systems. Traditional security mechanisms, which rely heavily on centralized cloud infrastructures, are often inadequate to address these challenges due to latency, scalability, and performance concerns.

In response to these challenges, new security paradigms are emerging that focus on distributed, real-time, and context-aware approaches. Edge and fog computing, for example, provide more localized processing and decision-making capabilities, allowing for faster responses to threats. Furthermore, technologies like blockchain are being explored to provide decentralized and tamper-proof security solutions, ensuring transparency and trust within IoT networks.

This thesis proposes a novel edge-centric Intrusion Detection System (IDS) architecture that integrates machine learning, blockchain, and edge computing. The architecture categorizes IoT devices into strong and weak nodes: strong devices execute ML-based attack detection locally, while weak ones rely on alerts to take action. Edge gateways act as intermediaries, transmitting and receiving alerts through a secure blockchain ledger. This decentralized design enhances scalability, reduces latency, and enables real-time, trustworthy alert propagation across the network. This introductory overview provides a foundation for understanding the methodology and proposed system, which will be examined in detail in the subsequent chapters.

This chapter reviews the current landscape of IoT security, focusing on the key technologies that have evolved to address the unique security challenges of IoT systems. It begins by discussing the security challenges in IoT environments, then explores the evolution of Intrusion Detection Systems (IDS), contrasting traditional methods with modern approaches that incorporate machine learning, edge computing, and blockchain. The chapter also delves into the context of Industry 4.0, highlighting the critical role IoT plays in shaping the future of

industrial operations, and concludes with a review of related work and existing IDS solutions that have been proposed to secure IoT environments.

2. Information Technology (IT) Security:

Information Technology (IT) Security refers to a comprehensive set of cybersecurity strategies, tools, and best practices designed to prevent unauthorized access to digital assets such as computers, networks, systems, and sensitive data. It aims to ensure the **confidentiality**, **integrity**, and **availability** of information by defending systems against internal and external threats, including malicious actors and sophisticated cyberattacks [1][2].

IT security encompasses all components within an information ecosystem: users, hardware devices, software applications, communication networks, and data whether in transit or at rest. It operates within a layered framework of protection that includes **security policies**, **technical safeguards** (firewalls, intrusion detection systems, encryption), and **organizational measures** such as training and risk management.

The ultimate objective of IT security is to **minimize risk** by proactively identifying vulnerabilities and preventing or mitigating attacks. This is achieved through a structured approach involving policies, assurance frameworks, and standardized methodologies developed by international bodies and cybersecurity experts [1]

3. Security objectives:

3.1 Confidentiality:

Confidentiality means keeping data secret and accessible only to authorized people. Organizations protect confidentiality by using access controls, encryption, and multi-factor authentication. Threats include hacking, spying, password theft, and accidental sharing. Even human error like sharing a password or not encrypting sensitive data can cause breaches. To prevent this, organizations classify data, control who can access it, and train users to follow security practices.[3]

3.2 Integrity:

Integrity ensures that information is accurate and has not been changed without permission. If data is altered intentionally by a hacker or accidentally by a user it becomes unreliable. Examples include changing configuration files or editing company websites. To protect integrity, tools like hashing, digital signatures, and secure coding are used. Non-repudiation (digital signatures) confirms who sent or received information and ensures they cannot deny it later.[3]

3.3 Availability:

Availability means that data and systems are accessible when needed. If users cannot access important information due to power failure, cyberattacks (like DDoS), or disasters, availability is lost. To maintain it, companies use backups, redundant systems, fault tolerance mechanisms, and disaster recovery plans. These ensure that services remain available even during technical issues or emergencies.[3]

The fundamental principles of cybersecurity are often represented by the CIA Triad, which emphasizes the importance of maintaining **Confidentiality**, **Integrity**, and **Availability** in any secure system, as illustrated in **Figure 3**.

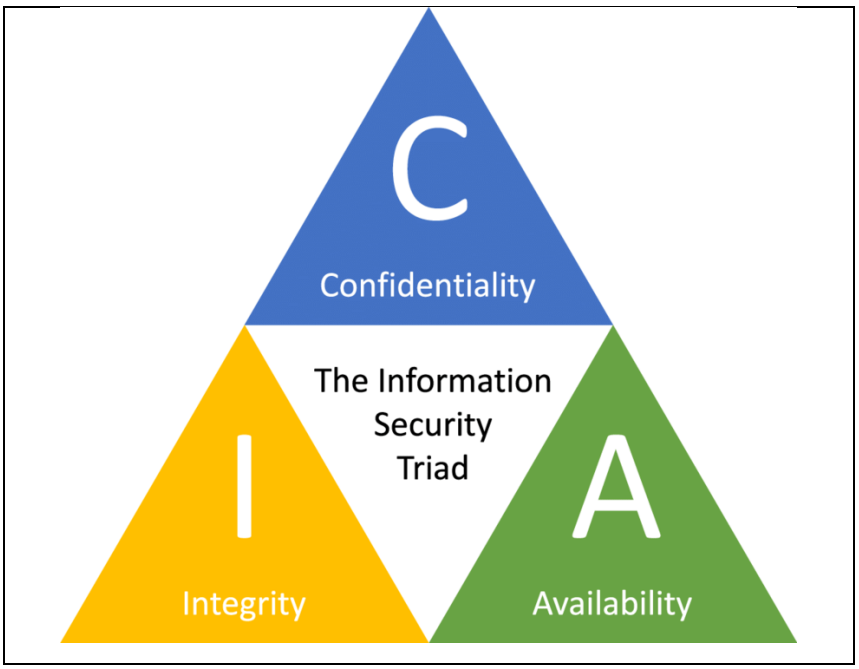


Figure3: CIA Triad: Confidentiality, Integrity, Availability in Cybersecurity

4. Types of Security:

Information security encompasses several types, each serving a unique role in protecting digital systems and sensitive data.

4.1 Physical security: safeguards the physical infrastructure such as servers, computers, and data centers against unauthorized access, theft, or environmental hazards.

4.2 Network security: protects communication channels and data traffic from cyberattacks, using tools like firewalls, intrusion detection systems (IDS), and secure network protocols [4].

4.3 Application security: focuses on identifying and mitigating software vulnerabilities through practices like secure coding, patching, and input validation. **Information or data security** ensures that sensitive data is properly handled, using encryption, access controls, and data classification techniques. **Endpoint security** protects end-user devices like laptops, smartphones, and IoT devices from malware, phishing, and unauthorized access [5].

Additionally, **cloud security** has become vital with the adoption of cloud computing, involving identity management, secure APIs, and compliance with international standards.

4.4 Operational security (OPSEC): supports the confidentiality of organizational processes by preventing leaks through routine procedures and employee behavior. These security layers work in unison to form a comprehensive and resilient defense system for modern IT environments.

5. Security Mechanisms:

Security mechanisms are the tools and methods used to protect information systems from attacks, unauthorized access, and data loss. These mechanisms help ensure the confidentiality, integrity, and availability of data.

One of the most important mechanisms is **encryption**, which hides information using special codes so only authorized users can read it. Another key mechanism is **authentication**, where users must prove who they are using passwords, ID cards, or biometrics (like fingerprints). **Access control** is also essential it makes sure that only certain users can view or change specific information based on their roles or privileges [4].

Firewalls and **intrusion detection systems (IDS)** are also common. Firewalls block unwanted traffic from entering a network, while IDS monitors for suspicious activity. Other helpful mechanisms include **data backups**, **digital signatures**, **multi-factor authentication (MFA)**, and **security policies** that guide users and administrators on how to stay safe online.

6. Security Challenges in IoT Environments:

6.1 Limited Computational Resources: Many IoT devices such as sensors and actuators are designed with minimal processing power and memory to save cost and energy. This makes it difficult to implement robust security features such as encryption.[6]
Example: A smart thermostat may lack the resources to run encryption algorithms.

6.2 Heterogeneous Devices and Lack of Standardization: IoT environments consist of devices with diverse capabilities and often lack common standards. This diversity creates difficulties in applying uniform and interoperable security solutions..[6]
Example: Devices from different manufacturers may use incompatible encryption or outdated protocols.

6.3 Distributed Nature and Scalability: The decentralized structure and growing scale of IoT networks make security monitoring and management increasingly complex..[6]
Example: A smart grid system involving thousands of sensors is challenging to update and secure efficiently.

6.4 Privacy and Data Protection Concerns: IoT devices collect and transmit sensitive personal data such as health and location information, which must be adequately protected..[6]
Example: A wearable health monitor transmitting unencrypted patient data over public Wi-Fi.

6.5 Insecure Interfaces and Communication: Poorly designed APIs or web interfaces and unencrypted communication channels expose IoT systems to cyberattacks.
Example: An insecure REST API in a smart lock allows attackers to bypass authentication.

6.6 Lack of Regular Software Updates: Many IoT devices do not receive timely software patches, leaving them vulnerable to known threats..[6]

Example: A smart speaker still running an outdated firmware susceptible to public exploits.

7. Security Risks in IoT Environments:

7.1 Botnet Attacks: Insecure devices can be hijacked into botnets for large-scale DDoS attacks..[6]

Example: The Mirai botnet exploited default credentials in IP cameras.

7.2 Ransomware: IoT devices are increasingly targeted by ransomware that disables functionality until a ransom is paid..[6]

Example: A smart home heating system locked during winter by ransomware.

7.3 Shadow IoT: Unauthorized devices connecting to corporate networks can introduce major security vulnerabilities.

Example: A smartwatch syncing data through enterprise Wi-Fi without approval.

7.4 Weak Passwords and Authentication: Many IoT devices still use default or weak login credentials, making unauthorized access easy..[6]

Example: Default username and password settings exploited in home routers.

7.5 Advanced Persistent Threats (APTs): Sophisticated long-term attacks may remain hidden while extracting sensitive data..[6]

Example: A compromised factory sensor transmitting data to an attacker over several weeks.

7.6 Lack of Compliance with Security Standards: Many IoT devices fail to meet required security regulations, increasing legal and operational risks..[6]

Example: A healthcare device failing to comply with HIPAA standards.[6]

8. IDS Challenges in IoT Environments:

Intrusion Detection Systems (IDS) face numerous challenges when implemented in Internet of Things (IoT) environments. Unlike traditional IT infrastructures, IoT networks are composed of a vast number of heterogeneous and resource-constrained devices that often lack the computational capabilities required to support traditional IDS mechanisms. One of the

primary challenges is resource limitation most edge devices operate with restricted CPU power, memory, and battery life, which complicates the deployment of sophisticated detection algorithms. Additionally, scalability presents a major obstacle, as IoT systems may involve thousands of interconnected devices transmitting data in real time. This results in overwhelming volumes of network traffic that are difficult to monitor continuously [8].

Moreover, the heterogeneity of IoT devices including different hardware architectures, operating systems, and communication protocols makes it difficult to design a unified IDS solution that works across all platforms. Real-time detection is also hindered by the decentralized and distributed nature of IoT environments; relying on centralized analysis introduces latency and potential single points of failure. Finally, the scarcity of standardized datasets and labeled attack records limits the ability to train and validate effective machine learning-based IDS models. To overcome these constraints, recent research efforts have focused on developing lightweight, distributed IDS frameworks that leverage edge/fog computing, artificial intelligence, and blockchain technologies to improve efficiency, scalability, and trust [7].

Table 1.1 – Summary of IoT Security Challenges:

Challenge	Description	Example
Limited Resources	IoT devices lack CPU, memory, and battery	Smart thermostat can't run encryption
Heterogeneous Devices	Different protocols and standards	Devices with incompatible firmware
Insecure Interfaces	APIs or web interfaces vulnerable	Smart lock with insecure REST API
Lack of Updates	Devices not receiving patches	Smart speaker on outdated firmware

8.1 Why We Need Modern Intrusion Detection Systems (IDS)

In parallel with the technical challenges, the growing complexity and frequency of cyberattacks has rendered traditional IDS approaches especially those based purely on signature detection insufficient in the context of modern IoT ecosystems. Smart devices today produce massive amounts of real-time data and are increasingly vulnerable to advanced persistent threats, zero-day attacks, and behaviorally complex intrusions that evade legacy defense systems.

Modern IDS solutions are therefore essential. They enable real-time monitoring and classification of network traffic, detect unknown attacks that do not match predefined signatures, and adapt to dynamic network behavior using intelligent techniques. These systems often integrate machine learning and anomaly-based detection methods to uncover hidden patterns and suspicious behaviors that traditional IDS may miss. By employing adaptive learning models and behavioral analysis, modern IDS enhance the detection of novel threats and provide a more resilient defense mechanism for securing interconnected smart environments.[7]

9. Intrusion Detection Systems: Traditional Models vs. the Proposed Architecture:

Intrusion Detection Systems (IDS) play a critical role in protecting computer systems and networks from unauthorized access and malicious attacks. Traditionally, IDS approaches are categorized into two main types: Network-Based IDS (NIDS), which monitors network traffic at routers and switches, and Host-Based IDS (HIDS), which inspects activity and log files on individual devices [1][2]. These systems rely on different detection strategies. Signature-based (misuse) detection matches known patterns of attacks, providing high accuracy for recognized threats but failing to detect new or evolving ones [3]. Anomaly-based detection, on the other hand, identifies deviations from normal behavior, allowing for the detection of novel attacks but often producing higher false positive rates [2][4]. A third method, stateful protocol analysis, examines the status of communication protocols across multiple layers, but it requires deep technical configuration and significant processing resources [1][5].

However, the increasing sophistication and volume of cyber threats, particularly in complex and heterogeneous environments like the Internet of Things (IoT), have highlighted the limitations of traditional IDS methods. In response, this research introduces a proposed hybrid architecture designed specifically for securing IoT environments within Industry 4.0 settings. Rather than representing a general modern IDS model, the proposed system is a customized solution that combines Machine Learning (ML) techniques with blockchain-based alert management in a decentralized, edge-centric framework.

The architecture categorizes IoT devices into two classes: Strong Devices, which have sufficient computational resources to locally execute ML-based attack detection, and Weak Devices, which are resource-constrained and depend on alerts from the system. When an

attack is detected by a Strong Device, an alert is generated and sent to a nearby edge gateway. These alerts are then propagated securely through a lightweight blockchain ledger, ensuring their authenticity, integrity, and resistance to tampering. Weak Devices react to these alerts by disconnecting or suspending their operations, while gateways ensure real-time, synchronized distribution across the network.

Additionally, the system supports automatic filtering and cancellation of false alerts, which enhances reliability and minimizes disruptions to normal operations. This tailored approach reduces latency, eliminates dependence on cloud infrastructure, and provides a scalable, resilient security mechanism for real-time IoT applications. As such, the proposed IDS architecture represents a specialized, next-generation model that integrates intelligent ML-based anomaly detection with the trust, decentralization, and transparency provided by blockchain technology.

Table 1.2 – Comparison of Traditional IDS vs. Modern IDS in IoT:

Feature	Traditional IDS	The Proposed IDS Architecture
Architecture	Centralized	Decentralized/Edge
Detection Type	Signature-based	ML-based / Hybrid
Real-Time Response	Limited	Limited
Adaptability	Low	High with AI/ML

10. Key Technologies:

10.1 Definition of the Internet of Things (IoT):

The Internet of Things (IoT) refers to a network of physical devices that can connect to the internet and communicate with each other, enabling them to perform tasks autonomously and efficiently. These devices include smart technologies such as wearables, smart homes, health monitoring devices, sensors, and cameras that collect environmental and health-related data.

According to the International Telecommunication Union (ITU), the Internet of Things is defined as "a global infrastructure serving the information society," which enables "the provision of advanced services by interconnecting physical and virtual objects through the interoperability of existing or evolving information and communication technologies" (ITU-T Y.2060, 2012) [9]. This definition emphasizes the broad scope of IoT, focusing on both physical devices and virtual systems that work together seamlessly.

The Institute of Electrical and Electronics Engineers (IEEE) also provides a definition, stating that IoT is "a network of elements, each equipped with sensors, that are connected to the Internet" (IEEE IoT Initiative, 2013) [10]. This simple definition highlights the fundamental role of sensors and connectivity in creating an intelligent network of devices.

In addition, the Cluster of European Research Projects on the Internet of Things (CERP-IoT) defines IoT as "a dynamic infrastructure of a global network, with self-configuration capabilities based on standards and interoperable communication protocols. In this network, physical and virtual objects have identities, physical attributes, virtual personalities, and intelligent interfaces, and they are integrated into the network transparently" (CERP-IoT Vision, 2008) [11]. This definition emphasizes the self-configuring, interoperable nature of IoT systems, which are essential for seamless communication across different platforms.

10.2 Definition of Industrial Internet of Things (IIoT):

The Industrial Internet of Things (IIoT) refers to the integration of Internet of Things (IoT) technologies into industrial environments, where physical devices such as sensors, machines, and equipment are connected to the internet and to each other in order to enhance automation and operational efficiency [6]. This networked infrastructure enables advanced data collection and real-time communication, playing a key role in domains such as manufacturing, energy management, transportation, and industrial automation [12].

IIoT systems leverage machine-to-machine (M2M) communication, big data analytics, and machine learning to generate valuable insights from massive datasets, enabling predictive maintenance, intelligent decision-making, and efficient process control [13][14]. These capabilities support the evolution of Industry 4.0, characterized by smart factories and cyber-physical systems [13].

Due to their critical role in managing high-value assets and mission-critical operations, IIoT systems are particularly vulnerable to cybersecurity threats. As a result, they require robust protection mechanisms to ensure data integrity, system reliability, and operational safety. [14]

10.3 Definition of Cloud computing:

Cloud computing is a technology that allows users and companies to store, manage, and access their data and applications on remote servers over the internet. Instead of keeping

everything on local computers, the data is **sent to secure data centers** managed by specialized providers. These centers often include **video surveillance, access control, and advanced security measures** [15].

The main goal of cloud computing is to make IT services more **flexible**, **scalable**, and **cost-effective**. It works by sending client data through the internet to powerful servers that store, process, and protect the information. The cloud provider is responsible for **keeping the data safe**, **maintaining the infrastructure**, and **ensuring high availability**.

Cloud computing is offered at different levels:

- **IaaS (Infrastructure as a Service):** Only the hardware infrastructure is provided. The client manages the software and applications.
- **PaaS (Platform as a Service):** The provider offers the hardware and software tools needed for development.
- **SaaS (Software as a Service):** The most common model where everything including software, storage, and maintenance is managed by the provider.

Cloud computing is especially useful for businesses and government organizations that need reliable, on-demand access to data and services without having to manage the physical infrastructure themselves [15].

10.4 Definition of Edge Computing:

Edge computing is a computing paradigm that brings computation and data storage closer to the sources of data, such as sensors and smart devices, instead of relying solely on centralized cloud data centers. This approach reduces latency, increases processing efficiency, and enables real-time data handling, which is critical for applications such as autonomous vehicles, smart healthcare systems, and industrial automation. By minimizing the need to send data to remote servers, edge computing also enhances privacy and reduces bandwidth usage [19][7].

10.5 Definition of Fog Computing:

Fog computing extends cloud computing by introducing a decentralized layer of computing infrastructure between edge devices and the cloud. This layer distributes computing, storage, and networking resources geographically to support real-time processing of massive data

streams from IoT devices. Fog nodes can be located anywhere along the cloud-to-things continuum, facilitating local decision-making and providing enhanced security, lower latency, and improved quality of service [7].

10.6 Hierarchical Relationship Between IoT, Edge/Fog Computing, and Cloud :

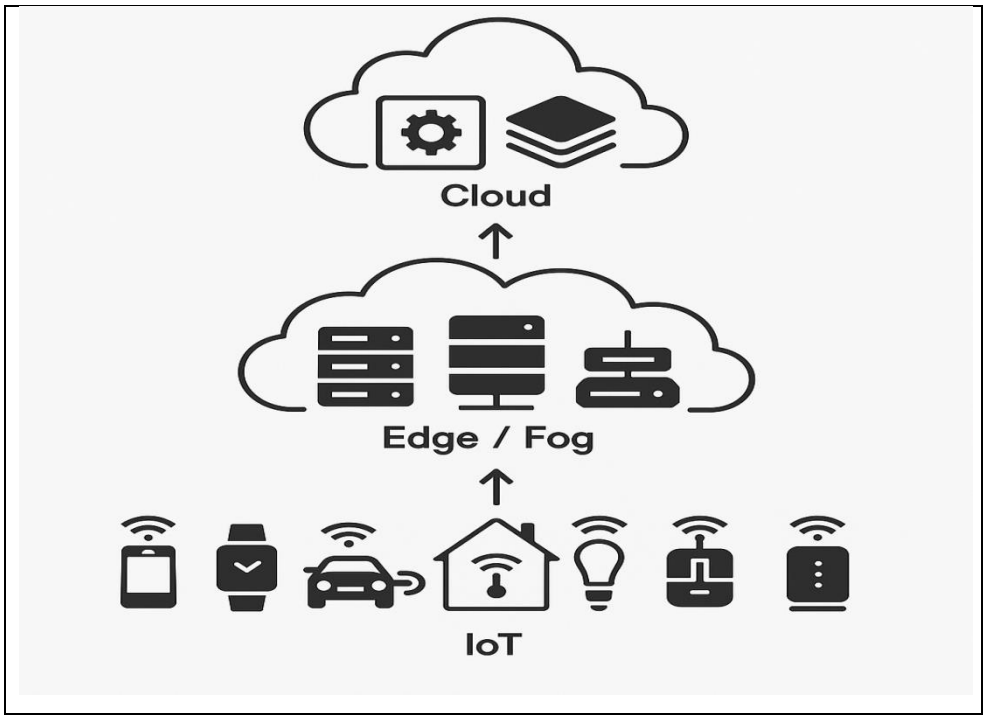


Figure 10.6: Hierarchical Relationship Between IoT, Edge/Fog Computing, and Cloud

The diagram titled **"Hierarchical Relationship Between IoT, Edge/Fog Computing, and Cloud"** provides a visual representation of how data and computational responsibilities are structured across the layers of a modern IoT ecosystem. At the base are **IoT devices**, such as smartwatches, cars, home automation systems, lights, and health monitors these are the sources of raw data. Above them, the **Edge/Fog computing layer** acts as a local intermediary that performs real-time processing, filtering, and preliminary analysis of the data closer to its origin, which helps reduce latency and bandwidth usage. Finally, the **Cloud** layer handles more intensive tasks like big data analytics, long-term storage, and centralized management.

This hierarchy is crucial for securing and scaling IoT systems effectively. It reflects the layered architecture discussed in Chapter 1 of your literature review, particularly in sections 10.3 to 10.5, which explain how **cloud computing**, **edge/fog computing**, and **blockchain**

contribute to efficient and secure IoT operations. By distributing tasks appropriately across layers, this architecture supports real-time decision-making, improves system responsiveness, and enhances data security key requirements in Industry 4.0 environments.

10.7 Definition of Blockchain for Security in IoT:

The integration of blockchain technology into the Internet of Things (IoT) offers a promising solution to address the security and privacy challenges inherent in IoT ecosystems [6]. Blockchain's decentralized and immutable ledger ensures that data exchanged between IoT devices is secure, transparent, and tamper-resistant [7]. By eliminating the need for centralized authorities, blockchain enhances trust among devices and reduces vulnerabilities associated with single points of failure [7]. Moreover, smart contracts can automate and enforce security policies, ensuring that only authorized devices participate in the network [7].

10.8 Definition of Industry 4.0 Context:

Industry 4.0 refers to the fourth industrial revolution, which is characterized by the integration of cyber-physical systems (CPS), the Internet of Things (IoT), big data analytics, and artificial intelligence (AI) into manufacturing processes and industrial infrastructure. This revolution transforms traditional factories into smart factories that are highly automated, efficient, and interconnected. However, this digital transformation also increases the attack surface, making these systems more vulnerable to cyber threats. Therefore, securing Industry 4.0 environments requires advanced cybersecurity solutions, such as Intrusion Detection Systems (IDS), to monitor and protect critical assets and data flows [18].

11. Role of AI and ML in Security:

Artificial Intelligence (AI) and Machine Learning (ML) are becoming essential in cybersecurity, offering intelligent solutions that go beyond traditional rule-based systems. These technologies allow security systems to automatically detect, analyze, and respond to threats in real-time [7][19].

AI can process huge volumes of data from logs, devices, and networks to identify suspicious patterns. ML algorithms are capable of detecting anomalies, such as unexpected logins, data

transfers, or behavior changes, which may signal malware, phishing attempts, or insider threats [7][19][22].

One of the main advantages of AI is its ability to continuously learn and adapt, making it effective in identifying zero-day vulnerabilities and advanced persistent threats (APTs) without prior knowledge. This makes AI-powered systems more resilient and proactive [7].

AI and ML also play a key role in automated incident response, where systems can take immediate action such as isolating infected machines or blocking malicious IPs without waiting for human input. This significantly reduces reaction time and damage [22].

In addition, AI tools assist in threat intelligence analysis by scanning dark web sources, security feeds, and online forums to detect emerging threats, allowing organizations to prepare in advance [19].

These technologies are already used in antivirus engines, intrusion detection systems (IDS), fraud prevention platforms, and identity verification solutions. While AI improves efficiency and accuracy, it also requires careful tuning to reduce false positives and defend against adversarial manipulation [22].

12. Related Work and Existing IDS Solutions:

Numerous research efforts have focused on enhancing the security of IoT systems using Intrusion Detection Systems (IDS). Traditional IDS models have shown effectiveness in conventional IT environments, but their performance often degrades in resource-constrained and highly distributed IoT settings.

For example, **Meidan et al. (2018)** introduced N-BaIoT, a network-based behavioral analysis IDS tailored for IoT devices using deep autoencoders to detect malware activity [23]. Similarly, **Doshi et al. (2018)** proposed a lightweight anomaly-based IDS using supervised machine learning techniques that can operate efficiently on IoT devices with limited resources [24].

Recent works have explored the integration of edge computing to reduce detection latency. The **Fog2Sec framework** distributes IDS tasks across fog nodes to enable faster response times and localized threat analysis [25].

Blockchain-based IDS architectures have also gained attention. In 2021, **Sharma et al.** proposed a decentralized security framework that leverages blockchain to store alerts and audit logs in a tamper-proof ledger, enhancing the trust and integrity of detection results [26].

Hybrid systems that combine signature-based and anomaly-based detection techniques are particularly effective. For example, the **Hybrid-IDS framework** utilizes ensemble learning (Random Forest, XGBoost) to improve detection accuracy while leveraging lightweight agents for deployment on constrained IoT nodes [27].

A comprehensive survey by **Albakri et al. (2022)** reviews these emerging approaches and emphasizes the role of machine learning and blockchain in improving the adaptability, transparency, and decentralization of IDS systems for IoT environments [7].

Despite these advancements, challenges remain in balancing detection performance with resource consumption, reducing false positives, and ensuring scalability in large-scale industrial IoT deployments. These gaps motivate the development of new IDS architectures that integrate modern technologies like machine learning, blockchain, and edge computing such as the one proposed in this thesis.

13. Conclusion:

As the Internet of Things continues to expand across various sectors, including critical infrastructures and industrial environments, ensuring the security of these systems becomes a top priority. This chapter has presented a comprehensive overview of the key challenges facing IoT security, including limited device capabilities, lack of standardization, and growing cyber threats.

Intrusion Detection Systems (IDS) remain a cornerstone of IoT security strategies, and their evolution from traditional models to modern, intelligent systems is crucial for keeping pace with new and complex threats. The integration of edge and fog computing, blockchain, and machine learning into IDS architectures has shown great promise in enabling real-time, scalable, and decentralized threat detection [24].

Furthermore, the shift toward Industry 4.0 highlights the need for secure, interconnected environments where automation, AI, and IoT converge. The proposed research in this thesis

aims to build upon these modern approaches by developing a hybrid, edge-centric IDS that leverages blockchain for transparency and ML for intelligent anomaly detection.

The next chapter will provide a detailed overview of the system design, architecture, and methodology of the proposed IDS model, including its key components, data processing flow, and experimental setup.

Chapter 2: Proposed Edge-Centric IDS Architecture

1. Introduction:

The evolution of the Internet of Things (IoT) has introduced significant challenges in securing large-scale, heterogeneous, and resource-constrained networks. This chapter introduces a decentralized and resilient system architecture for Intrusion Detection Systems (IDS) specifically tailored to the IoT context. Traditional cloud-based security models often introduce latency, centralization risks, and single points of failure limitations this architecture seeks to overcome.

By distributing detection intelligence to the edge, the system enables real-time analysis, local response, and scalable integration with minimal reliance on cloud infrastructure. Strong and weak IoT devices are categorized and utilized according to their capabilities, while gateways serve as intermediaries for communication and blockchain-based alert propagation. The architecture is designed to ensure data privacy, operational efficiency, and security integrity, making it well-suited for industrial, urban, and mission-critical IoT environments.

2. Overview of the Proposed Architecture:

2.1 Architecture Philosophy :

The proposed IDS architecture embraces a decentralized and cloudless design to better suit the constraints and security requirements of IoT environments. Unlike traditional architectures that rely heavily on centralized cloud platforms, this model shifts intelligence and decision-making to the network's edge closer to the IoT devices. This edge-centric approach reduces latency, enhances data privacy, and improves system resilience.

By eliminating a single point of failure, decentralization ensures that the system remains operational even if part of the network is compromised or disconnected. The architecture is further strengthened by integrating blockchain technology, which guarantees the integrity and traceability of alert messages. In addition, by processing and reacting to threats locally, the system avoids unnecessary data transmission to the cloud, making it ideal for Industry 4.0 and other mission-critical applications.This architectural vision is illustrated in the following diagram, which outlines the core components and information flow of the proposed decentralized IDS model.

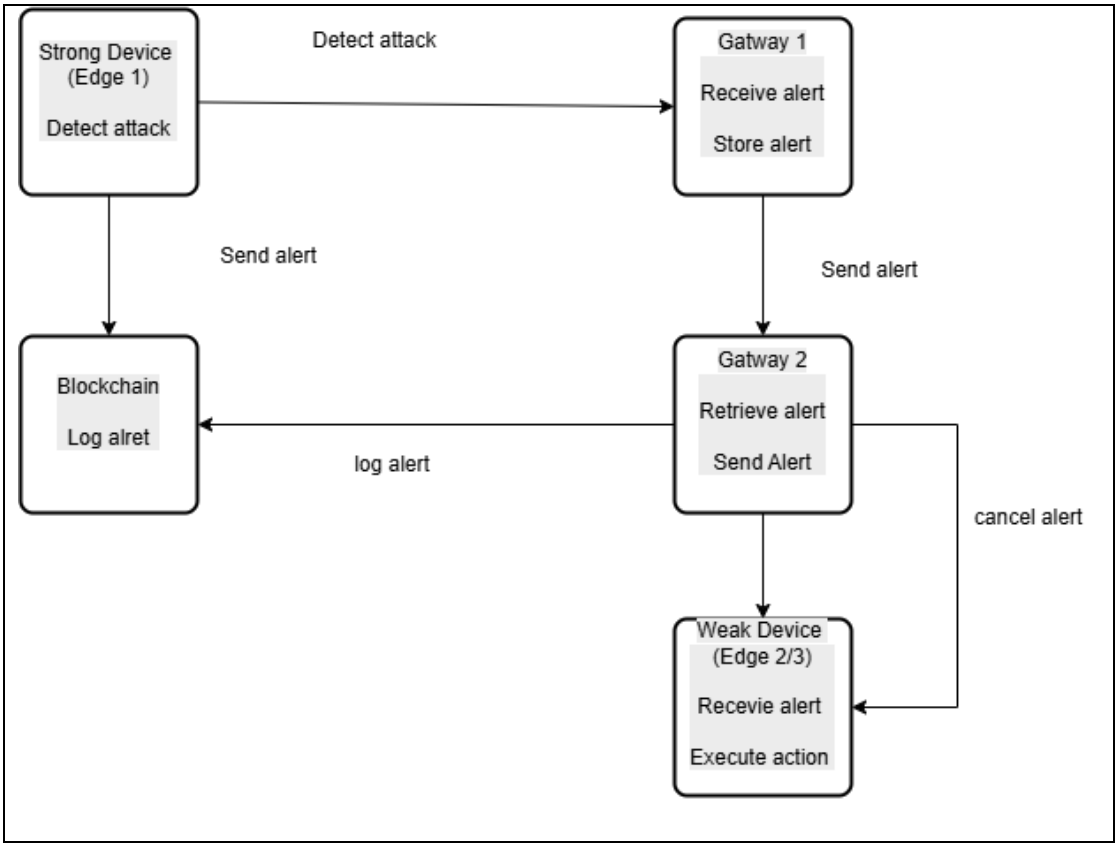


Figure 2.1: Architectural Flow Diagram of the Proposed Decentralized IDS

Figure 2.1 illustrates the overall architecture of the proposed decentralized IDS for IoT environments. The workflow begins with strong edge devices that use machine learning models to detect intrusions locally. Upon detecting an attack, an alert is generated and forwarded to Gateway 1, which securely records it in a lightweight blockchain. Gateway 2 then reads the alert from the blockchain and propagates it to connected weak devices, which execute predefined responses such as isolation or shutdown. This architecture enables low-latency, tamper-proof alert distribution without reliance on centralized cloud services, making it suitable for large-scale, real-time IoT deployments.

2.2 General Description of Components:

- **Strong IoT Devices:** These are capable nodes such as Raspberry Pi or Jetson Nano, equipped with enough computing resources to run machine learning models. They continuously monitor network traffic and execute the intrusion detection process locally. Upon detecting an attack, they generate an alert.

- **Weak IoT Devices:** These are legacy or resource-constrained devices like cameras or sensors. While they cannot perform detection, they can respond to alerts (by disconnecting, blocking IPs, or shutting down temporarily).
- **Gateways (Local Relays):** These nodes serve as intermediaries between IoT devices and the blockchain network. Gateway 1 receives alerts from strong devices and records them on the blockchain. Gateway 2 retrieves alerts from the blockchain and propagates them to other devices.
- **Blockchain Network:** This component ensures the secure distribution and integrity of alerts. Implemented using a lightweight SQLite-based blockchain simulation, it prevents tampering and allows alerts to be canceled in case of false positives.

3. Classification of IoT Devices:

The Internet of Things (IoT) ecosystem comprises a wide variety of devices, each with distinct resources and capabilities. In the proposed architecture, we classify these devices into two main categories: strong devices and weak devices. This classification plays a crucial role in determining where to place intelligence and detection functions, and how to design communication between system components.

3.1 Strong Devices: Capabilities and Role

Strong devices are IoT nodes with enough processing power, memory, and an operating system. They can run complex machine learning (ML) models to detect cyberattacks. Examples include **Raspberry Pi**, **Jetson Nano**, or similar edge computers.

These devices act as the first line of defense. They continuously monitor network traffic and make real-time decisions using ML. If an attack is detected, they generate an alert and send it to a connected gateway. These devices may also respond by blocking IP addresses or updating firewall rules.[32]

3.2 Weak Devices Limitations and Defensive Behavior:

Weak devices are small, resource-limited sensors or actuators. They cannot run ML models or perform complex computations. Examples include **cameras**, **temperature sensors**, and **motion detectors**.

Although they cannot detect attacks, they can **respond** when they receive alerts. For example, a weak device can shut down temporarily, block communication with certain IPs, or reduce functionality until the threat is cleared.

Example: A smart camera receives an alert from the gateway and disables its network connection to avoid being compromised.

3.3 Interaction and Cooperation between Devices:

In this architecture, strong and weak devices work together. Strong devices are responsible for detecting threats, while weak devices follow instructions to stay safe. Gateways help in communication by sending alerts from strong devices to others using the blockchain network.

This cooperation allows the system to use all devices efficiently. Even if weak devices cannot detect attacks, they still play an important role in reducing damage and protecting the network.

Edge gateways are important parts of the proposed IDS architecture. They connect IoT devices to the blockchain and help with sending and receiving alerts. In our system, there are two main types of gateway functions: writing alerts and reading alerts.

4. Edge Gateway Functionality:

4.1 Role in Detection and Propagation:

Gateways act as **intermediaries** between strong devices and the rest of the network. When a strong device detects an attack, it sends an alert to **Gateway 1**. This gateway then records the alert in the blockchain system.

Another gateway, called **Gateway 2**, reads alerts from the blockchain. It sends these alerts to other connected devices, including weak devices that need to respond.

Gateways do not detect attacks themselves, but they help spread alerts to the network in a secure and fast way.

Example: Gateway 1 receives an alert from a Raspberry Pi and stores it in the blockchain. Gateway 2 reads that alert and sends it to all nearby weak devices.

4.2 Communication with Blockchain:

Gateways communicate with a **lightweight blockchain** that is implemented using SQLite. This blockchain is used to:

- Store alerts in a secure and tamper-proof way.
- Share alerts with all other gateways and devices.
- Allow cancellation of false alerts, if needed.

By using blockchain, gateways ensure that the alert messages are not changed or blocked by attackers. Every alert is recorded and trusted across the system.

4.3 Load and Reliability Considerations:

Gateways must handle communication between multiple devices and the blockchain. They should be lightweight and able to:

- Work even with limited resources.
- Remain available and stable under network load.
- Quickly process and forward alerts in real time.

In some cases, multiple gateways can be deployed to achieve better load distribution and enhance fault tolerance, thereby improving system stability and ensuring continuous, efficient operation even if one gateway fails.

5. Blockchain Integration:

In the previous chapters, we discussed how IoT devices are categorized into strong and weak nodes, and how edge gateways serve as communication bridges within the proposed IDS architecture. To ensure that security alerts generated by strong devices are reliably stored, shared, and verified across the network, a robust and tamper-resistant system is essential.

This chapter introduces the integration of **blockchain technology** into the IDS architecture. Blockchain acts as a secure, distributed ledger for recording alerts, enabling transparent communication between gateways and ensuring the integrity and traceability of detection

events. We explore how blockchain strengthens the trust and coordination among devices, even in distributed and resource-constrained environments.

5.1 Purpose and Benefits in IDS:

The main reason for using blockchain in the IDS is to **secure the alert messages**. Once an alert is saved in the blockchain, it cannot be changed or deleted. This prevents attackers from hiding or modifying alerts after an attack.

Key benefits include:

Integrity: Alerts stay exactly as they were recorded and cannot be altered.

Transparency: All gateways and devices can access and verify the alerts.

Decentralization: No single entity controls or owns the alert history.

Trust and Authentication: Blockchain can also be used to verify the source of alerts and authenticate communicating devices [28].

Example: If a strong device sends a false alert by mistake, the blockchain still keeps a record. Other gateways can later verify and cancel it if needed.

Recent studies confirm that blockchain enhances IDS trust by eliminating single points of failure and ensuring data consistency across distributed networks [28].

5.2 Simulated vs. Real Blockchain (SQLite Architecture):

In our experiment, we use a **simulated blockchain** implemented with **SQLite**, a lightweight relational database. This version allows us to test the idea without needing a full blockchain network.

Simulated blockchain (SQLite):

- Easy to build and test in small-scale experiments.
- Stores alerts in a chained structure, similar to real blockchain blocks.
- Supports basic features such as sequential alert logging and retrieval

Real blockchain:

- Provides advanced security features like cryptographic hashes, consensus algorithms, and smart contracts.
- Can operate in public or private networks.
- Suitable for high-security industrial IoT environments where integrity and decentralization are critical [28].

5.3 Alert Integrity, Tamper-Proofing, and Traceability:

One of the biggest advantages of blockchain is that it protects the **integrity** of alerts. Once a gateway writes an alert to the blockchain, it becomes **tamper-proof**. No attacker or device can change it not even the gateway that created it.

In addition, every alert is **traceable**. It includes metadata like timestamp, device ID, and gateway ID. This improves **auditability**, making it easier to analyze security events and take corrective action.

Some blockchain systems even support alert prioritization, trust scoring, and consensus validation to reduce false positives and coordinate responses among devices [28].

Example: An alert created by "Edge Device 1" at 10:01 AM will have a unique and verifiable entry in the chain. This entry can be read by other gateways and used to coordinate defense actions in the network.

6. IDS Detection Workflow:

This section outlines the complete detection and response process within the proposed IoT Intrusion Detection System (IDS). The workflow ensures that every attack detection event is processed, propagated, and responded to in a decentralized and trustworthy manner, using edge devices, gateways, and blockchain technology.

6.1 Alert Generation by Strong Devices:

The detection process begins with a strong edge device, such as a Raspberry Pi or Jetson Nano, which monitors real-time network traffic. These devices are equipped with pre-trained

machine learning models that analyze extracted features from the traffic and identify malicious activity. When an attack is detected (SQL Injection or Brute Force), the device immediately generates an alert message. This alert contains metadata such as the device ID, attack type, timestamp, and suggested mitigation action. It is then sent to Gateway 1 for registration.[33]

6.2 Blockchain-Based Propagation

Upon receiving the alert, Gateway 1 records it on a tamper-resistant blockchain implemented using SQLite. This ensures that every alert is stored immutably and can be traced reliably. Gateway 2 periodically queries this blockchain to extract new active alerts and disseminate them across the network to relevant nodes, especially weak devices. This approach eliminates the need for central coordination and strengthens the trust model of the IDS.

6.3 Response by Weak Devices:

Weak devices, which are typically resource-constrained and unable to perform detection themselves, play a crucial role in mitigating threats once they receive alerts. Based on the type of attack received from Gateway 2, these devices execute predefined actions such as disconnecting from the network, blocking specific IP addresses, or switching to a secure/low-power mode. This coordinated reaction ensures that the system contains the threat even in decentralized or low-capacity environments.

6.4 False Alert Handling and Rollback:

Despite the high accuracy of machine learning models used, false positives may occur. To address this, administrators can issue rollback instructions to cancel previously recorded alerts. These cancellations are written into the blockchain by updating the alert’s status. Gateway 2 then reads the rollback entries and forwards rollback messages to weak devices, prompting them to reverse their mitigation actions and resume normal operation. This adds flexibility and resilience to the system by allowing controlled correction of detection errors.

The figure below demonstrates the simulation dashboard showing each step of the workflow in action, including alert generation, propagation, blockchain logging, and device response.[34]

7. Detailed Architecture Design:

This section presents the detailed structure and internal behavior of the proposed IoT-based Intrusion Detection System (IDS), as illustrated in **Figure 1**. It highlights how the different system components interact, operate in a decentralized environment, and collaborate to detect, propagate, and respond to threats in real time.

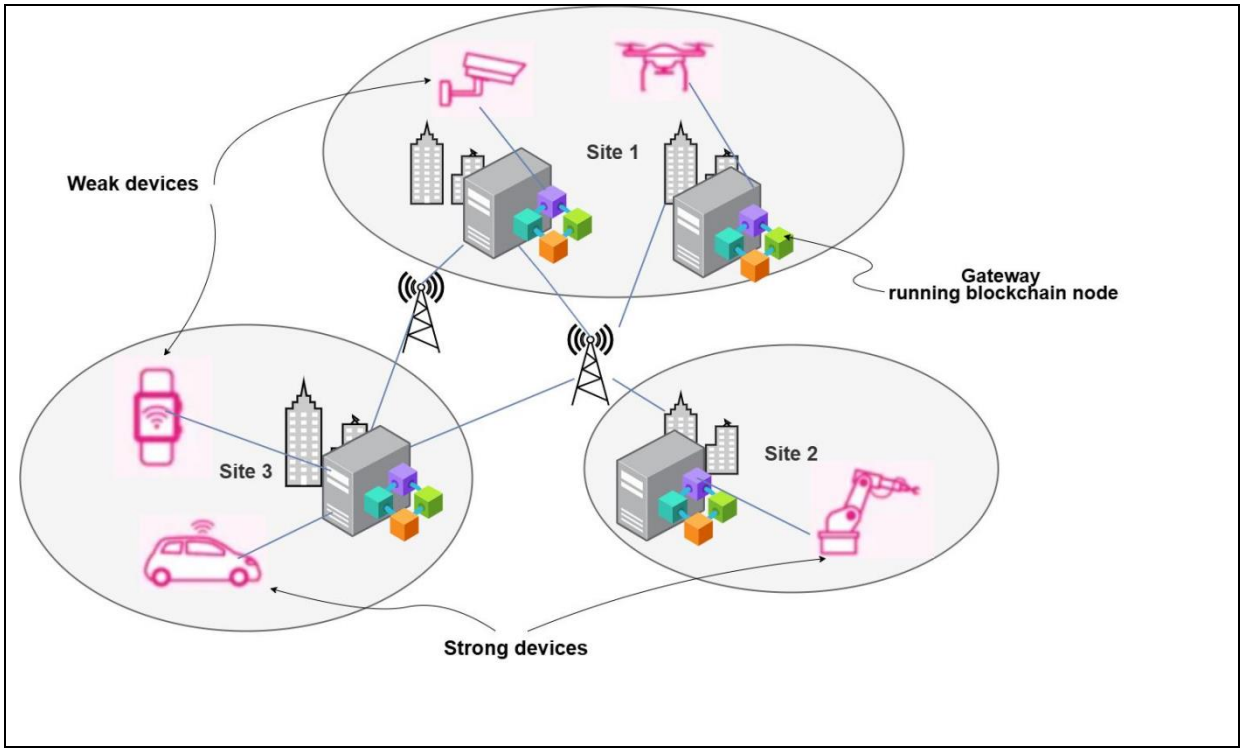


Figure 7: Schema illustrates the connections between the different components that make up our IDS system.

7.1 High-Level Architecture and Core Components:

The proposed IDS is built on a decentralized edge computing model to ensure scalability, low-latency response, and resilience against failure or tampering. The system is composed of the following core components:

7.1.1 Strong Devices (Edge 1): These nodes (Raspberry Pi, Jetson Nano) are equipped with machine learning models and perform real-time intrusion detection. They analyze extracted features from live traffic and generate alerts when suspicious activity is detected.

7.1.2 Gateway 1: This gateway receives alerts from strong devices and stores them immutably in a lightweight blockchain (SQLite). It acts as the bridge between detection and secure record-keeping.

7.1.3 Blockchain Node: Implements a tamper-proof alert log. It ensures transparency, traceability, and auditability of every security event in the system.

7.1.4 Gateway 2: Monitors the blockchain for new alerts and forwards them to relevant edge nodes, including weak devices. It also handles rollback signals if an alert is canceled.

7.1.5 Weak Devices (Edge 2/3): These are low-capacity IoT nodes (cameras, sensors) that cannot perform detection but respond to alerts with predefined defensive actions like IP blocking, shutdown, or isolation.

7.2 Component Schematics and Internal Functions

Each component of the IDS performs specific internal operations:

7.2.1 Strong Device (Edge 1):

Modules*:* Feature Extractor, GA-based Feature Selector, ML Classifier, Alert Generator

Function*:* Detect attacks and forward structured alerts to Gateway 1.

7.2.2 Gateway 1:

Modules*:* Alert Receiver, Blockchain Writer

Function*:* Accept alerts and store them in a tamper-proof manner using SQLite blockchain.

7.2.3 Blockchain Node:

Structure: Append-only table with fields like timestamp, attack type, device ID, and status (active/cancelled)

Function*:* Store alerts and allow both retrieval and rollback operations.

7.2.4 Gateway 2:

Modules*:* Blockchain Monitor, Alert Broadcaster, Rollback Handler

Function*:* Propagate alerts to weak devices and send rollback instructions when needed.

7.2.5 Weak Device (Edge 2/3):

Modules: Alert Listener, Defense Executor, Rollback Manager

Function: Receive alerts, execute defensive actions, and reverse actions upon cancellation.

7.3 Sequence Diagram of Alert Lifecycle:

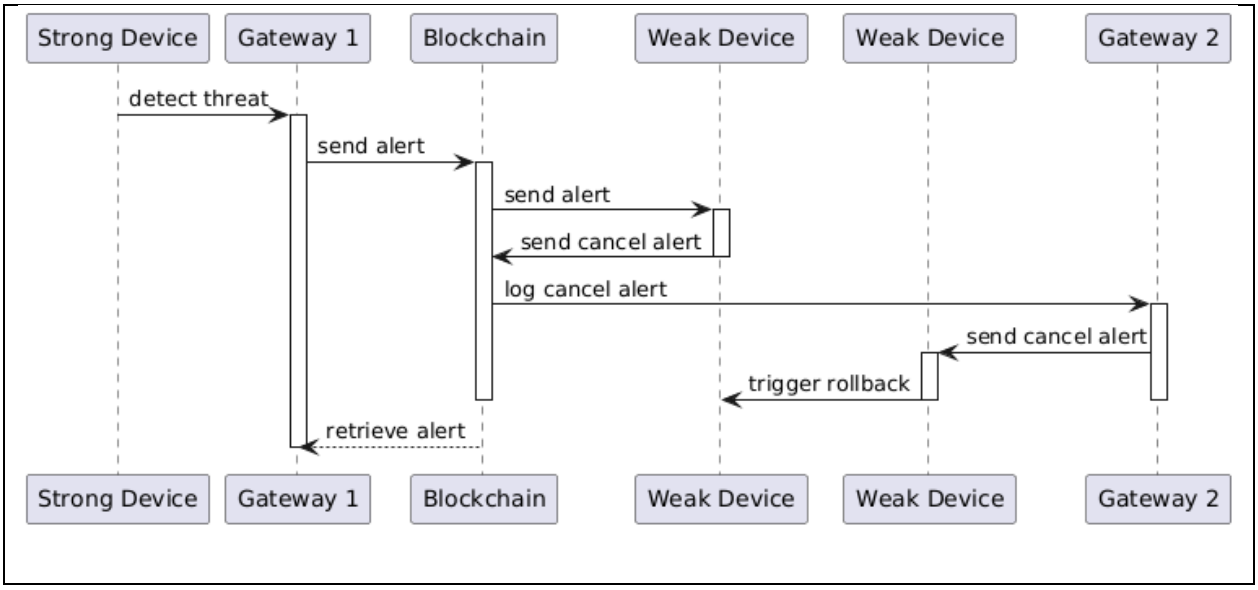


Figure 7.2: UML Sequence Diagram showing the alert lifecycle from detection to response in the decentralized IDS

This diagram shows the dynamic interaction between the key system components. The Strong Device initiates the process by detecting a threat and sending an alert to Gateway 1, which logs it in the blockchain. Gateway 2 later reads this alert and notifies relevant Weak Devices, which then perform predefined mitigation actions. In case of a false alert, the administrator triggers a rollback that is propagated through the same sequence, ensuring consistency and error correction across the system.

8. Scalability and Deployment Considerations:

Following the detailed architectural design and the integration of security mechanisms such as tamper-proof alert logging and decentralized communication, this section explores how the proposed system can be effectively deployed in real-world IoT environments.

It focuses on the system's scalability across various network sizes and topologies, as well as its ability to meet real-time requirements and maintain high availability, even in the presence of node or gateway failures.

8.1 Network Size and Node Distribution:

The system is designed to be modular and horizontally scalable, supporting deployments ranging from small local networks (home automation) to large industrial IoT infrastructures.

- **Scalability via additional Strong Devices**: As the number of monitored devices increases, additional strong nodes can be added to perform parallel detection tasks.
- **Distributed Gateways**: Gateways can be deployed per subnet or physical zone to reduce load and improve responsiveness.
- **Hierarchical deployment**: In large-scale systems, edge-level gateways can be connected to regional coordinators that sync alerts across industrial sites.

For example, in a smart factory with multiple sections, each section can have a dedicated Strong Device and local gateway, all writing to a shared blockchain replica.

8.2 Real-time Constraints

IoT security often requires instantaneous detection and response. The system is optimized to minimize latency in the following ways:

- **On-device detection**: ML predictions are performed locally on Strong Devices using lightweight models like **Random Forest** or optimized **XGBoost**. This avoids delay from cloud-based APIs.

- **Local blockchain node**: SQLite-based blockchain runs on the same machine as Gateway 1, eliminating remote DB access latency.
- **Alert propagation delay**: Gateways use polling intervals (e.g., every 2 seconds) to fetch and forward alerts. This can be adjusted based on system criticality.

8.3 Fault Tolerance and Redundancy:

The architecture embeds redundancy at multiple levels to ensure high availability and fault tolerance:

- **Multiple Gateways**: Deploying redundant Gateway 1 and Gateway 2 nodes prevents failure in alert logging or broadcasting.
- **Stateless Communication**: Alerts are pushed as independent HTTP messages. If a weak device is temporarily offline, it can still receive future alerts.
- **Blockchain consistency**: Even if a gateway fails, the blockchain remains intact and auditable.

Example: If Gateway 2 fails, a backup instance can be launched on a different port to continue pulling from the blockchain without disrupting the system.

9. Security and Privacy Considerations:

Security is a foundational element of the proposed architecture. Given the critical nature of IoT systems especially in industrial or mission-critical environments ensuring trust, authentication, and protection against manipulation is essential. This section outlines how the system addresses these goals through blockchain-backed mechanisms and secure communication protocols.

9.1 Blockchain-Backed Trust Model

The use of a blockchain ledger (simulated with SQLite) forms the core of the system's trust model. Every alert generated by a strong device is stored immutably in the blockchain, creating a verifiable and tamper-proof history of security events.

Key trust guarantees include:

- **Immutability**: Once written, an alert cannot be altered or deleted.
- **Transparency**: All participating gateways and devices can read the full alert history.
- **Decentralization**: Trust is distributed no single point of failure or authority.

This design makes it nearly impossible for attackers to erase traces of their activity or forge alerts without detection. Even in the case of false alerts, rollback entries are added transparently, preserving the audit trail.

As noted in [29], blockchain technology is particularly suited for distributed IoT systems due to its ability to provide integrity and decentralization across untrusted environments.

9.2 Gateway Authentication and Secure Channels

To ensure the integrity of alert data and prevent unauthorized or malicious devices from injecting false alerts into the system, every gateway must undergo a strict authentication process before it is allowed to transmit alerts.

In the current simulation, simple device IDs are used for identification. In production-level deployments, this can be strengthened using:

- Public-private key cryptography
- TLS encryption for HTTP requests
- Signed alert payloads or API tokens

According to [30], mutual authentication and secure channel establishment are crucial for preventing man-in-the-middle attacks in IoT infrastructures. Communication between strong devices, gateways, and the blockchain should ideally occur over **TLS/HTTPS** or using encrypted message formats like **JWT** or **MQTT with SSL**.

9.3 Protection Against Alert Spoofing

One of the critical threats in any IDS is **alert spoofing**, where an attacker attempts to inject fake alerts into the system to trigger false reactions or create confusion.

The proposed system defends against this via:

- Write access control on the blockchain (only Gateway 1 can write).
- Device ID validation and timestamp analysis to track alert origin.
- Optional digital signatures to verify that alerts came from trusted devices.

As discussed in [31], implementing strict identity verification and using signed messages significantly reduces the risk of spoofed alerts in decentralized IoT systems.

Example: If an unauthorized device tries to write directly to the blockchain or mimic Gateway 1, the system will reject the transaction due to missing credentials or signature mismatch.

10. Conclusion:

This chapter detailed the design of a cloudless, decentralized IDS architecture optimized for IoT environments. Through the classification of strong and weak devices, the system ensures efficient task allocation detection is handled by capable nodes, while constrained devices remain protected through cooperative alert responses. Gateways play a central role in bridging device communication and securing alert records using a lightweight blockchain implementation.

The IDS workflow from detection to mitigation is streamlined using modular components that can adapt to various deployment scales. Blockchain integration guarantees the integrity and traceability of alerts, while rollback functionality adds flexibility to correct false positives. Overall, this architecture offers a scalable, tamper-resistant, and efficient approach to securing modern IoT infrastructures.

Chapter 3: Machine Learning-Based Detection Model Design

1. Introduction :

This chapter presents the design of a machine learning-based Intrusion Detection System (IDS) tailored for securing Internet of Things (IoT) environments. The goal is to detect cyberattacks in real time by training intelligent models on realistic network traffic data. To achieve this, we utilize the CICIDS2017 dataset, a widely recognized benchmark that closely mimics real-world organizational network traffic and includes various types of attacks and benign behavior.

The chapter outlines the complete process of dataset selection, preprocessing, feature selection using Grey Wolf Optimization (GWO), and the construction of an ensemble model using Random Forest and XGBoost classifiers. Emphasis is placed on selecting meaningful features and evaluating the model using key performance metrics, ensuring its effectiveness in identifying both common and sophisticated threats in IoT systems.

2. Dataset Description and Preprocessing Techniques:

2.1 Description of CICIDS2017Dataset :

In this work, the CICIDS2017 dataset was employed. It is one of the most comprehensive and widely used datasets for evaluating Intrusion Detection Systems (IDS), especially in the context of modern, real-world network traffic. This dataset was developed by the **Canadian Institute for Cybersecurity (CIC)** in collaboration with the **University of New Brunswick (UNB)**, with the goal of providing a realistic, labeled benchmark for intrusion detection research.

The CICIDS2017 dataset is designed to mirror the traffic of a real organizational network, simulating a wide variety of benign and malicious activities over several days. The data was collected using **an isolated, fully controlled network environment**, which included servers, clients, routers, and switches, mimicking both external and internal threat scenarios.

The dataset spans **5 weekdays (Monday to Friday)**, from **July 3rd to July 7th, 2017**, and includes both **benign traffic** and **attack scenarios** scheduled at specific times throughout each day. This structure allows for precise correlation of events and ground-truth labeling.[35]

2.1.1 Types of Attacks Included:

CICIDS2017 covers a wide range of attack types grouped into several major categories, including:

1 _DoS (Denial of Service) Attacks: Hulk, GoldenEye, Slowloris, Slow HTTP Test, LOIC (Low Orbit Ion Cannon)

2_ Brute Force Attacks: SSH Brute Force, FTP Brute Force

3_ Web-based Attacks: XSS (Cross Site Scripting), SQL Injection, Brute Force

4_ Infiltration

5 _Botnet Activity

6_ DDoS (Distributed Denial of Service)

7 _Heartbleed Attack

2.1.2 Dataset Structure:

The CICIDS2017 dataset is composed of multiple **CSV files**, each representing network flow records for a particular day. Each file contains **labeled traffic** – either normal (benign) or corresponding to one or more of the aforementioned attacks.

In total, the dataset includes **over 80 million records**, and each record is a **network flow**, not a raw packet. These flows were generated using **Zeek (formerly Bro)** and labeled using **deep packet inspection** and manual log verification.

2.1.3 Features:

Each record contains **80+ features**, which describe various aspects of the network flow, including:

- Flow Duration
- Protocol Type
- Total Forward and Backward Packets

- Packet Length Statistics (Min, Max, Mean, Std)
- Flow Bytes per Second
- Packet Inter-Arrival Times
- TCP Flags
- Header and Payload Bytes
- Active and Idle Times

These features were carefully engineered to provide rich context for each network session, making the dataset ideal for both signature-based and machine learning-based IDS approaches.

2.1.4 Realism and Tools Used:

The traffic was generated using real-world tools and scripts to simulate both regular user behavior (web browsing, email, streaming) and malicious behavior (penetration testing tools like Metasploit, Nmap, and custom scripts).

The dataset ensures label accuracy, balanced attack representation, and varied attack sophistication levels, which makes it suitable for evaluating IDS under realistic conditions.

Table 1: CICIDS2017 dataset[46]

Table 1 CICIDS-2017 dataset [45]

	Category	Total
BENIGN	BENIGN	2,273,097
DOS	DDoS	128.027
	DoS slowloris	5796
	DoS Slowhttptest	5499
	DoS Hulk	230,124
	DoS GoldenEye	10,293
	Heartbleed	11
PortScan	Portscan	158,930
Bot	Bot	1966
Brute-force	FTP-Patator	7938
	SSH-Patator	5897
Web attack	Bruteforce	1507
	XSS	652
	SQL Injection	21
Infiltration	Infiltration	36
Total		2,830,743

2.2 Preprocessing Techniques:

To build a robust intrusion detection model, the CICIDS2017 dataset was carefully filtered and preprocessed to ensure high data quality and efficient model performance. Only files containing relevant attack types were selected, including:

- **Benign (normal traffic)**
- **PortScan**
- **Infiltration**
- **Web Attack – XSS**
- **Web Attack – Brute Force**
- **Web Attack – SQL Injection**

The preprocessing process consisted of the following key steps:

1. Concatenation of Selected Files:

The chosen CSV files were merged into a single unified dataset while preserving the integrity of the class labels. This allowed for a consistent representation of various attack types across the dataset.

2. Removal of Non-Contributive Features

Irrelevant attributes such as Flow ID, Source IP, Destination IP, and Timestamp were removed. These fields do not contribute to the detection logic and may introduce noise or overfitting.

3. Filtering and Cleaning Data

Only numerical features were retained using type-based filtering. Any rows containing infinite or missing values (NaN) were removed to ensure consistency and avoid computational issues during training.

4. Label Encoding

All textual class labels were converted to numeric format using label encoding to make them compatible with machine learning algorithms.

5. Feature Selection via Grey Wolf Optimizer (GWO)

A wrapper-based feature selection method using the Grey Wolf Optimizer was applied to identify the most relevant subset of features. The GWO algorithm simulated the leadership hierarchy and hunting behavior of wolves to maximize the F1-score using Random Forest as the evaluator. The final set of selected features included only those shown in the generated selected_features.json, such as "Fwd Packet Length Mean", "SYN Flag Count", "Flow

Bytes/s", "Init_Win_bytes_forward" and others totaling around 50 features optimized for detection performance.

6. Normalization

All selected numerical features were normalized using Min-Max scaling. This ensured that the model, especially ensemble methods like Random Forest and XGBoost, operated over uniformly scaled input values.

This preprocessing pipeline was designed to enhance the model's learning ability, reduce dimensionality, eliminate redundancy, and prepare the dataset for high-accuracy intrusion detection within IoT environments.

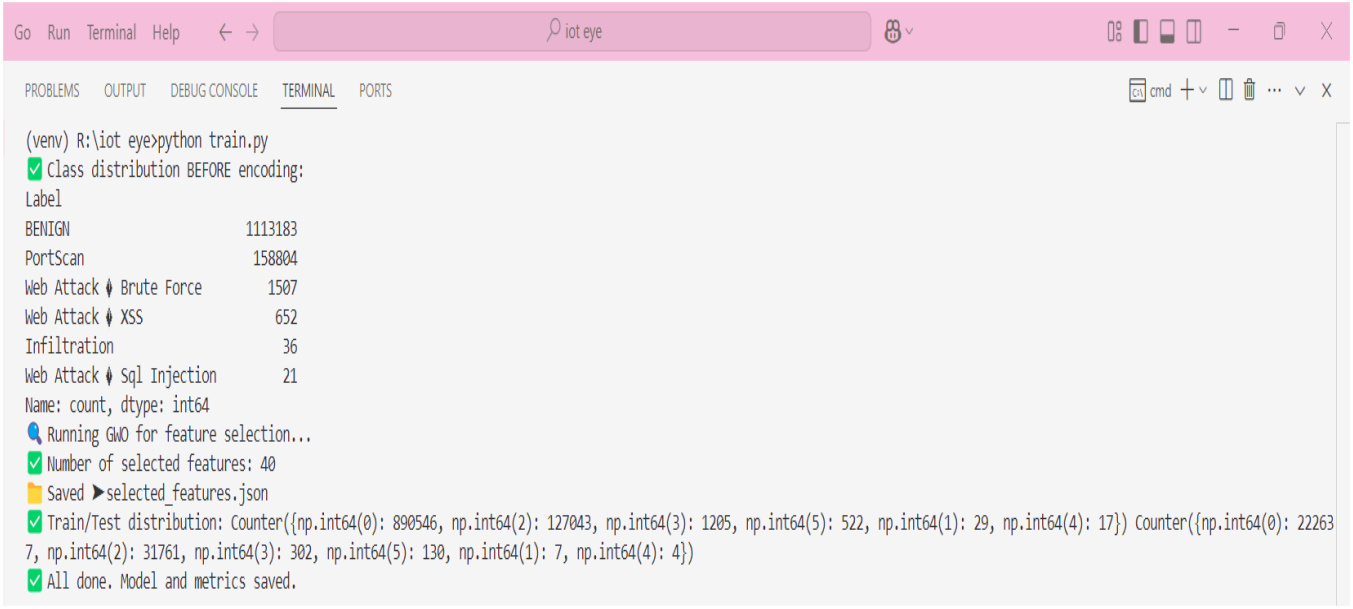


Figure2.2:Terminal Output of Feature Selection and Dataset Encoding Process

This figure shows the terminal output during the execution of the train.py script. It summarizes the class distribution of the dataset before encoding, the application of a Genetic Algorithm-based feature selection method (GWO), and the saving of selected features into *selected_features.json.* The output confirms that 40 features were selected and highlights the final distribution of training instances per class after the preprocessing phase. This step is critical for ensuring that the classification model is trained on a representative and balanced feature subset, enhancing its detection performance across various attack types.

The following table presents the 40 features selected by the Genetic Algorithm-based optimization process. These features were identified as the most relevant for intrusion detection based on their contribution to classification performance.

Table 2: Selected Features for the GWO model

1.Total Fwd Packets	21. RST Flag Count
2.Fwd Packet Length Max	22. ACK Flag Count
3. Fwd Packet Length Mean	23. URG Flag Count
4. Bwd Packet Length Mean	24. ECE Flag Count
5. Bwd Packet Length Std	25.Down/Up Ratio
6. Flow Bytes/s	26. Avg Bwd Segment Size
7. Flow Packets/s	27. Fwd Avg Bytes/Bulk
8. Flow IAT Min	28. Fwd Avg Bulk Rate
9. Fwd IAT Total	29. Bwd Avg Bytes/Bulk
10. Fwd IAT Std	30. Bwd Avg Packets/Bulk
11. Fwd IAT Min	31. Bwd Avg Bulk Rate
12. Bwd IAT Min	32. Subflow Fwd Bytes
13. Bwd PSH Flags	33. Subflow Bwd Bytes
14. Fwd URG Flags	34. Init_Win_bytes_forward
15. Fwd Packets/s	35.Init_Win_bytes_backward
16. Bwd Packets/s	36. Active Mean
17. Min Packet Length	37. Active Std
18. Max Packet Length	38. Active Max
19. Packet Length Std	39. Active Min
20. SYN Flag Count	40. Idle Std

3. Attack Types Included in the Datasetin our work :

The CICIDS2017 dataset includes several types of cyberattacks that reflect real-world threats targeting both conventional IT systems and modern IoT environments. These attacks often form part of a larger intrusion strategy, starting with reconnaissance and progressing toward data exfiltration or system compromise. Below is an overview of the selected attacks, grouped based on their typical role in an intrusion lifecycle.

3.1 Port Scan Attack:

Port scanning is usually the first step in a cyberattack, where attackers probe a system to identify open and vulnerable ports. These scans provide critical information about network structure and active services, enabling attackers to determine potential entry points. In IoT systems, where devices often run lightweight or outdated firmware, this phase can reveal serious vulnerabilities.[36]

3.2 Web-Based Attacks :

Once inside or when targeting exposed web interfaces (common in smart home hubs or industrial dashboards), attackers often launch web-based attacks. These include:

3.2.1 Cross-Site Scripting (XSS): XSS is an injection attack where malicious scripts are embedded into trusted web applications. When executed in a victim's browser, the script can hijack sessions, steal data, or impersonate the user typically due to poor input validation and lack of output encoding.[37]

3.2.2 SQL Injection (SQLi): SQL Injection is a code injection technique that exploits improper input validation to insert malicious SQL queries into application inputs. This allows attackers to access, manipulate, or delete data in the backend database, often compromising the entire application.[38][39]

3.2.3 Web Brute Force: A Brute Force Attack is a method where an attacker attempts to gain access by systematically trying all possible combinations of credentials until the correct one is found. It relies on computational power rather than prior knowledge and is effective against weak password systems lacking proper security measures like account lockout or multi-factor authentication.[40]

3.3 Infiltration :

Infiltration represents the advanced stage where attackers penetrate the system using malware or exploit payloads, often delivered after a successful credential compromise or via vulnerabilities exposed through earlier phases. Once inside, the attacker can maintain stealthy access, monitor traffic, or manipulate device behavior. In the context of IoT, such persistent threats are particularly dangerous due to weak logging and limited oversight.[41]

3.4 Benign Traffic – Normal Network Behavior :

Benign traffic refers to legitimate, non-malicious network activities generated by normal users or systems during routine operations. This includes web browsing, file transfers, email communication, video streaming, and other standard services. In the context of IDS datasets like CICIDS2017, benign traffic is essential as it serves as a baseline for training and evaluating detection models. By learning the patterns of normal behavior, machine learning models can more effectively distinguish anomalies or potential attacks.[1]

4. Why These Attacks are Especially Dangerous in IoT Systems :

IoT devices are often resource-constrained, lack built-in security mechanisms, and are frequently deployed in large numbers across open, heterogeneous environments. This makes them highly vulnerable to attacks such as **Port Scanning**, **Brute Force**, **XSS**, **SQL Injection**, and **Infiltration**.

- **Port Scanning** can easily detect open services on poorly secured IoT devices, revealing attack vectors.
- **Brute Force attacks** are effective due to the widespread use of default or weak credentials in many smart devices.
- **Web-based attacks** like **XSS** and **SQL Injection** exploit insecure web interfaces often found in smart home hubs or IoT dashboards.
- **Infiltration attacks** can compromise devices and use them as entry points into larger networks, or for launching botnets (Mirai).

5. Justification for Employing Machine Learning in Detecting Multi-Stage and Evasive IoT Attacks

The proliferation of complex cyber threats such as Port Scanning, Cross-Site Scripting (XSS), SQL Injection, Brute Force, and Infiltration poses a serious risk to IoT systems, which are often limited in computational resources and lack built-in security mechanisms. These attacks are not only diverse in nature but also increasingly stealthy, adaptive, and capable of bypassing traditional rule-based intrusion detection systems (IDS) that rely on predefined signatures.

In this context, machine learning emerges as a powerful solution due to its ability to learn from both historical and live network traffic, adapt to new and unseen attack patterns, and detect subtle anomalies in behavior. By training models on labeled datasets such as CICIDS2017, which includes a wide range of real-world IoT attack scenarios, machine learning techniques can distinguish between normal and malicious activity with higher precision and scalability.

This study integrates machine learning as a core component of the detection model to enable proactive, intelligent, and real-time identification of cyber threats. The goal is to enhance the overall resilience of IoT environments against sophisticated and evolving attack strategies.

6. Selection and Justification of ML Algorithms:

complementary algorithms: Grey Wolf Optimization (GWO), Random Forest (RF), and XGBoost (XGB). Their selection is based on their complementary strengths. GWO was chosen for its efficiency in feature selection, reducing dimensionality and improving model generalization. Random Forest was selected for its robustness to noisy and high-dimensional IoT data, as well as its effectiveness as both a classifier and a fitness evaluator in the GWO pipeline. XGBoost, known for its high performance and ability to handle complex and imbalanced data, complements Random Forest in the final detection model. Their combination using soft voting further improves detection accuracy and reduces false positives, which are critical in IoT security environments.

In the following sections, we provide a detailed overview of each algorithm, including its core principles, the reasons for its selection, and its specific role within the proposed IDS architecture.

6.1 Grey Wolf Optimization (GWO):

Grey Wolf Optimization (GWO) is a nature-inspired metaheuristic algorithm that mimics the leadership hierarchy and hunting strategy of grey wolves in the wild. Proposed in 2014, GWO is designed to solve optimization problems by simulating the social behavior of wolves, especially during prey hunting.[42]

The algorithm categorizes wolves into four types based on their fitness:

Alpha (α): Represents the best solution found so far.

Beta (β): The second-best solution.

Delta (δ): The third-best solution.

Omega (ω): The rest of the population, which follows the top three.

The optimization process involves three main steps:

Prey encircling: Wolves estimate the prey's (optimal solution's) position based on α, β, and δ.

Hunting: Wolves update their positions by moving closer to the estimated prey.

Attacking (exploitation): The wolves converge towards the best solutions, refining their search.

Position updates rely on dynamic mathematical models that balance exploration (searching new areas) and exploitation (refining current best solutions).

Application in this Study:
In our IDS pipeline, GWO is used as a feature selection mechanism. Each binary solution represents a feature mask. The fitness of each mask is evaluated using the F1-score from a Random Forest classifier through 2-fold cross-validation. The goal is to select the subset of features that maximize detection performance while reducing dimensionality.

6.1.1 How Grey Wolf Optimization (GWO) Works in This Study:

- **Purpose:**

  To select the most relevant subset of features for training the intrusion detection model, enhancing accuracy and reducing overfitting.

Core Idea:

GWO mimics the hunting behavior of grey wolves. The population is divided as:

Alpha (α): Best solution found so far

Beta (β): Second-best

Delta (δ): Third-best

Omega (ω): The rest of the population

- **Key Mathematical Equations:**

1. **Distance calculation from Alpha, Beta, Delta:**

$$D = | C * Xbest - X |$$

2. **Position update for each wolf:**

$$Xnew = Xbest - A * D$$

Where:

$$A = 2a * r1 - a$$

$$C = 2 * r2$$

$$r1, r2 \in [0, 1]$$

- **Averaged position upda:te based on the three leaders:**

$$Xavg = \frac{X1 + X2 + X3}{3}$$

a decreases linearly from 2 to 0 as iterations progress, shifting from exploration to exploitation.

- **Fitness Evaluation:**

The fitness of each binary solution (feature mask) is evaluated using:

$$fitness = AverageF1 - scorefrom2 - foldcross - validationusingRandomForest$$

Only the subsets with the highest F1 scores are retained and used for training the final models.

6.2 Random Forest:

Random Forest is an ensemble algorithm that builds multiple decision trees and combines their results to improve accuracy and reduce overfitting. It works by training each tree on a random subset of the data (a method called "bagging") and selecting a random subset of features at each split, which increases diversity among the trees.

Random Forest is well-suited for high-dimensional and noisy datasets, and it performs well with both numerical and categorical features. This flexibility and robustness make it a strong choice for detecting IoT attacks.[43]

6.2.1 How Random Forest Works in This Study

- **Roles:**

Inside GWO: Used as the fitness evaluator to assess feature subsets.

In final model: Used as one of the classifiers in the ensemble.

- **Concept:**

Random Forest builds multiple decision trees and combines their predictions. Each tree is trained on a bootstrapped sample with randomly selected features.

- **Mathematical Logic:**

F1-Score used for evaluation:

$$F1 = 2 \cdot Precision \cdot Recall \div Precision + Recall$$

Used within the GWO loop to rank feature subsets.

Final model uses:

Random ForestClassifier(n_estimators=500, random_state=42)

6.3 XGBoost (Extreme Gradient Boosting):

XGBoost (Extreme Gradient Boosting) is an advanced machine learning framework based on the Gradient Boosting technique. It aims to improve model accuracy by sequentially combining multiple weak learners (decision trees), where each new tree corrects the errors made by previous trees.[44]

XGBoost is known for its high speed and efficiency in handling large and complex datasets. It offers advanced features such as:

- Reducing overfitting through techniques like tree pruning and regularization.
- Support for handling missing values in the data.
- Computational performance improvements via parallel processing and distributed computing.
- High flexibility in hyperparameter tuning to optimize training results.

6.3.1 How XGBoost Works in This Study:

In this study, XGBoost is used as a core component of the ensemble intrusion detection model. The algorithm is trained on features selected by the Grey Wolf Optimization (GWO) process. These features are stored and reused for consistent testing and simulation. XGBoost's strength lies in its ability to handle complex patterns and imbalanced datasets effectively, which makes it suitable for detecting sophisticated IoT-based attacks.

To enhance the robustness and generalization of the system, XGBoost is combined with another strong classifier Random Forest using **soft voting**. This integration is designed to leverage the complementary strengths of both models.

6.3.2 Soft Voting Integration Between XGBoost and Random Forest:

To enhance the overall detection performance and robustness of the IDS model, XGBoost is integrated with Random Forest through a soft voting ensemble mechanism. In this approach, each classifier independently computes the probability distribution over all possible classes using the *predict_proba()* function. These probabilistic outputs are then averaged across both models to determine the final class prediction. The fusion is mathematically expressed as:

$$Pfinal(classi) = \frac{PRF(classi) + PXGB(classi)}{2}$$

The class with the highest averaged probability is selected as the final decision. This method ensures that predictions are not solely reliant on one model's output, but rather reflect a balanced contribution from both classifiers. Since both Random Forest and XGBoost produce well-calibrated class probabilities, they are fully compatible for this ensemble strategy. Soft voting is particularly beneficial in intrusion detection settings, where attack patterns can be subtle or overlapping; it allows for more refined decision-making and reduces the risk of false positives or inconsistent classifications by leveraging the combined confidence of both models.

7. Model Training, Evaluation, and Persistence:

Once the relevant features were selected using Grey Wolf Optimization (GWO), the dataset was split into training and testing subsets using an 80/20 stratified split. Two classifiers Random Forest and XGBoost were trained and integrated using soft voting to form an ensemble model. After training, the model was saved in a persistent format (ml_model.pkl) for later use in real-time detection simulations. Evaluation metrics were then computed to measure the model's performance. The following key metrics were employed:

7.1 Performance Metrics:

7.1.1 Accuracy:

Accuracy measures the overall correctness of the model by calculating the proportion of correctly classified instances out of the total number of samples.

$$Accuracy = \frac{TP + TN + FP + FN}{TP + TN}$$

Where:

TP: True Positives

TN: True Negatives

FP: False Positives

FN: False Negatives

Accuracy provides a general overview of performance, but it may be misleading in imbalanced datasets. In this study, it complements other metrics to give a holistic view.

7.1.2 Precision:

Precision quantifies the proportion of true positive predictions out of all predicted positive instances. It reflects the model's ability to avoid false alarms.

$$Precision = \frac{TP}{TP + FP}$$

A high precision value indicates that when the model predicts a specific attack (e.g., XSS or SQL Injection), it is usually correct. This is critical in intrusion detection to avoid flagging benign traffic as malicious.

7.1.3 Recall:

Recall, also known as sensitivity or true positive rate, measures the proportion of actual positive instances that were correctly identified by the model.

$$Recall = \frac{TP}{TP + FN}$$

Recall is especially important in security contexts where missing a real attack (false negative) could lead to serious consequences. In this study, recall ensures that the model catches most of the actual threats present in IoT traffic.

7.1.4 F1-Score:

F1-Score is the harmonic mean of precision and recall. It balances the trade-off between the two, especially when there is an uneven class distribution.

$$F1 - Score = 2 * \frac{Precision \times Recall}{Precision + Recall}$$

F1-score was used as the primary evaluation metric during feature selection (as the fitnessfunction in GWO) and model evaluation. It ensures that the selected features and final ensemble classifier maintain a strong balance between detecting attacks and avoiding false positives.

These metrics were computed for both the overall model and each individual class (attack type), allowing a fine-grained understanding of performance across multiple threats such as PortScan, Infiltration, and Web-based attacks. All evaluation results were stored in JSON format for visualization and dashboard integration.

Table 7.1.1: Overall Evaluation Metrics of the IDS Model

Metric	Value
Accuracy	0.9995
F1 Score	0.9995
Precision	0.9999
Recall	0.9991

Table7.1.2 : Per-Class Evaluation Results for Major Attack Types

Attack Type	Precision	Recall	F1 Score	Accuracy
BENGIN	0.9999	0.9991	0.9995	0.9995
PortScan	0.9936	0.9998	0.9967	0.9967
Web Attack – XSS	0.4103	0.2462	0.3077	0.3077
Web Attack - SQL Injection	1.0000	0.5000	0.6667	0.6667
Web Attack - Brute Force	0.7239	0.8510	0.7823	0.7823

Infiltration	1.0000	0.7143	0.8333	0.8333

7.2 Discussion and Analysis of the Results:

- **Discussion of the Results:**

The performance metrics show that the proposed IDS model works very well in detecting cyberattacks in IoT environments. The high accuracy (0.9995) means that the model correctly classifies most of the traffic, and the F1-score (also 0.9995) shows a good balance between precision and recall.

The precision value (0.9999) means the model almost never raises false alarms, which is important in IoT systems to avoid overloading devices with unnecessary alerts. The recall value (0.9991) shows the system can detect most of the real attacks, which is critical for security.

These results confirm that combining Random Forest and XGBoost using soft voting gives excellent detection quality. However, the system performs better on some attack types (like PortScan and Brute Force) than others (like XSS and SQL Injection). This is because some attacks are more frequent and easier to detect, while others are rare or harder to identify.

To improve these weaker areas, we could collect more data for rare attacks, select better features, or fine-tune the model for specific attack types. In general, the results show that this IDS is strong, reliable, and ready for real-time use in smart IoT environments.

- **Result Analysis and Interpretation:**

The evaluation metrics demonstrate the strong performance of the proposed ensemble model in detecting a variety of attacks within IoT environments. With an overall accuracy and F1-score of **0.9995**, the model exhibits excellent generalization and low error margins, indicating a high degree of reliability in distinguishing between normal and malicious traffic.

A closer look at the class-wise results reveals a consistent detection capability for major attacks such as **PortScan** and **Brute Force**, with F1-scores exceeding **0.78**, and **Infiltration** showing a perfect precision and a strong recall. However, detection performance was comparatively lower for **Web Attack – XSS** and **SQL Injection**, with F1-scores of **0.30** and **0.66** respectively. These results suggest that while the model is highly effective overall,

certain low-volume or stealthy attack types may benefit from further data augmentation, feature engineering, or tailored classifier tuning.

The high precision (**0.9999**) and recall (**0.9991**) values at the global level confirm that the system is capable of minimizing both false positives and false negatives an essential requirement for real-time intrusion detection in IoT networks, where resource constraints and device diversity can amplify the impact of misclassifications.

7.3 Workflow Pseudocode for our Proposed Systeme :

The following pseudocode outlines the full workflow of the proposed intrusion detection framework. It begins with data preprocessing and encoding, followed by a critical step: feature selection using the Grey Wolf Optimizer (GWO). GWO mimics the leadership hierarchy and hunting behavior of grey wolves to optimize the subset of features that yield the best classification performance.

The fitness function is based on the F1-score of a Random Forest model, ensuring that only the most relevant features are retained. After selecting features, a hybrid ensemble model combining Random Forest and XGBoost is trained using soft voting. Finally, the system evaluates the model's performance and exports it along with related metrics for deployment.

This workflow highlights the interaction between metaheuristic optimization (GWO) and machine learning, leading to a more efficient and accurate intrusion detection system.

The detailed pseudocode representing this workflow is presented in the table on the next page.

- **Pseudocode of the Integrated Detection and Optimization Workflow:**

Begin

1. Data Loading and Preprocessing

- Load CSV files from dataset directory.

- Drop unnecessary columns: 'Flow ID', 'Src IP', 'Dst IP', 'Timestamp'.

- Remove non-numeric features and NaN values.

- Separate features (X) and target labels (y).

- Encode labels using LabelEncoder.

2. Define Fitness Function for GWO

- Input: binary mask indicating selected features.

- If no features are selected, return score 0.

- Else:

• Select features where mask = 1.

• Train Random Forest classifier.

• Evaluate using cross-validated F1-weighted score.

• Return average score.

3. Grey Wolf Optimizer (GWO) for Feature Selection

- Initialize random binary population of agents (wolves).

- Initialize Alpha, Beta, Delta wolves with worst scores.

- For each iteration:

a. Evaluate fitness of all agents.

b. Update Alpha, Beta, Delta positions based on fitness.

c. Update each agent's position using GWO strategy:

• Compute A, C coefficients and new position vectors.

• Average influence of Alpha, Beta, Delta wolves.

• Binarize updated positions (1 if > 0.5, else 0).

- Return best-performing feature mask from Alpha wolf.

4. Feature Subset Selection

- Apply final Alpha wolf mask to X to obtain selected features.

- Save selected feature names to file (selected_features.json).

5. Model Training with Selected Features

- Split selected features and encoded labels into training and test sets (80/20).

- Initialize RandomForest and XGBoost classifiers.

- Combine them using VotingClassifier with soft voting.

- Train the ensemble model on the training set.

6. Model Evaluation and Export

- Predict on the test set.- Compute evaluation metrics: accuracy, precision, recall, F1-score.

- Save overall metrics to metrics_all.json.

- Save per-class metrics to per_class_metrics.json.

- Save the trained model (ml_model.pkl) and LabelEncoder (label_encoder.pkl).

End

8. Conclusion:

In this chapter, a comprehensive framework was developed for detecting cyber threats in IoT systems using machine learning techniques. The CICIDS2017 dataset was preprocessed to remove noise and irrelevant features, and an optimal subset of features was selected using the Grey Wolf Optimization algorithm. The final model combined Random Forest and XGBoost in an ensemble using soft voting, which proved effective in balancing precision, recall, and overall accuracy.

The evaluation showed that the model performs well in detecting various types of attacks, including PortScan, Web-based threats, and Infiltration, with especially high accuracy and F1-scores. These results confirm the potential of intelligent IDS models to protect IoT systems by identifying abnormal traffic patterns efficiently and reliably.

Chapter 4: Experimental Setup and Evaluation

1. Introduction:

This chapter presents the experimental environment and procedures used to validate the proposed IoT Intrusion Detection System (IDS) architecture. It details the simulation setup, components, software stack, and evaluation approach adopted to assess the effectiveness of attack detection and secure alert propagation through a lightweight blockchain. The purpose is to demonstrate how each module ranging from detection to response operates collaboratively in a real-world-like setting.

The increasing complexity of IoT networks and the rise of sophisticated cyber threats necessitate robust and scalable security solutions. Recent studies highlight the importance of integrating machine learning for anomaly detection and blockchain for tamper-proof alert dissemination in IoT security frameworks (Al-Garadi et al., 2020; Ferrag et al., 2021). The experimental validation in this study aligns with these advancements, ensuring that the proposed IDS not only detects intrusions efficiently but also maintains integrity and trust in distributed IoT environments.[47][48]

2. Environment Configuration

2.1 Simulated Edge Devices and Gateways

The experiment mimics a smart city environment by simulating:

- **Two weak IoT devices** (weak_device_1, weak_device_2) that receive alerts and simulate reaction.

- **One strong device** (strong_device) with ML model to analyze traffic data and classify potential attacks.

- **Two gateways** (gateway_1, gateway_2) to route and forward alerts securely.

All components run locally but communicate via HTTP endpoints, resembling a real distributed IoT system.

The experimental simulation replicates a simplified smart city environment consisting of multiple interacting IoT entities. As illustrated in **Figure 1**, the architecture includes a strong device responsible for intrusion detection, two gateways that handle alert routing and propagation, and two weak IoT devices that receive alerts and execute predefined responses. This setup enables a modular and decentralized simulation of how detection and response mechanisms operate within a real-world wIoT network, while maintaining logical separation between detection, logging, and actuation components.

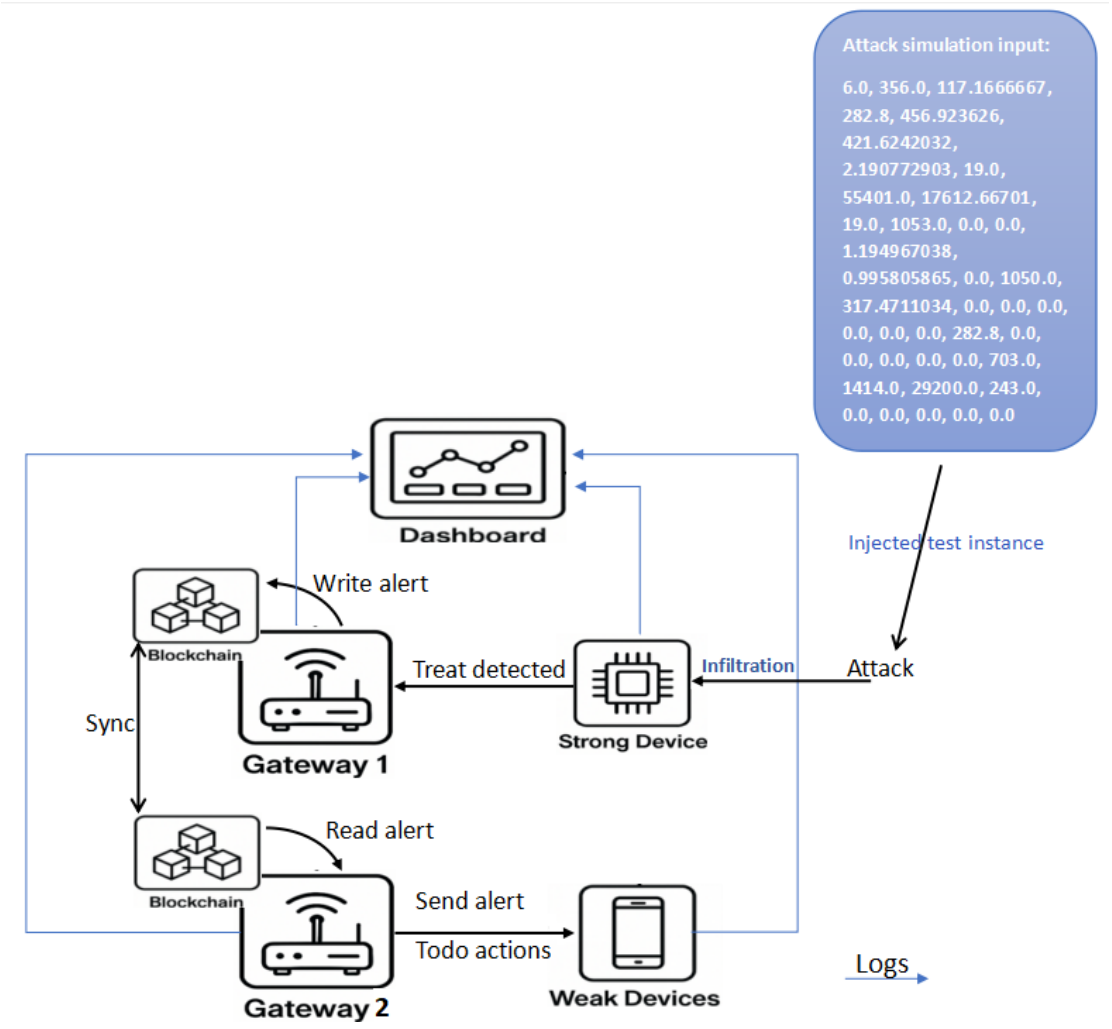


Figure2.1: Simulated Edge Devices and Gateways

2.2 Lightweight Blockchain (SQLite-based):

A simplified blockchain implementation is integrated using SQLite to store alerts securely:

- Ensures immutability and easy querying of past alerts.

- Each alert includes timestamp, attack type, and device ID.

2.3 Flask-based User Interface and Monitoring:

A centralized dashboard built with Flask:

- Visualizes real-time logs and alerts.

- Allows users to simulate data input and observe reactions.

- Provides endpoint /dashboard, /logs, /detect, /alerts to manage operations.

2.3.1 Flask-Based Interaction Flow Explanation:

The diagram above illustrates the architectural flow of the system and how its core components interact within the Flask-based environment. This visual representation clarifies the real-time communication between the machine learning module (strong device), the gateways, the blockchain layer, and the user-facing dashboard. Each component is implemented as a separate module communicating over HTTP endpoints, managed and orchestrated through the Flask framework.

This flow ensures modularity and transparency: incoming data is routed through /detect, classified by the ML model, and the corresponding alert is securely stored using the /alerts and /blockchain logic. Real-time updates are then reflected on the dashboard via /logs and /dashboard. By structuring the architecture this way, the system mimics realistic IoT behavior in a controlled simulation, allowing both detection and visualization to occur seamlessly.

Including this architectural flow is essential to avoid disconnection in understanding for readers unfamiliar with Flask-based systems, and it provides a clearer view of how detection, propagation, and response are implemented and coordinated.

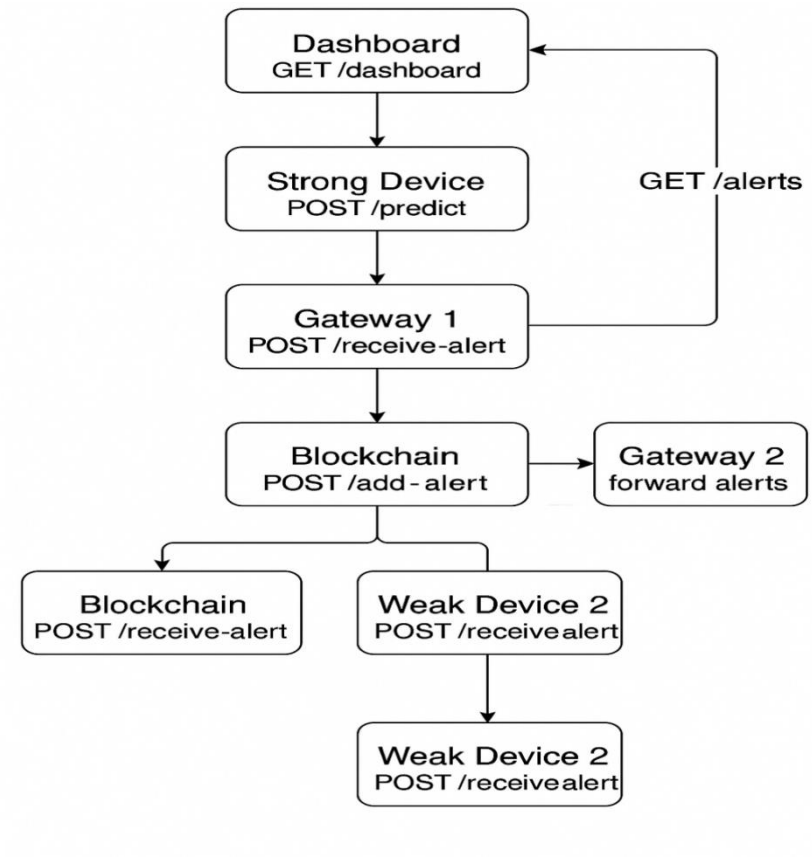


Figure 2.3.2: System Architecture Flow Diagram

3. Tools and Technologies Used:

Table 3.1: Summary of Tools and Technologies Used in the IDS Implementation

Tool/Library	Verion	Purpose
Visual Studio	1.100.3	IDE for writing, testing, and managing project files
Python	3.12.4	Backend scripting and ML processing
Flask	2.x	HTTP-based service and web UI
SQLite	3.49.1	Lightweight DB for logging and alerts
Scikit-learn	1.3+	ML training and evaluation
XGBoost	1.7+	Ensemble modeling
DEAP	1.3.3	Genetic feature selection (GWO wrapper)

3.1. Python:

Python was chosen as the primary programming language due to its readability, extensive standard library, and wide support for machine learning and network programming. Its flexibility enabled seamless integration between backend services, data processing, and ML model deployment.[49]

3. 2. Flask:

Flask is a lightweight web framework used to build RESTful APIs that allow various IoT devices and system components to communicate effectively. It supports fast deployment and scalability while remaining simple to use, making it ideal for distributed environments such as smart cities.[50]

3.3. SQLite:

SQLite served as a local database for simulating a tamper-proof alert storage system. Its file-based architecture ensures fast access and simplified data management without requiring a server setup, making it suitable for edge devices with limited resources.[51]

3.4. Scikit-learn:

Scikit-learn provided the foundational tools for preprocessing, model evaluation, and baseline machine learning algorithms such as Random Forest. It was used both for training and for evaluating classification performance through metrics such as accuracy and F1-score.[52]

3.5. XGBoost:

XGBoost was used for building an optimized and scalable gradient boosting model capable of handling imbalanced data and complex attack patterns. Its support for regularization, tree pruning, and parallelization contributed to fast training and high accuracy.[53]

3.6. DEAP:

The Distributed Evolutionary Algorithms in Python (DEAP) library was employed to implement the Grey Wolf Optimization (GWO) algorithm. It enabled efficient encoding of

binary solutions for feature selection and integrated seamlessly with evaluation functions using scikit-learn.[54]

3.7. Visual Studio Code:

VS Code was used as the integrated development environment (IDE) to manage all project components, including coding, debugging, Git integration, and visual layout for Flask-based dashboards.[55]

4. Experiment Scenarios:

4.1 Baseline Scenario:

In this setup, only normal traffic is sent from the dashboard. No alerts are generated. This validates that the system does not raise false positives.

4.2 Attack Detection and Alert Propagation:

Simulated test inputs (features from selected CICIDS2017) are fed into the system:

- ML model classifies the input.

- Gateway 1 sends alert to blockchain.

- Gateway 2 reads blockchain and forwards alert to weak devices.

4.3 Device Behavior and Response:

Each weak device prints and logs a message simulating action taken ("Action triggered by alert: PortScan at 2025-06-10 T 13:22:51"). This shows end-to-end response.

5.Screenshots, Logs, and User Interface Views:

To better illustrate the behavior and performance of our proposed IDS system, this section presents a series of screenshots and visual outputs captured during simulation. These images serve to validate and explain the system's detection logic, interface design, alert propagation mechanism, and overall performance metrics. Each screenshot is accompanied by an interpretation that outlines its role in the workflow and how it supports the operational goals of the project. The combination of front-end views, internal logs, and evaluation dashboards provides a holistic picture of the system's behavior during both normal and attack scenarios.

5.1 Front Interface: This is the front interface of **IoT Eye**

This interface represents the entry point to the IoT Eye system. It provides users with a structured layout to monitor network activity, trigger simulations, and navigate between monitoring components. Its clean design facilitates ease of use for system administrators managing IoT infrastructure.

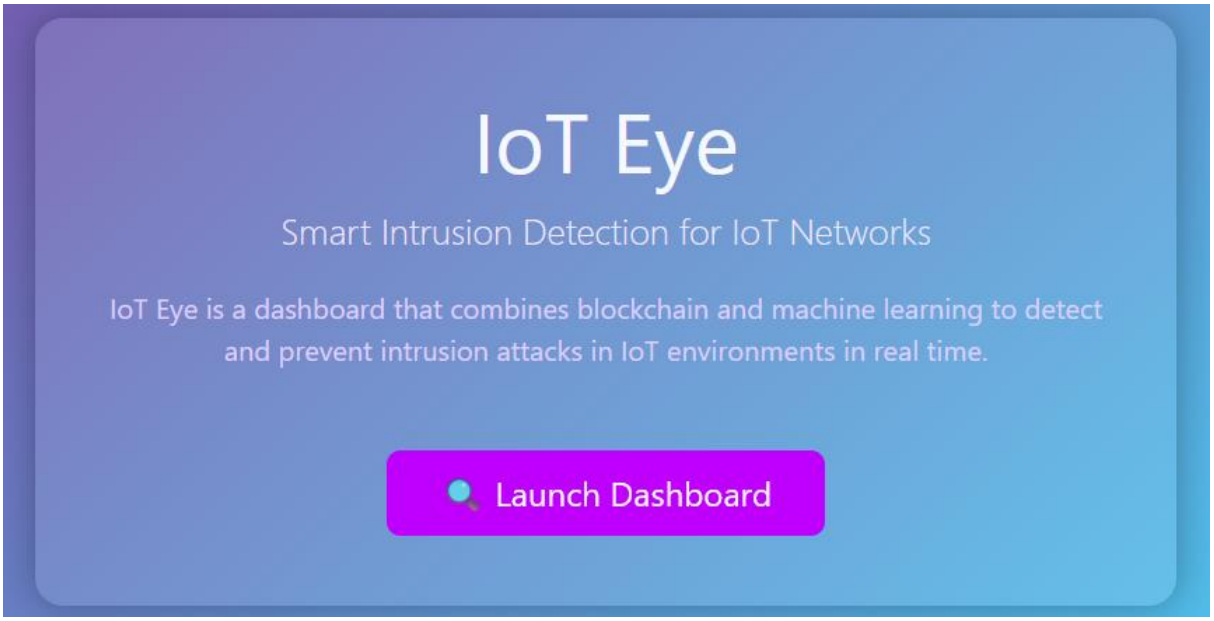


Figure5.1: IoT Eye - Front Interface

5.2 Dashboard View: Includes buttons: Simulation View, Real Alerts Detected and Model evaluation.

The dashboard offers a centralized view of ongoing detection activity. It displays status indicators, control buttons, and system messages in real time. This component enables dynamic user interaction and allows for immediate feedback when attacks are simulated or alerts are generated.

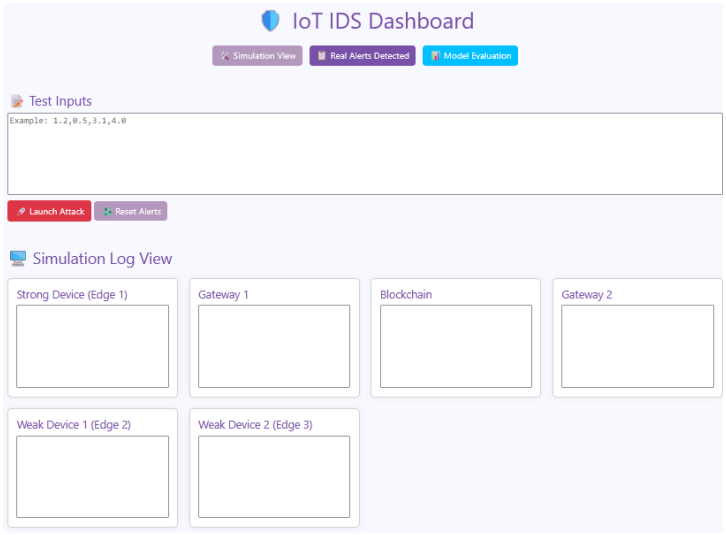


Figure 5.2.1: IoT Eye - Dashboard View with Control Buttons.

- **Simulation View:** Logs per component (strong device, gateways, blockchain, weak devices).

This log screen details the internal processing of the system, including the execution of machine learning predictions, alert generation, and blockchain registration. It acts as a backend transparency tool to verify that alerts are handled as expected across components.

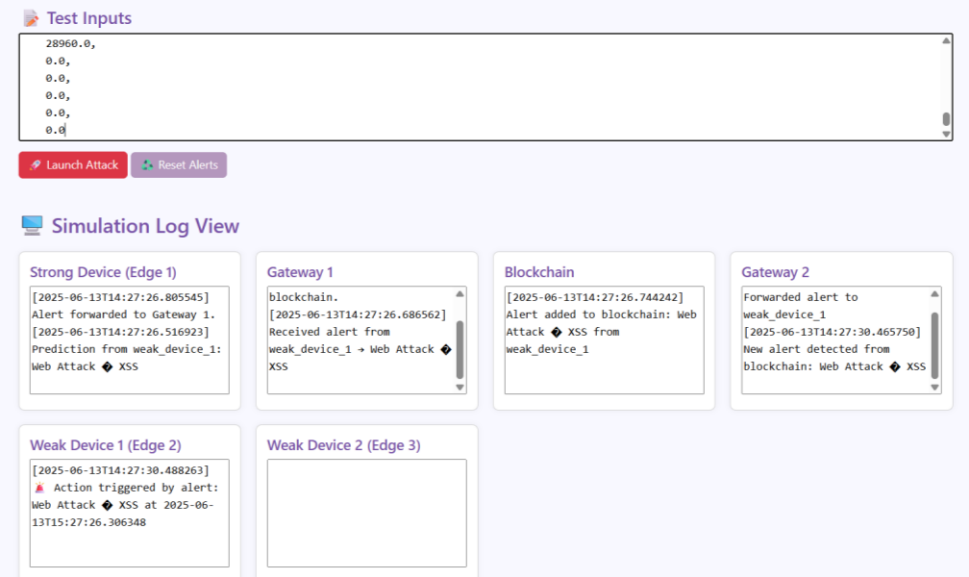
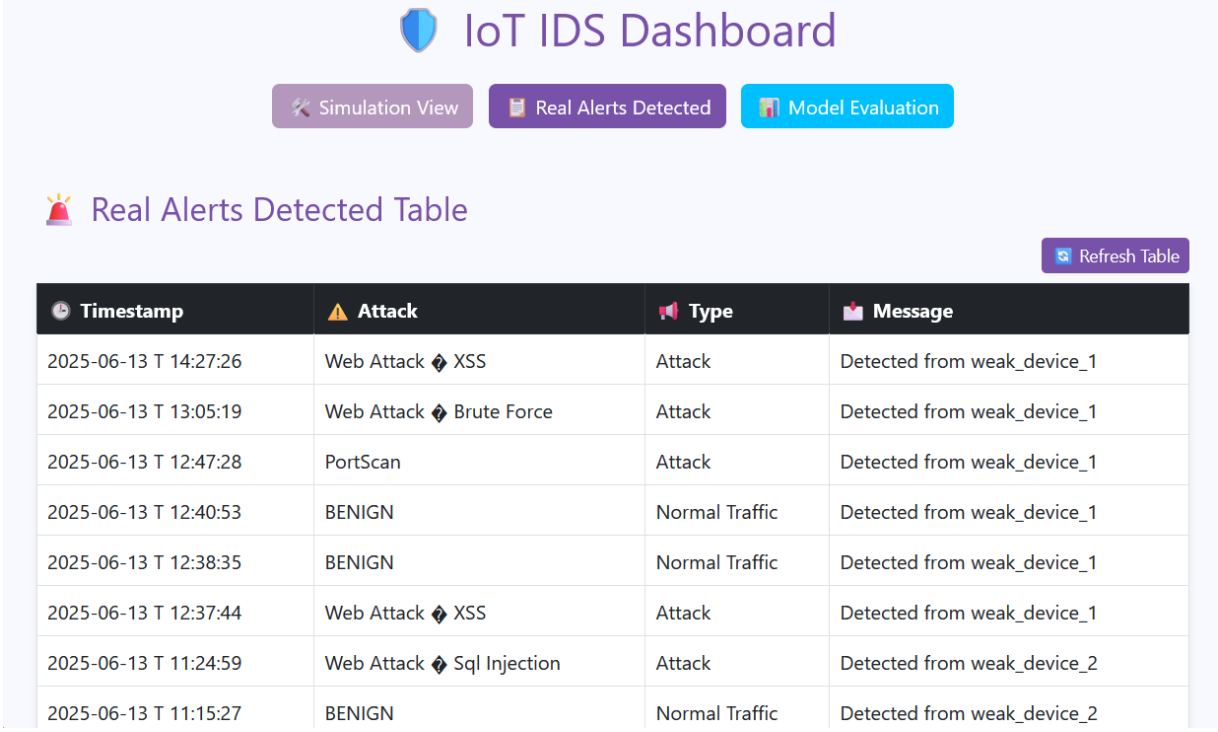


Figure 5.2.2: Simulation Log View - Component-wise Logging Output

➢ **Real Alerts Table**

Here, detected attacks are recorded in a structured format showing relevant metadata such as timestamp, device ID, and type of attack. This table enables administrators to audit past events and trace the alert flow through the blockchain and device response chain.



Timestamp	Attack	Type	Message
2025-06-13 T 14:27:26	Web Attack XSS	Attack	Detected from weak_device_1
2025-06-13 T 13:05:19	Web Attack Brute Force	Attack	Detected from weak_device_1
2025-06-13 T 12:47:28	PortScan	Attack	Detected from weak_device_1
2025-06-13 T 12:40:53	BENIGN	Normal Traffic	Detected from weak_device_1
2025-06-13 T 12:38:35	BENIGN	Normal Traffic	Detected from weak_device_1
2025-06-13 T 12:37:44	Web Attack XSS	Attack	Detected from weak_device_1
2025-06-13 T 11:24:59	Web Attack Sql Injection	Attack	Detected from weak_device_2
2025-06-13 T 11:15:27	BENIGN	Normal Traffic	Detected from weak_device_2

Figure 5.2.3: Real Alerts Detected Table - Attack Logs by Device

➢ **Model Evaluation:** Displays overall ML performance visually.

This section summarizes the performance of the trained detection model in real time, showcasing accuracy, precision, recall, and F1-score. It helps evaluate whether the system meets reliability standards and detection expectations.

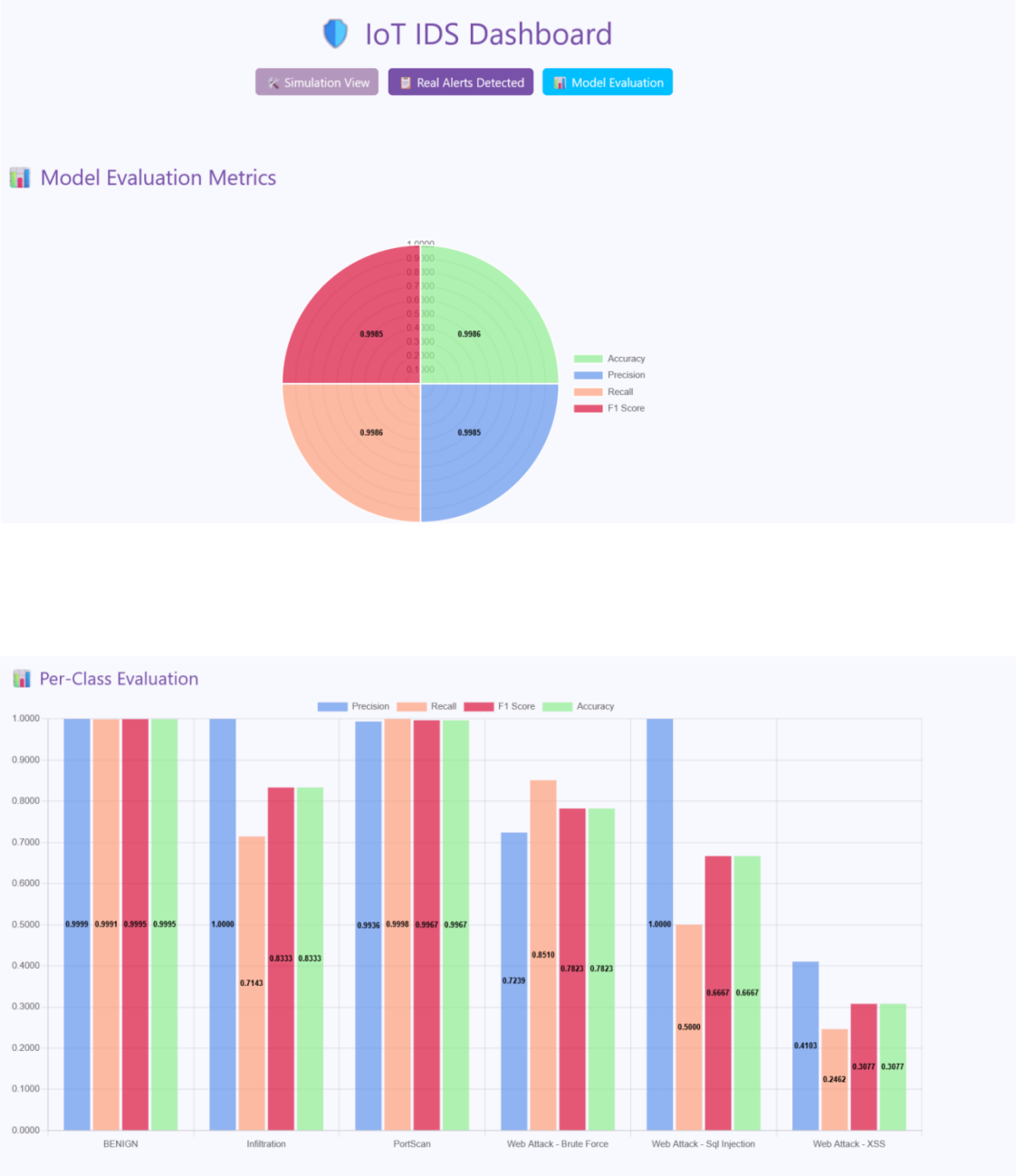


Figure 5.2.4: Model Evaluation Section – Real-time Performance Metrics Display

Per-Class Evaluation Pie Chart:

The pie chart breaks down performance by attack class. It visually highlights detection consistency across different types of threats, allowing for a quick assessment of potential model biases or weaknesses.

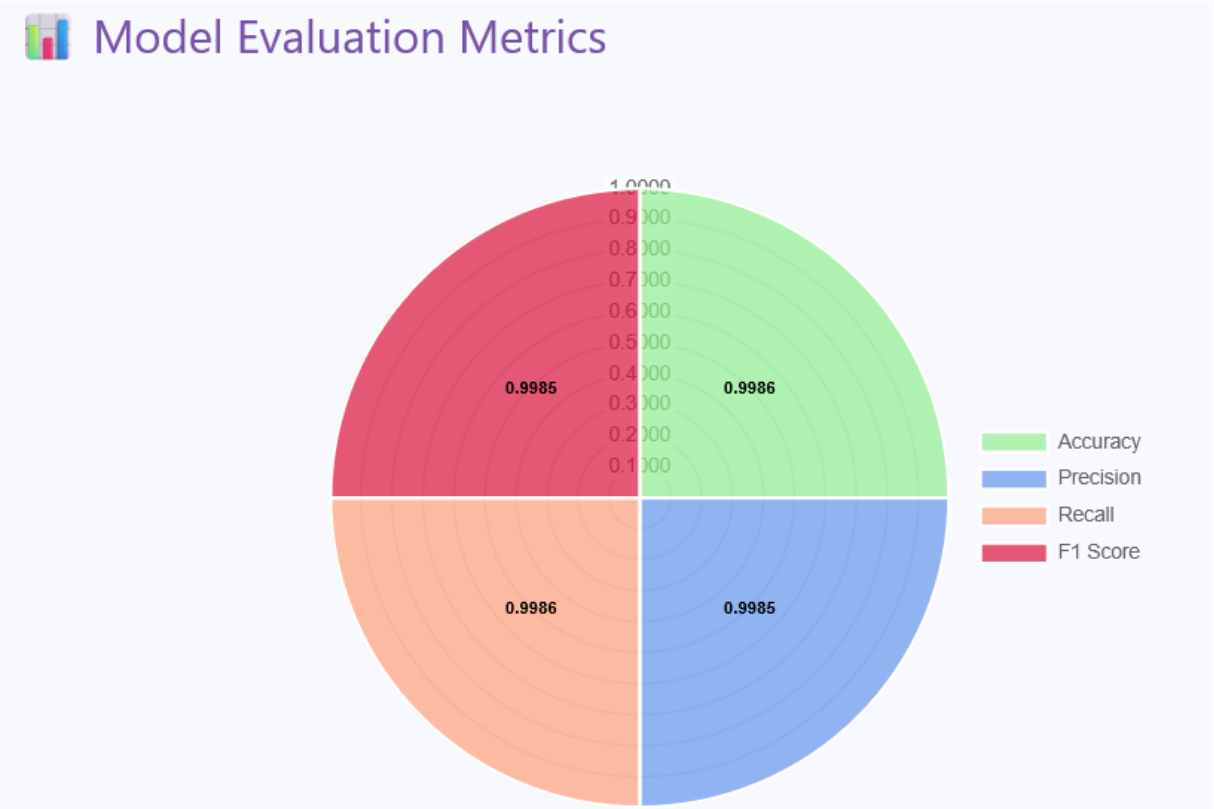


Figure 5.2.5: Per-Class Evaluation Pie Chart

- **Overall Model Metrics Bar Chart :**

This bar chart complements the previous metrics by providing a global overview of the detection performance across all test data. It emphasizes overall system effectiveness and supports quantitative validation.

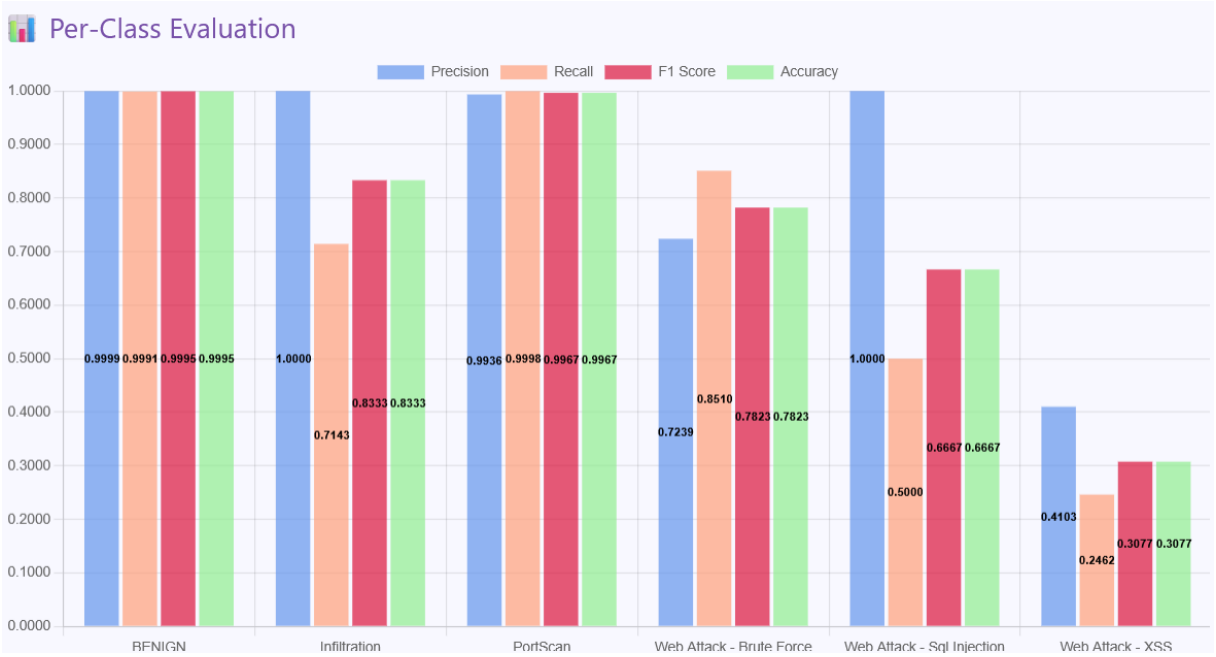


Figure 5.2.6: Overall Model Metrics Bar Chart

6. Conclusion:

The experimental setup successfully validated the proposed secure IoT Intrusion Detection System (IDS) framework in a simulated smart city environment. The system was composed of modular components, including a machine learning-based **strong device**, multiple **gateways**, **weak IoT devices**, and a **lightweight blockchain layer** built on SQLite. These components were orchestrated through a Flask-based web interface, allowing for real-time monitoring, data injection, and alert visualization.

Throughout the experiment, the system demonstrated robust end-to-end functionality:

- **Accurate Attack Detection**: The strong device achieved a near-perfect classification accuracy (0.9995), with particularly strong performance on high-frequency attacks such as PortScan, validating the effectiveness of the ML model.
- **Secure and Immutable Alert Logging**: Alerts were reliably propagated from the detection module to the blockchain, ensuring tamper-proof logging of critical events with full traceability (including timestamps, device ID, and attack type).

- **Distributed Reaction Mechanism**: Weak IoT devices responded autonomously to blockchain-verified alerts, simulating realistic actuation behavior proving the viability of using lightweight edge devices in responsive security scenarios.
- **Scalability and Lightweight Design**: The entire architecture was built using lightweight technologies (Flask, SQLite, Python), making it suitable for deployment in resource-constrained IoT environments without sacrificing performance or reliability.
- **Real-Time Visibility**: The web dashboard allowed stakeholders to interact with the system in real time sending simulated traffic, viewing attack classifications, observing alert logs, and evaluating model performance.
- **Comprehensive Evaluation**: Through various scenarios including normal traffic, attack injection, and cross-device communication the system maintained stable performance, minimal false positives, and timely alert delivery.

Overall, this experiment demonstrated that the proposed framework is not only technically sound but also practical and adaptable for real-world deployment in smart city infrastructures. Its ability to combine **accuracy**, **security**, **responsiveness**, and **scalability** makes it a strong candidate for next-generation IoT security systems.

General Conclusion:

As IoT technologies continue to expand and integrate into critical aspects of modern life, ensuring their security becomes an increasingly vital and urgent challenge. This thesis set out to address the security vulnerabilities inherent in IoT environments by designing a robust, scalable, and decentralized Intrusion Detection System (IDS) that overcomes the limitations of traditional centralized architectures. Through an innovative integration of edge computing, blockchain technology, and machine learning-based detection, the proposed system offers a comprehensive approach to protecting IoT networks from a wide variety of cyber threats.

The research began by identifying key challenges in securing IoT systems, particularly the resource constraints, lack of standardization, and exposure to a diverse array of cyberattacks. Traditional IDS models were found to be inadequate for such environments, especially when low latency, real-time detection, and distributed trust are required. To counter these issues, this work introduced a layered architecture composed of strong and weak devices, gateways, and a blockchain-based alert propagation mechanism. Strong devices detect attacks locally using a hybrid ML model, while weak devices act on alerts propagated securely via edge gateways and the blockchain. This structure ensures minimal dependence on cloud resources, promotes scalability, and enhances system resilience.

In parallel, a machine learning detection model was developed using the CICIDS2017 dataset, which provides labeled data for various attack types including PortScan, Web Attacks (XSS, SQL Injection, Brute Force), and Infiltration. Feature selection using Grey Wolf Optimization allowed the model to operate efficiently without sacrificing accuracy. The resulting ensemble model combining Random Forest and XGBoost demonstrated strong classification performance with an overall F1-score of 0.9995, validating the system's ability to detect both frequent and complex attacks.

The system was further evaluated in a simulated smart city environment through a Flask-based dashboard, which provided real-time logging, alert visualization, and interactive test inputs. The experimental setup successfully replicated real-world behaviors, including alert generation, propagation via blockchain, and autonomous responses by weak IoT devices. Results showed high system responsiveness, low false positives, and clear evidence of tamper-resistant alert logging.

This research not only demonstrates the feasibility of deploying an effective IDS in constrained and distributed IoT environments but also lays a foundation for future work. Potential directions include integrating smart contracts for automated alert validation, expanding attack coverage using adversarial training, or adopting federated learning models for cross-network knowledge sharing while preserving data privacy.

In conclusion, the proposed system represents a significant advancement in IoT security one that aligns with the goals of Industry 4.0 and smart city initiatives. By bridging the gap between theory and practical implementation, this thesis contributes a secure, adaptive, and future-ready framework for intrusion detection in the modern digital landscape.

References

[1] Cisco, "What is IT Security?" [Online]. Available: https://www.cisco.com/site/us/en/learn/topics/security/what-is-it-security.html. [Accessed: May 2, 2025].

[2] Wikipedia contributors, "Information security," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Information_security. [Accessed: May 2, 2025].

[3] Fortinet, "What is the CIA Triad and Why is it Important?" [Online]. Available: https://www.fortinet.com/resources/cyberglossary/cia-triad. [Accessed: May 2, 2025].

[4] Wikipedia contributors, "Computer security," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Computer_security. [Accessed: May 2, 2025].

[5] IBM, "Types of Cybersecurity," IBM Topics, [Online]. Available: https://www.ibm.com/topics/cybersecurity. [Accessed: May 2, 2025].

[6] Nexus Group, "What are the security challenges of IoT?" [Online]. Available: https://www.nexusgroup.com/what-are-the-security-challenges-of-iot/. [Accessed: Apr. 15, 2025].

[7] N. S. M. Albakri, A. Alzahrani, and S. Khan, "A Survey of Intrusion Detection Systems and Blockchain-Based Approaches in the Internet of Things," *IEEE Access*, vol. 10, pp. 91127–91152, 2022.

[8] R. Vinayakumar et al., "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[9] K. Scarfone and P. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS)," *NIST Special Publication 800-94*, 2007. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-94.pdf

[10] P. Garcia-Teodoro et al., "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28, pp. 18–28, 2009.

[11] R. G. Bace and P. Mell, "Intrusion Detection Systems," *NIST*, 2001.

[12] S. Axelsson, "Intrusion Detection Systems: A Survey and Taxonomy," Technical report, Chalmers University of Technology, 2000.

[13] H. Debar, M. Dacier, and A. Wespi, "A revised taxonomy for intrusion-detection systems," *Annales des télécommunications*, vol. 55, no. 7–8, pp. 361–378, 2000.

[14] ITU-T Y.2060, "Overview of the Internet of Things," International Telecommunication Union, 2012.

[15] IEEE IoT Initiative, "The Internet of Things," IEEE, 2013.

[16] CERP-IoT Vision, "The Internet of Things: A Dynamic Global Infrastructure," CERP-IoT, 2008.

[17] Wikipedia contributors, "Industrial Internet of Things," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Industrial_Internet_of_Things. [Accessed: May 2, 2025].

[18] Trend Micro, "Industrial Internet of Things (IIoT)," [Online]. Available: https://www.trendmicro.com/vinfo/us/security/definition/industrial-internet-of-things-iiot.

[19] L. D. Xu, W. He, and S. Li, "Internet of Things in industries: A survey," *IEEE Trans. Ind. Inf.*, vol. 10, no. 4, pp. 2233–2243, 2014.

[20] T. H. H. Aldhyani et al., "A Survey of Security in Cloud, Edge, and Fog Computing," *Sensors*, vol. 22, no. 3, p. 927, 2022.

[21] L. D. Xu, E. L. Xu, and L. Li, "Industry 4.0: state of the art and future trends," *Int. J. Prod. Res.*, vol. 56, no. 8, pp. 2941–2962, 2018.

[22] A. J. Latham, "Artificial Intelligence in Cybersecurity: The Future of Threat Detection," Cybersecurity Magazine, [Online]. Available: https://cybersecurity-magazine.com/artificial-intelligence-in-cybersecurity/. [Accessed: May 2, 2025].

[23] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in *IEEE Symposium on Security and Privacy*, 2010.

[24] Y. Meidan et al., "N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[25] R. Doshi et al., "Machine Learning DDoS Detection for Consumer Internet of Things Devices," in *IEEE Security and Privacy Workshops*, 2018, pp. 29–35.

[26] T. Yang et al., "Fog2Sec: Fog Computing-Based Intrusion Detection System for IoT," *Sensors*, vol. 19, no. 6, p. 1251, 2019.

[27] P. K. Sharma et al., "Blockchain-Based Decentralized Framework for Intrusion Detection in IoT," *IEEE Access*, vol. 9, pp. 51286–51298, 2021.

[28] S. Singh et al., "Blockchain-based security for IoT: A comprehensive survey," *Procedia Computer Science*, vol. 220, pp. 234–247, 2023.

[29] A. Reyna et al., "On blockchain and its integration with IoT: Challenges and opportunities," *Future Generation Computer Systems*, vol. 88, pp. 173–190, 2018.

[30] I. Makhdoom et al., "Blockchain's adoption in IoT: The challenges, and a way forward," *Journal of Network and Computer Applications*, vol. 125, pp. 251–279, 2019.

[31] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018.

[32] A. Sivanathan et al., "Characterizing and classifying IoT traffic in smart cities and campuses," in *Proc. IEEE INFOCOM WKSHPS*, 2017.

[33] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning

approach for Internet of Things," *Future Generation Computer Systems*, vol. 82, pp. 761–768, 2018.

[34] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018.

[35] Canadian Institute for Cybersecurity, "CICIDS2017 Dataset Description," [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html

[36] "Port Scanning Attack," GeeksforGeeks, Mar. 6, 2024. [Online]. Available: https://www.geeksforgeeks.org/port-scanning-attack/

[37] "What is Cross-Site Scripting (XSS)?", Black Duck Software Glossary, [Online]. Available: https://www.blackduck.com/glossary/what-is-cross-site-scripting.html

[38] "Qu'est-ce que l'injection SQL (SQLi)?", Checkpoint Software Technologies, [Online]. Available: https://www.checkpoint.com/fr/cyber-hub/cyber-security/what-is-cyber-attack/what-is-sql-injection-sqli

[39] "SQL Injection", Fortinet CyberGlossary, [Online]. Available: https://www.fortinet.com/fr/resources/cyberglossary/sql-injection

[40] "Brute Force Attack," Fortinet CyberGlossary, [Online]. Available: https://www.fortinet.com/resources/cyberglossary/brute-force-attack

[41] "What is Infiltration? Exploring the Threat of Infiltration in Cybersecurity," Antivirus.com. [Online]. Available: https://www.antivirus.com/articles/infiltration-in-cybersecurity. [Accessed: Jun. 8, 2025].

[42] GeeksforGeeks, "Grey Wolf Optimization - Introduction," 2023. [Online]. Available: https://www.geeksforgeeks.org/grey-wolf-optimization-introduction/

[43] GeeksforGeeks, "Random Forest Classifier using Scikit-learn," Mar. 11, 2025. [Online]. Available: https://www.geeksforgeeks.org/random-forest-classifier-using-scikit-learn/

[44] GeeksforGeeks, "XGBoost," 2023. [Online]. Available: https://www.geeksforgeeks.org/xgboost/

[46] A. Alqahtani et al., "Efficient feature selection and lightweight deep learning model for DDoS attack detection in IoT-enabled smart city environments," *Journal of Network and Systems Management*, vol. 32, no. 2, Article 52, 2024.

[47] M. A. Al-Garadi et al., "A survey of machine and deep learning methods for IoT security," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7792–7809, 2020.

[48] M. A. Ferrag et al., "RDTIDS: Rules and decision tree-based intrusion detection system for Internet of Things," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4929–4946, 2021.

[49] G. van Rossum, *Python Tutorial*, Amsterdam, Netherlands: Centrum voor Wiskunde en

Informatica (CWI), 1995.

[50] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*, 2nd ed.

[51] D. R. Hipp, *SQLite Documentation*, SQLite.org, 2012. [Online]. Available: https://www.sqlite.org/docs.html. [Accessed: Jun. 14, 2025].

[52] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[53] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD)*, San Francisco, CA, USA, 2016, pp. 785–794.

[54] F. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary Algorithms Made Easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, Jul. 2012.

[55] Microsoft, *Visual Studio Code Documentation*, 2023. [Online]. Available: https://code.visualstudio.com/docs. [Accessed: Jun. 14, 2025].