# SECURING IoT DEVICES IN EDGE COMPUTING THROUGH REINFORCEMENT LEARNING

Thesis Submitted for the Award of the Degree of

## DOCTOR OF PHILOSOPHY

**in**

**Computer Science & Engineering**

**By**

**Anit Kumar**

**Registration Number: 41800692**

**Supervised By**

**Dr. Dhanpratap Singh (25706)**

**Computer Sc. & Engg. (Professor)**

**Lovely Professional University, Punjab**


**LOVELY PROFESSIONAL UNIVERSITY, PUNJAB**

**2026**

# DECLARATION

I, hereby declared that the presented work in the thesis entitled "Securing IoT Devices in Edge Computing through Reinforcement Learning" in fulfilment of degree of **Doctor of Philosophy (Ph. D.)** is outcome of research work carried out by me under the supervision of Dr. Dhanpratap Singh, working as a Professor, in the Computer Science & Engineering Department of Lovely Professional University, Punjab, India. In keeping with the general practice of reporting scientific observations, due acknowledgements have been made whenever work described here has been based on findings of another investigator. This work has not been submitted in part or in full to any other University or Institute for the award of any degree.

**(Signature of Scholar)**

Name of the scholar: Anit Kumar

Registration No.: 41800692

Department/school: Computer Science & Engineering

Lovely Professional University,

Punjab, India

# CERTIFICATE

This is to certify that the work reported in the Ph. D. thesis entitled "Securing IoT Devices in Edge Computing through Reinforcement Learning" submitted in fulfillment of the requirement for the award of the degree of **Doctor of Philosophy (Ph.D.)** in the Computer Science & Engineering Department is a research work carried out by Anit Kumar, (41800692), is a bonafide record of his original work carried out under my supervision and that no part of thesis has been submitted for any other degree, diploma, or equivalent course.

**(Signature of Supervisor)**

Name of supervisor: Dr. Dhanpratap Singh

Designation: Professor

Department/School: Computer Science & Engineering

University: Lovely Professional University, Punjab, India

## PREFACE

In this PhD thesis document titled “Securing IoT Devices in Edge Computing through Reinforcement Learning”, we provided an in-depth introduction and extensive literature review to highlight the need for security of the data collected by an IoT device for Edge computing. Again, we proposed a novel Reinforcement Learning based security method for securing the edge computing of IoT devices. The proposed method acts as a middleware between IoT devices and edge devices, and outperforms similar existing benchmark methods.

The objective of the proposed work is to ensure the utmost security of IoT data when it is sent to the edge server for further processing to extract or filter the relevant data for the intended purpose. The processed data, in general, is also sent to cloud storage for global access.

I want to express my heartfelt gratitude to my respected supervisor, Dr. Dhanpratap Singh, for his guidance and support throughout this research work.

## ABSTRACT

Exponentially increasing demand for IoT devices, with the expectation of maximum user fulfilment needed to integrate an edge server with the IoT devices. The small size of an IoT device, but the need for complex computation and high-end software requirements, demanded an additional hardware setup that could never be possible without an edge server. The Edge server collects data from an IoT device to reduce the computational burden and data transmission requirements. With the Edge server integration, the prime responsibility of the IoT device is to sense the data from the surroundings and send it to the edge server through the internet or via some other communication channel for further computation. After processing the data on the edge server, it is generally sent for permanent storage locally or on a cloud server for global access.

As the IoT-Edge computing practice advances, it has become a target for attackers seeking to intercept and extract sensitive data transmitted between IoT devices and edge servers. The dependence of IoT systems on numerous edge devices to process and gather large critical data has generated the need for the security of IoT-Edge computation. Gradually, Smart hackers also started using artificial intelligence tools to attack the edge server with some malicious intent. The IoT network is susceptible to various types of malware attacks, but Distributed Denial-of-Service (DDoS) attacks have shown a notable increase in recent years, rising by about one-third annually.

The increasing incidents of such serious attacks on the edge server motivated the researchers to model a security layer on the edge server that will protect it from malware attacks. To counter such a major threat, we have proposed a scalable and robust security model to protect an edge server using a novel reinforcement learning method that is configurable with multiple IoT devices and the edge server, providing consistency even in high volumes of malicious data.

Reinforcement Learning (RL) is a widely used machine learning approach that enables an autonomous agent to make decisions by interacting with a dynamic environment, aiming to learn optimal actions that maximize cumulative rewards over time. The proposed work is designed and implemented by leveraging a novel reinforcement learning method, and it is verified in two phases. The first phase involved simulating IoT-Edge computation, while the second and final phase extended the proposed security model to real-time data communication and was evaluated sequentially using a public IoT dataset and an In-house IoT device dataset.

The proposed model is found to be superior in terms of response time, detection rate, scalability, robustness, and consistency. The proposed secure framework provided the expected

results in detecting and preventing DDoS attacks in IoT-Edge computing, with a very low error rate. Extensive experiments conducted on Simulation, Public, and In-house IoT datasets demonstrate that the proposed security method achieves a much better detection accuracy, surpassing benchmark methods including Support Vector Machines (SVM), Random Forest (RF), and Deep Reinforcement Learning (DRL) in terms of accuracy, scalability, consistency, and robustness. These results highlight the proposed method as an effective and adaptive solution for securing next-generation IoT-Edge systems.

**Keywords:** IoT device, Edge server, Cloud server, Novel Reinforcement Learning, Network security, Scalability, Robustness

# TABLE OF CONTENTS

# List of Tables

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| **AI** | Artificial Intelligence |
| **ML** | Machine Learning |
| **RL** | Reinforcement Learning |
| **IDS** | Intrusion Detection System |
| **MDRL** | Message-Driven Based Reinforcement Learning |
| **AEGRL** | Adaptive Epsilon Greedy Reinforcement Learning Method |
| **DDoS** | Distributed Denial-of-Service |
| **IoT** | Internet of Things |
| **IIoT** | Industrial Internet of Things |
| **MN** | Malicious Node |
| **AODV** | Ad hoc On-Demand Distance Vector |
| **MQTT** | Message Queuing Telemetry Transport |
| **PDR** | Packet Delivery Ratio |
| **E2E** | End-to-End |
| **TPR** | True Positive Rate |
| **FPR** | False Positive Rate |
| **DR** | Detection Rate |

# CHAPTER 1

# INTRODUCTION

The evolution of IoT devices stems from the need to sense data from the surrounding environment implicitly. This requires generating IoT devices with sensing, communication, and microcontroller capabilities. It makes IoT devices a bridge between the physical and cyber worlds. Moving from personal needs to complex real-time user requirements requires a network of IoT systems where each has dedicated tasks and collaborates whenever needed to achieve a common goal. The exponential growth of the IoT network generates vast amounts of data that cannot be preprocessed within an IoT device. At this time, Engineers began to use and deploy a separate computing machine that could do the preprocessing of the data on behalf of IoT devices. This need for a separate computing machine arose in the form of an Edge server [1] - [10].

The Internet of Things (IoT) is reshaping contemporary life in multiple sectors, ranging from personal healthcare to industrial production. However, the widespread integration of IoT into daily life also amplifies the risks of data privacy breaches and cybersecurity threats, necessitating robust security frameworks to mitigate potential vulnerabilities. A typical IoT setup consists of affordable, resource-limited devices linked to a nearby IoT gateway. This gateway facilitates bidirectional data flow between the connected devices and the Internet, enabling seamless communication and remote access while also serving as a potential security checkpoint [11] - [17].

Edge computing makes security and privacy techniques critical for IoT communication systems. **Figure 1** is specifying how edge servers can be used with IoT devices used in a Smart Home, Smart Industry, or Smart City to improve the user experience with the use of the Internet. The intention of introducing an edge server with an IoT device is to provide fast response, preprocessing of the data before sending it to the cloud, and dealing with IoT device resource constraints to enable the device to do complex computation at the corresponding edge server end [18] - [23].

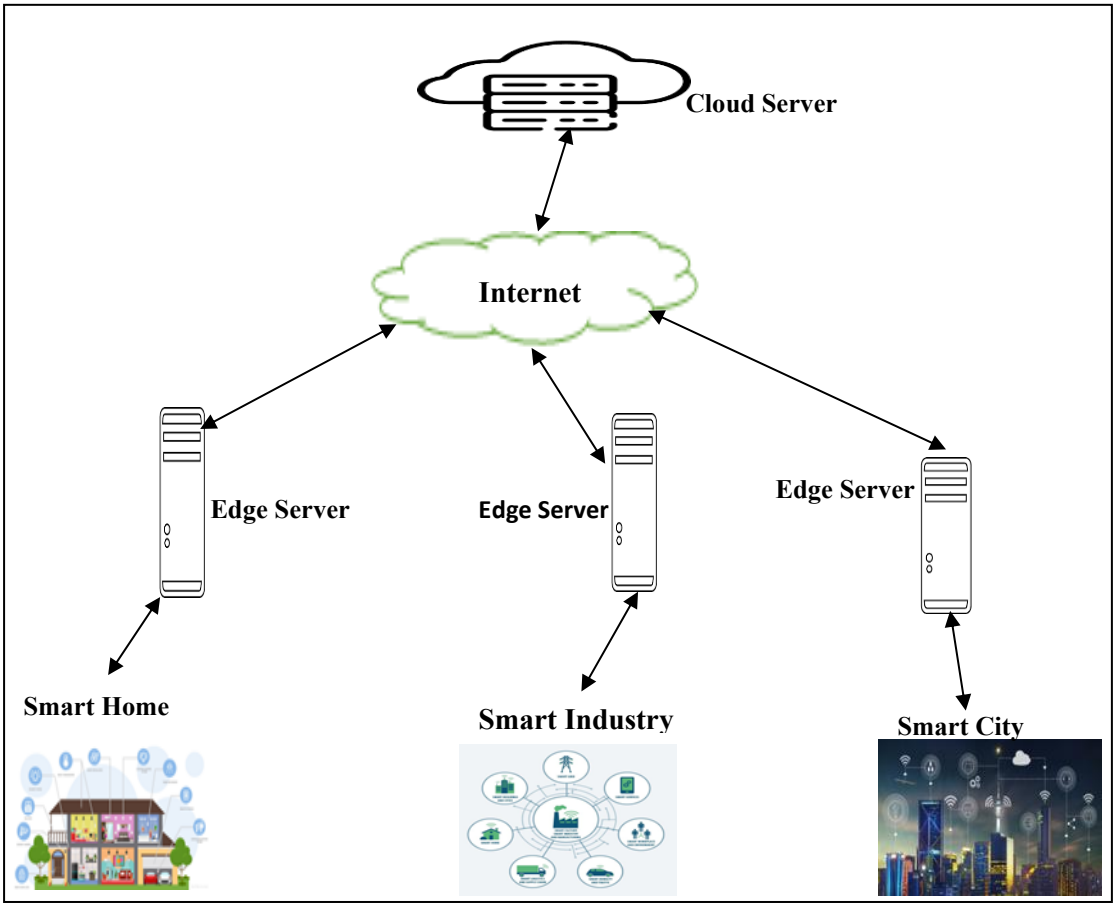



**Figure 1:** Edge Computing Basic Architecture for IoT devices

An IoT-Edge Communication Network with real-time data communication involves deploying edge computing devices that interact with IoT sensors, process data locally, and transmit only relevant insights to the cloud or centralized systems. This setup reduces latency, bandwidth efficiency, and security while enabling real-time decision-making [24] - [26]. Edge computing is a subset of Fog computing. Fog computing is broader and includes the Edge layer plus additional network-layer processing [27]. **Figure 2** illustrates how an Edge server collects data from IoT devices in real time for processing and forwards the processed output to the Cloud server for global access. Edge computing is a distributed architectural paradigm in Internet of Things (IoT) infrastructure that decentralizes computation and data storage by relocating these functions closer to the data-generating sources, such as sensors, devices, and gateways. Unlike traditional cloud-centric models that require transmitting all data to centralized data centers, edge computing processes data locally or near the edge of the network, significantly reducing latency and bandwidth usage. Bringing computation closer to the

edge offers multiple benefits, particularly in terms of performance, security, and system resilience.

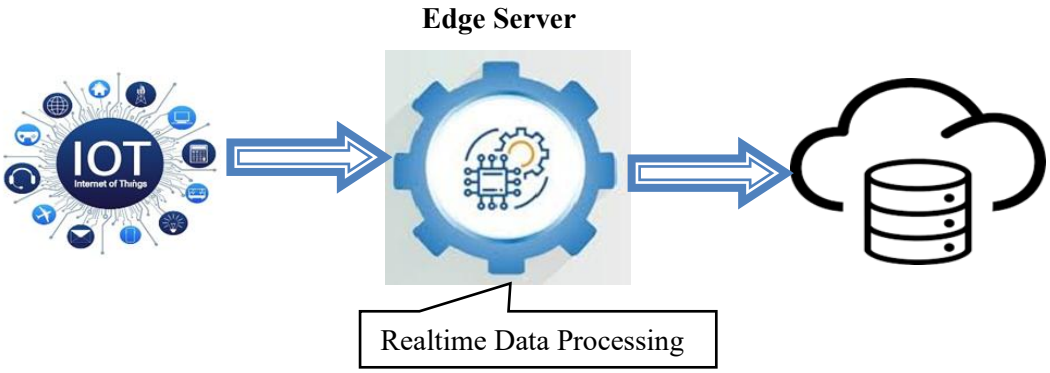



**Figure 2:** IoT-Edge-Cloud Communication

The Internet of Things (IoT) is a network of interconnected physical devices that collect and exchange data over the Internet. These devices include sensors, actuators, smart appliances, and other embedded systems. The communication between IoT devices, edge computing nodes, and cloud servers forms the backbone of many modern distributed computing architectures. IoT devices are typically resource-constrained regarding computing power, energy, and storage. They collect real-time data from the physical environment and transmit it to more capable nodes (Edge or Cloud) for processing and decision-making. The rapid advances in IoT devices have made human lives easier in many ways, including smart healthcare, innovative industries, smart cities, and intelligent transportation [27] [28].

Edge computing involves processing data near the point of generation. Edge nodes may be gateways, routers, or microdata centers. Edge computing reduces latency, saves bandwidth, and enhances real-time decision-making. It acts as an intermediary between IoT devices and cloud servers by filtering, aggregating, or pre-processing data before it is sent to the cloud. Cloud servers offer scalable, centralized infrastructure for large-scale storage, advanced analytics, and machine learning. Data that does not require immediate action or has long-term value is sent to the cloud for deeper processing and storage [34].

However, it also increases the security concerns from these devices, which may lead to sensitive data falling into the wrong hands for self-benefit. The intruders take advantage

of the resource constraints of the IoT devices and enter into IoT dynamics to steal sensitive information from them [35] - [38].

Security in IoT devices is not limited to wired or wireless communication but depends on device control and authentication issues. Many IoT devices use low power and network bandwidth, mainly through low-voltage batteries. Hence, the improper supply of energy to IoT devices may cause malfunction of their hardware and software, and provide chances for an outsider to enter the IoT communication with bad intent [39] [40].

The open and dynamic characteristics of IoT devices utilizing wireless sensor technologies expose inter-sensor and sensor-to-controller communications to various security threats, which can degrade device performance and hinder future development. In these scenarios, ensuring anonymous authentication for IoT sensors despite their strict resource limitations becomes a critical security challenge [56] - [60].

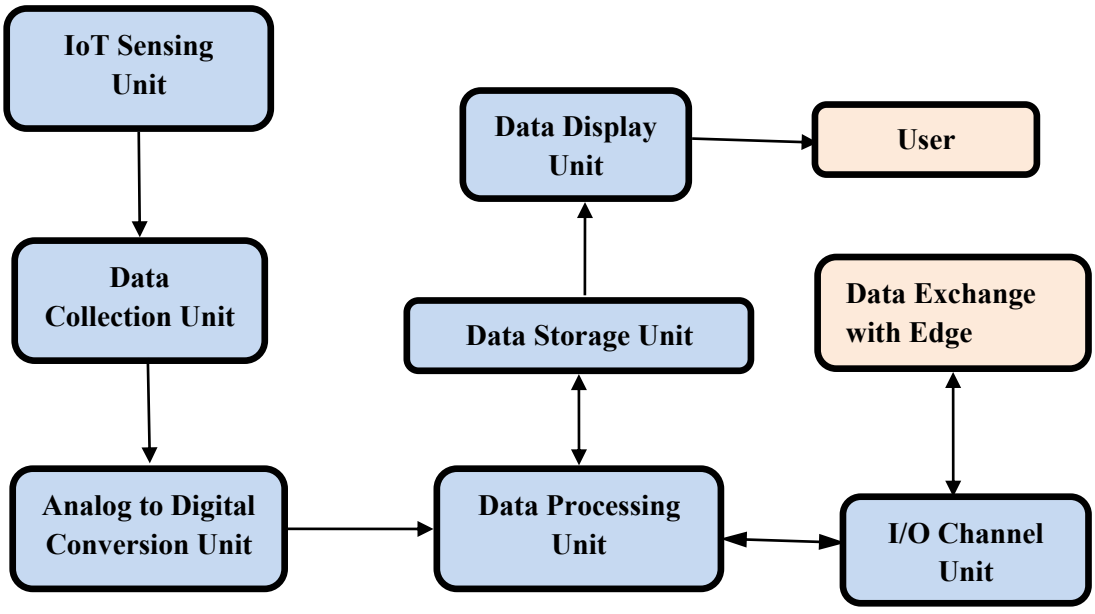



**Figure 3:** Data Communication Block Diagram between an IoT Device and the Edge Server

Here, **Figure 3** represents the data communication block diagram between an IoT device and the Edge server. An IoT sensing unit (or IoT sensor unit) is a core component in an Internet of Things (IoT) system that detects and measures physical or environmental changes in the real world and converts them into data that can be

processed or transmitted. IoT-Edge computing works over several components comprising the following main units [11].

1. **IoT Sensing Unit**: This unit collects data from the surrounding environment for the intended purpose.
2. **Data Collection Unit**: This unit facilitates streamlining the data collected from the IoT sensing unit.
3. **Analog to Digital Conversion Unit**: The data collected by the IoT device from the surroundings are analog signals, which are converted to digital data by using this unit for later purposes.
4. **Data Processing Unit**: This unit processes digital data as input to generate the desired output data.
5. **I/O Channel Unit**: This unit acts as a bridge to exchange data between the IoT device and any other external device.
6. **Data Storage Unit**: Inside the IoT device, this unit stores the temporary or permanent data in the built-in memory device.
7. **Data Display Unit**: This is the built-in display unit for the end users to visualize the processed output data.

In our day-to-day communication network, the data comes from a central cloud storage. In cloud storage, data comes from various sources, where each source might be an IoT device, and several IoT devices may communicate with each other before sending the data for permanent storage at the cloud layer. These IoT devices might be the same or different depending on the purpose of their use. Also, the increasing demand for computing services from IoT devices requires heterogeneous tasks to be accommodated within a single IoT processor, which eventually extends the IoT computing services from the Cloud server to the Edge server, but at the same time, it invites new network security challenges and concerns over IoT computation. This broad heterogeneity of IoT networks often does not come under a unified secure communication protocol. This comes as a benefit for the malicious outsider to be available to represent itself as a legal body of contact with the underlying security layer of the data provider, which could be an IoT device [61] - [63].

IoT devices in general use and deploy Edge devices to serve as intermediaries among sensors, servers, or cloud services, which enables the Edge server to gather a vast amount

of data from various sources. Edge Computing provides fast computation and communication in the IoT network, but it also produces privacy and security challenges due to the existence of IoT devices. IoT has emerged as a prominent technology, and its application is tremendous in various sectors, including smart healthcare, smart transport, smart homes, and smart cities. IoT devices are generally heterogeneous in physical and functional means that they show very few similar physical and functional features in common. IoT devices communicate with one another through either a wireless or wired channel [64] - [67].

Given the limited computational and storage capacities of edge servers, achieving efficient and equitable resource allocation within the detection system presents a significant challenge. The absence of trust in IoT edge devices is a key factor limiting the widespread adoption of IoT edge computing as an outsourced computing solution. In the past few years, IoT computing has made significant advancements, incorporating advanced communication and data sensing features that can be accessed anywhere, at any time. It results in an increasing number of cyberattacks on the edge server. This has led to the necessity of deploying an intrusion detection system (IDS) to counter cybersecurity threats in edge computing environments [36] [38]. The edge server in IoT computing remains at the central point where certain IoT devices request data transfer with the Edge server in a real-time, dynamic data communication phase. Once an IoT device gets authenticated, it gets admitted into the IoT-Edge network dynamics. Even in some networks, Edge servers may have limited resources and computational capabilities, due to which continuous load on the edge server in peak hours can benefit the attackers and allow them to succeed as legitimate senders for the Edge server [68] - [71].

Edge servers play a crucial role in modern IoT architectures by performing local data preprocessing before transmitting refined information to cloud servers for long-term storage and advanced analytics. This localised computation minimises latency, reduces bandwidth consumption, and enhances system efficiency. However, as edge servers handle sensitive and mission-critical data, they have increasingly become attractive targets for cybercriminals seeking to exploit vulnerabilities within the IoT-Edge-Cloud communication pathway. This growing threat landscape has prompted the necessity for

implementing robust, multilayered security frameworks at the edge layer to authenticate users, detect anomalies, and prevent unauthorised access. In recent years, edge computing has evolved as an essential extension of traditional cloud computing, bringing computational power and intelligence closer to IoT devices, thereby enabling faster decision-making and improved data privacy in distributed environments [72] - [76].

Some of the most notorious cyberattacks involving compromised IoT devices have had severe consequences. One such incident occurred in a German steel mill in 2015, where hackers infiltrated the production network, as confirmed by the German Federal Office for Information Security. The attack disabled critical systems, preventing personnel from shutting down a blast furnace and ultimately causing catastrophic damage. This is not an isolated case of IoT-targeted cyberattacks. Significant incidents have included the 2015 cyberattack on Ukraine's power grid, the remote deactivation of an offshore oil rig, the compromise of kitchen appliances in the UK, and the exploitation of smart refrigerators to send spam messages. One of the most infamous IoT-related cyberattacks in history was the Mirai malware attack in 2017. Mirai infected over 380,000 IoT devices, creating vast networks of compromised devices remotely controlled to execute large-scale distributed denial-of-service (DDoS) attacks. As reported by the U.S. Computer Emergency Readiness Team (CERT), Mirai stands as the most extensive DDoS attack recorded, highlighting the persistent security vulnerabilities in IoT systems [77] [78].

**Figure 4** represents the hierarchical structure of imposed IoT vulnerabilities on IoT-Edge computation. In the context of IoT-Edge computing, the primary target of these attacks is the Edge server, which plays a critical role in managing and preprocessing data from various IoT devices. The importance of the Edge server makes it an attractive target for attackers, as its disruption can cripple the entire IoT-Edge-Cloud communication chain. When an Edge server is compromised by a DDoS attack, the entire IoT communication process is interrupted, potentially leading to significant data loss or delay in services. What makes these attacks even more dangerous is that once the legitimate Edge server is overwhelmed and taken offline, malicious nodes from the pool of distributed attack sources may pose as legitimate Edge servers. These rogue

servers can then intercept sensitive data from IoT devices, leading to unauthorised access and data theft. This type of attack not only undermines trust in IoT-Edge infrastructures but also exposes both users and service providers to severe financial and reputational damage [80] - [85].

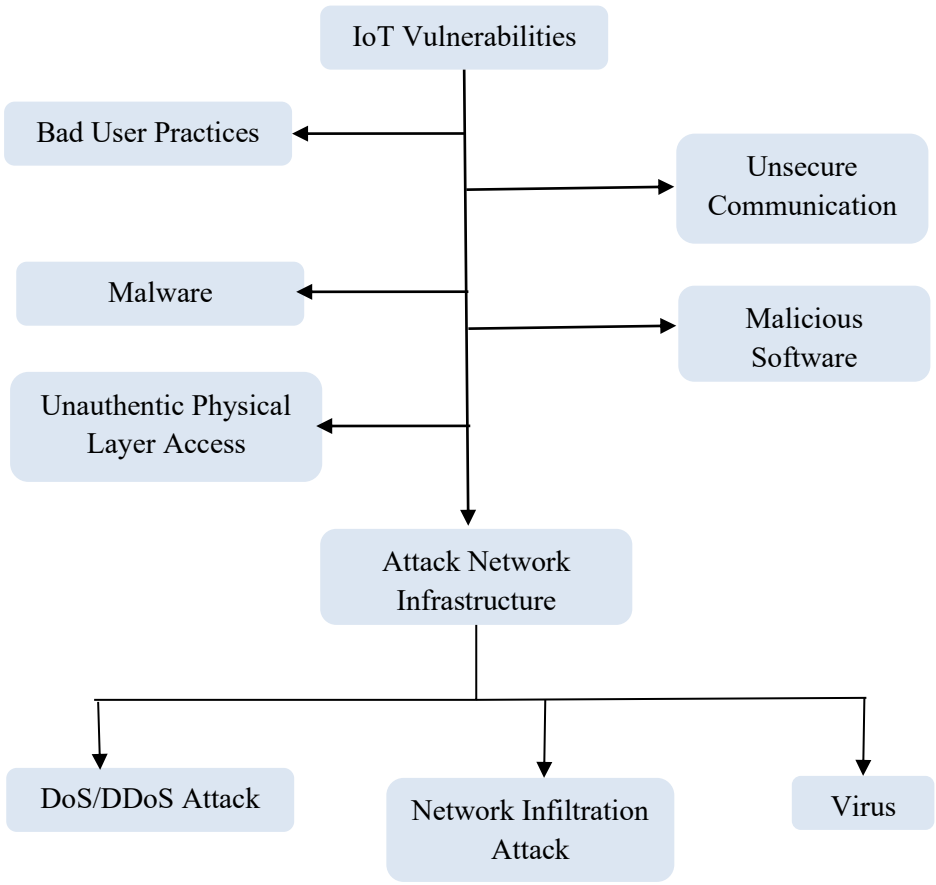



**Figure 4:** Hierarchical Tree Structure for Different IoT Attacks

IoT vulnerabilities are security weaknesses in connected devices that attackers can exploit. These vulnerabilities can affect everything from smart home devices and wearables to industrial systems and medical equipment. IoT devices often rely on cloud infrastructure. A breach in the cloud service can affect all connected devices [88].

Edge computing has significantly enhanced the deployment of networked services by enabling faster response times and reduced bandwidth consumption. This is accomplished by moving computation and storage to edge servers positioned near data sources, such as IoT devices and sensors, thereby reducing latency caused by transmitting data to distant cloud centres. Despite these advantages, securing edge

servers remains a formidable challenge. Unlike centralized data centres, which benefit from robust and layered security architectures, edge servers are typically deployed in less controlled, more exposed environments. This distributed nature not only increases the attack surface but also limits the implementation of consistent and scalable security policies. Distributed Denial-of-Service (DDoS) attacks pose one of the most significant security risks to edge servers. Edge computing environments often lack the centralized monitoring and orchestration systems found in traditional data centers, making real-time threat detection and coordinated defense mechanisms more difficult to implement. The heterogeneity of edge hardware and software stacks further complicates the deployment of uniform security solutions across all nodes. In light of these challenges, there is a pressing need for adaptive, lightweight, and distributed security mechanisms tailored for edge environments. Techniques such as reinforcement learning, anomaly detection, and secure data offloading are being increasingly explored to safeguard edge infrastructures against evolving threats, including DDoS attacks, spoofing, and malware intrusions [89] - [93].

**Figure 5** illustrates the increasing number of cyberattacks that are compromising IoT Edge computing infrastructure in today's digital landscape [79]. The analysis presented in this thesis primarily focuses on Distributed Denial-of-Service (DDoS) attacks, which have emerged as one of the most significant security threats within the IoT Edge ecosystem. DDoS attacks are particularly challenging to manage because they are executed by many distributed sources, making their detection and prevention complex for service providers. This type of attack is commonly aimed at network communication channels for malicious gain, and it is notorious for its ability to overwhelm networks with excessive traffic. The widespread occurrence of DDoS attacks can be attributed to their dual impact. First, they disrupt normal network operations by flooding the communication channels with an overwhelming amount of traffic, which not only slows down the system but can also bring it to a halt. Second, by targeting the network from multiple distributed sources, these attacks make it exceptionally difficult for security measures to identify and neutralize the threat in real-time. This disruption often results in the denial of vital services, severely affecting the network's ability to function and communicate effectively [94] - [97].

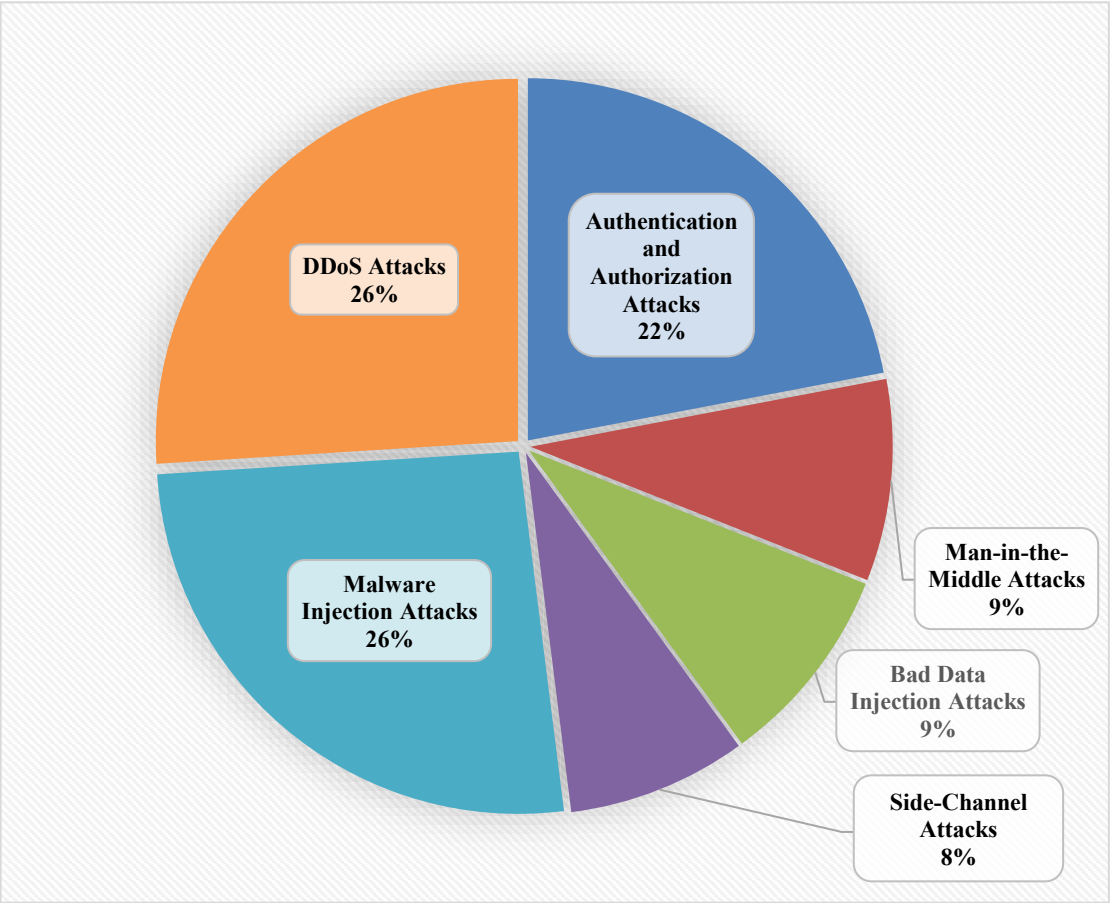



**Figure 5**: Percentages of Different Types of Attacks on Edge Computing Infrastructures in the Real World

The Open Systems Interconnection (OSI) model is a foundational framework in conventional networking, widely adopted for its structured approach to communication. It divides network functions into seven distinct layers, beginning with the Physical Layer at the bottom and culminating with the Application Layer at the top. Each layer handles specific responsibilities and offers standardized interfaces to adjacent layers, promoting interoperability, modularity, and scalable network architecture. Although the OSI model serves as a foundational framework in traditional networking, its direct application in Internet of Things (IoT) environments, particularly those that are resource-constrained, poses several challenges. Most IoT devices are designed to be lightweight, with limited processing power, memory, and energy resources, to maintain cost efficiency and prolong battery life. Implementing the full OSI stack in such devices

introduces significant overhead, resulting in increased code complexity, higher energy consumption, and inefficient resource utilization. Moreover, certain OSI layers may offer functionality that exceeds the practical needs of basic IoT operations, further contributing to unnecessary computational burden. As a result, many IoT architectures opt for simplified or custom communication stacks tailored to the constraints and specific requirements of IoT ecosystems. Each layer in the OSI model introduces its framing, control messages, and processing requirements, which can result in excessive data overhead, increased latency, and higher energy demand, particularly detrimental for battery-operated devices [98] - [101].

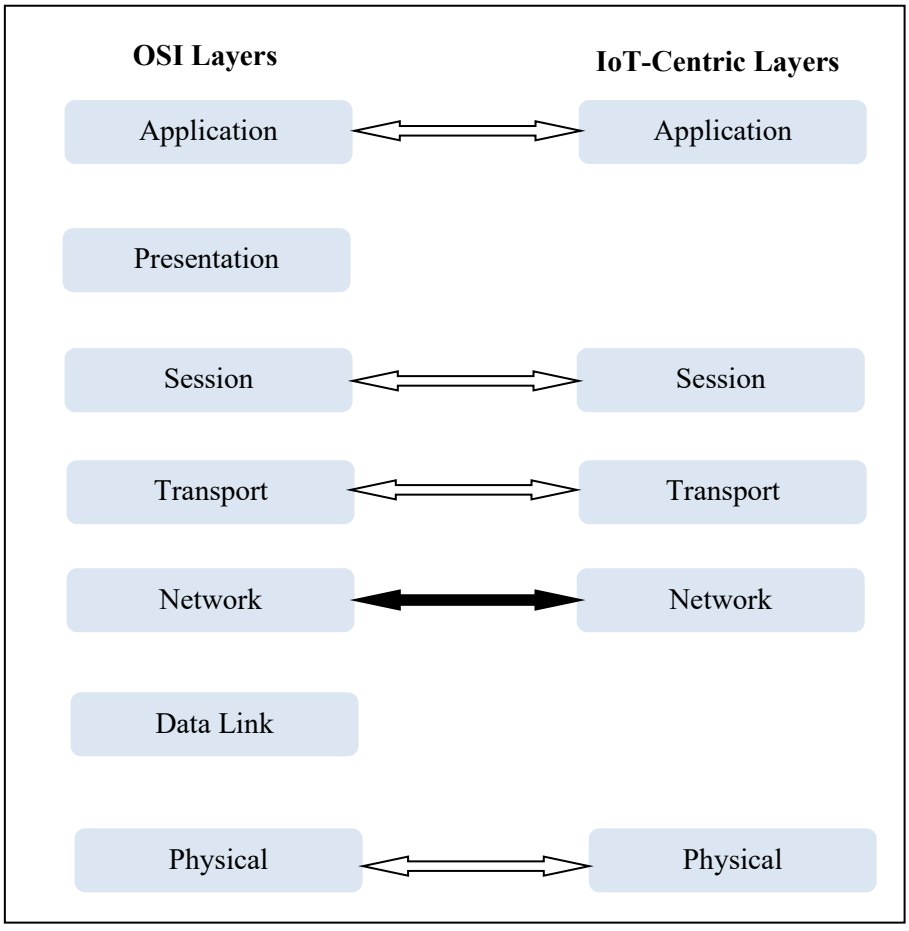



**Figure 6:** IoT Centric Network Communication Model

The proposed security framework encounters DoS/DDoS attacks on IoT-Edge computing, which occurs at the network layer of the IoT-Centric Network Communication Model. **Figure 6** illustrates the OSI layers and IoT-centric layers

involved in data communication. The Open Systems Interconnection (OSI) model is one of the most widely used frameworks for structuring network communications. It divides communication processes into seven distinct functional layers, each responsible for specific tasks while interacting with adjacent layers. This modular approach simplifies the development of scalable and interoperable networks. While the OSI model applies to the Internet of Things (IoT), it presents challenges, particularly for low-power, resource-constrained devices. The layered structure increases complexity, requiring additional code, memory, and processing power, which can lead to higher energy consumption. Furthermore, each layer introduces data overhead due to extra framing and control messages. These factors may not be ideal for IoT deployments involving simple, battery-operated devices. However, despite these limitations, the layered architecture offers flexibility, scalability, and enhanced interoperability, making it a valuable model for many network applications [100] - [102].

The proposed novel reinforcement learning method is implemented over two widely used network communication protocols for IoT devices, which are as follows.

1) **Ad hoc On-Demand Distance Vector (AODV)**

The Ad hoc On-Demand Distance Vector (AODV) routing protocol is a widely recognised and popular routing protocol used in Mobile Ad Hoc Networks (MANETs). MANETs are dynamic, infrastructure-less networks where nodes can freely move and communicate directly with each other over wireless links. In such networks, maintaining reliable and efficient routes becomes a challenge due to frequent topology changes, node mobility, and limited energy resources. AODV addresses these challenges by establishing routes only when they are needed, thus minimising unnecessary overhead and conserving bandwidth and energy. It operates using a route discovery mechanism that broadcasts route request (RREQ) packets throughout the network, and when a valid path is found, it responds with a route reply (RREP). This reactive approach ensures that routes are always up-to-date, which is critical in highly dynamic network environments. The protocol also employs sequence numbers to ensure loop-free routing and the freshness of route information, enhancing its reliability [37] [67] [68]. Due to its simplicity, low processing requirements, and efficient use of bandwidth, AODV has been

extensively adopted and evaluated in academic research and practical implementations of MANETs, vehicular ad hoc networks (VANETs), and even in some IoT-based scenarios. Many recent studies have proposed enhancements to AODV, such as incorporating energy-aware metrics, security mechanisms, and QoS parameters to improve its performance under various conditions. Overall, AODV remains a foundational protocol in the study of dynamic routing in wireless networks and continues to inspire new protocols and optimizations for mobile and distributed network environments [38] [39] [69] [101].

2) **Message Queuing Telemetry Transport (MQTT)**

MQTT was invented at IBM in 1999 as a messaging protocol. It started to help communication between Satellites and Oil pipeline sensors [59]. It is a lightweight, publish-subscribe messaging protocol widely used in the Internet of Things (IoT) for communication among devices. In 2014, OASIS accepted MQTT as an ISO Standard, making it a universal acceptance for IoT messaging protocols. The MQTT protocol is very suitable for IoT devices that have minimal computational hardware resources. Small microcomputers work perfectly over the MQTT protocol [86]. With all these advantages and popularity, we decided to use the MQTT protocol for underlying data communication. We set up the Node-RED server [47] to establish and enable MQTT protocol-based communication. **Figure 7** shows the Node-RED server running on our implemented Edge server to send/receive data packets.

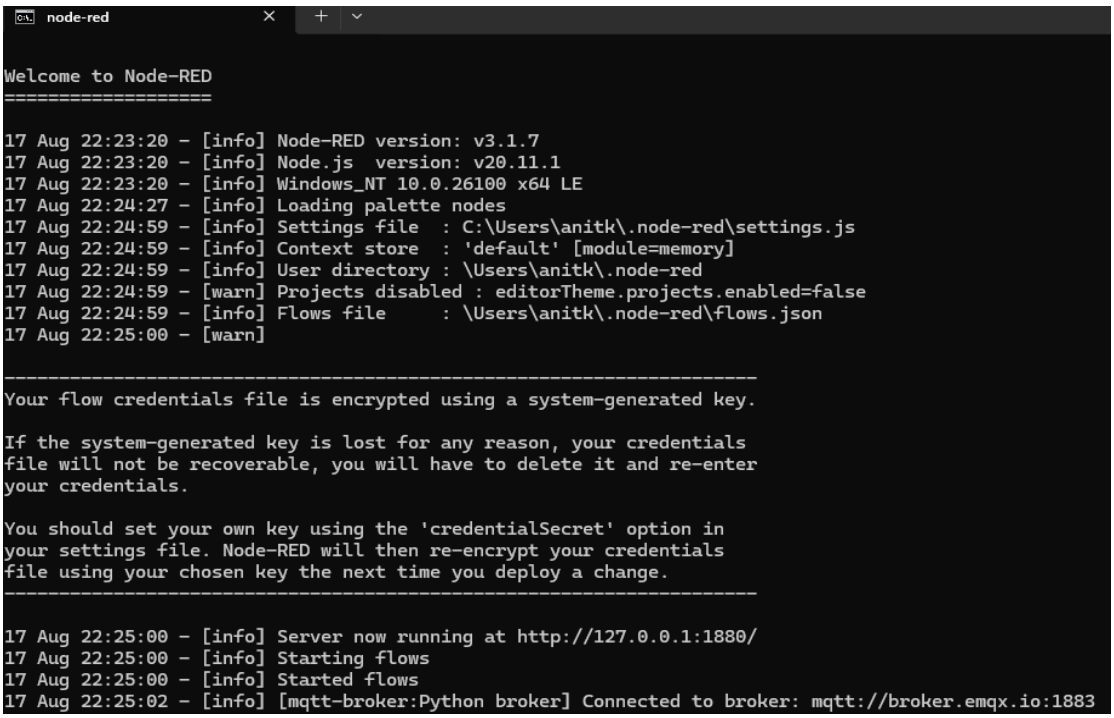



**Figure 7:** Node-RED Server for MQTT Communication

## 1.1 Background

Edge computing greatly assists computation in its dedicated tasks for lightweight devices. Hence, whenever we expect outstanding functions from a device with limited resource capabilities, we must engage the edge server and the IoT device. Until and unless this edge server follows a strict protocol that can prevent security threats, the employment of the edge server is significantly meaningless to its users [38] - [40].

Cyber-threat protection remains one of the most critical and evolving research areas in information technology. The rapid proliferation of interconnected IoT devices, many of which continuously transmit personal data over the Internet, has significantly escalated the cybersecurity landscape. This exponential growth expands the attack surface and intensifies the ongoing conflict between cybersecurity professionals and malicious actors. This protection is paramount in Internet-of-Things (IoT) ecosystems, where numerous IoT-enabled data sources interact with the physical world across diverse application domains. These include agriculture, healthcare, home automation, and critical industrial processes, each demanding robust security measures to safeguard sensitive data, ensure system integrity, and prevent unauthorized access. Unfortunately, modern IoT devices often incorporate minimal security features, making them highly

vulnerable to increasingly sophisticated cyberattacks. This not only hampers the widespread adoption of IoT technologies but also raises significant security concerns, particularly given the millions of already deployed IoT devices that lack dedicated security mechanisms [70] [71] [88]. The tremendous use of IoT devices increases the system's vulnerabilities to cyberattacks due to the immense number of devices and access points outside the traditional administrative domain. This massive interconnection of IoT devices can push personal data to the Internet, which is not permissible in many contexts [89].

IoT devices can be considered as agents that decide whether the edge devices are required to gain computing tasks from an underlying network or not. The pervasive application of IoT devices in today's communication world to provide more business opportunities in line also raises many security challenges. To overcome such security challenges needs to acknowledge the peculiar conditions under which IoT devices fail to operate as expected. These failover conditions can be as follows [25] [26] [39].

1) Low computational capacity
2) Low power capacity in the battery enables the IoT device
3) Unauthorized access of the IoT device by the Edge server
4) Distributed decision-making system in the IoT Network
5) High latency in the Edge server node and the IoT device
6) High response wait time

The IoT devices with heavy data streams are generally under security threats for the following reasons [56] [72] [73] [76] [77] [78].

a) Restricted computation capacity of the underlying processor
b) Limited memory
c) Limited radio bandwidth
d) Inefficient power supply from the battery
e) Lack of computationally intensive and latency-sensitive security jobs

The introduction of edge computing in IoT computation is decentralized and mostly placed near IoT devices. This brings many challenges to edge security and the ultimate target, an IoT device. The following are the Edge Computing Security Risks that are most common so far.

i. Malicious Hardware/Software Injections
ii. Physical Tampering & Attacks
iii. Routing Information Attacks
iv. Distributed Denial of Service (DDoS) Attacks

In IoT Edge computing, one of the significant challenges arises from the differences in communication protocols between the IoT layer and the Cloud layer. Communication between the Edge server and the Cloud layer typically relies on protocols like HTTP(S), SMTP, or FTP, whereas communication between the IoT devices and the Edge server uses protocols such as UDP, CoAP, and MQTT. These disparities create a non-uniform communication landscape, making the network more susceptible to Distributed Denial-of-Service (DDoS) attacks. The lack of a standardized communication protocol across all layers introduces vulnerabilities, as the Edge server often struggles to efficiently interpret and process data transmitted through multiple protocols, especially from a single machine resource. Even when various protocol services are configured to run simultaneously on the Edge server, the server's ability to switch seamlessly between these protocols can become compromised during a DDoS attack. The latency in switching from one protocol to another can create windows of vulnerability, wherein data packets may be intercepted or misrouted, potentially ending up in the hands of an attacker instead of being delivered to the intended Cloud server. This problem is exacerbated when the attack overwhelms the server's resources, causing further delays and opening the door for malicious nodes to exploit the network's weak points [79].

The integration of IoT devices with Edge computing and Cloud infrastructure forms a powerful communication architecture that supports real-time data processing, reduced latency, and enhanced scalability. However, this distributed architecture also introduces significant vulnerabilities, particularly to Distributed Denial-of-Service (DDoS) attacks. A DDoS attack occurs when a large volume of malicious traffic is directed toward a network service or infrastructure, overwhelming its capacity and rendering it unavailable to legitimate users [80].

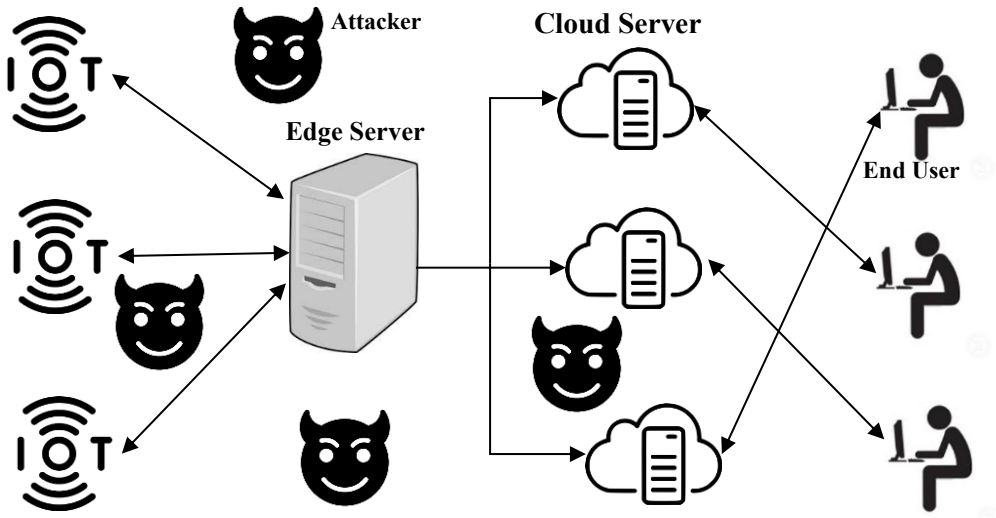



**Figure 8:** DDoS attack on IoT-Edge Cloud Communication Network

**Figure 8** shows how DDoS attacks destroy the IoT-Edge-Cloud communication. Attackers aim to identify vulnerabilities in the underlying IoT network and exploit them to steal sensitive data from an IoT device. The most common process of IoT devices is as follows [81].

1) **Sense**: Intercepts the relevant information and gathers and processes the data.
2) **Transport**: Transmits the data to the outside world through the Internet.
3) **Store**: Stores the gathered data at some temporary or permanent storage for processing.
4) **Analyze**: Provide insights into the system from the data gathered from storage by the IoT device.
5) **Control**: The IoT device uses the commands to operate.
6) **Share**: IoT data can be exchanged with a third-party system.

Reinforcement learning has become a crucial method in machine learning (ML) algorithms that utilize and apply feedback architectures in the fields of computer security and artificial intelligence. It provides sequential or dynamic responses to malware attacks that require a limited or advanced knowledge of the surrounding environment. IoT devices are not a new paradigm for anyone. In our day-to-day lives, we use millions of devices that generate data and are also capable of receiving information through sensors. These IoT data are sent to the Cloud and become an

integral part of the day-to-day life of the users in many ways. Essentially, this volume of informational IoT data is processed in a cloud environment, and, as needed, it is also stored in a cloud database. This helps improve current knowledge, enhances the process, and adds valuable information for organizations [82]. The IoT infrastructural setup enhances the integration between the real communicating world and the outer communication network. Edge Computing has emerged to minimize the costs associated with transferring, processing, and storing data from IoT environments in the Cloud. Data are pre-processed at the edge server and then only sent to the Cloud. It helps to obtain shorter response times and makes the service consistent even in the case of communication breakdowns between the IoT user and the Cloud interface. It needs to make security and privacy crucial for the existence of IoT device communication systems [83].

## 1.2 Motivation

Nowadays, IoT-Edge computations are widely used, and almost every automation involves IoT-Edge computations for better response and to achieve high-end computation. On the other hand, malware attacks on such IoT-Edge computations are increasing day by day, as edge servers carry sensitive data from various IoT devices [84]. The motivation behind writing this research work is also to explore the IoT-Edge network layer data communication and establish a security model to protect IoT-Edge communication. Hence, we worked to design and implement a Novel Reinforcement Learning approach to secure IoT-Edge computation. The main contributions of this research work are as follows.

1) Proposed a Novel Reinforcement Learning-based security model installed on the Edge server for Data Security in IoT-Edge computing.
2) Enhanced the ability of the proposed model for a much more scalable IoT network than similar existing models.
3) The proposed model is much more robust, resilient, and consistent than similar existing models.
4) Examine all the network data breaches with the collected Simulation dataset, Public dataset, and Real-time In-house IoT dataset.

5) Evaluate the performance of our network security layer upon the parameters packet delivery ratio (PDR), Average throughput, and End-to-End (E2E) delay.
6) Determined and compared the accuracy of the proposed IoT-Edge security model with existing similar benchmark security models.
7) Test and validate the proposed security approach with other existing similar approaches.

## 1.3 Benefits of Edge Computing on IoT Devices

The practice and inclusion of Edge computing on IoT devices are of utmost importance nowadays for many reasons [85] [88] [106] [108] - [111]. Some of these are listed below.

1) Modern IoT devices need to run a program that has complex calculations, but have limited processing capacity and memory. Hence, to fulfil this, it requires an Edge device to integrate with itself.
2) It provides faster response times for time-sensitive applications like autonomous vehicles, industrial automation, and healthcare monitoring.
3) It reduces the need for constant high-bandwidth connectivity with the IoT devices, which is especially useful in bandwidth-constrained environments.
4) It enables seamless operation in remote or even disconnected environments.
5) It supports the growing number of IoT devices and high data volume, sometimes without the need for cloud infrastructure.
6) Edge Computing significantly reduces network traffic and communication delays by decentralizing computational tasks, shifting the processing, data storage, and service delivery from centralized cloud infrastructures to localized Edge Servers situated closer to the IoT device.

## 1.4 Benefits of Reinforcement Learning in IoT Data Security

Reinforcement Learning (RL) offers several key benefits for securing IoT data, especially in dynamic and large-scale IoT-Edge-Cloud environments where static security mechanisms are often ineffective [69] - [74].

1) Reinforcement Learning (RL) offers several key benefits for securing IoT data, especially in dynamic and large-scale IoT-Edge-Cloud environments where static security mechanisms are often ineffective [8] – [10].

2) IoT networks are highly dynamic due to device mobility, changing traffic patterns, and evolving attack strategies. RL agents continuously learn from the environment and adapt security policies in real-time, making them effective against evolving malware attacks.
3) Unlike traditional rule-based or supervised ML approaches, RL does not rely solely on labelled datasets. It learns optimal behaviour through interaction, which helps in reducing false positives and false negatives in intrusion detection systems (IDS).
4) When deployed at the edge server, RL allows security decisions to be made closer to IoT devices. This reduces response latency, limits data exposure to the cloud, and enhances protection against DDoS, spoofing, and data injection attacks.
5) As the number of IoT devices increases, manual or static security policies fail to scale. RL supports scalable security orchestration by learning optimal strategies in large networks.
6) RL excels in unknown and stealthy attack detection, as it learns from reward feedback rather than predefined attack signatures.

## 1.5 Security Challenges of IoT Devices in Edge Computing

Manufacturers, organizations, and infrastructures are increasingly adopting the Internet of Things (IoT) to enhance both economic and technological efficiency. This rapid expansion of IoT devices across various domains has necessitated the integration of Edge servers, which support computational offloading and real-time data processing to meet user demands effectively. However, as Edge servers began handling IoT data for preprocessing and analytics, they became attractive targets for cyber attackers aiming to exploit vulnerabilities and steal sensitive information for malicious purposes. This introduces abnormal behaviour in IoT-Edge computing environments, which can compromise the integrity, confidentiality, and availability of IoT systems, ultimately defeating their intended purpose. Securing Edge devices is critical to safeguarding the entire IoT ecosystem. Since Edge devices maintain tight interaction with IoT devices, cloud infrastructure, and other system components, they significantly influence network connectivity, system scalability, and overall performance. A compromised Edge device

can act as a gateway for adversaries to infiltrate the broader IoT network, emphasizing the urgent need for robust security mechanisms in IoT-Edge computing architectures [111] - [118].

The growing inclination toward edge computing stems from the need to process data closer to its source, reducing latency and enhancing efficiency. This paradigm shift is essential because transmitting raw data directly to recipients is often inefficient. Instead, edge computing ensures that only relevant and meaningful information is processed and forwarded, optimizing bandwidth and response times. Edge computing follows a distributed computing model, where data processing occurs on local premises or near the originating IoT devices rather than relying solely on centralized cloud infrastructure. Many manufacturers develop IoT devices that depend on edge servers or personal computers for functionality and real-time data processing. The proliferation of IoT-enabled electronic devices, many of which are embedded with sensors, has accelerated the need for Edge computing, enabling faster decision-making and real-time analytics. The integration of edge computing into critical sectors such as finance, banking, healthcare, and telecommunications has revolutionized data processing by ensuring low latency, reliable, and secure operations. However, this increasing reliance on edge computing has also made Edge servers a prime target for cyber intrusions, as attackers seek to exploit vulnerabilities and steal sensitive information for malicious activities. As a result, robust security mechanisms for edge computing environments are essential to safeguard sensitive IoT data and maintain system integrity [89] [90].

A security breach in edge devices can compromise the entire IoT system through one or more of the following attack scenarios [91] - [93].

1) Attackers exploiting vulnerabilities in edge devices can gain unauthorized access to sensitive infrastructure data, leading to data theft, espionage, or system manipulation.

2) A compromised edge device within an IoT platform can trigger system failures, causing temporary or prolonged disruptions in production lines, critical services, or business operations.

3) Malicious control of edge robots can endanger worker's safety, while hackers infiltrating a production system through compromised edge devices can disrupt manufacturing processes, resulting in financial and operational losses.

4) The expansion of edge devices in an IoT ecosystem increases the risk of IP address exposure, making it easier for attackers to locate and exploit vulnerable devices within the network.

5) Many edge devices have constrained processing power, memory, and storage, limiting their ability to implement strong encryption, intrusion detection systems, and advanced security mechanisms.

6) IoT edge devices often rely on third-party components and firmware. A supply chain attack can introduce vulnerabilities before deployment, allowing attackers to gain a backdoor entry into critical infrastructure.

## 1.6 Types of Machine Learning

Machine learning is broadly categorized into three primary paradigms: Supervised learning, Unsupervised learning, and Reinforcement learning. These approaches define distinct ways in which machines learn from data, ranging from learning with labelled datasets (Supervised learning), discovering hidden patterns in unlabeled data (Unsupervised learning), to learning optimal actions through trial-and-error interactions with an environment (Reinforcement learning). Together, they form the foundation for developing intelligent systems capable of decision-making, prediction, and pattern recognition [119] - [120].

**Table 1**: Different Machine Learning Methods

| Supervised Learning | Unsupervised Learning | Reinforcement Learning |
|---|---|---|
| The model learns from a labelled dataset where each input has a known output. | The model learns from unlabeled data with no explicit output information available. | The model in which an agent makes decisions based on trial and error. |
| **Goal**: Find patterns that relate to those known outcomes. | **Goal**: Discover hidden patterns to identify Clusters. | **Goal**: Learn a policy that maximizes cumulative reward over time. |

| | | |
|---|---|---|
| **Ex**:- Image classification (Cat vs. Dog), Spam email detection, etc. | **Ex**:- Clustering customers by some characteristics, Anomaly data detection, etc. | **Ex**:- Game Playing, Anomaly Data detection, etc. |

### 1.6.1 Existing Machine Learning Based Security Methods

In this thesis, we will compare our proposed security method with some state-of-the-art Machine Learning IoT security methods. So, in this section, we briefly introduce each of these models before discussing our proposed method in the subsequent chapter, which is as follows.

1) **Decision Tree**: A Decision Tree is a popular supervised learning algorithm used for both classification and regression tasks in machine learning. It splits data based on feature conditions to form a tree structure. It is a tree-like structure in which the root node represents the starting point that contains the entire dataset and is split based on the best feature. Each internal node represents a decision based on a feature, each branch represents an outcome of the decision, and each leaf node represents a class label for classification (Classification Decision Tree) or a continuous value for regression (Regression Decision Tree) [94].
2) **Support Vector Machine (SVM)**: SVM is a supervised learning algorithm used for classification, regression, and outlier detection tasks. It is particularly powerful for high-dimensional data and small to medium-sized datasets. It aims to find the best decision boundary, which is also known as the hyperplane, that separates data points of different classes in an n-dimensional space. The hyperplane is chosen such that it maximizes the margin, which is the distance between the nearest data points known as support vectors from each class. Support vectors define the margin and directly impact the classification decision [95].
3) **k-Nearest Neighbors (k-NN)**: k-Nearest Neighbors (k-NN) is a supervised learning algorithm utilized for both classification and regression tasks. It is a non-parametric, instance-based learning method, meaning it does not assume any specific data distribution but instead makes predictions based on stored training instances. The algorithm classifies a data point by analyzing the

majority class of its k-nearest neighbours in the feature space. Due to its lazy learning nature, k-NN does not build an explicit model during training; rather, it stores the dataset and performs computations only at the time of classification or prediction. This characteristic makes k-NN computationally efficient for training but potentially expensive during inference, especially with large datasets [96].

4) **Naive Bayes**: Naive Bayes is a supervised learning algorithm based on Bayes Theorem. It is widely used for classification tasks and is particularly effective for text classification, spam filtering, and sentiment analysis due to its simplicity and efficiency. It utilizes the Naive Bayes classifier, a probabilistic learning algorithm based on Bayes' theorem, which assumes strong independence between features. It has proven to be an efficient and effective classifier for various applications, including text classification, sports event prediction, and medical diagnosis, etc. Naive Bayes is particularly well-suited for high-dimensional text classification due to its computational efficiency and simplicity. The classifier operates on categorical variables and leverages conditional probability to make predictions. Additionally, it requires a few parameters, making it a lightweight and fast algorithm compared to other classification techniques, particularly when dealing with large datasets [97].
5) **Logistic Regression**: Logistic Regression is a supervised learning technique primarily used for classification tasks, especially in binary classification, where data is assigned to one of two categories. Although its name includes "regression," it functions as a classification algorithm rather than a regression model. It estimates the probability that a given input belongs to a specific class, utilizing the sigmoid function to transform predicted values into a range between 0 and 1. This characteristic makes it particularly suitable for classification problems. Despite certain limitations, its simplicity, interpretability, and efficiency make it a strong baseline model for binary and multiclass classification. Additionally, incorporating regularization techniques and feature engineering can enhance its effectiveness, making it applicable to real-world scenarios [98].

6) **Artificial Neural Networks (ANN)**: Artificial Neural Networks are widely utilized in machine learning for tasks such as classification, regression, clustering, and decision-making. Inspired by the structure of the human brain, ANNs consist of interconnected artificial neurons organized into layers. These models process data through an input layer, extract features using hidden layers with activation functions, and generate predictions in the output layer. Weights and biases are adjusted during training through backpropagation and optimization techniques like gradient descent to minimize errors. ANN-based models have diverse applications, including image recognition, natural language processing (NLP), autonomous systems, cybersecurity, and healthcare. They power deep learning architectures such as Convolutional Neural Networks (CNNs) for image processing, aid NLP tasks like speech recognition and sentiment analysis, and enhance cybersecurity by detecting anomalies and fraud. Additionally, they contribute to medical diagnostics and personalised healthcare solutions [99].

## 1.7 Recent Works on IoT-Edge Security Implementation

In [90], Researchers presented an ensemble model for threat hunting in edge devices for Industrial Internet of Things (IIoT) systems. Their proposed model is flexible, allowing the integration of various state-of-the-art classifiers as base learners. It effectively detects and classifies multi-class anomalies by leveraging Multi-class AdaBoost with a majority voting mechanism, ensuring robust and accurate anomaly detection. The detection of multi-class anomalies in the network data packets is the basis of anomaly detection for all error detection methods. The mechanism of detecting malicious payloads is known as Threat Hunting. They utilized this mechanism, and in our proposed security method, we also extended it, incorporating the reinforcement learning technique. In [89], Researchers introduced Passban, an intelligent anomaly-based Intrusion Detection System (IDS) specifically designed for direct deployment and execution on edge devices, such as low-cost IoT gateways. To assess its effectiveness, they developed an IoT testbed that emulates a typical smart home automation environment. Passban was implemented using one-class classification techniques and rigorously evaluated against various malicious activities to measure its

performance in detecting anomalies. In [88], Researchers proposed an unsupervised anomaly detection method for identifying anomalies in smart grids. Their approach leverages a scalable technique that reduces computational overhead through data reduction, thereby enhancing efficiency in large-scale deployments. The proposed method follows a model-free approach, making it adaptable for hierarchical and topological networks across various cyber-attack scenarios in smart grid infrastructures. Our proposed security method aligns with the approach presented in [88], particularly in its ability to scale efficiently by integrating multiple Edge servers. This design reduces the computational overhead on any single-edge server, thereby enhancing the overall performance, load balancing, and resilience of the system. In [91], Researchers proposed an IoT security scheme that integrates machine learning and cryptography within an edge computing framework. This approach enables real-time monitoring and anomaly detection in IoT systems, allowing for automated early-stage threat response. Additionally, they designed a lightweight key management scheme that operates without third-party involvement, enhancing security and reducing dependency on external entities. The proposed method effectively combines active detection and passive protection mechanisms, providing a comprehensive security solution for IoT environments. Similar to [91], we trained our security method to protect IoT-Edge computation by utilizing both simulation-based and real-world datasets. This training enhances its capability to automatically detect and respond to threats at an early stage, ensuring a more resilient and adaptive security mechanism for IoT-Edge environments. In [92], Researchers proposed a general Smart Contracts-based IoT monitoring framework that leverages the security features of public blockchains while minimizing the associated monetary costs. The framework incorporates a reinforcement learning agent that dynamically adapts to user requirements and operates in real-time to optimize the data submission rate of IoT sensors, enhancing both efficiency and security. Smart Contracts (SCs) are general-purpose software applications deployed on a distributed network. They function as digital counterparts of real-world contractual agreements, providing key security-enhancing features such as transparency, immutability, and automated execution of contract terms without intermediaries. Similar to [92], our proposed security model also fulfills the Smart Contracts-based IoT monitoring system requirement to enhance IoT security.

## 1.8 Key Highlights of the Proposed Research Work

The proposed research uses the popular reinforcement learning method to secure the Edge server when it collects data from the IoT devices for further processing. Securing IoT Devices in Edge Computing is an emerging and impactful research topic that integrates cybersecurity, machine learning, and distributed computing. The proposed security model to secure IoT-Edge communication has the following novel contributions.

1) Implemented the novel reinforcement learning method using a message-driven approach to secure the IoT-Edge computation, which we termed "Message Driven-based Reinforcement Learning" (MDRL) security method of IoT-Edge computing. We used the NS-2.35 simulator to validate this approach experimentally.
2) To validate and extend the capabilities of the proposed novel reinforcement learning method to work perfectly under real-time constraints and conditions, we extended the proposed reinforcement learning-based security method by using the "Adaptive Epsilon Greedy Reinforcement Learning" (AEGRL) approach, which is an extension of the traditional Epsilon (Ɛ) greedy reinforcement learning method. The proposed method now works efficiently for both static and dynamic environments.
3) The proposed novel reinforcement learning security method to secure IoT-Edge computation extracts the data packet features to distinguish between malicious and non-malicious ones.
4) The proposed security method is validated and strengthened by using simulation, public, and real-time in-house datasets.
5) The proposed security method is more robust, resilient, consistent, and scalable than similar existing security methods.

## 1.9 Summary

The IoT-Edge server has become a frequent target for intruders seeking to exploit sensitive data for personal, financial, or even geopolitical gain. This threat is particularly critical in high-stakes sectors such as finance, banking, healthcare, and telecommunications, where the compromise of confidential data can lead to severe

economic losses, privacy violations, or disruption of critical services. Unfortunately, many traditional security mechanisms are no longer sufficient in the face of sophisticated and evolving cyberattacks, as modern hackers often leverage advanced tools and strategies to bypass outdated defenses. Moreover, the Edge server, being a critical intermediary setup that bridges IoT devices with the cloud server, is frequently exposed to the network through dedicated IP addresses, making it highly visible and thus a prime target for attackers. This exposure increases its susceptibility to a variety of network-layer attacks, including Distributed Denial of Service (DDoS), spoofing, man-in-the-middle (MITM), and unauthorized access attempts, all of which threaten the availability, integrity, and confidentiality of IoT data. Given the rising dependency on IoT-Edge infrastructures in mission-critical applications, strengthening the security posture of Edge servers has become a prime requirement for ensuring trust and resilience in next-generation IoT ecosystems.

Our research work is primarily motivated by the urgent need to explore and address the vulnerabilities in IoT-Edge network layer communication, where the exposure of devices and servers to dynamic and distributed environments makes them highly susceptible to cyberattacks. With the rapid expansion of IoT ecosystems, traditional static security measures have proven inadequate in countering the scale, complexity, and adaptability of modern threats. To bridge this gap, we propose a novel reinforcement learning (RL)–based security model that is specifically designed to secure IoT-Edge computation. Unlike conventional methods, the proposed RL approach introduces a dynamic and adaptive defense mechanism that continuously learns from environmental interactions, enabling it to evolve alongside emerging attack strategies. The model emphasizes real-time monitoring, anomaly detection, and autonomous decision-making to mitigate network-layer threats such as Distributed Denial-of-Service (DDoS) and unauthorized access attempts. By integrating RL's ability to balance exploration and exploitation, this security framework not only enhances resilience but also ensures reliable, efficient, and secure IoT-Edge communication in highly dynamic environments.

# CHAPTER 2

# LITERATURE REVIEW

Smart city and smart home boom, facilitation of smart transport and smart health care, etc., where IoT devices are an integral part of the user needs, require an edge server because the IoT device or the network of interconnected IoT devices alone cannot perform the necessary computation and high-speed data transmission to the cloud server whenever needed. This edge server is typically a high-performance computing platform that enables IoT devices to offload their computing tasks smoothly, allowing them to focus on their primary intended purpose. Consequently, the number of Internet of Things (IoT) devices connected to the Internet has surged significantly, along with the volume of data generated by these devices. This growth necessitates Edge computing to relieve the burden of intensive computation and storage. It gives rise to the extensive use of IoT-Edge computation worldwide. By this time, it has been observed that a secure layer over an IoT-Edge network would efficiently provide security to IoT-Edge computation. This secure interface obviously must use some machine learning algorithm that can intelligently defend from malware attacks. Since IoT devices are physical objects interconnected through the Internet and have sensors to collect data from the surrounding environment, or installed software to generate data. It makes them always a target vector for intruders who steal sensitive data for personal benefit. IoT device manufacturers focus much on the quantity of production and profit margin, hence giving less concern about strengthening the security layer [15] [17] [22] [23] [25] [26] [56] [62] [63].

**Figure 9** represents how an attacker can take advantage of the Internet and make malicious attacks on the Edge server. The lack of computing efficiency of IoT devices also endangers their use and invites hackers to either cause it or steal some sensitive data from their private storage. Today, almost all organizations that use a communication network need a Network Intrusion Detection System (NIDS) to maintain the security and safety of their communication system.

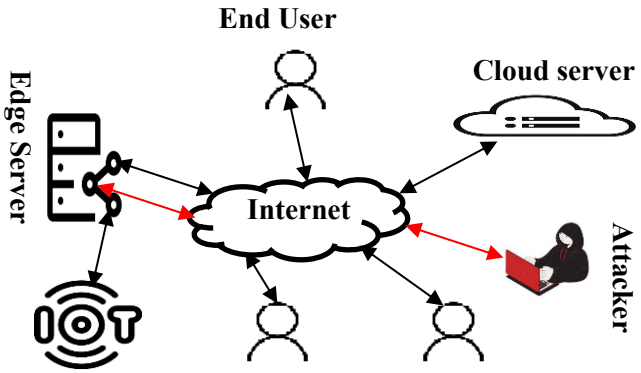



**Figure 9:** Malicious Attack on IoT-Edge Communication

IoT devices have limited memory, even though their physical layer must perform the following tasks [84] [85].

1) Detect Channel Impulse Response
2) Detect Signal Strength
3) Examine Channel State Information
4) Maintain Communication Protocol
5) Send/Receive Data

Due to the above essential tasks of IoT devices, IoT devices are left with a minimal amount of computing resources that can be utilized for security measures for the device. IoT devices generally have limited memory, computation resources, and battery power, which invites intermediate attacks in the underlying network. It subsequently adds serious security threats to the device, allowing thieves to steal sensitive information from the device. Attackers are also enriched with more advanced and intelligent tools, which is also a serious concern for the IoT manufacturers. With the integration of Edge computation with IoT devices, the limitation of resource constraints on IoT devices has been largely solved. Some of the current challenges where still more and more researches are needed are as follows [127] - [130].

1) Security challenges

2) Computation challenges

3) Software challenges

4) Communication challenges

5) Data storage challenges

Deploying the edge server (s) on the edge of the IoT network solves computation and data storage challenges. Software upgrades on computing devices have nowadays become an automated process where updated software is automatically downloaded and installed into the devices, especially at the time of the restart. Hence, Software challenges nowadays are no big concerns for IoT manufacturers and their users [133] [134].

Among the above challenges mentioned, our proposed research work is mainly aimed at the Security challenges. From our personal views, security challenges are the top concern for IoT manufacturers and their users. Any gaps in addressing other challenges can exacerbate security challenges, and this is an implicit concern when performing hardware and software upgrades or functional changes to IoT devices.

## 2.1 Related Works on IoT Security

Bikila et al. (2025) [118] presented a machine learning-based framework for detecting cyberattacks in Internet of Things (IoT) environments, aiming to address the rising threat of security breaches due to the heterogeneous and resource-constrained nature of IoT devices. They proposed a lightweight and efficient attack detection model that leverages supervised learning algorithms such as Random Forest and Support Vector Machines, trained on benchmark datasets to identify anomalies in network traffic. The framework is designed to optimize detection accuracy while maintaining minimal computational overhead, making it suitable for deployment on edge or fog nodes in real-world IoT settings. Experimental results demonstrate that the proposed approach achieves high detection rates and low false positive rates, underscoring its potential to enhance IoT security through intelligent, real-time threat monitoring.

Malik et al. (2025) [117] introduced a novel hybrid threat intelligence framework designed to secure Industrial IoT (IIoT) environments. They presented a robust and practical deep learning-based solution for threat intelligence in industrial IoT, merging high detection precision with efficient runtime characteristics, thereby paving the way for deployment in resource-constrained, mission-critical IIoT environments. The

system is built to be highly scalable and self-optimizing, leveraging advanced deep learning models to detect and mitigate a wide spectrum of sophisticated IoT malware and persistent cyber threats. They benchmarked their hybrid architecture against existing deep learning models and standard IDS algorithms, using both conventional and extended performance metrics. Their framework consistently delivered superior detection accuracy while maintaining minimal latency overhead, thus ensuring speed-efficient deployment in real-time IIoT systems.

Ren et al. (2024) [128] design a multi-agent deep reinforcement learning (MADRL) system in which distributed IoT agents learn defensive policies autonomously, rather than relying solely on human rules or a central controller. They implemented an attack defense model as a finite random game and defined agent observations, actions, and reward shaping so that agents can adapt their policies to evolving attacks. The work emphasizes decentralized decision-making, where the agents operate locally, which is a key architectural contribution for scalable and heterogeneous IoT deployments where central control is impractical. They employed a decentralized attack defense simulation environment built with the CyberBattleSim framework and introduced an adversarial learning strategy to train both attacker and defender agents, allowing the defense agent to learn adaptive and dynamic decision-making behaviors.

Geetha et al. (2024) [130] introduced an intrusion detection system (IDS) for Internet of Things (IoT) networks called the Adaptive Weighted Kernel Support Vector Machine with Circle Search (AWSVM-CS). This framework integrates an adaptive weighted kernel-based SVM classifier with a Circle Search Algorithm (CSA) to improve the accuracy and efficiency of anomaly detection. The adaptive kernel allows the model to effectively manage heterogeneous and noisy IoT data, while the CSA optimizes the feature selection and extraction process. The proposed IDS operates through several stages, including data analysis, preprocessing, feature selection, and attack prediction. In the preprocessing phase, raw network data are cleaned and transformed into a more suitable form for analysis. Feature selection then removes redundant or irrelevant attributes, generating a refined subset of useful features. Using a set of defined extraction rules, the system differentiates between normal and

anomalous traffic behavior. Ultimately, the AWSVM-CS classifier accurately identifies and classifies network traffic as either legitimate or malicious.

Oh et al. (2024) [115] made a significant contribution to the field of cybersecurity by meticulously adapting Deep Reinforcement Learning (DRL) algorithms to meet the complex demands of modern threat landscapes. Their approach integrated customized reward structures, adversarial training, and responsiveness to dynamic cyber environments, resulting in a robust framework that substantially improves upon traditional reinforcement learning methods. They presented a comprehensive comparative analysis of three advanced Deep Reinforcement Learning (DRL) algorithms, Deep Q-Network (DQN), Actor-Critic, and Proximal Policy Optimization (PPO), against the classical Q-learning algorithm. They evaluated the experimental results within a controlled simulation environment designed to reflect real-world cybersecurity threats. They insisted that their findings were noteworthy and the proposed Actor-Critic algorithm outperformed all counterparts by achieving a success rate of 0.78, requiring the fewest iterations (171) to complete an episode, and attaining the highest average reward of 4.8. They simulate cyber threats effectively and also implemented the proposed model, scalable and adaptable for a range of cybersecurity applications.

Arif et al. (2024) [120] introduced DQQS, a deep reinforcement learning (DRL)-driven framework designed to optimize routing decisions in Software-Defined Networking (SDN)-enabled Internet of Things (IoT) environments. Recognizing the need to address the trade-offs between Quality of Service (QoS), Quality of Experience (QoE), and network security, DQQS formulates adaptive routing policies by continuously learning from historical network behaviour. Unlike static or rule-based methods, DQQS operates in a model-free reinforcement learning setup, where the SDN controller acts as the learning agent, interacting with a dynamic IoT environment to make intelligent routing choices. Their model leverages a Q-learning-based architecture, where the agent receives a reward based on throughput maximization, latency minimization, and the successful avoidance of compromised or malicious nodes. DQQS integrates environmental state variables such as packet loss rate, bandwidth availability, link delay, and node trust scores into its state-action formulation. The policy updates occur

through a deep Q-network (DQN), enabling the agent to generalize across large state spaces typically encountered in scalable SDN-IoT deployments.

Louai A. Maghrabi (2024) [129] addresses the crucial security challenges of Internet of Things (IoT) networks, which are the high volume of heterogeneous traffic, constrained device resources, and the difficulty of detecting rare attack instances, by proposing an automated network intrusion detection (NID) system tailored for IoT environments. His approach employs a Random Forest classifier applied to the UNSW-NB15 dataset, augmented by resampling techniques to mitigate class imbalance, pre-processing, and feature selection to improve detection effectiveness. Experimental results show that the model achieves an accuracy of 90.17% and, under a balanced resampled version of the dataset, achieves 98.83% of accuracy.

Yazdinejad et al. (2023) [90] proposed a parallel ensemble-based threat hunting model designed to detect anomalies in the behaviour of Industrial Internet of Things (IIoT) edge devices. Their model exhibits a high degree of flexibility by enabling the integration of multiple state-of-the-art classifiers as base learners. They employed a combination of Multi-class AdaBoost and majority voting mechanisms to perform multi-class anomaly classification effectively. One of the distinguishing features of their approach was its ability to handle diverse anomaly types from heterogeneous data sources. They conducted comprehensive experimental evaluations using a dataset composed of multi-source normal behaviour records and multiple types of anomalies. Their experimental results demonstrate that the ensemble framework outperforms existing benchmark models across key performance metrics, including Accuracy, F1-score, Recall, and Precision, indicating its robustness and efficiency in real-world IIoT threat detection scenarios.

Li et al. (2023) [109] address the challenge of conservative exploration in reinforcement learning (RL), aiming to ensure that an agent's performance consistently exceeds a predefined threshold throughout the learning process. They introduce an algorithm named StepMix, which operates within the framework of tabular episodic Markov Decision Processes (MDPs) characterized by finite states and actions. StepMix leverages a known safe baseline policy and dynamically interpolates between this

baseline and an optimistic policy. This adaptive mixture policy ensures that the conservative performance constraint is upheld in each episode with high probability. Theoretical analyses demonstrate that StepMix achieves near-optimal regret bounds comparable to those in unconstrained settings, indicating that maintaining conservative constraints does not compromise learning efficiency. Additionally, the authors propose a randomized variant called EpsMix, which attains similar performance guarantees. Their work extends to scenarios where the baseline policy is not predefined but learned from offline datasets, showing that with sufficiently large datasets, the conservative guarantees and regret bounds remain intact. Empirical evaluations corroborate the theoretical findings, highlighting the effectiveness of these conservative exploration strategies in RL.

T. Zhang et al. (2023) [14] focused on designing an energy-efficient and secure communication system for IoT devices. They employed Deep Reinforcement Learning (DRL) to propose a cooperative jamming scheme that ensures secure communication within a cooperative network, protecting it from eavesdroppers. By leveraging the proximity of end devices, their approach enables low-latency communication, offering real-time control and support for IoT devices. To further enhance system performance, an edge server was utilized to process tasks quickly, significantly reducing service delays. Their proposed approach also demonstrated high robustness when handling uncertain or dynamic conditions in the network. The novelty of their algorithm lies in using the edge computing device not only for computational purposes but also as a learning agent. The edge server continually updates its model policy based on the rewards obtained from interactions, investigating the environment. This learning agent reconstructs and optimizes strategies to tackle interconnected subproblems, ultimately forming a comprehensive global model for efficient decision-making and security in IoT networks.

Mishra et al. (2022) [40] worked on providing security and privacy in a Mobile-Edge based distributive environment. Their proposed model maintains data security by using a consensus approach of blockchain and machine learning techniques, which are based on several classifiers and optimization techniques. They worked to mitigate the security risk raised by vulnerable data in the Mobile-Edge-based distributed computing

environment. Hence, they proposed a combination of edge computing and a consensus approach based on blockchain, which utilizes machine learning methods that have been shown to enhance data security and reduce the risk of data breaches.

Sudheera et al. (2021) [135] proposed ADEPT, a distributed framework aimed at detecting and recognizing individual stages of coordinated multi-stage attacks in IoT networks by correlating anomalous behaviors across temporal and spatial dimensions. The framework is structured on a hierarchical Edge gateway management architecture, where gateways conduct local anomaly detection on device-level traffic. A central security manager then aggregates these gateway alerts to identify cross-time and cross-space attack patterns. Leveraging both alert-level and pattern-level features, the authors applied a machine learning model to classify and detect distinct stages of the identified attacks.

Ullah et al. (2021) [136] developed a hybrid deep learning model for anomaly detection in IoT networks, combining convolutional neural networks (CNNs) and long short-term memory (LSTM) layers to capture spatial–temporal traffic features. Evaluated on BoT-IoT, IoT Network Intrusion, MQTT-IoT-IDS2020, and IoT-23 intrusion detection datasets, the model effectively detects attacks such as DoS/DDoS, reconnaissance, and data exfiltration, achieving over 99% accuracy with a low false positive rate. Through rigorous preprocessing and feature optimization, the approach outperforms conventional machine learning classifiers, demonstrating that deep neural architectures can provide scalable, real-time intrusion detection in heterogeneous IoT environments.

Zhang et al. (2020) [38] proposed a Virtual Security Network (VSN) model specifically designed for edge computing environments, taking into account the heterogeneous network characteristics of edge servers. In this model, the edge network is logically divided into multiple hierarchical layers, where each layer is assigned a distinct security function such as authentication, anomaly detection, access control, and policy enforcement. This layered architecture allows for specialized threat detection mechanisms to operate at the most appropriate level, ensuring both security granularity and responsiveness. By introducing intelligent virtualization techniques, the model enables dynamic resource allocation based on real-time security risk assessment and

workload distribution. This, in turn, enhances the resilience of edge servers against Distributed Denial-of-Service (DDoS) and spoofing attacks, while maintaining low-latency performance for legitimate services. Experimental results in their study demonstrated that the VSN model significantly improved resource allocation efficiency and reduced system vulnerability under various simulated cyberattack scenarios.

Li et al. (2019) [25] proposed a secure multi-user, multi-task computation offloading model for Mobile Edge Computing (MEC) in mobile IoT devices. Their approach enhanced the traditional Advanced Encryption Standard (AES) cryptographic scheme by integrating it with a genetic algorithm, creating a fortified security layer. This security enhancement aimed to protect private information, such as personal photos, medical records, shopping details, and payment data, as it is offloaded to the MEC server via wireless channels. By combining AES with the genetic algorithm, they ensured the secure transmission of sensitive data while improving computational efficiency in IoT environments.

Khoda et al. (2019) [17] developed a machine learning-based adversarial defense model to enhance the robustness of classifiers against adversarial attacks, which pose significant threats to machine learning systems in security-sensitive environments such as IoT. Their model incorporates two novel techniques for adversarial sample detection and classifier retraining. The first technique focuses on sample selection based on distance from the malware cluster centroid. By identifying data points located near the decision boundary or at atypical positions within the feature space, this approach allows the system to pinpoint potentially adversarial samples. These are then used to retrain the classifier, improving its decision boundary and resilience. The second technique utilizes a probability measure derived from kernel-based learning (KBL). This statistical method estimates the likelihood of a given input being adversarial by modelling the data distribution. If a sample exhibits a low probability of fitting within the known benign/malicious distributions, it is flagged for adversarial behaviour. By combining these approaches, geometric distance-based selection and probabilistic inference through KBL, the proposed framework significantly enhances the classifier's robustness, improving its detection accuracy and resilience under adversarial conditions.

Arfaoui et al. (2019) [24] proposed a context-aware and lightweight anonymous authentication and key agreement scheme tailored specifically for Wireless Body Area Network (WBAN) applications in next-generation ubiquitous healthcare systems. Their scheme addresses the stringent requirements of WBANs, such as low power consumption, lightweight computation, high mobility, and privacy preservation, which are critical for wearable medical devices. The proposed model introduces a selective anonymous authentication mechanism that allows WBAN nodes to authenticate one another based on dynamic context changes (e.g., physiological data, device movement, or environment). This dynamic context-awareness ensures that the authentication process adapts to real-time health monitoring scenarios, improving system responsiveness and security robustness. Additionally, the scheme includes a lightweight key agreement protocol that minimizes computational overhead, making it suitable for resource-constrained medical sensors. Security analysis and formal proofs show that the protocol resists common attacks such as replay, impersonation, and man-in-the-middle attacks, while preserving user anonymity and data confidentiality.

Khatun et al. (2019) [35] proposed a deep learning-based detection mechanism using Artificial Neural Networks (ANN) to identify infected nodes within IoT networks, with a primary focus on mitigating Distributed Denial-of-Service (DDoS) attacks. Their approach leverages the ability of ANN to learn complex, nonlinear patterns in traffic data, enabling the model to distinguish between normal and anomalous communication behaviours in real-time. By training the neural network on labelled datasets that include both benign and malicious traffic, their method effectively classifies IoT device behaviour into safe or potentially compromised categories. The model achieves high accuracy in identifying attack patterns, including traffic spikes, irregular packet intervals, and asymmetric flow patterns, which are indicative of DDoS activities. The system is designed to operate autonomously, making it suitable for deployment in resource-constrained IoT environments, where traditional signature-based intrusion detection systems (IDS) are less effective.

Chaudhary et al. (2019) [36] developed a Support Vector Machine (SVM)-based anomaly detection model aimed at identifying routing attacks and abnormal behaviour in IoT devices. To enhance the performance of the model, they applied feature scaling

techniques to both the training and evaluation datasets, ensuring that all input parameters contributed equally to the classification decision. The SVM algorithm constructs an optimal classification hyperplane that separates benign and malicious data points with maximum margin. Once trained, the model classifies real-time sensor data inputs based on their proximity to this hyperplane. Their approach demonstrated high accuracy in detecting routing attacks such as sinkhole, blackhole, and selective forwarding attacks, making it effective for deployment in resource-constrained IoT environments.

Li et al. (2018) [20] proposed an anonymous mutual authentication scheme tailored for three-tier mobile healthcare (mHealth) systems, addressing the security and privacy concerns of user data in mobile medical environments. Their scheme comprises three specialized protocols. The first protocol, Protocol-1, facilitates mutual anonymous authentication among mobile users, controller nodes, and the medical server. In the second protocol, Protocol-2, specifically handles authentication between mobile users and controller nodes, ensuring user privacy in initial data handoff. Finally, the third protocol, Protocol-3, supports authentication between controller nodes and wearable body sensors, considering the extreme resource limitations of body sensor networks. The protocols are designed with resource-awareness, ensuring low computational and communication overhead for wearable sensors while maintaining high security standards. Evaluation results showed that the proposed scheme strikes a balanced trade-off between security, computational efficiency, and real-world feasibility, making it highly applicable for ubiquitous healthcare infrastructures.

Hsu et al. (2018) [22] proposed a reconfigurable security framework that leverages edge computing to meet the evolving security demands of IoT environments. Their framework integrates a near-user edge server to offload complex security operations from resource-constrained IoT devices. At the core of the system is a dedicated security agent, deployed at the edge, which enforces robust security mechanisms across all layers of IoT communication protocols. They addressed several pressing challenges in IoT security, like High computational overhead of advanced cryptographic schemes, Limited flexibility in key management across heterogeneous devices, and Compatibility issues with diverse IoT hardware and software ecosystems. By utilizing

edge resources for security processing and reconfigurable cryptographic primitives, their model enhances security, scalability, and efficiency for various IoT applications, including those with strict latency and energy constraints.

Yuan et al. (2018) [69] focused on enhancing the adoption and trustworthiness of IoT-Edge computing architectures by proposing a lightweight and reliable trust evaluation mechanism. Their approach addresses the fundamental issue of trust establishment between heterogeneous IoT devices and Edge servers. To this end, they introduced a multi-source feedback information fusion technique, which aggregates trust evaluations from multiple sources to compute a global trust score. The trust model proposed ensures that Malicious or unreliable nodes can be effectively identified and isolated. The model ensures that Trust values are dynamically updated based on historical interactions and feedback, and the system remains lightweight enough to be deployed on resource-constrained IoT edge devices. This model offers a scalable solution to enhance reliability and user confidence in IoT-Edge environments while minimizing computational overhead.

Yan et al. (2016) [113] worked to evaluate the vulnerability of transmission grids under sequential topology attacks. They introduced a Q-learning-based approach aimed at identifying critical attack sequences by considering the physical behaviours of the power system. The method leverages a realistic power flow cascading outage model to simulate the system's dynamic response to attack-induced disruptions. Within this framework, the attacker acts as a reinforcement learning agent using the Q-learning algorithm to iteratively learn optimal attack strategies that maximize damage while minimizing the number of attacks required to induce system failure. Their proposed method enables a data-driven exploration of attack sequences that may not be easily anticipated through traditional rule-based or static analytical approaches. They implemented a reinforcement learning agent that continuously updates its policy based on feedback from the environment, enabling adaptive learning and improved decision-making under dynamic conditions. They conducted case studies on three IEEE benchmark power systems to demonstrate the learning capabilities of the Q-learning agent and the effectiveness of their approach in identifying critical system vulnerabilities. Results confirm that the Q-learning model can uncover previously

unknown weak points in the grid's topology and optimize the efficiency of attack execution, making it a powerful tool for both vulnerability assessment and the design of resilient power infrastructures.

Xiao et al. (2016) [19] introduced a novel physical-layer (PHY-layer) authentication framework to combat spoofing attacks in wireless sensor networks (WSNs). Their approach leverages radio channel characteristics as a means of identity verification, under the premise that wireless channel responses are location-specific and difficult to replicate precisely by adversaries. At the core of their method lies a game-theoretic formulation called the zero-sum authentication game, which models the strategic interaction between a legitimate receiver (defender) and spoofers (attackers). In this setup, the receiver aims to maximize its utility by selecting an optimal threshold in a Bayesian hypothesis testing framework, balancing the trade-off between false alarms and missed detections. Conversely, the spoofers strive to minimize the utility of the receiver by strategically adjusting their attack frequencies, thereby evading detection. To dynamically adapt to the time-varying and unpredictable nature of wireless environments, the researchers implemented reinforcement learning (RL) techniques, specifically Q-learning and Dyna-Q methods. Q-learning is used to estimate the expected utility over different radio conditions without relying on a predefined model of the environment, whereas Dyna-Q is an advanced RL approach combining model-free learning with simulated experiences, accelerating convergence to the optimal detection strategy. By continuously learning from the channel state information (CSI) and attacker behaviour, the receiver autonomously refines its detection policy and dynamically generates the optimal decision threshold for identifying spoofing attempts. The proposed method demonstrates high adaptability and robustness, making it particularly suitable for resource-constrained IoT and WSN scenarios.

Malialis et al. (2015) [16] proposed a Coordinated Team Learning (CTL) framework as an innovative method to counter Distributed Denial-of-Service (DDoS) attacks through a Multiagent Router Throttling mechanism. The key advancement of this approach lies in its decentralized coordination, which allows multiple reinforcement learning agents to collaboratively detect and mitigate DDoS attacks across large-scale network infrastructures. By enabling local agents to share learned behaviours and

cooperate on throttling malicious traffic, the system adapts dynamically without relying on a centralized controller. Their research focused particularly on the scalability of DDoS defense mechanisms, a well-known bottleneck in existing centralized solutions. Extensive experiments showed that CTL can operate effectively with up to 100 distributed learning agents, sustaining system performance and demonstrating high resilience against large-scale DDoS scenarios.

Michel Tokic (2010) [108] introduced "Value-Difference Based Exploration" (VDBE), a technique designed to address the exploration-exploitation dilemma commonly encountered in reinforcement learning. VDBE dynamically adjusts the exploration parameter of the Ɛ-greedy policy based on the temporal-difference error, which serves as an indicator of the agent's uncertainty about the environment. Their method evaluates using a multi-armed bandit problem, offering insights into its operational behaviour. Their experimental results suggest that VDBE exhibits greater parameter robustness compared to traditional heuristic methods like Ɛ-greedy or softmax.

**Table 2** in below summarizes the literature review done by us throughout this research work.

**Table 2:** Literature Review on Prevention of Security Attacks

| **Author (s), Journal, Year, Research Title** | **Type of Security Attacks Encountered** | **(+) Contributions**<br>**(-) Limitations** |
|---|---|---|
| Bikila et al. [118],<br>Future Generation Computer Systems, Elsevier<br>2025,<br>Machine Learning-Based Attack Detection for the Internet of Things | DoS/DDoS,<br>Spoofing Attack | (+) They proposed an optimized machine learning model that combines unsupervised feature extraction and supervised classification, specifically designed for IoT intrusion detection, addressing the high dimensionality and noise in IoT datasets. |

| | | |
|---|---|---|
| | | (-) Their work uses recent datasets, but they do not cover all possible real-world IoT environments, device types, or emerging zero-day attacks. This could limit the model's effectiveness in unseen scenarios. |
| Malik et al. [117],<br>Journal of Industrial Information Integration, Elsevier<br>2025,<br>Hybrid deep learning based threat intelligence framework for industrial IoT systems | DoS/DDoS | (+) They proposed a framework that pairs a Convolutional Neural Network (CNN) for spatial feature extraction with a Long Short-Term Memory (LSTM) network for temporal modelling, enabling robust identification of dynamic IIoT threats.<br>(-) The proposed framework operates on large volumes of telemetry data, raising concerns over data privacy and potential misuse. No built-in mechanisms, such as federated learning or encryption, were integrated to address these issues. |
| Arif et al. [120],<br>IEEE Access<br>2024,<br>DQQS: Deep Reinforcement Learning-Based Technique for Enhancing Security and Performance in SDN-IoT Environments | DoS/DDoS | (+) The proposed model introduces a customised Deep Q-Learning agent integrated within the SDN controller that dynamically learns optimal routing policies by observing network state, QoS, and security metrics, improving end-to-end performance in IoT environments. |

| | | |
|---|---|---|
| | | (-) The evaluation is limited to synthetic network environments. Real-world deployment with actual IoT devices, heterogeneity, and unpredictability remains untested, which could impact generalizability. |
| Geetha et al. [130],<br>Signal, Image and Video Processing, Springer<br>2024,<br>Adaptive weighted kernel support vector machine-based circle search approach for intrusion detection in IoT environments | DoS/DDoS | (+) They proposed an Adaptive Weighted Kernel Support Vector Machine with Circle Search (AWSVM-CS) that combines a weighted kernel-based SVM with a Circle Search Algorithm (CSA) to enhance detection performance in heterogeneous IoT environments.<br>(-) The approach was primarily tested offline by using public datasets, and its real-time deployment and performance under continuous streaming data were not explored. |
| Louai A. Maghrabi [129],<br>IEEE Access,<br>2024,<br>Automated network intrusion detection for internet of things: Security enhancements | DoS/DDoS | (+) He proposed an intrusion detection system for IoT networks addressing the challenges posed by performance constraints and class imbalance in datasets, which hinder effective attack detection. They introduced an end-to-end pipeline comprising data preprocessing, feature selection, class imbalance mitigation, and malware detection using a |

| | | |
|---|---|---|
| | | Random Forest classifier. Finally, the proposed NIDS framework was evaluated to assess its detection accuracy and overall performance.<br>(-) The use of synthetic balancing techniques, such as random resampling, can lead to inflated accuracy results, as it does not represent the inherently imbalanced distribution of attack types typically observed in real IoT networks. Consequently, the performance obtained on balanced test sets may appear overly optimistic compared to practical deployment scenarios. |
| Ren et al. [128],<br>IEEE Internet of Things,<br>2024,<br>A Multiagent Deep Reinforcement Learning Autonomous Security Management Approach for Internet of Things | DoS/DDoS | (+) They proposed an autonomous defense agent by leveraging reinforcement learning methodologies, which is capable of dynamically formulating and refining defense strategies in response to evolving threats. To mitigate environmental instability resulting from the decoupling of observation and action spaces among multiple concurrently operating offensive and defensive agents, they adopted a MINIMAX Q-learning framework integrated with a |

| | | |
|---|---|---|
| | | synchronized interactive training mechanism.<br>(-) The experiments were conducted within simulated environments, without large-scale real-world implementation or simulator-to-reality transfer. This limitation reduces the external validity of the results when applied to actual IoT network scenarios. |
| Oh et al. [115],<br>Electronics, MDPI<br>2024,<br>Employing deep reinforcement learning to Cyber-Attack simulation for enhancing cybersecurity | Credential Access,<br>Command and Control (C2) | (+) They introduced an advanced DRL system that simulates realistic red-team attack sequences by integrating some real-world attack scenarios and customizing reward structures, actions, and adversarial training dynamics for implementing cybersecurity use cases.<br>(-) The entire framework is validated in synthetic environments, lacking real-world or production-scale deployment. This may create a gap that may undermine generalizability and effectiveness in operational networks. |
| Yazdinejad et al. [90],<br>Digital Communications and Networks, Elsevier<br>2023, | DoS/DDoS | (+) They introduced a parallel ensemble machine-learning approach for threat hunting, combining multiple classifiers and deep models to improve |

| | | |
|---|---|---|
| Accurate threat hunting in industrial internet of things edge devices | | detection accuracy for malicious and abnormal behaviours in industrial IoT Edge environments.<br>(-) Their evaluation relies on curated or synthetic datasets and may not fully reflect the diversity of real industrial traffic, device heterogeneity, encryption, or adversary behavior in real-time deployment of IIoT devices. |
| Zhang et al. [14],<br>IEEE Internet of Things Journal,<br>2023,<br>Deep Reinforcement Learning Based IRS for Cooperative Jamming Networks under Edge Computing | Eavesdropping Attack | (+) They proposed a deep reinforcement learning based framework to jointly optimize transmit beamforming, jamming beamforming, and IRS phase-shifts in an intelligent reflecting surface-aided cooperative jamming network to maximize secrecy rate or energy efficiency under secrecy constraints.<br>(-) The simulations are conducted under specific channel models and system parameters. They did not explore the robustness of the learned policy under model mismatch, such as varying eavesdropper behavior, mobility, or dynamic channel conditions. |
| Huang et al. [27],<br>Annual reviews in control, Elsevier | ➢ Reward Manipulation | (+) Their work introduces the concept of a Cyber-Resilient Mechanism (CRM), an adaptive defense architecture that uses |

| | | |
|---|---|---|
| 2022,<br>Reinforcement learning for feedback-enabled cyber resilience | ➢ Observation Spoofing<br>➢ Action Command Hijacking | feedback loops (sensing, reasoning, and actuation) to respond to cyberattacks in real-time. Reinforcement Learning was positioned as the ideal tool for designing CRMs, as it adapts without prior knowledge of attacker dynamics or system models.<br>(-) Challenges remain unaddressed in scaling CRM frameworks to large-scale, heterogeneous cyber systems and ensuring generalization across diverse attack scenarios. |
| Mishra et al. [40],<br>Cluster Computing, Springer<br>2022,<br>Security provisions in smart edge computing devices using blockchain and machine learning algorithms: a novel approach | DoS/DDoS | (+) They proposed a security approach that combines blockchain, federated learning, machine learning classifiers, and differential privacy to secure edge computing on distributed environments. This hybrid setup aims to protect data at edge devices, during transit, and during training.<br>(-) Their experiments are based on simulating edge-client and server interactions. Their proposed model does not appear to deploy on actual edge devices. This limits insights into real-world latency, energy, communication overhead, etc. |

| | | |
|---|---|---|
| Sudheera et al. [135],<br>IEEE Internet of Things Journal,<br>2021,<br>ADEPT: Detection and identification of correlated attack stages in IoT networks | DoS/DDoS | (+) They introduced a distributed framework, ADEPT, in which lightweight anomaly detectors operate at edge gateways while a central security manager aggregates and correlates alerts across gateways, aligning with IoT deployment constraints. By correlating anomalous events both spatially across distributed gateways and temporally across sequential alerts, the framework effectively uncovers coordinated multi-stage attacks that conventional single-point detection methods fail to identify.<br>(-) The evaluation of the proposed method relies on limited datasets or synthetic traffic, which does not comprehensively represent the diversity of real-world device behaviors, encrypted communications, or evolving attack strategies, thereby constraining its external validity. |
| Ullah et al. [136],<br>IEEE Access,<br>2021,<br>Design and development of a deep learning-based model for anomaly detection in IoT networks | DoS/DDoS | (+) They proposed a convolutional neural network (CNN) model for both binary and multiclass attacks tailored to IoT network traffic. It uses a transfer-learning approach to create models that can be adapted to both binary and multiclass tasks, |

| | | |
|---|---|---|
| | | improving training efficiency and the reuse of learned features.<br>(-) They provided a little insight into why certain network flows were labelled as anomalous and did not provide interpretable outputs or feature-level insights, which are important for helping analysts understand, verify, and trust the system's decisions. |
| Qiao et al. [26],<br>IEEE Transactions on Intelligent Transportation Systems,<br>2020,<br>Trustworthy Edge Storage Orchestration in Intelligent Transportation Systems Using Reinforcement Learning | ➢ Identity Spoofing<br>➢ Data Integrity Attacks | (+) They proposed an intelligent, dynamic edge–cloud storage orchestration framework designed specifically for ITS. The architecture addresses two major challenges:<br>1) Rising data volumes generated by ITS devices, which overload traditional cloud-only storage.<br>2) A lack of clear trust between ITS devices and edge servers poses security risks in data storage and retrieval.<br>(-) The proposed model is tested only in simulated settings. There are no real-world deployments or empirical benchmarks in live ITS environments. Real-world constraints such as unpredictable connectivity, device |

| | | |
|---|---|---|
| | | heterogeneity, and mobility impacts are not evaluated. |
| Qu et al. [67],<br>IEEE Transactions on Cognitive and Developmental Systems,<br>2020,<br>Minimalistic Attacks: How Little it Takes to Fool Deep Reinforcement Learning Policies | Adversarial Attack | (+) The researchers sought to demonstrate that tiny, sparse perturbations that modify as little as 0.01% of the input pixels or attack just 1% of frames can significantly degrade the performance of DRL agents, even those trained with state-of-the-art algorithms on Atari games.<br>(-) Limited to simulation in Atari games; lacks real-world domain exploration, defense proposals, or richer attack modeling. |
| J. Park et al. [70],<br>IEEE Open Access Journal,<br>2020,<br>Privacy-Preserving Reinforcement Learning Using Homomorphic Encryption in Cloud Computing Infrastructures | Eavesdropping Attack | (+) The researchers introduced a Privacy-Preserving Reinforcement Learning (PPRL) framework that enables users to outsource RL services to the cloud without revealing sensitive data by only transmitting encrypted ciphertexts. They adapt Q-learning to operate over encrypted state-action values. Through homomorphic evaluation, the cloud can perform Q-value updates, comparisons, and policy evaluation without ever accessing plaintext data. This preserves privacy even in the cloud-based decision service. |

| | | |
|---|---|---|
| | | (-) While the framework protects data confidentiality via encryption, it doesn't address side channels, metadata leakage, or attacks during decryption or key handling. The threat model is narrow and assumes ideal cryptographic trust. |
| Hsu et al. [72],<br>IEEE 9th International Conference on Cloud Networking,<br>2020,<br>A Deep Reinforcement Learning Approach for Anomaly Network Intrusion Detection System | DoS/DDoS | (+) They worked to train a DRL model on standard intrusion-detection datasets to classify network traffic and detect anomalies indicating cyberattacks. The DRL agent incrementally updates its policy based on feedback (rewards) for correct/incorrect detection decisions, adjusting to new or evolving patterns. The model is designed for online deployment, and the system continuously classifies incoming flows as benign or malicious without human intervention.<br>(-) The proposed model does not account for adversarial attacks on the DRL pipeline, insider threats, or novel zero-day variants beyond the dataset taxonomy. |
| Zhang et al. [76],<br>IEEE Internet of Things Journal<br>2020, | DoS/DDoS,<br>Eavesdropping Attack | (+) They introduce a mechanism to measure security levels of devices or traffic flows, and then map communications via virtual |

| | | |
|---|---|---|
| STEC-IoT: A security tactic by virtualizing edge computing on IoT | | networks so that higher-security traffic is handled in safer virtual domains. This mapping helps enforce differentiated protection.<br>(-) The experimentation is entirely simulation-based. There is no real-world deployment or validation on actual IoT or edge hardware. This limits confidence in practical applicability. |
| Ioannou et al. [34],<br>International conference on distributed computing in sensor systems (DCOSS), IEEE<br>2019,<br>Classifying security attacks in IoT networks using supervised learning | DoS/DDoS,<br>Routing Attacks | (+) They worked on a support vector machine-based model that uses generic or public datasets, and implemented specific network-layer attacks to prevent malware attacks on the underlying IoT network.<br>(-) The attacker is assumed to follow known attack behaviours implemented, rather than adapt dynamically or attempt to counter unknown IoT threats. |
| Khatun et al. [35],<br>22nd International Conference on Computer and Information Technology (ICCIT), IEEE<br>2019,<br>Malicious nodes detection based on artificial neural network in iot environments | DoS/DDoS | (+) They proposed an Artificial Neural Network (ANN) model to classify IoT devices as either malicious or benign, based on the patterns of incoming network traffic.<br>(-) Their proposed model evaluation is based on a dataset from a specific IoT setting; however, there may be a variety of device types, attack types, or |

| | | |
|---|---|---|
| | | topologies that have not been tested. |
| Yuan et al. [69],<br>IEEE Transactions on Neural Networks and Learning Systems,<br>2018,<br>Adversarial Examples: Attacks and Defenses for Deep Learning | DoS/DDoS,<br>Adversarial Attack | (+) They proposed a taxonomy on DNNs, with recent findings on adversarial attacks and their potential solutions being addressed.<br>(-) They illustrated several representative methods for generating adversarial attacks. However, very few of these methods are found to be successful in later applications for attack prevention. The existence of these methods requires bringing more robustness. |
| Azmoodeh et al. [71],<br>IEEE Transactions on Sustainable Computing,<br>2018,<br>Robust Malware Detection for Internet Of (Battlefield) Things Devices Using Deep Eigenspace Learning | Code insertion attacks | (+) They represented a deep learning-based framework through the Operational Code (OpCode) sequence of the device to detect Internet of Battlefield Things (IoBT) device malware and then applied a deep Eigenspace learning method to classify Benign and Malicious applications.<br>(-) The proposed plan needs to be evaluated against larger and broader datasets to implement a prototype in a real-world setting of IoT and IoBT systems for refinement and evaluation. |

| | | |
|---|---|---|
| Shakeel et al. [73],<br>Journal of Medical Systems, Springer<br>2018,<br>Maintaining Security and Privacy in Health Care System Using Learning Based Deep-Q-Networks | DoS/DDoS | (+) They proposed a learning-based Deep-Q-Networks to reduce intermediate attacks to secure the health information by reducing malware attacks. The proposed method analyzes and examines the different layers of health and medical data and information with a Q-learning concept, with comparatively low complexity.<br>(-) As proposed learning-based Deep-Q Network-based Security Analysis in IoT-Health Data Detection Structure was designed by using medical data-related traffic information. Therefore, such data contains very little malware data, and hence, the proposed model needed to be validated with another set of training data. |
| Doshi et al. [74],<br>IEEE Symposium on Security and Privacy Workshops<br>2018,<br>Machine Learning DDoS Detection for Consumer Internet of Things Devices | DoS/DDoS | (+) They worked on packet-level machine learning to automatically detect consumer IoT attack traffic to distinguish between normal and DoS attacks on the traffic of consumers of IoT devices.<br>(-) The proposed method was based on a hypothesis that requires more research on real-world scenarios in anomaly |

| | | |
|---|---|---|
| | | detection to protect networks from insecure IoT devices. |
| Diro et al. [28],<br>Future Generation Computer Systems, Elsevier<br>2017,<br>Distributed Attack Detection Scheme using Deep Learning Approach for Internet of Things | IoT/Fog network attack | (+) They designed an IoT/Fog network attack-detection system using a distributed deep-learning method to enable attack detection in cybersecurity for social IoT devices.<br>(-) The proposed attack detection system has a narrow performance comparison and needs to be evaluated on different datasets and different traditional machine learning algorithms, such as decision trees, SVM, and other neural networks, to achieve better accuracy. |
| Xiao et al. [19],<br>IEEE Transactions on Vehicular Technology, IEEE<br>2016,<br>PHY-layer spoofing detection with reinforcement learning in wireless networks | Spoofing,<br>DoS/DDoS | (+) They proposed a valuable contribution by combining game theory, hypothesis testing, and reinforcement learning to realize a PHY-layer spoofing detector, and even validated it via software-defined radio (SDR) experiments. Their work is a constructive step toward adaptive, learning-based physical-layer security.<br>(-) The experiments and techniques are focused on a particular wireless setting, and their transferability to modern wireless systems or under varying channel models is not discussed. |

| | | |
|---|---|---|
| Malialis et al. [16],<br>Engineering Applications of Artificial Intelligence, Elsevier<br>2015,<br>Distributed response to network intrusions using multiagent reinforcement learning | DoS/DDoS | (+) They propose a system in which multiple RL agents, each installed on routers, act independently to throttle or rate-limit traffic toward a victim server under DDoS attack. This decentralized method enables a response closer to the edge of the network.<br>(-) Most results are based on simulated network topologies and traffic patterns, which lack the ability to fully address real-world issues. This can influence how the approach works in practice. |
| Bhunia et al. [39],<br>IEEE Military Communications Conference, IEEE<br>2014,<br>Cr-honeynet: A learning & decoy based sustenance mechanism against jamming attack in crn | Jamming,<br>DDoS | (+) They suggested a stochastic learning method using CR-honeynet that allows the cognitive radio network (CRN) to learn the attacker's strategy, specifically, how the attacker selects important communications to target based on past attack patterns.<br>(-) Their proposed model evaluation is simulation-based. There is no real-world hardware implementation or field testing to account for radio channel imperfections, unintended interference, or practical limitations like measurement noise. |

## 2.2 Different Types of Attacks on IoT Devices

Due to the limited size, dimensions, power, and resource constraints inherent in IoT devices, these devices frequently become targets for security attacks. Attackers aim to either disrupt the functionality of these devices or steal sensitive data for their benefit. The constrained resources make it challenging to implement robust security measures, leaving IoT devices vulnerable to a wide range of threats, from service interruptions to data breaches. As the number of connected IoT devices continues to grow, so does the frequency and sophistication of these security attacks [67] [69] [109]. Some common types of attacks encountered are by such devices as shown in **Table 3** and described as follows,

**Table 3:** Attacks on Different Layers of IoT-Edge Computing

| Layer | Type of Attacks |
|---|---|
| Physical | Jamming, Tampering, Adversarial Attack |
| Link | Collision, Exhausting, Jamming |
| Network | Wormhole, Sybil, Sinkhole, Selective Forwarding Attack, Wormhole Attack, Man-in-the-Middle Attack, DoS/DDoS, |
| Transport | Flooding, Session Hijacking, Routing and Forwarding Attacks, DoS/DDoS |
| Application | Code Injection, Weak Authentication and Authorization, Phishing and Social Engineering, SQL Injection, DoS/DDoS |

### 2.2.1 DDoS Attacks on IoT-Edge Computation

One of the most serious threats facing the modern internet is distributed denial-of-service (DDoS) attacks, which primarily target the availability of systems and networks. A DDoS attack is a highly coordinated effort in which an attacker takes control of many compromised devices. These devices are typically infected with malware and remain dormant until activated by the attacker. Once instructed, such compromised devices launch a simultaneous and overwhelming flood of requests or traffic to the target system,

bombarding it with data at a rate far exceeding its handling capacity. The ultimate aim of a DDoS attack is to deplete a server's computational resources or saturate the bandwidth of a network's infrastructure, rendering the system incapable of functioning. This exhaustion of resources prevents legitimate users or customers from accessing the services they rely on, causing severe disruption. Such attacks are particularly harmful to businesses and service providers, as they can result in significant financial losses, damage to reputation, and corrosion of user trust. In more severe cases, critical infrastructures such as banking systems, healthcare services, and government networks can be brought to a standstill, exacerbating the consequences of the attack [36] [74] [79] [110].

The highly distributed and often decentralised nature of the victim servers makes it extremely difficult to mitigate DDoS attacks in real-time, as the malicious traffic originates from numerous sources spread across the globe. The threat of DDoS attacks has only grown in recent years with the rise of IoT devices, many of which have weak security measures, making them easy targets for attackers looking to expand their target network. Addressing this growing threat requires the development of more sophisticated detection and mitigation techniques, including real-time monitoring, machine learning models, and proactive defence mechanisms designed to counter the evolving tactics of DDoS attackers [80] [81] [111].

## 2.3 IoT-Edge Security Solutions with Machine Learning Approaches

Nowadays, machine learning has been proven as the best method to solve any type of problem and to provide solutions. Smart hackers or attackers even use this method to take control of any external entities in an unauthorised way. In light of the growing security challenges, machine learning (ML) has emerged as a powerful tool for enhancing IoT security. As IoT ecosystems become more complex, there is a need for advanced security systems. ML provides the necessary intelligence by utilising sophisticated algorithms and analysing collected data. It accomplishes this by detecting patterns, identifying anomalies, and predicting potential threats in real time, allowing for proactive responses to vulnerabilities and attacks [17] [40] [44] [112]. ML algorithms analyse the normal behaviour of IoT devices, networks, and communication channels to create a baseline. This helps them quickly detect deviations or unusual

activities, marking them as potential security threats. By doing so, ML facilitates early detection and response to cyberattacks like DDoS or malware, reducing the risk of significant damage [78].

### 2.3.1 IoT-Edge Security Provided by Reinforcement Learning Method

Reinforcement learning (RL) has emerged as a powerful tool for enhancing IoT-Edge security by enabling adaptive and dynamic responses to threats. With IoT devices continuously generating large volumes of data, RL can help secure communication between edge servers and IoT devices, providing a robust defence against cyberattacks. RL-based security models excel in dynamic and real-time environments, as they can learn and adapt to evolving attack patterns without requiring large pre-existing datasets. One key advantage of using RL in IoT is its ability to interact with the environment, learning from trial and error, which is particularly useful in complex, fast-changing IoT networks [10] [16] [113].

In IoT-edge computing, RL models can be deployed to optimise resource management, secure data transmission, and reduce latency while dealing with adversarial learning environments. For example, deep reinforcement learning (DRL) methods have been used to enhance energy efficiency and secure communication channels against eavesdropping and other cyber threats by leveraging edge servers for real-time control and processing of IoT device data [114]. Additionally, researchers have explored combining RL with other technologies like blockchain for secure resource management and privacy protection in IoT systems, ensuring greater resilience to attacks and improving overall system security. Typically, the following types of attacks are examined for the utility of reinforcement learning and deep reinforcement learning techniques in malware detection [115].

1) Adversarial Attack
2) Code Insertion Attack
3) IoT/Fog network attack
4) Internet Network attack
5) Attack on the Edge server

These attacks can affect any part of the process of working with IoT devices and result in unreliable and inconvenient ways of working. To counter such malware threats over IoT devices, a machine learning trained security model is needed to protect such devices. Self-taught and learning-based RL techniques can play a vital role in distinguishing malware from benign traffic. Since the invention and beginning of the initial application of the Reinforcement Learning method, which is described by its mathematical representation, Q-Learning has played a foundational role in measuring and preventing malware attacks on an organisation's communication network [54].
Reinforcement Learning is one of the best machine learning methods to learn from the surrounding environment, where many things are unknown to the acting agent, and the agent can make decisions on taking some actions in the environment. Action taken by the RL agent changes its state, which ultimately helps to reach its destination state. At each state change agent either gets a reward or a penalty based on the decision taken by it according to the RL policy. Based on the outcomes of each movement of the agent, the RL policy gets updated to optimise the best solution of the given problem [26] [27].

The Reinforcement Learning (RL) method can also be used to implement and provide the security layer for IoT-Edge computation. The Reinforcement Learning method allows an entity in the system to act as an Agent that keeps doing some actions in the surrounding environment, which leads to a state transition for it. Due to regular interactions between the Agent and the Environment, the familiarity with the environment related to its diversity and implications keeps growing and provides sufficient data to determine the malware attacks that happen in the environment to steal sensitive data from the IoT devices. Until and unless the Agent in an Environment does not get the full information about the Environment, it keeps taking some action in the Environment. Once it has full knowledge about its state transition while taking actions in the Environment, it stops making further actions in the Environment upon reaching this saturation level [13] [14] [18] [19].

Reinforcement Learning (RL) is a type of machine learning paradigm where an agent learns to make decisions by interacting with an environment in order to maximise a notion of cumulative reward. Unlike supervised learning, which relies on labelled data, RL learns through trial and error, receiving feedback in the form of rewards or penalties.

In Reinforcement Learning, agents are created to regularly interact with the surrounding environment, as shown in **Figure 10**. During the interaction with the environment, the Agent learns what to do and how to map the situations into actions. The most important thing in this machine learning method is that the Agent is not explicitly told what actions should be taken in some circumstances, whereas the agent takes the action based upon its past experiences through trial and error, by which it can maximise total cumulative reward. It is inspired by behavioural psychology and is particularly useful for solving sequential decision-making problems, especially when the model of the environment is unknown or difficult to model explicitly. Each action taken by the agent is defined with a value. This value is determined by using the action-value function. As the value of taking an action is not known before the Agent, the sample-average method is used to estimate the value of the action taken [10] [12] [16].

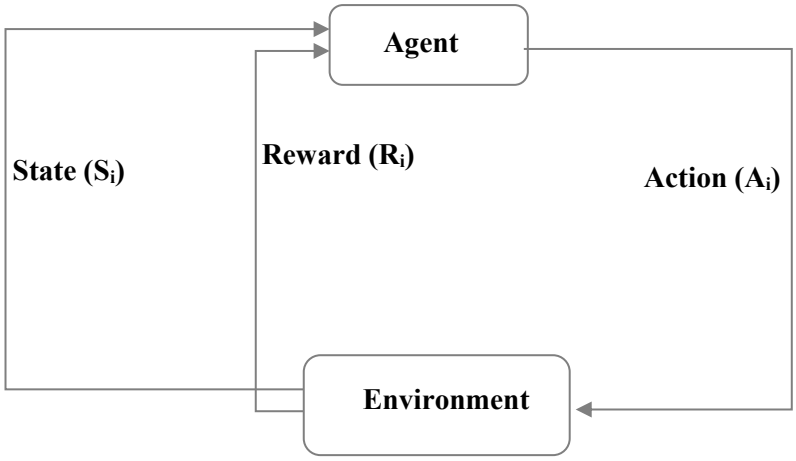



**Figure 10:** A Basic Reinforcement Learning Model

### 2.3.2 Epsilon (ε) Greedy Reinforcement Learning Method

A widely used epsilon **(ε)** greedy reinforcement learning method is also used to detect malware attacks. It is based mainly on two processes: 1) **Exploration** and 2) **Exploitation**. This method starts with exploration and continues till the Agent gathers sufficient information about the environment and becomes efficient enough to decide which action can yield maximum reward in the due course of time. Hence, Exploration improves the current knowledge about each action the Agent takes. On the other hand,

Exploitation helps the Agent to find the greedy action that can provide the maximum or optimal reward by exploiting the already accumulated action-value estimates from the exploration phase. Hence, during the exploration, the agent refines the action set, and during the exploitation, the agent applies an action to get an optimal or maximum reward. The agent cannot run both these processes simultaneously, which is known as the exploration-exploitation dilemma [10] [12] [13]. We are motivated to use the Ɛ-greedy reinforcement learning method because of its simplicity and focus on Ɛ-value, which is the probability of choosing to explore. Hence, the (1 - Ɛ) value will be the probability of exploiting the environment. In general, Ɛ-value is a very small value for most of the time exploitation, and with a small amount of duration for exploration. We have designed and implemented adaptive Ɛ-greedy reinforcement learning to protect IoT-Edge computation and experimentally verified that our proposed adaptive Ɛ-greedy reinforcement learning is much better than other similar methods regarding IoT-Edge computation security.

Balancing exploration and exploitation remain a significant challenge in reinforcement learning, critically affecting both learning efficiency and the quality of the derived policies. Excessive exploration can hinder the accumulation of immediate rewards, as exploratory actions may lead to unfavourable outcomes. Conversely, relying too heavily on the exploitation of uncertain or incomplete knowledge can prevent the agent from discovering more optimal actions, thereby limiting long-term reward. This fundamental trade-off is widely recognized as the "Exploration-Exploitation Dilemma" [108].

## 2.4 Limitations of IoT-Edge Computation

IoT-Edge computing has numerous benefits, such as reduced latency, bandwidth savings, and improved real-time data processing. However, it also has several limitations and challenges, which are as follows [21] - [28].

1) Edge devices typically have less processing power, memory, and storage than cloud servers.
2) Edge devices are often more vulnerable to physical tampering and cyberattacks due to distributed deployment.

3) Managing and updating a vast number of edge devices in a large distributed network can be complex.
4) Diverse hardware platforms and operating systems are needed across different IoT-Edge computing environments.
5) Edge servers often don't have backup or failover systems like cloud infrastructure.
6) Data processed at the Edge server may fall under different legal or regulatory jurisdictions.
7) The lack of trust between edge servers and IoT devices has created a lot of hindrances to adopting edge computing that is universally accepted. It is giving attackers plenty of opportunities to intervene and steal sensitive data from IoT devices.
8) The number of IoT users is increasing exponentially each day. For our many day-to-day living needs, we depend on IoT devices, either directly with handheld devices or indirectly with integrated IoT devices into Smart homes, Smart transport, Automotive vehicles, etc., providing a large attack surface to cybercriminals.
9) Traditional security mechanisms such as Firewalls, Intrusion detection, and cryptography are insufficient to counter sophisticated, intelligent, and resourceful malware attacks.
10) Intelligent attackers might create a proxy user in front of an IoT device that seems like an authentic user instead of a hacker.
11) Many Smart IoT devices generate huge amounts of data within a fraction of time, which is also a challenging task to recollect from the edge device and then push over to the Cloud. This limitation of the edge device invites attackers to steal sensitive data from the IoT devices.

## 2.5 Research Gaps

Based on our literature review, we have identified the following research gaps that still require attention and must be addressed [54] - [58].

1) Due to the limited size, dimensions, power, and resource constraints inherent in IoT devices, these devices are unable to establish secure communication with Edge devices, leading to them frequently becoming targets for security attacks.
2) Once an Edge server is down, IoT devices lose coordination, leading to retries and message storms that amplify the DoS/DDoS attacks.
3) Deployment of Edge Server for IoT devices often lacks centralised monitoring to detect malware attacks.
4) Ensuring consistent security policies across all these nodes is difficult, especially when different teams or vendors manage them.
5) Establishing trust between IoT devices, edge servers, and cloud is challenging due to heterogeneity and scale. Hence, maintaining secure identity management and key distribution is a complex and error-prone process.
6) As the number of connected IoT devices in Edge computation continues to grow, so does the frequency and sophistication of security attacks, increasing day by day.

## 2.6 Research Objectives

Our research aims to secure IoT-Edge computation. The research papers in [1] - [8] are our earlier published works on IoT-Edge security. In the current research, we propose a novel reinforcement learning method for the security of IoT-Edge computation with more accuracy, robustness, and scalability. The proposed research work in this PhD thesis aims to achieve the following objectives.

1) **To pre-process the data gathered from IoT devices on the edge layer.**

   With this first objective, we collected datasets from various sources, including the ESP32 Real-time Dataset, IoT-23 Dataset [29], and the Simulation dataset.

2) **To design and implement a novel reinforcement-based secure approach between IoT devices and Edge nodes to prevent them from security attacks.**

We proposed an RL-based security method that protects IoT data arriving on the Edge server from any external malware threats.

3) **To test and validate the proposed approach.**

We validated and compared our proposed security model with similar security methods to prove better efficiency and reliability.

## 2.7 Applicability of the Proposed Model in Real World Applications

Secure IoT-Edge computing models are highly applicable in real-world systems where low latency, privacy preservation, resilience, and trust are critical. Their adoption strengthens cybersecurity while enabling scalable, real-time applications across multiple domains. The proposed security model is installed at the Edge device near an IoT premises, so it has a practical necessity in the real world, as IoT devices are deeply integrated into smart homes, healthcare, industry, and transportation [78] [80] - [85], and influence the following domains.

1) **Healthcare**: Secure IoT-Edge models protect sensitive patient data while enabling real-time analysis of vital signs from wearable devices at the Edge device. This ensures faster decision-making without exposing data to centralised cloud vulnerabilities.
2) **Industrial IoT (IIoT)**: In manufacturing and critical infrastructure, secure Edge models prevent cyberattacks such as DDoS or Ransomware, ensuring uninterrupted operations, predictive maintenance, and safe automation.
3) **Smart Cities**: Applications like traffic management, surveillance, and energy distribution rely on a rapid local area decision-making system. Secure IoT-Edge ensures that these systems are resilient against attacks and safeguard citizen privacy.
4) **Autonomous Vehicles**: Edge security models help vehicles communicate and process data locally with minimal latency, while preventing malicious manipulation of navigation or sensor data.
5) **Smart Homes**: Enhances security for connected home devices such as smart locks, cameras, and voice assistants by performing local, Edge-based

authentication and anomaly detection, thereby reducing the risks of intrusion, data leakage, or privacy violations.

6) **Agriculture & Environmental Monitoring**: Edge security protects remote sensors deployed in unattended environments and supports offline operation.

## 2.8 Summary

We have conducted an extensive literature review of existing IoT security methods and, simultaneously, a detailed discussion of IoT-Edge computation. We have observed that existing IoT security has specific contributions to address certain security issues, but at the same time, certain limitations still leave the confidentiality or integrity of IoT data vulnerable to manipulation. We have discussed the same in detail by conducting a comparative study on the available IoT security methods. We have also discussed different types of prevalent security attacks on IoT devices, which need researchers' attention to protect IoT data from these attacks. We have also discussed the impact of DDoS attacks on IoT-Edge computation. The mitigation of DDoS attacks from IoT-Edge computation is the main objective of our proposed research work. We have also discussed the recent IoT-Edge security solutions with existing machine-learning methods, and at the same time, we also discussed research gaps in these available IoT security solutions that need attention. We emphasized the use of the Reinforcement Learning (RL) method to provide security for IoT-Edge computation. We also discussed how we used the Epsilon (Ɛ) Greedy Reinforcement Learning Method to protect IoT-Edge computation in a much better way. At the end, we discussed the research objectives, which are the overall goals of our research work. Finally, we also discussed the applicability of the proposed model in the various domains of real-world applications, where our proposed model would make a significant impact to mitigate the existing security concerns.

# CHAPTER 3

# PLAN OF RESEARCH WORK

The proposed research work aims to provide security to the Edge server, as the Edge server is the very first initial data temporary data repository for the data coming from the IoT device (s). Hence, it is the most valuable entity that intruders want to take control of. Our purpose is to maintain packet delivery ratio (PDR), average throughput, and end-to-end (E2E) delay with minimal effects from external threats. Hence, we are safeguarding the communication channel among IoT devices (s), the Edge server, and the Cloud server. The proposed security approach uses a novel reinforcement learning method to enable secure data communication across the underlying communication channel, but it does not have any relevance to the physical layer of the connected IoT device or the nature of the Cloud server, whether it is a public, private, or hybrid server. It is only a software plugin that will be installed and set up for our proposed security scheme. Hence, no extra hardware setup is required to use our security policy.

## 3.1 Statement of the Problem

With the rapid proliferation of Internet of Things (IoT) devices and their integration into Edge computing environments, ensuring the security of these devices has become a critical challenge. This rise of the Internet of Things (IoT) has led to the deployment of billions of interconnected devices across various sectors, including healthcare, smart cities, transportation, and industry. These devices generate enormous volumes of data and often operate in distributed environments close to end-users, known as edge computing. Edge computing offers several advantages, such as reduced latency, improved bandwidth utilization, and localized data processing. However, it also introduces new security vulnerabilities due to the decentralized and resource-constrained nature of IoT devices. Traditional security solutions, such as rule-based firewalls or signature-based intrusion detection systems, are often ineffective [61] [62]. Followings are the major impediments to securing IoT devices in Edge computing [63] - [70].

1) Heterogeneity and scale of IoT devices in a large, distributed edge computing environment. The exponentially increasing user demands increase the heterogeneity.

2) IoT devices have stringent resource constraints, for example, limited processing power, memory storage, and battery life, etc. It weakens the built-in security layer of IoT devices and invites hackers to enter the IoT network to steal sensitive data from it.
3) Edge nodes in a large IoT network are mostly decentralized, which prevents keeping them updated with new software installations. It increases the vulnerabilities in the entire IoT network infrastructure.
4) Evolving intelligent cyber threats on the IoT network require a regular, adaptive, and intelligent defence mechanism in IoT-Edge computation.
5) Existing security methods lack in operating with low latency to respond promptly to such security threats.

## 3.2 Research Outline

The proposed research focuses on developing a reinforcement learning (RL)-based security framework to protect IoT devices operating in edge computing environments. The work begins by exploring the need for rapid growth of IoT devices and the increasing reliance on Edge computing. We studied with an in-depth analysis of the unique security challenges faced by IoT users nowadays, particularly in dynamic, distributed, and resource-constrained contexts. A comprehensive literature review highlights the limitations of traditional security mechanisms and identifies reinforcement learning as a promising, adaptive solution. We proposed a research methodology that involves designing a custom RL model tailored to IoT-Edge environments, where agents learn to detect and respond to cyber threats such as DoS/DDoS, Spoofing, Eavesdropping, and Data tampering. The proposed framework will be implemented and evaluated in a simulated platform that reflects real-world constraints, using metrics such as detection accuracy, latency, and resource efficiency. In order to make the proposed security meet real-time security constraints, we again validated and strengthened the proposed security model with public and in-house IoT device datasets. Results will be compared with baseline security solutions to demonstrate the effectiveness of the proposed RL-based security approach. Ultimately, our research aims to bridge the gap between intelligent threat mitigation and the practical deployment of security solutions in emerging IoT-Edge infrastructures.

The scale of Internet-connected systems has expanded significantly in recent years, encompassing a vast ecosystem of IoT, IIoT, and Edge computing devices. As a consequence, these systems are becoming increasingly vulnerable to sophisticated and frequent cyberattacks. Traditional security mechanisms are often insufficient to deal with the dynamic, multi-vector nature of modern threats, which evolve rapidly and exploit even minor vulnerabilities in real-time environments. To address these challenges, security mechanisms must be responsive, adaptive, scalable, and resilient under the following conditions [112].

1) Responsiveness ensures minimal reaction time to new or ongoing attacks.
2) Adaptability allows systems to learn from emerging threat patterns and evolve defence strategies accordingly.
3) Scalability ensures protection can be extended across thousands or even millions of interconnected devices without degrading performance.
4) Resiliency ensures that the defence system works perfectly over low to high varying system overheads.

This demand has driven the exploration of intelligent, learning-based security frameworks, such as those employing reinforcement learning based anomaly detection intrusion detection systems, which can continuously refine their defence policies in response to changing threat landscapes.

## 3.3 Research Scope

Our research work is focused on enhancing the security of Edge servers, which act as immediate intermediaries between IoT devices and Cloud infrastructure. Specifically, we target the protection of incoming IoT data on the Edge server during its preprocessing phase before it is forwarded to the cloud for permanent storage and broader analytics. This is a critical stage, as any compromise here can undermine the confidentiality, integrity, or availability of the entire IoT-Edge-Cloud communication channel. Most manufacturers prefer the complex computation to happen at the Edge server rather than at the IoT device itself or the Cloud server. This strategy saves much cost because it avoids integrating advanced physical layers into the IoT device and also avoids purchasing higher physical resources at the connected Cloud server [15] [21] [23] [25]. We are focused on preventing DDoS attacks on the IoT-Edge computation.

Our proposed security method protects IoT-Edge computation at the application and transport layers from external malware attacks.

### 3.4 Research Plan Roadmap

The research methodology was comprehensively designed and validated through two distinct experimental setups to evaluate the robustness, scalability, and real-world applicability of the proposed novel reinforcement learning based security model to protect IoT-Edge computation. These experimental setups are as follows.

1) Network Simulator-Based Experiment
2) Real-Time Data Communication-Based Experiment
    a) Real-Time Public Domain Data-Based Experiment
    b) Real-Time In-house IoT Device Communication-Based Experiment

Each experimental setup experiments with the following three sequential network model designs and implementations.

i. A baseline network model formation without any attacks.
ii. The same network model under the cyberattacks.
iii. Again, the same network is secured using the proposed security model.

### 3.5 Proposed Research Workflow

The proposed research work uses three kinds of datasets, namely, **1)** Simulation dataset, **2)** Public dataset, and **3)** In-house Device dataset. The steps involved in the proposed research work are illustrated in **Figure 11**. The following are the three main routes in the proposed research work.

1) Collect and evaluate the model using the Simulation dataset
2) Collect and evaluate the model using the public dataset
3) Collect and evaluate the model using the In-house Device dataset

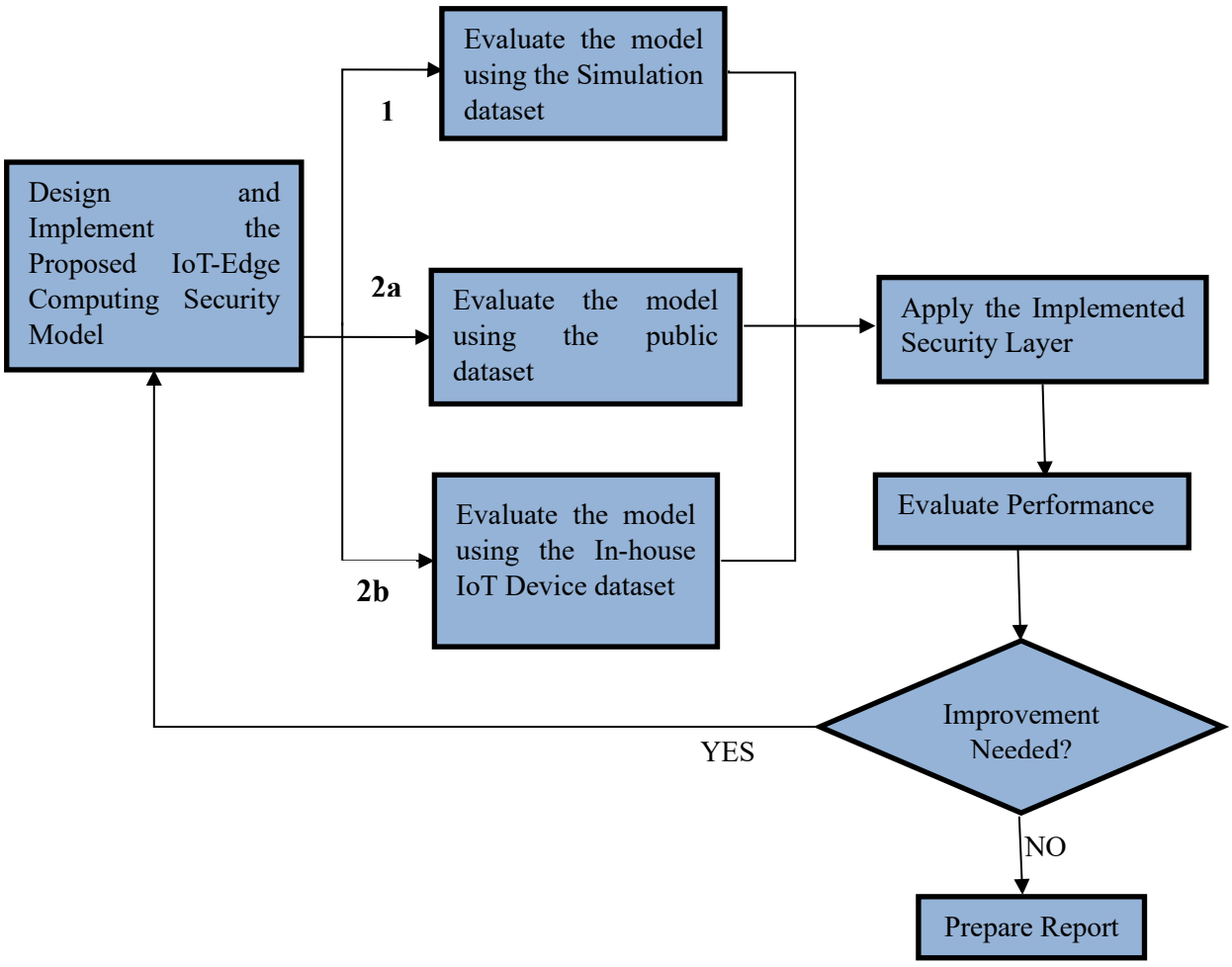


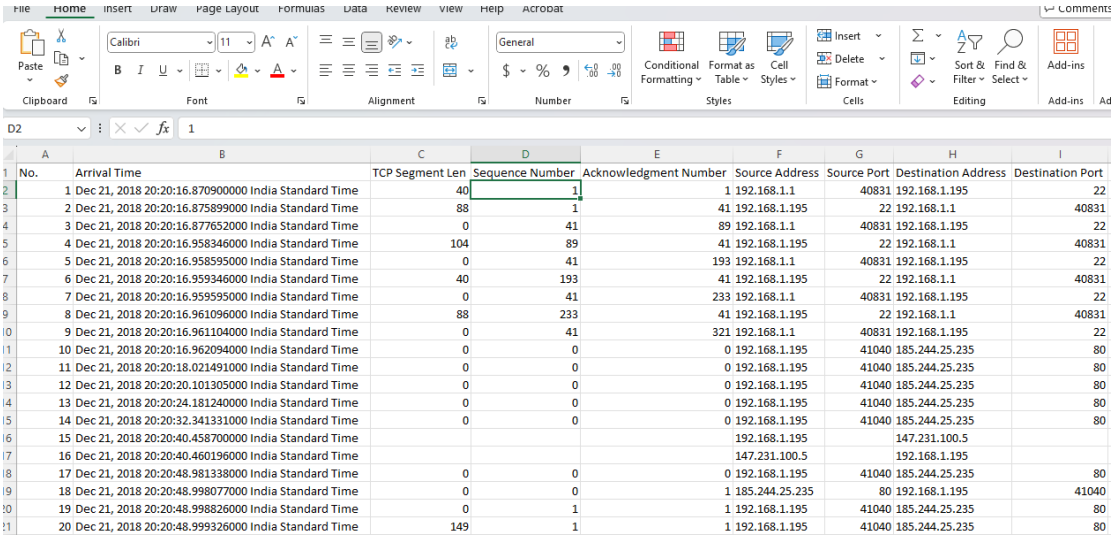

**Figure 11:** Proposed Research Work Flow Block Diagram

First of all, we designed and implemented the proposed IoT-Edge Computing Security model using the NS-2.35 simulator [32] [33]. At this time, we evaluated our model using the NS-2.35 simulator-generated dataset, as shown in **Figure 12**. Once we achieve the desired performance of the security model, the same model is assessed again against the dataset collected through various sensor data from our personal computer using the tool "HWiNFO" [74] and public datasets [29]. We have used the IoT-23 public dataset [28], as shown in **Figure 13**, to validate our proposed security model. At this time, we again improved our model to meet our security expectations. Finally, we evaluated our proposed model using in-house real-time datasets, which include the ESP-WROOM-32 IoT device [41] [42] connected with a DHT22 Temperature and Humidity sensor module [43]. **Figure 14**, and **Figure 15** show the real-time non-malicious and malicious IoT device datasets. We have experimentally worked until the proposed IoT-Edge

computing security model provides the expected security requirements on real-time IoT devices.

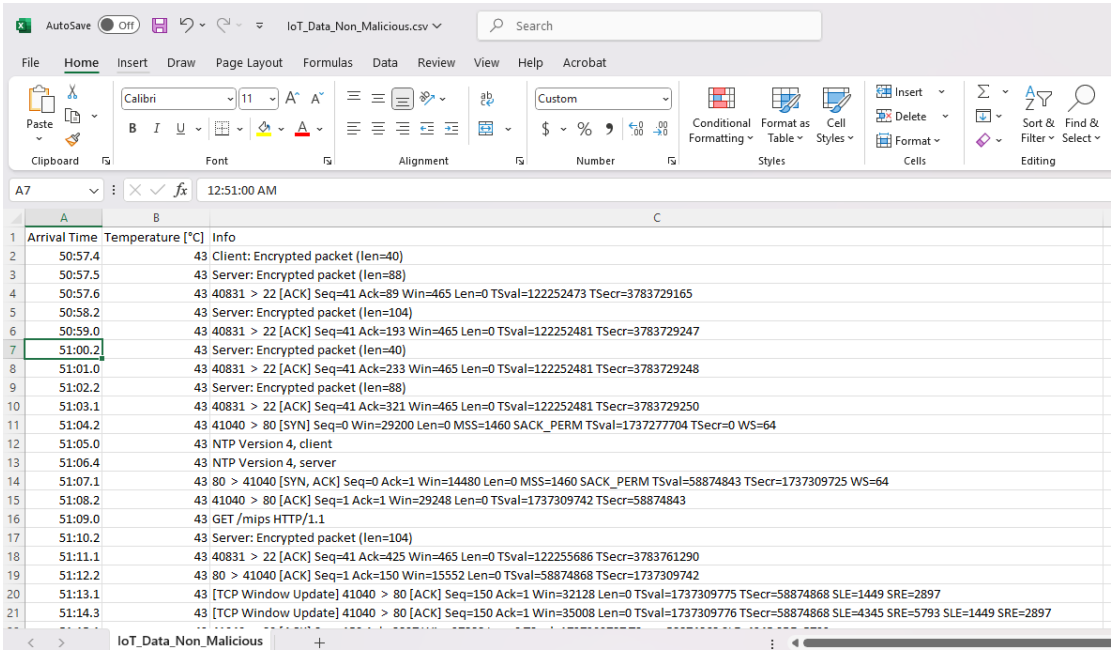
```
151469 s 41.167680000 _0_ AGT  --- 31381 cbr 40 [0 0 0 0] ------- [0:0 3:0 32 0] [31381] 0 0
151470 r 41.167680000 _0_ RTR  --- 31381 cbr 40 [0 0 0 0] ------- [0:0 3:0 32 0] [31381] 0 0
151471 s 41.167680000 _0_ RTR  --- 31381 cbr 60 [0 0 0 0] ------- [0:0 3:0 30 7] [31381] 0 0
151472 D 41.167705000 _0_ IFQ  --- 31381 cbr 60 [0 7 0 800] ------- [0:0 3:0 30 7] [31381] 0 0
151473 s 41.168960000 _0_ AGT  --- 31382 cbr 40 [0 0 0 0] ------- [0:0 3:0 32 0] [31382] 0 0
151474 r 41.168960000 _0_ RTR  --- 31382 cbr 40 [0 0 0 0] ------- [0:0 3:0 32 0] [31382] 0 0
151475 s 41.168960000 _0_ RTR  --- 31382 cbr 60 [0 0 0 0] ------- [0:0 3:0 30 7] [31382] 0 0
151476 D 41.168985000 _0_ IFQ  --- 31382 cbr 60 [0 7 0 800] ------- [0:0 3:0 30 7] [31382] 0 0
151477 r 41.169528447 _7_ RTR  --- 31202 cbr 60 [13a 7 0 800] ------- [0:0 3:0 30 7] [31202] 1 0
151478 f 41.169528447 _7_ RTR  --- 31202 cbr 60 [13a 7 0 800] ------- [0:0 3:0 29 4] [31202] 1 0
151479 s 41.170240000 _0_ AGT  --- 31383 cbr 40 [0 0 0 0] ------- [0:0 3:0 32 0] [31383] 0 0
151480 r 41.170240000 _0_ RTR  --- 31383 cbr 40 [0 0 0 0] ------- [0:0 3:0 32 0] [31383] 0 0
151481 s 41.170240000 _0_ RTR  --- 31383 cbr 60 [0 0 0 0] ------- [0:0 3:0 30 7] [31383] 0 0
151482 s 41.171520000 _0_ AGT  --- 31384 cbr 40 [0 0 0 0] ------- [0:0 3:0 32 0] [31384] 0 0
151483 r 41.171520000 _0_ RTR  --- 31384 cbr 40 [0 0 0 0] ------- [0:0 3:0 32 0] [31384] 0 0
151484 s 41.171520000 _0_ RTR  --- 31384 cbr 60 [0 0 0 0] ------- [0:0 3:0 30 7] [31384] 0 0
151485 r 41.171534353 _4_ RTR  --- 31200 cbr 60 [13a 4 7 800] ------- [0:0 3:0 29 4] [31200] 2 0
151486 D 41.171534353 _4_ RTR LOOP 31200 cbr 60 [13a 4 7 800] ------- [0:0 3:0 28 4] [31200] 2 0
151487 D 41.171545000 _0_ IFQ  --- 31384 cbr 60 [0 7 0 800] ------- [0:0 3:0 30 7] [31384] 0 0
151488 s 41.172800000 _0_ AGT  --- 31385 cbr 40 [0 0 0 0] ------- [0:0 3:0 32 0] [31385] 0 0
151489 r 41.172800000 _0_ RTR  --- 31385 cbr 40 [0 0 0 0] ------- [0:0 3:0 32 0] [31385] 0 0
151490 s 41.172800000 _0_ RTR  --- 31385 cbr 60 [0 0 0 0] ------- [0:0 3:0 30 7] [31385] 0 0
151491 D 41.172825000 _0_ IFQ  --- 31385 cbr 60 [0 7 0 800] ------- [0:0 3:0 30 7] [31385] 0 0
151492 r 41.173620895 _4_ RTR  --- 31202 cbr 60 [13a 4 7 800] ------- [0:0 3:0 29 4] [31202] 2 0
151493 D 41.173620895 _4_ RTR LOOP 31202 cbr 60 [13a 4 7 800] ------- [0:0 3:0 28 4] [31202] 2 0
151494 s 41.174080000 _0_ AGT  --- 31386 cbr 40 [0 0 0 0] ------- [0:0 3:0 32 0] [31386] 0 0
151495 r 41.174080000 _0_ RTR  --- 31386 cbr 40 [0 0 0 0] ------- [0:0 3:0 32 0] [31386] 0 0
151496 s 41.174080000 _0_ RTR  --- 31386 cbr 60 [0 0 0 0] ------- [0:0 3:0 30 7] [31386] 0 0
151497 D 41.174105000  0  IFQ  --- 31386 cbr 60 [0 7 0 800] ------- [0:0 3:0 30 7] [31386] 0 0
```

**Figure 12**: NS-2.35 Simulator-generated Dataset



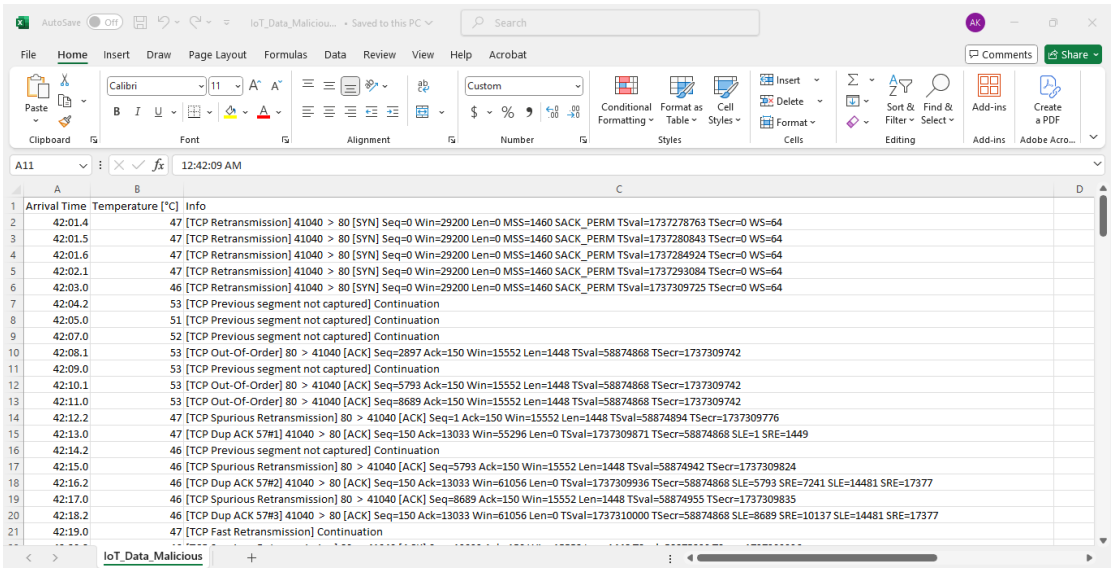
| No. | Arrival Time | TCP Segment Len | Sequence Number | Acknowledgment Number | Source Address | Source Port | Destination Address | Destination Port |
|---|---|---|---|---|---|---|---|---|
| 1 | Dec 21, 2018 20:20:16.870900000 India Standard Time | 40 | 1 | 1 | 192.168.1.1 | 40831 | 192.168.1.195 | 22 |
| 2 | Dec 21, 2018 20:20:16.875899000 India Standard Time | 88 | 1 | 41 | 192.168.1.195 | 22 | 192.168.1.1 | 40831 |
| 3 | Dec 21, 2018 20:20:16.877652000 India Standard Time | 0 | 41 | 89 | 192.168.1.1 | 40831 | 192.168.1.195 | 22 |
| 4 | Dec 21, 2018 20:20:16.958346000 India Standard Time | 104 | 89 | 41 | 192.168.1.195 | 22 | 192.168.1.1 | 40831 |
| 5 | Dec 21, 2018 20:20:16.958595000 India Standard Time | 0 | 41 | 193 | 192.168.1.1 | 40831 | 192.168.1.195 | 22 |
| 6 | Dec 21, 2018 20:20:16.959346000 India Standard Time | 40 | 193 | 41 | 192.168.1.195 | 22 | 192.168.1.1 | 40831 |
| 7 | Dec 21, 2018 20:20:16.959595000 India Standard Time | 0 | 41 | 233 | 192.168.1.1 | 40831 | 192.168.1.195 | 22 |
| 8 | Dec 21, 2018 20:20:16.961096000 India Standard Time | 88 | 233 | 41 | 192.168.1.195 | 22 | 192.168.1.1 | 40831 |
| 9 | Dec 21, 2018 20:20:16.961104000 India Standard Time | 0 | 41 | 321 | 192.168.1.1 | 40831 | 192.168.1.195 | 22 |
| 10 | Dec 21, 2018 20:20:16.962094000 India Standard Time | 0 | 0 | 0 | 192.168.1.195 | 41040 | 185.244.25.235 | 80 |
| 11 | Dec 21, 2018 20:20:18.021491000 India Standard Time | 0 | 0 | 0 | 192.168.1.195 | 41040 | 185.244.25.235 | 80 |
| 12 | Dec 21, 2018 20:20:20.101305000 India Standard Time | 0 | 0 | 0 | 192.168.1.195 | 41040 | 185.244.25.235 | 80 |
| 13 | Dec 21, 2018 20:20:24.181240000 India Standard Time | 0 | 0 | 0 | 192.168.1.195 | 41040 | 185.244.25.235 | 80 |
| 14 | Dec 21, 2018 20:20:32.341331000 India Standard Time | 0 | 0 | 0 | 192.168.1.195 | 41040 | 185.244.25.235 | 80 |
| 15 | Dec 21, 2018 20:20:40.458700000 India Standard Time | | | | 192.168.1.195 | | 147.231.100.5 | |
| 16 | Dec 21, 2018 20:20:40.460196000 India Standard Time | | | | 147.231.100.5 | | 192.168.1.195 | |
| 17 | Dec 21, 2018 20:20:48.981338000 India Standard Time | 0 | 0 | 0 | 192.168.1.195 | 41040 | 185.244.25.235 | 80 |
| 18 | Dec 21, 2018 20:20:48.998077000 India Standard Time | 0 | 0 | 1 | 185.244.25.235 | 80 | 192.168.1.195 | 41040 |
| 19 | Dec 21, 2018 20:20:48.998826000 India Standard Time | 0 | 1 | 1 | 192.168.1.195 | 41040 | 185.244.25.235 | 80 |
| 20 | Dec 21, 2018 20:20:48.999326000 India Standard Time | 149 | 1 | 1 | 192.168.1.195 | 41040 | 185.244.25.235 | 80 |

**Figure 13**: IoT-23 Public Dataset



| Arrival Time | Temperature [°C] | Info |
|---|---|---|
| 50:57.4 | 43 | Client: Encrypted packet (len=40) |
| 50:57.5 | 43 | Server: Encrypted packet (len=88) |
| 50:57.6 | 43 | 40831 > 22 [ACK] Seq=41 Ack=89 Win=465 Len=0 TSval=122252473 TSecr=3783729165 |
| 50:58.2 | 43 | Server: Encrypted packet (len=104) |
| 50:59.0 | 43 | 40831 > 22 [ACK] Seq=41 Ack=193 Win=465 Len=0 TSval=122252481 TSecr=3783729247 |
| 51:00.2 | 43 | Server: Encrypted packet (len=40) |
| 51:01.0 | 43 | 40831 > 22 [ACK] Seq=41 Ack=233 Win=465 Len=0 TSval=122252481 TSecr=3783729248 |
| 51:02.2 | 43 | Server: Encrypted packet (len=88) |
| 51:03.1 | 43 | 40831 > 22 [ACK] Seq=41 Ack=321 Win=465 Len=0 TSval=122252481 TSecr=3783729250 |
| 51:04.2 | 43 | 41040 > 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM TSval=1737277704 TSecr=0 WS=64 |
| 51:05.0 | 43 | NTP Version 4, client |
| 51:06.4 | 43 | NTP Version 4, server |
| 51:07.1 | 43 | 80 > 41040 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM TSval=58874843 TSecr=1737309725 WS=64 |
| 51:08.2 | 43 | 41040 > 80 [ACK] Seq=1 Ack=1 Win=29248 Len=0 TSval=1737309742 TSecr=58874843 |
| 51:09.0 | 43 | GET /mips HTTP/1.1 |
| 51:10.2 | 43 | Server: Encrypted packet (len=104) |
| 51:11.1 | 43 | 40831 > 22 [ACK] Seq=41 Ack=425 Win=465 Len=0 TSval=122255686 TSecr=3783761290 |
| 51:12.2 | 43 | 80 > 41040 [ACK] Seq=1 Ack=150 Win=15552 Len=0 TSval=58874868 TSecr=1737309742 |
| 51:13.1 | 43 | [TCP Window Update] 41040 > 80 [ACK] Seq=150 Ack=1 Win=32128 Len=0 TSval=1737309775 TSecr=58874868 SLE=1449 SRE=2897 |
| 51:14.3 | 43 | [TCP Window Update] 41040 > 80 [ACK] Seq=150 Ack=1 Win=35008 Len=0 TSval=1737309776 TSecr=58874868 SLE=4345 SRE=5793 SLE=1449 SRE=2897 |

**Figure 14**: Real-time IoT Device Non-Malicious Dataset



| Arrival Time | Temperature [°C] | Info |
|---|---|---|
| 42:01.4 | 47 | [TCP Retransmission] 41040 > 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM TSval=1737278763 TSecr=0 WS=64 |
| 42:01.5 | 47 | [TCP Retransmission] 41040 > 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM TSval=1737280843 TSecr=0 WS=64 |
| 42:01.6 | 47 | [TCP Retransmission] 41040 > 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM TSval=1737284924 TSecr=0 WS=64 |
| 42:02.1 | 47 | [TCP Retransmission] 41040 > 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM TSval=1737293084 TSecr=0 WS=64 |
| 42:03.0 | 46 | [TCP Retransmission] 41040 > 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM TSval=1737309725 TSecr=0 WS=64 |
| 42:04.2 | 53 | [TCP Previous segment not captured] Continuation |
| 42:05.0 | 51 | [TCP Previous segment not captured] Continuation |
| 42:07.0 | 52 | [TCP Previous segment not captured] Continuation |
| 42:08.1 | 53 | [TCP Out-Of-Order] 80 > 41040 [ACK] Seq=2897 Ack=150 Win=15552 Len=1448 TSval=58874868 TSecr=1737309742 |
| 42:09.0 | 53 | [TCP Previous segment not captured] Continuation |
| 42:10.1 | 53 | [TCP Out-Of-Order] 80 > 41040 [ACK] Seq=5793 Ack=150 Win=15552 Len=1448 TSval=58874868 TSecr=1737309742 |
| 42:11.0 | 53 | [TCP Out-Of-Order] 80 > 41040 [ACK] Seq=8689 Ack=150 Win=15552 Len=1448 TSval=58874868 TSecr=1737309742 |
| 42:12.2 | 47 | [TCP Spurious Retransmission] 80 > 41040 [ACK] Seq=1 Ack=150 Win=15552 Len=1448 TSval=58874894 TSecr=1737309776 |
| 42:13.0 | 47 | [TCP Dup ACK 57#1] 41040 > 80 [ACK] Seq=150 Ack=13033 Win=55296 Len=0 TSval=1737309871 TSecr=58874868 SLE=1 SRE=1449 |
| 42:14.2 | 46 | [TCP Previous segment not captured] Continuation |
| 42:15.0 | 46 | [TCP Spurious Retransmission] 80 > 41040 [ACK] Seq=5793 Ack=150 Win=15552 Len=1448 TSval=58874942 TSecr=1737309824 |
| 42:16.2 | 46 | [TCP Dup ACK 57#2] 41040 > 80 [ACK] Seq=150 Ack=13033 Win=61056 Len=0 TSval=1737309936 TSecr=58874868 SLE=5793 SRE=7241 SLE=14481 SRE=17377 |
| 42:17.0 | 46 | [TCP Spurious Retransmission] 80 > 41040 [ACK] Seq=8689 Ack=150 Win=15552 Len=1448 TSval=58874955 TSecr=1737309835 |
| 42:18.2 | 46 | [TCP Dup ACK 57#3] 41040 > 80 [ACK] Seq=150 Ack=13033 Win=61056 Len=0 TSval=1737310000 TSecr=58874868 SLE=8689 SRE=10137 SLE=14481 SRE=17377 |
| 42:19.0 | 47 | [TCP Fast Retransmission] Continuation |

**Figure 15**: Real-time IoT Device Malicious Dataset

The novelty in the proposed reinforcement learning based security framework is achieved by doing the following activities.

1) Implemented an acknowledgement-based malware detection technique between the Sender (IoT device) and the Receiver (Edge device).

2) Integrated checksum error detection technique [52] [53] to efficiently track all possible malicious data packets at the Edge device side.

Implemented a reinforcement learning based security policy that is based on taking actions based on the extraction of data packet features to distinguish between a malicious and non-malicious data packet.

### 3.6 Data Communication into Proposed Security Model

In the proposed security model, data communication is enhanced through a novel reinforcement learning (RL)-based approach, wherein an intelligent agent autonomously learns and adapts optimal strategies for securing the information exchange between IoT devices and Edge servers. The proposed RL model operates in a real-time dynamic environment, where the agent continuously observes communication behaviour, traffic patterns, and contextual features to identify potential anomalies or malicious activity. The agent receives feedback from the environment in the form of rewards or penalties, allowing it to iteratively refine its policy to maximise long-term security performance while minimising computational overhead. Furthermore, the RL agent not only detects irregularities but is also capable of triggering preventive actions, such as blocking suspicious packets or malicious data sources, thereby supporting proactive threat mitigation. The RL Agent is trained to select actions to detect anomalies in a real-time dynamic environment, optimising for both security and performance. This adaptive approach enables proactive threat mitigation, as the system evolves and improves through experience. By integrating the novel reinforcement learning model into the network layer, the model introduces a self-improving, intelligent framework that ensures secure, resilient, and efficient data transmission in increasingly complex cyber environments. We divided the communication task of sending a data packet from the IoT device to the Cloud server into several sub-tasks. The subtasks for our RL agent are visualised in **Figure 16**.

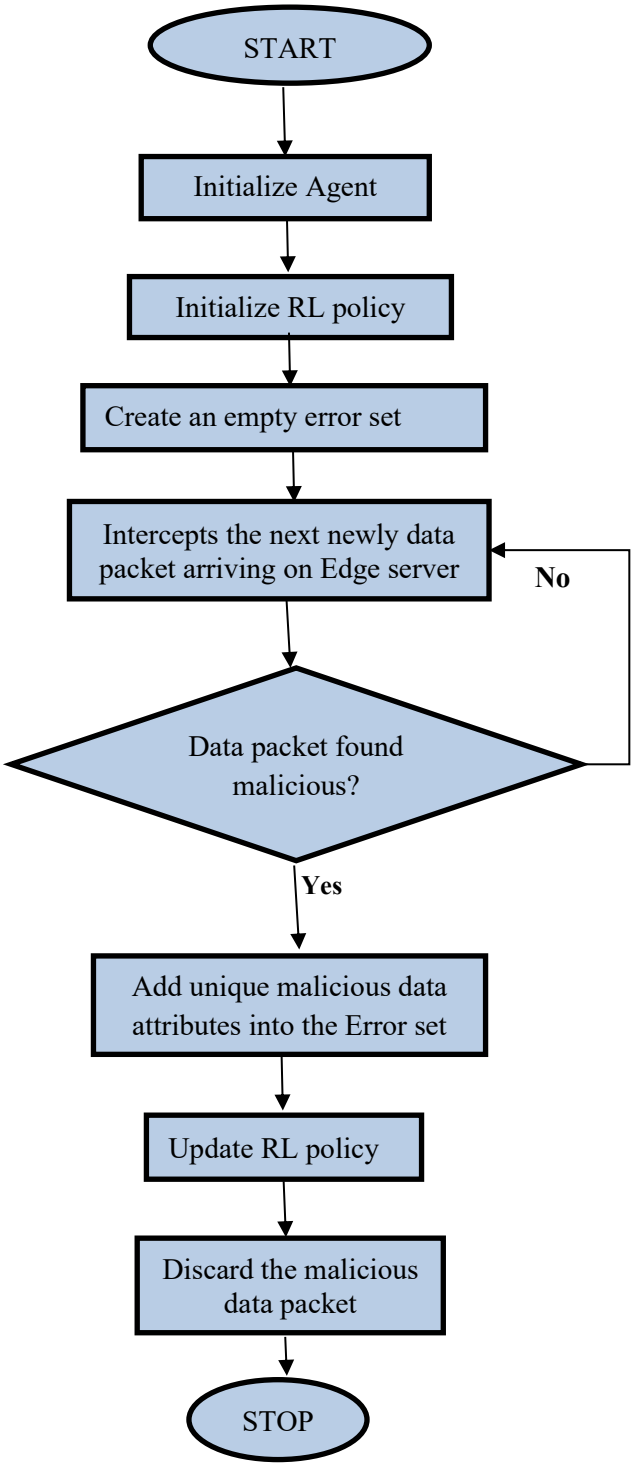



**Figure 16:** Flow-chart of the Proposed Security Method

## 3.7 Checksum-Based Error Detection Technique

We also have an additional integrated Checksum Error Detection Technique [52] [53] to empower the proposed security methodology to protect the data from malware attacks during transmission. The checksum is a simple and widely used error detection technique employed in digital networks and storage systems to ensure the integrity of transmitted or stored data. The core idea is to generate a small, fixed-length binary value (the checksum) based on the sum of the data bits. This checksum is sent along with the data to help the receiver verify whether the data has been altered during transmission. This error detection technique uses a Checksum Generator at the sender side, which processes the data and produces a compact, fixed-size binary value known as the checksum. This checksum is appended to the data and sent to the receiver. On the receiving end, a corresponding Checksum Checker recalculates the checksum from the received data and compares it to the transmitted checksum. If both match, the data is assumed to be error-free; otherwise, an error is detected. We used the 8-bit checksum method. Following this error detection technique, it is applied at two places.

1) **At the Sender side:**
    a) The data to be transferred is divided into segments of 8 bits.
    b) All these divided segments are added to get the resultant sum.
    c) The resultant sum is now complemented using 1's complement arithmetic.
    d) The final resultant sum value so obtained is called the checksum.
    e) Finally, the data that was to be transferred along with this checksum value is transmitted to the receiver.
2) **At the Receiver side:**
    a) The received data at the receiver side is divided into segments of 8 bits.
    b) All these segments are added along with the checksum value.
    c) The value so obtained is again 1's complemented, and the result is verified.

Now, at the final verification stage, if the result is zero, it indicates that no error occurred in the data during its transmission from the sender to the receiver. We allow

this data for further processing. But if the result is found to be non-zero, then the receiver assumes that the transmitted data has been disrupted during the transmission, and then the receiver discards the data.

## 3.8 Data Collection

As we have the very first research objective, "**To pre-process the data gathered from IoT devices on the edge layer**", for experimental purposes, we have collected IoT/Sensor data from various sources. These experimental data, as input, are categorized as follows:

1) **Simulation Dataset**: We have also used the network simulator tool to generate the test data for IoT-Edge-Cloud communication security. This dataset is generated during the simulation of the proposed approach using the NS 2.35 simulator [32] [33].
2) **Secondary Dataset**: The secondary dataset we used is a public IoT dataset available for researchers from IoT devices connected across the globe, which is the IoT-23 dataset [29]. This dataset was created by Stratosphere Lab (Czech Technical University) and focuses exclusively on IoT devices and their traffic patterns. It captures realistic, malicious, and benign traffic from IoT devices, such as smart cameras and smart home appliances. The IoT-23 [29] dataset better reflects actual IoT-Edge scenarios as compared to other existing IoT datasets, CICIDS2017 [125], KDD-99 [139], and UNSW-NB15 [126]. IoT-23 is superior because it is IoT-specific, attack-relevant, based on real device traffic, provides versatile data formats, and captures modern IoT malware behaviour. KDD-99, CICIDS2017, and UNSW-NB15 are still valuable, but they reflect more traditional enterprise network threats rather than IoT ecosystems [127]. The IoT-23 dataset contains twenty-three labelled network traffic captures representing both benign and malicious behaviours. These records were collected from IoT environments affected by various malware attacks between 2018 and 2019. It serves as a significant resource for research, as it reflects realistic IoT network

activity and offers a substantial volume of annotated malicious traffic data [134].

3) **Primary Dataset**: The primary dataset is collected from various sources, some of which are listed below.
    - Collected sensor data from our personal computer by using the tool HWiNFO [75].
    - Real-time datasets were gathered from the ESP-WROOM-32 [41] [42] IoT device. We took this primary data from ESP-WROOM-32, an IoT device connected with a DHT22 Temperature and Humidity sensor module [43] arriving on the Edge server. We collected room temperature and humidity data from the DHT22 sensor module, whereas the touch sensor data was collected using the ESP-WROOM-32 device's built-in touch sensor pin (GPIO4). **Figure 17** shows how the ESP32 and DHT22 are connected. **Figure 18**, and **Figure 19** describe the pin layouts of the DHT22 sensor and ESP-WROOM-32, respectively.

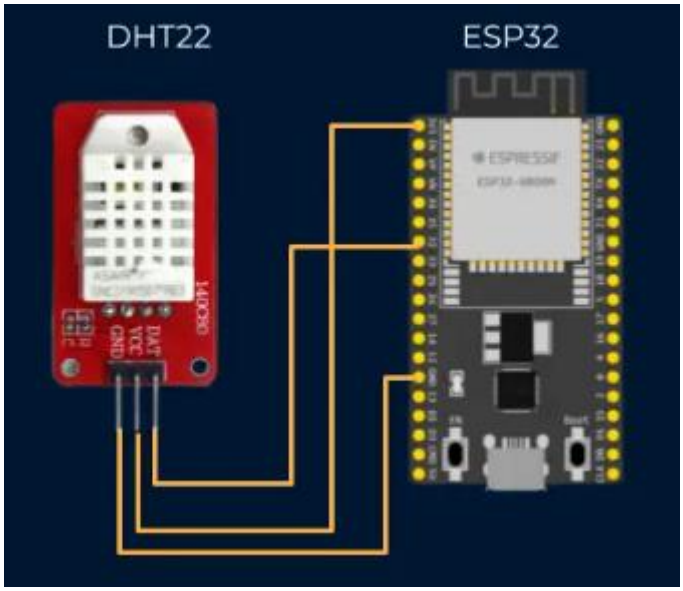



**Figure 17:** Connection between ESP-WROOM-32 and DHT22


**Figure 18:** DHT22 Pin Layout

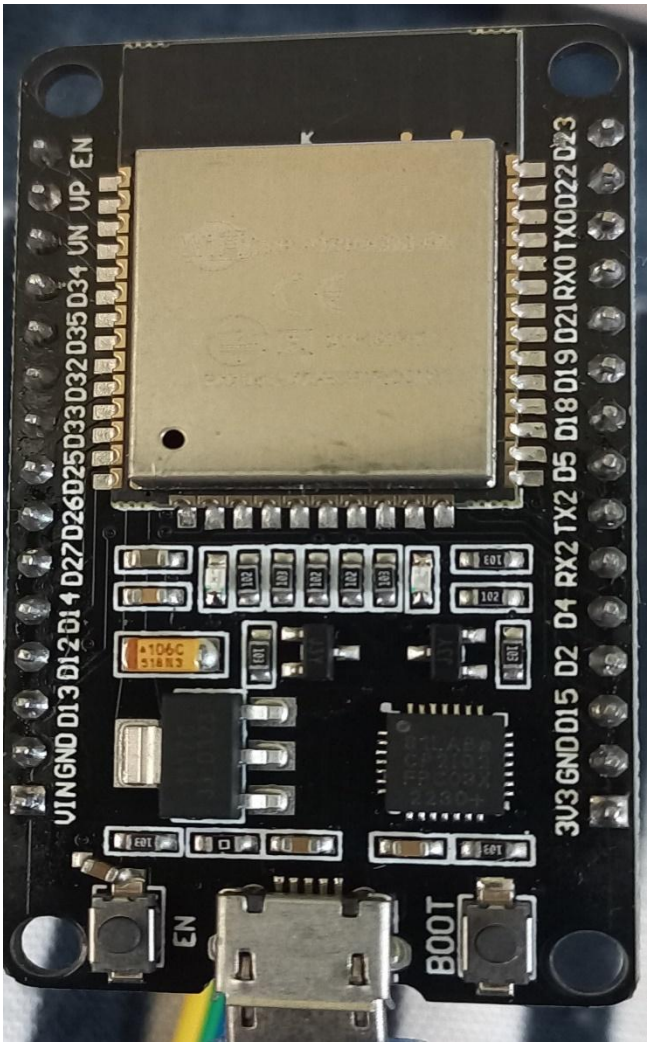



**Figure 19:** ESP-WROOM-32 Pin Layout

## 3.9 Hardware Tools Used for the Experiment

We used the following hardware tools to set up the experiment, carry out our experiment, and evaluate the proposed security method.

1) **ESP32-WROOM-32**: We used ESP32-WROOM-32 as an IoT device in our experiment, which is generally regarded as superior to other IoT devices because it features a more powerful dual-core processor, built-in Bluetooth connectivity, an in-built touch sensor, and enhanced versatility through additional GPIO pins, analog-to-digital converters (ADCs), and advanced low-power modes. These improvements make it suitable for a broad spectrum of applications, ranging from basic smart home systems to more demanding projects that require multitasking and wireless communication [137].

   The ESP32 microcontroller is often preferred over the Raspberry Pi for low-cost, energy-efficient, and real-time IoT applications. It integrates Wi-Fi and Bluetooth modules on-chip, consumes very low power, and supports deep sleep modes, making it ideal for battery-powered or remote IoT devices. Unlike the Raspberry Pi, which runs a full operating system (Linux-based) and requires higher power and boot time, the ESP32 executes real-time operations instantly through firmware-level control. Its dual-core processor and numerous GPIO pins allow flexible interfacing with sensors and actuators without the overhead of an operating system. Furthermore, the ESP32's compact size, low latency, and cost-effectiveness make it superior for edge-based sensing, home automation, and wireless communication projects where lightweight computation is needed [140].
2) **DHT22**: We used a DHT22 sensor module in our experiment, which is a low-cost digital sensor used for accurately measuring temperature and humidity in environmental monitoring applications. It integrates a capacitive humidity sensor and a thermistor to provide calibrated digital output signals via a single data line, making it simple to interface with microcontrollers like Arduino or ESP32. Compared to its predecessor, the DHT11, the DHT22 offers higher accuracy, a wider measurement range, and better long-term stability, making it

suitable for IoT-based weather stations, smart homes, and industrial automation systems [138].

## 3.10 Software Tools Used for the Experiment

We used the following software tools to collect data, carry out our experiment, and evaluate the experimental results.

1) **NS-2.35:** We started our experiment with the tool NS-2.35 [32] [33] as an experimental tool to implement, validate, and test our proposed methodology. It is a very powerful and popular open-source event-driven network simulation platform. It is written in C++ and Otcl. The simulation program is written in TCL Script. This network simulator tool provides support for protocols such as TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) at the network and transport layers. We used an Ad-hoc On-demand Distance Vector (AODV) [37] routing protocol to transfer data among virtual network simulation nodes. We adopted and used it due to its rich set of features and flexibility in the network simulation. It uses the GCC compiler to build the source code written by the network developer.
2) **Node-RED:** Node-RED is a flow-based programming tool, originally developed by the IBM Emerging Technology Services team and now a part of the OpenJS Foundation [47]. It provides a browser-based flow editor that makes it easy to wire together flows using the wide range of nodes in the palette. Once we completed the experiment with the NS-2.35 simulator, and obtained our desired and target results. Then, we used the MQTT protocol [31], which is a lightweight protocol for carrying out information using a "publish or subscribe" model. Node-RED is used to enable the MQTT broker that runs at the Edge server and allows data to communicate between IoT device (s) and the Cloud server. It makes a suitable choice for the Internet of Things to transfer data more easily. It is a programming tool for wiring together hardware devices to send data as publishers and receive data as subscribers [59] [86]. The agenda is to use the MQTT protocol to apply our proposed security approach in real-world IoT applications.

3) **Wireshark:** Wireshark, the most widely used open-source network protocol analyzer [45], plays a critical role in the packet-level inspection and analysis of communication data in networked environments. In this research, it is employed to capture and analyze network-layer datasets, specifically focusing on data packets transmitted by the ESP-WROOM-32 [41] [42], a popular microcontroller module with integrated Wi-Fi and Bluetooth capabilities. The tool enables the monitoring of both inbound and outbound traffic in real-time, providing comprehensive visibility into network activities, including source and destination IPs, port usage, protocol distributions, and packet sizes. Wireshark's ability to decode and dissect hundreds of protocols (such as TCP, UDP, HTTP, and MQTT) allows researchers to investigate the precise behavior of IoT devices under various network conditions.

   Furthermore, Wireshark supports the application of advanced display filters, custom capture rules, and statistical analysis tools, which aid in the categorization and extraction of relevant features for subsequent processing. This functionality is particularly useful for analyzing the lightweight and sometimes intermittent traffic patterns generated by IoT nodes [47] like ESP-WROOM-32, which often operate in constrained network environments. By leveraging Wireshark's capabilities, the captured traffic can be exported in various formats (e.g., .pcap, .csv) for offline analysis, feature engineering, and use in machine learning models designed for intrusion detection, anomaly detection, or energy-efficient communication protocol evaluation.

   Overall, Wireshark provides an essential foundation for empirical validation, enabling researchers to ensure accurate, secure, and optimized communication between IoT devices, Edge devices, and Cloud infrastructures. Its role is indispensable in understanding the network behavior of embedded systems, thus contributing significantly to the broader goals of IoT security, performance tuning, and protocol development.

4) **Arduino IDE:** We used open-source Arduino Software (IDE) to simplify the process of writing code and uploading it to microcontroller boards such as the ESP-WROOM-32. We used this IDE to collect real-time datasets from the ESP-WROOM-32 device. This software provides an easy-to-use environment for

creating programs, commonly referred to as sketches, which are written using a language based on C and C++. To upload a sketch to the ESP-WROOM-32 board, the board must first be connected to the computer running the Arduino IDE. The sketch, which is saved with a '**.ino**' file extension, is then compiled and uploaded to the microcontroller via a USB connection. The IDE offers features like syntax highlighting, error detection, and a built-in serial monitor for debugging, making it accessible for beginners and efficient for experienced developers. This approach supports rapid prototyping, allowing developers to implement and test functionalities such as reading sensor data, controlling devices, and integrating IoT applications quickly [49].

5) **Python Programming Language:** Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. It emphasizes clear syntax and code structure, making it an excellent choice for beginners while remaining a powerful tool for researchers. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming. It comes with an extensive standard library and a vast ecosystem of third-party packages, enabling applications in web development, data analysis, artificial intelligence, scientific computing, automation, and more. Its portability and active community support have made Python one of the most popular and widely used programming languages in the world today [48].
6) **HWiNFO:** HWiNFO is a powerful and comprehensive hardware monitoring tool designed for Windows platforms. It provides detailed real-time diagnostics for nearly every hardware component within a computer system, including CPU, GPU, RAM, storage devices, motherboard chipsets, and network interfaces. This software monitors key parameters such as temperature, voltage, frequency, power consumption, fan speeds, and thermal performance, offering granular insights into the health and status of the system. Its real-time updating capability makes it exceptionally suitable for dynamic system diagnostics, stress testing, and performance benchmarking in both consumer and enterprise environments. It is widely used by researchers and developers to collect baseline hardware performance metrics and monitor system load under various

computational conditions. In cybersecurity or edge computing experiments, for example, it can be employed to assess resource consumption of edge devices or server nodes during malware detection, intrusion simulation, or reinforcement learning agent training. The tool supports custom logging, system alerts, and integration with third-party dashboards, making it highly extensible for professional use [75].

7) **Eclipse IDE:** In the implementation of our proposed security model, we have also utilized Eclipse IDE [50], a widely adopted, open-source Integrated Development Environment (IDE) particularly suited for Java development. Eclipse offers a powerful and flexible framework for building applications and is well-known for its modular architecture, extensive plugin ecosystem, and active community support. These features made it an ideal environment for developing and testing components of our IoT-Edge security framework, especially when integrating multiple functionalities such as message handling, data classification, and real-time monitoring.
8) **Java Programming Language:** To develop the incoming data listener at the Edge server, we used the Java programming language, a high-level, object-oriented language originally introduced by Sun Microsystems in 1995 [51]. Java's platform independence, strong memory management, and built-in security mechanisms make it a reliable choice for network-based and multithreaded application development, which is crucial for edge computing scenarios. Java's mature libraries and real-time execution capabilities allowed us to implement a responsive and secure listener module that continuously monitors data packets arriving from IoT devices, evaluates them using RL-based security rules, and interacts with the RL agent deployed on the Edge server. This integration ensures that the Edge server can effectively handle large volumes of data while maintaining system integrity and security in dynamic IoT environments [46].

## 3.11 Summary

In this chapter, we presented a comprehensive overview of the research plan adopted for the design and implementation of our proposed novel reinforcement learning-based

security model tailored for IoT-Edge computing environments. The chapter begins with an in-depth discussion of the research scope, focusing on the critical security challenges in modern IoT-Edge ecosystems, such as vulnerability to DDoS attacks, limited computational resources of edge devices, and the need for real-time threat detection. We also identified the inherent limitations of the proposed model, including potential scalability constraints and dependency on network topology and traffic variability. Subsequently, the research objectives were represented, which include enhancing the resilience of IoT-Edge systems against network-layer attacks, reducing end-to-end delay while maintaining a high packet delivery ratio (PDR), and leveraging reinforcement learning to adaptively secure dynamic network conditions. These objectives serve as the foundational goals for the research and guide the algorithm design and experimental validation. Further, we elaborated on the datasets used to validate and benchmark the performance of the proposed model. These include both publicly available datasets, such as the IoT-23 dataset [29] collected and labelled by the Stratosphere Laboratory, as well as real-time data captured using hardware implementations. For real-world experimentation, the ESP-WROOM-32 microcontroller was interfaced with sensors like the DHT22 temperature and humidity module [43] to simulate authentic IoT device behaviour and generate environmental data for security analysis.

Lastly, we detailed the software tools, simulation platforms, and programming environments employed during the implementation phase of our proposed security framework. The NS-2.35 network simulator served as the primary simulation platform for conducting network-level experiments, offering robust support for widely used communication protocols such as AODV, TCP, and UDP, which were critical in modelling realistic IoT-Edge communication scenarios. To facilitate Edge-based data handling, JAVA [46] and Python [48] were utilized for implementing the reinforcement learning algorithms and processing experimental datasets. Furthermore, MQTT acted as the lightweight messaging protocol for efficient device-to-Edge and Edge-to-Cloud communication, while Node-RED [47] was leveraged as a visual programming tool to integrate and orchestrate hardware devices, APIs, and services in real-time workflows. The combined use of these platforms provided a comprehensive development and testing environment, ensuring that our proposed security protocol was rigorously validated under both controlled simulation conditions and practical, real-world IoT deployments.

# CHAPTER 4

# PROPOSED METHODOLOGY

We have designed and implemented a Novel Reinforcement Learning security method to prevent malicious attacks on IoT-Edge computation. Our research work consists of

1) Generate IoT datasets from the simulation with non-malicious and malicious network settings.
2) Collect a publicly available dataset, which is IoT-23 [29], which is a compromised dataset with some anomaly patterns.
3) Refine the unusual malicious data patterns in the IoT network simulation based on the anomaly pattern found in IoT-23 [29].
4) Once the malicious data pattern is finalised, then by using it, train the proposed IoT-Edge security model.
5) Discard the data packets with malicious patterns from the underlying network traffic.
6) Finally, allow only non-malicious data packets to be preprocessed on the edge server, which can then be transferred to the cloud server for global access by users.

We ensure that the Edge server is protected from all types of network layer attacks, especially DDoS attacks. Many researchers have used the reinforcement learning approach for more than a decade to provide a solution for their chosen research problem. Still, none can provide the needed security without the data being compromised [9] [10] [11]. Hence, at present, attackers can exploit these shortcomings and attempt to access sensitive information from IoT devices. On the other hand, Reinforcement learning is a widely used machine learning method to detect the behaviour of any malicious node. So, we aimed to design and implement a novel reinforcement learning method that deals with how an RL agent in an IoT-Edge-Cloud environment should act to provide the optimal solution for the security of the sensor data. There must be a security policy that should keep updating based on the nature and volume of the attack data. In our secure reinforcement learning environment, the RL agent installed on the Edge server performs

some action and updates the data security policy whenever needed. With our proposed security approach, if, by any chance, any course of action in an RL-enabled environment is made by a malicious Edge node, then its action will be determined as unauthorised and will not bring any significant result. It results in the state change from "Unknown" to "Unauthenticated". There will be a reward, like providing permission to access data from an IoT device in case of a state change to "Authenticate"; otherwise, there will be a penalty for a state change to "Unauthenticated" and consequently "Blocked" for any time in the future. Hence, it will block the malicious entity that requests it at all times in the future. The proposed RL-based security interface over the IoT-Edge environment is depicted in **Figure 20**, which consistently prevents malware attacks on IoT-Edge dynamics.

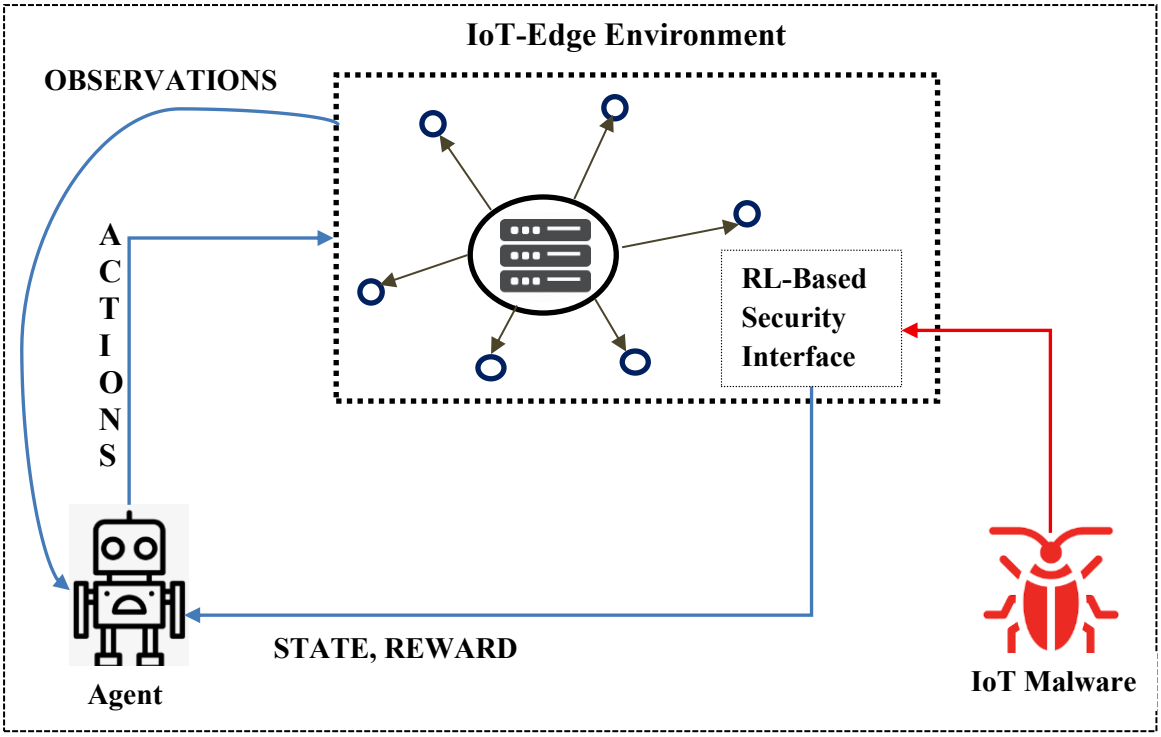



**Figure 20**: RL-Based Security Interface in IoT-Edge Computing

In a large-scale global IoT network, the dynamic nature of device participation significantly influences system behaviour. IoT devices frequently join and leave the network based on user-specific operational cycles, such as time-based triggers, environmental conditions, or task completion. This fluctuation results in variable

workloads on the Edge servers, ranging from heavy processing loads during peak device activity to periods of light or idle activity when fewer devices are active. A single Edge server may be tasked with managing communications from multiple IoT devices across different network segments. It may even interface with multiple network channels to facilitate the offloading and routing of data from heterogeneous IoT ecosystems. This dynamic and distributed environment, while improving scalability and latency performance, also introduces increased attack surfaces. The inconsistent presence of devices and irregular traffic patterns can mask malicious behaviours, making it easier for attackers to exploit system vulnerabilities. For instance, attackers may time their DDoS attacks or spoofing attempts during periods of network transition, or when Edge servers are under heavy stress and more prone to delayed detection and response. Such vulnerabilities are often amplified by resource constraints on Edge nodes and a lack of uniform security policies across distributed devices and communication protocols [76] [77].

We have proposed an IoT security method based on edge computation, which implements a security layer over a large, heterogeneous network consisting of multiple IoT devices communicating with multiple Edge devices, as shown in **Figure 21**. The security layer uses the data packet features extraction technique to distinguish between a non-malicious and a malicious data packet. The proposed security allows only non-malicious data packets (B) to be pre-processed on the Edge server and then allowed to move to the Cloud server, whereas malicious data packets (A) are discarded by the proposed security layer.

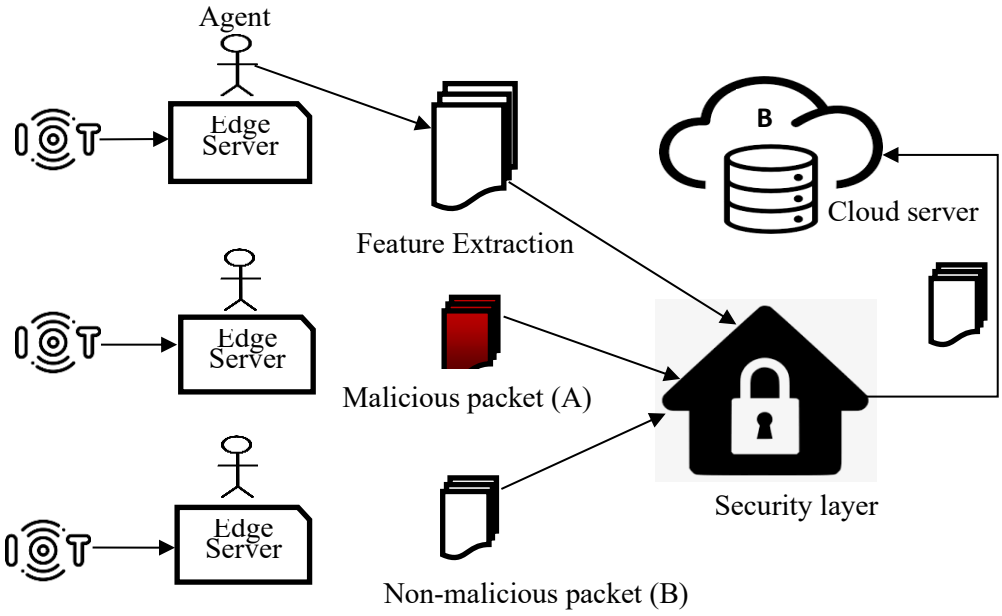



**Figure 21**: Data Protection in IoT Edge Computation

Data Protection in IoT Edge Computation is a critical concern due to the distributed nature of the architecture and the sensitivity of the data involved. Edge computing pushes data processing closer to IoT devices, which improves performance and reduces latency but also introduces new security and privacy risks. Unlike centralised cloud systems, edge servers are distributed and more exposed to physical and cyber threats. IoT devices and Edge servers often have limited computing, memory, and power, restricting the implementation of efficient intrusion detection systems [89] [90].

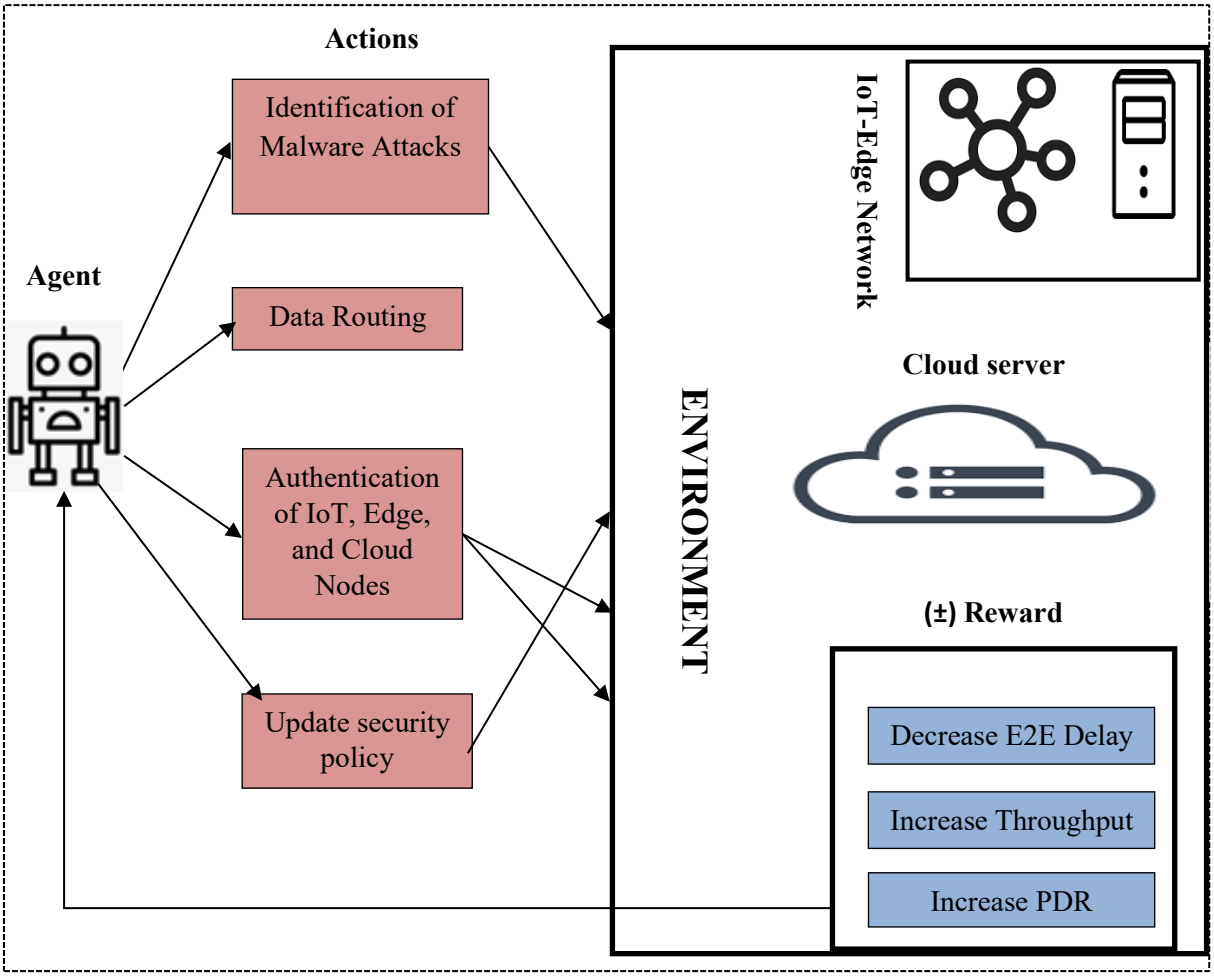



**Figure 22:** Proposed Research Model Components

We have trained the model to trace malicious data packets within network traffic intelligently. Each action the RL agent takes yields a reward, which is described in **Figure 22**. We have observed that our Reinforcement Learning (RL) based policy is highly efficient in detecting and preventing security attacks over the network, especially when DDOS attacks are imposed. In a DDOS attack, attackers are distributed over the network. Hence, mitigating one attacker is not sufficient, and predetermining the next possible attacker's location is far more important [36] [74]. With our literature review, the existing RL-based security approaches are not in such a dynamic environment.

## 4.1 Novelty of the Proposed Research Work

The proposed method for securing IoT devices in Edge computing has the following novelty implemented by us.

1) Implemented the novel reinforcement learning method using a message-driven approach to secure the IoT-Edge computation, which we termed "Message Driven-based Reinforcement Learning" (MDRL) security method for IoT-Edge computing.
2) To validate and extend the capabilities of the proposed novel reinforcement learning method to work perfectly under real-time constraints and conditions, we extended the proposed reinforcement learning-based security method by using the Adaptive Epsilon Greedy Reinforcement Learning (AEGRL) approach, which is an extension of the traditional Epsilon (Ɛ) greedy reinforcement learning method. The proposed method now works efficiently for both static and dynamic environments.
3) The proposed novel reinforcement learning security method extracts the data packet metadata and also uses the Checksum Error Detection Technique to distinguish between malicious and non-malicious ones.
4) The proposed novel reinforcement method uses a malware detection classifier, which accurately classifies the malware data packets based on different abnormalities in their metadata. The proposed method maintains the efficiency, resilience, and consistency of the IoT data communication in terms of packet delivery ratio (PDR), average throughput, and end-to-end (E2E) delay.
5) The proposed security method is validated and strengthened by using Simulation, Public, and Real-time In-House IoT device datasets.
6) The public dataset IoT-23 [29] is used to validate the proposed security model, and the proposed model outperforms the existing similar security models using the same dataset implemented by Sudheera et al. (2021) [135], and Ullah et al. (2021) [136].
7) The proposed security method outperforms benchmark security models, including Deep Reinforcement Learning based machine learning models [35] [38], Random Forest (RF) based Machine learning models [129], Support

Vector Machine (SVM) [34] [130], and other similar Reinforcement Learning based models [16] [19].

## 4.2 Comparison Between the Proposed Method and Benchmark Methods

The proposed IoT-Edge security method outperforms benchmark methods in several aspects. **Table 4** represents the summarised comparison of the proposed security model with existing benchmark security methods.

**Table 4**: Comparison of the Proposed Model with Benchmark Models

| Existing Security Method | Attack Prevention Model | Performance Metrics | Experiment Environment |
|---|---|---|---|
| Proposed Model | MDRL with Adaptive Epsilon Greedy | Network Security, Scalability, Adaptivity, Consistency | ➢ Simulated environment<br>➢ Public IoT-23 Dataset-based [29] Communication environment<br>➢ Real-time In-house IoT Device communication environment |
| Geetha et al. (2024) [130] | Support Vector Machine (SVM) | Network Security | ➢ Public datasets UNSW-NB15 [126] and KDD-99 [139] based communication environment |
| Louai A. Maghrabi (2024) [129] | Random Forest (RF) | Network Security | Public UNSW-NB15 Dataset-based Communication Environment |
| Zhang et al. (2023) [38] | Deep Reinforcement Learning | Network Security, | Simulated Environment |

| | | | |
|---|---|---|---|
| | | Resource Allocation | |
| Ullah et al. (2021) [136] | Convolutional Neural Network (CNN) | Network Security | Public IoT-23 Dataset-based Communication Environment |
| Sudheera et al. (2021) [135] | Machine Learning-based Classifiers | Network Security | Public IoT-23 Dataset-based Communication Environment |
| Ioannou et al. (2019) [34] | Support Vector Machine (SVM) | Network Security, Scalability | Public KDD-99 Dataset-based Communication Environment |
| Khatun et al. (2019) [35] | Artificial Neural Network (ANN) | Network Security | Real-time In-house IoT Device Communication Environment |
| Xiao et al. (2016) [19] | Reinforcement Learning | Network Security | ➢ Simulated Environment<br>➢ Real-time In-house IoT Device Communication Environment |
| Malialis et al. (2015) [16] | Coordinated Team Learning (CTL) | Network Security, Scalability, Adaptivity | Simulated Environment |

## 4.2 Network Simulator-Based Security Method

In the Network Simulator-based method, we utilized the NS-2.35 simulator [31] [32], a widely used, open-source, event-driven simulation tool that enables the detailed modelling of communication protocols and network behaviour. This simulator supports various networking protocols, including TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) [116] at the transport and network layers. The objective

was to assess the model's effectiveness in detecting and mitigating attacks under varying conditions. To evaluate the proposed model's scalability, the number of simulation nodes was varied systematically during the experiment. The routing between nodes was managed using the Ad-hoc On-Demand Distance Vector (AODV) routing protocol [37], chosen for its dynamic route discovery capabilities and suitability for mobile and ad hoc network scenarios. The simulation code was written in TCL (Tool Command Language) and compiled using the GCC compiler, allowing for efficient testing and debugging. **Table 5** presents the detailed network configuration data used in designing our malicious and non-malicious networks.

**Table 5**: Network Configuration Data used in Non-Malicious and Malicious Networks

| Option | Available Values | Default Value | Selected Value for Experiment | Remarks |
|---|---|---|---|---|
| addressType | flat, hierarchical | Flat | flat | IP Address Type |
| MPLS | ON, OFF | OFF | OFF | Multi-Protocol Label Switching |
| wiredRouting | ON, OFF | OFF | OFF | Wired Routing Status |
| llType | LL, LL/Sat | Blank ("") | LL | Link Layer |
| macType | Mac/802_11, Mac/Csma/Ca, Mac/Sat, Mac/Sat/UnslottedAloha, Mac/Tdma | Blank ("") | Mac/802_11 | Medium Access Control |
| ifqType | Queue/DropTail, Queue/DropTail/PriQueue | Blank ("") | Queue/DropTail/PriQueue | Interface Queue type |
| phyType | Phy/WirelessPhy, Phy/Sat | Blank ("") | Phy/WirelessPhy | Physical Layer Type |

| | | | | |
|---|---|---|---|---|
| adhocRouting | DIFFUSION/RATE, DIFFUSION/PROB, DSDV, DSR, FLOODING, OMNIMCAST, AODV, TORA, PUMA | Blank (“”) | AODV | Adhoc Routing Protocol |
| propType | Propagation/ TwoRayGround, Propagation/ Shadowing | Blank (“”) | Propagation/ TwoRayGround | Radio Propagation Model |
| antType | Antenna/ OmniAntenna | Blank (“”) | Antenna/OmniAntenna | Antenna Type |
| Channel | Channel/ WirelessChannel, Channel/Sat | Blank (“”) | Channel/WirelessChannel | Channel to be used |
| mobileIP | ON, OFF | OFF | OFF | IP for Mobile |
| energyModel | EnergyModel | Blank (“”) | EnergyModel | Energy Model to be enabled or not |
| agentTrace | ON, OFF | OFF | ON | Enable Agent Tracing |
| routerTrace | ON, OFF | OFF | ON | Enable Router Tracing |
| macTrace | ON, OFF | OFF | OFF | Enable MAC Tracing |
| movementTrace | ON, OFF | OFF | ON | Enable Movement Tracing |

| | | | | |
|---|---|---|---|---|
| topoInstance | <topology file> | Blank (“”) | load_flatgrid | Network Topography |

### 4.2.1 Message Driven Reinforcement Learning (MDRL) Security for IoT-Edge Computing

By utilizing the NS-2.35 simulator experiment setup, we have developed and implemented a novel reinforcement learning-based security layer protocol, specifically designed for the dynamic nature of IoT-Edge computing environments. The proposed method, termed “Message Driven Reinforcement Learning” (MDRL) Security for IoT-Edge Computing, aims to enhance network resilience by intelligently detecting and mitigating malicious behaviours through continuous learning from environmental interactions. MDRL introduces an intelligent, adaptive mechanism that reinforces secure communication paths based on real-time message exchanges and network conditions, thereby providing a robust defence against evolving cyber threats in decentralized IoT infrastructures. The proposed algorithm introduces a custom security protocol implemented over the Ad-hoc On-demand Distance Vector (AODV) routing protocol, specifically designed for IoT-Edge computing environments. The network layer configuration is initialized through a separate configuration file. **Figure 23** is the screenshot of the network layer configuration setting file used in the experiment. We have used the User Datagram Protocol (UDP), one of the core communication protocols of the Internet protocol suite, to send messages as data packets [116]. The UDP communication-based network layer has a packet size limit of 1500 bytes. The data packets generated in the simulation are 40 bytes in size. The data packets transmission rate was set to 0.25 MB per second. These Application configurations in the NS-2.35 Simulator are listed in **Table 6**.

```
#====================================
#         Agents Definition
#====================================
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0
set null1 [new Agent/Null]
$ns attach-agent $n(3) $null1
$ns connect $udp0 $null1
# $udp0 set packetSize_ 1500

#====================================
#         Applications Definition
#====================================
#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 40
$cbr0 set rate_ 0.25Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 100.0 "$cbr0 stop"
```

**Figure 23:** Network Layer Configuration of the Simulator

**Table 6**: Application Configurations in NS-2.35 Simulator

| **Application Settings** | **Available Values** |
|---|---|
| Network Layer Protocol | AODV |
| Transport Layer Protocol | UDP |
| Type of Network Traffic | Application/Traffic/CBR |
| Maximum Packet Size | 150 bytes |
| Packets Transmission Rate | 0.25 Megabytes |
| Total No. of Nodes | 25, 50, 75, 100 |
| Malicious Nodes | 4 |
| Simulation Time | 1800 sec |

**Table 6** represents the following application configuration settings, which are applied in the NS-2.35 simulator-based communication of IoT-Edge-Cloud communication [32] [33] [37].

1) **AODV** is a routing protocol at the network layer, responsible for finding paths between nodes in a mobile ad hoc network (MANET).

**2) UDP** is a transport layer protocol that provides connectionless data delivery services to applications, which can then use AODV to establish those paths.

**3) Application/Traffic/CBR** represents a Constant Bit Rate (CBR) traffic source, a built-in application used to generate predictable network traffic for simulations.

The following components are implemented in the MDRL methodology.

1) **SNT (Status Notification Token)**: This header is used to broadcast local node status to neighbouring agents or edge nodes. It includes real-time metrics such as Packet metadata, CPU and Memory usage, Transmission details, and Anomaly score derived from a lightweight intrusion detection system. The SNT acts as a proactive signal for detecting early signs of suspicious traffic surges.

2) **ACK (Acknowledgment Token)**: The ACK header is used to confirm the reception and validity of the SNT from neighbouring nodes. It serves as a trust handshake that allows the reinforcement learning agent to build a situational awareness graph, identify unresponsive or malicious peers, and adjust its policy accordingly.

3) **Message Data Buffer (MDB)**: Each data packets transmitted by the IoT device are first sent for temporary storage into a Message Data Buffer (MDB) until it is safely received on the Edge Processing Unit (EPU) at the Edge server. The benefit of having MDB is to retransmit the data packet from MDB to EPU when the same packet is dropped/lost before reaching EPU due to security attacks on the Edge server.

4) **Reinforcement Learning Agent (RL Agent)**: A Reinforcement Learning Agent (RL Agent) is implemented to continuously monitor the data packets communicated from the IoT device to the Edge server. It observes the IoT-Edge computing environment and takes the necessary actions on the data packets to safeguard them from external security attacks.

MDRL is proposed for highly dynamic and distributed network scenarios, making it particularly effective against Distributed Denial-of-Service (DDoS) attacks, which typically originate from multiple malicious sources within an IoT infrastructure. To effectively detect and counteract such attacks, MDRL utilizes a dual-header message

mechanism comprising: 1) SNT (Status Notification Token), and 2) ACK (Acknowledgement Token). The functioning of the MDRL security layer on the Edge server is shown in **Figure 24**.

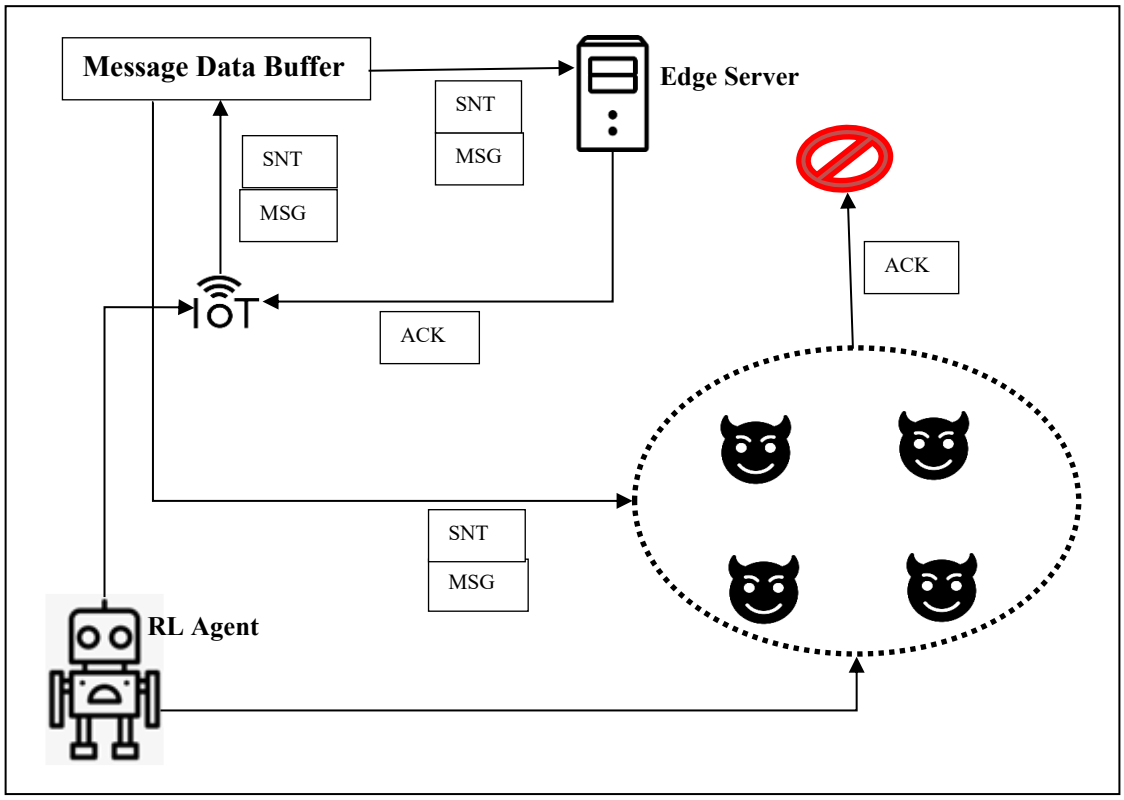



**Figure 24:** MDRL Security Layer on IoT-Edge Computation

These headers enable secure tracking and acknowledgement of transmitted data packets. To enhance reliability and responsiveness, the source node maintains a backup of the transmitted data packet. In case a malicious node intercepts or drops the packet and falsely acknowledges its receipt, the source node can quickly re-route and resend the original packet via a different, more trustworthy path. This is achieved by identifying an alternate benign neighbour node through reinforcement learning. The core of MDRL is powered by a Q-learning algorithm, wherein the agent, which is an intelligent decision-making entity embedded within the protocol, learns optimal routing strategies. It continuously updates its policy based on environmental feedback, guiding data packets securely from the IoT source node to the cloud-based destination node, thereby ensuring data integrity and enhancing the robustness of the IoT-Edge network against DoS/DDoS/Adversarial threats.

```
 1 s 1.000000000 _0_ AGT  --- 0 cbr 40 [0 0 0 0] ------- [0:0 3:0 32 0] [0] 0 0
 2 r 1.000000000 _0_ RTR  --- 0 cbr 40 [0 0 0 0] ------- [0:0 3:0 32 0] [0] 0 0
 3 s 1.000000000 _0_ RTR  --- 0 AODV 48 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 1 1 [3 0] [0 4]]
   (REQUEST)
 4 s 1.001280000 _0_ AGT  --- 1 cbr 40 [0 0 0 0] ------- [0:0 3:0 32 0] [1] 0 0
 5 r 1.001280000 _0_ RTR  --- 1 cbr 40 [0 0 0 0] ------- [0:0 3:0 32 0] [1] 0 0
 6 r 1.001408085 _28_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [3
   0] [0 4]] (REQUEST)
 7 s 1.001408085 _28_ RTR  --- 0 AODV 44 [0 0 0 0] ------- [28:255 0:255 30 0] [0x4 1 [3 4]
   10.000000] (REPLY)
 8 r 1.001408250 _43_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [3
   0] [0 4]] (REQUEST)
 9 s 1.001408250 _43_ RTR  --- 0 AODV 44 [0 0 0 0] ------- [43:255 0:255 30 0] [0x4 1 [3 4]
   10.000000] (REPLY)
10 r 1.001408311 _23_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [3
   0] [0 4]] (REQUEST)
11 s 1.001408311 _23_ RTR  --- 0 AODV 44 [0 0 0 0] ------- [23:255 0:255 30 0] [0x4 1 [3 4]
   10.000000] (REPLY)
12 r 1.001408332 _10_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [3
   0] [0 4]] (REQUEST)
13 s 1.001408332 _10_ RTR  --- 0 AODV 44 [0 0 0 0] ------- [10:255 0:255 30 0] [0x4 1 [3 4]
   10.000000] (REPLY)
14 r 1.001408406 _12_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [3
   0] [0 4]] (REQUEST)
15 s 1.001408406 _12_ RTR  --- 0 AODV 44 [0 0 0 0] ------- [12:255 0:255 30 0] [0x4 1 [3 4]
   10.000000] (REPLY)
16 r 1.001408457 _46_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [3
   0] [0 4]] (REQUEST)
17 s 1.001408457 _46_ RTR  --- 0 AODV 44 [0 0 0 0] ------- [46:255 0:255 30 0] [0x4 1 [3 4]
   10.000000] (REPLY)
18 r 1.001408479 _27_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [3
```

**Figure 25:** NS-2.35 Simulation Data Trace File

We have published the research work related to the MDRL method in the Springer Nature Publishing journal [4]. The proposed model, MDRL, was validated using the simulation datasets. This simulation dataset is accumulated in a separate trace file. The content of the trace file describes the movement of data packets during the simulation time. **Figure 25** is a screenshot of the network simulator trace file. Below is one of the simulation data points in the trace file.

"**s 1.234240000 _0_ RTR  --- 183 cbr 60 [0 0 0 0] ------- [0:0 3:0 30 12] [183] 0 0**"

It has the following meaningful components.

- s: Means "Send"
- 1.234240000: Timestamp
- _0_:  Node id
- RTR: Means "Request to Routing"
- AGT: Agent
- IFQ : Interface Priority Queue
- 183:  Id of the packet
- cbr: CBR agent (Constant Bit Rate)

- 60: size of the Header
- [0 0 0 0]: MAC detail
- [0:0 3:0 30 12]: Source IP address
    - 0 means Source IP Address 0.0.0.0
    - 0 means port number 0
    - 3 means Destination IP Address 0.0.0.3
    - 0 means port number 0
    - TTL: 30
    - Next-Hop: 12
- 183: ID of the packet
- 0: Sequence number
- 0: Acknowledgement number

MDRL is specifically tailored to operate within highly dynamic and heterogeneous network environments, such as those commonly found in IoT-Edge infrastructures. It is particularly effective against Distributed Denial-of-Service (DDoS) attacks, where malicious traffic originates from multiple, dispersed sources with the intent to overwhelm network resources and disrupt legitimate communication. In the context of DDoS attacks targeting IoT networks, multiple attacker nodes can simultaneously launch coordinated packet floods, making detection and mitigation more challenging. MDRL addresses this by continuously adapting its policy through regular interaction with the environment, learning to distinguish normal communication patterns from anomalous behaviours. For simulation purposes, the network topology and the behaviour of each node are defined in a dedicated scenario file. This file includes the spatial coordinates and roles of all nodes within the simulation environment. Specifically, it categorizes the nodes into three primary groups: 1) IoT nodes, 2) Edge nodes, and 3) Attacker nodes. **Figure 26** is the scenario file used in the Simulation-based experiment. The scenario file defines the total number of nodes and their coordinates on the simulator platform. To identify those sources, we implemented our

security protocol over such a network where two types of message headers are used: **1)** SNT, and **2)** ACK.

```
#=====================================
#         Nodes Definition
#=====================================
#Create mobile nodes
set n0 [$ns node]
$n0 set X_ 99
$n0 set Y_ 299
$n0 set Z_ 0.0
$ns initial_node_pos $n0 40
set n1 [$ns node]
$n1 set X_ 209
$n1 set Y_ 337
$n1 set Z_ 0.0
$ns initial_node_pos $n1 40
set n2 [$ns node]
$n2 set X_ 499
$n2 set Y_ 298
$n2 set Z_ 0.0
$ns initial_node_pos $n2 40
set n3 [$ns node]
$n3 set X_ 700
$n3 set Y_ 299
$n3 set Z_ 0.0
$ns initial_node_pos $n3 40
set n4 [$ns node]
$n4 set X_ 475
$n4 set Y_ 203
$n4 set Z_ 0.0
$ns initial_node_pos $n4 40
set n5 [$ns node]
$n5 set X_ 599
$n5 set Y_ 350
$n5 set Z_ 0.0
$ns initial_node_pos $n5 40
```

**Figure 26**: Scenario configuration of Network Simulator

**Figure 27**, and **Figure 28** show the simulated non-malicious and malicious networks, respectively.

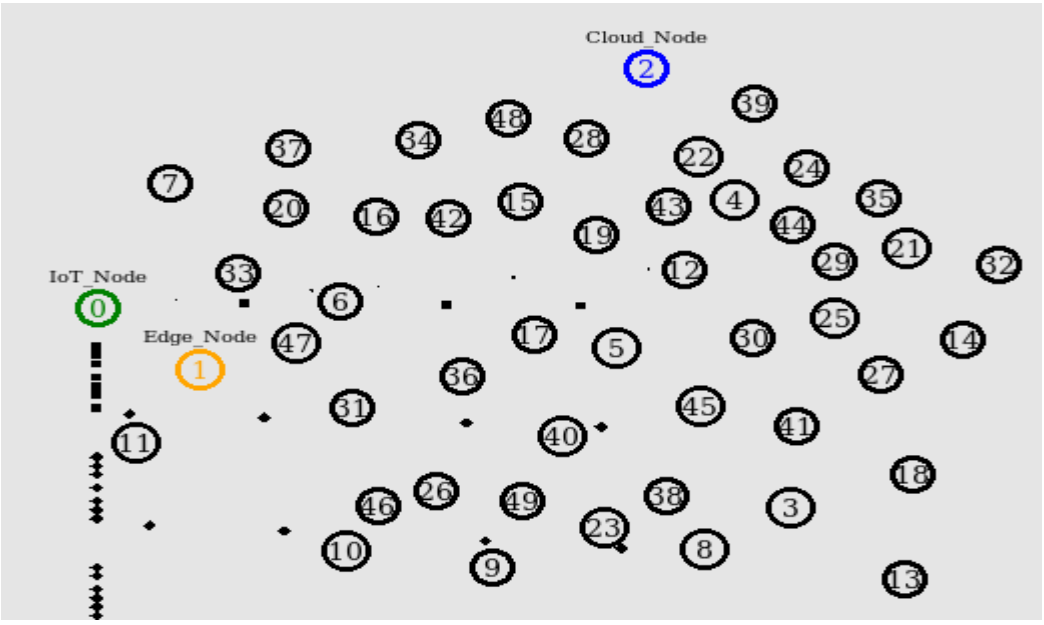



**Figure 27:** Simulated Non-malicious Network Using NS-2.35 Simulator

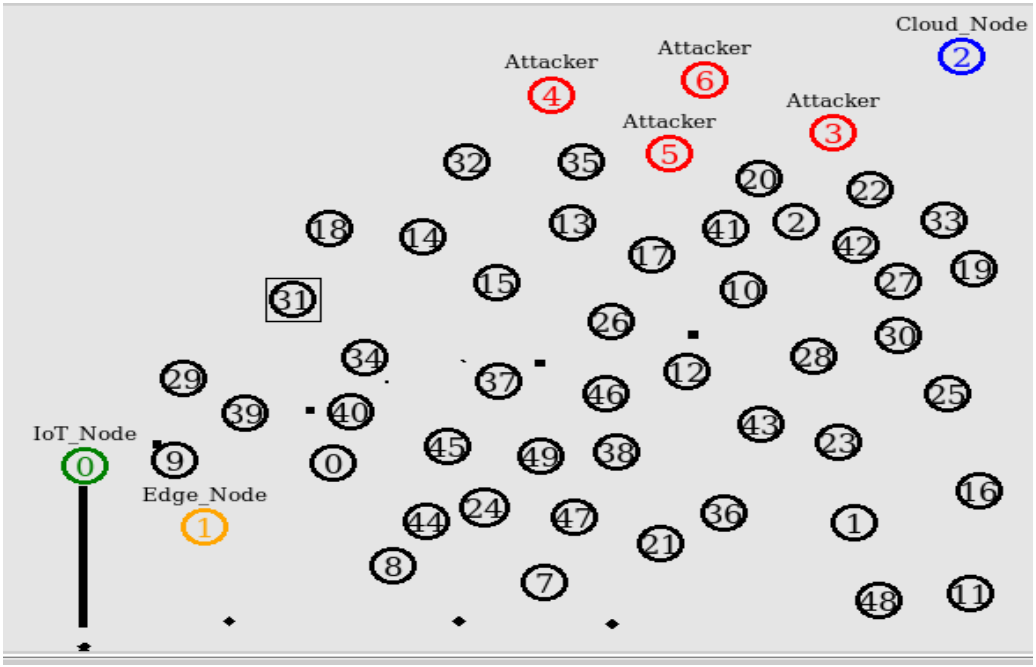



**Figure 28:** Simulated Malicious Network Using NS-2.35 Simulator

We are keeping a copy of the data packet on the source node to boost the speed and performance, so that if a malicious receiver node destroys the data packet, then the source node acknowledges it, and then the source node will first identify the most suitable next benign node, and then resend the same data packet. In our proposed method, the agent is a programming thread that implements the method using the Q-learning algorithm [55].

The following criteria are used to create a malicious data traffic configuration in the simulated IoT-Edge-Cloud network.

1) Implemented the attack space with four attacker nodes similar to an ideal DDoS Attack scenario, where attackers collectively generate malicious traffic of 40% from the overall network traffic [16] [131] [132] [133].
2) In our setup, each attacker node contributed (1/10) of the aggregate network traffic. For this controlled attack intensity, we fixed the ratio of malicious communication links to benign communication links at 1:4. Each attacker node establishes communication links with multiple benign nodes to send malicious data packets to them at the same time.

3) Started attack at low levels, but linearly vary the amount of attack from each attacker till their maximum attack capacity.

The proposed method determines how to traverse a data packet through a secure path from the source node (IoT device) to the destination node (Edge server). It is also shown in **Figure 29,** how the data packets traverse in the message-driven network topology. To give an overview and understanding of our approach, the simulated network has the nodes that are created as below.

1) Node (**0**) represents an IoT device that captures the data from the surroundings for the intended purpose.
2) Node (**1**) represents an Edge server node with which the IoT device communicates directly and typically presents at the IoT premise.
3) Node (**2**) represents a Cloud server node in the underlying network, which is the final destination of the data to provide access to the data.
4) Node (**3**), Node (**4**), Node (**5**), and Node (**6**) are the attacker nodes that either disrupt the network communication or steal the sensitive data from the IoT device.
5) Node (**7**), Node (**8**), Node (**9**), and so on are the intermediate nodes between the Source and Destination nodes and are part of the network topology.

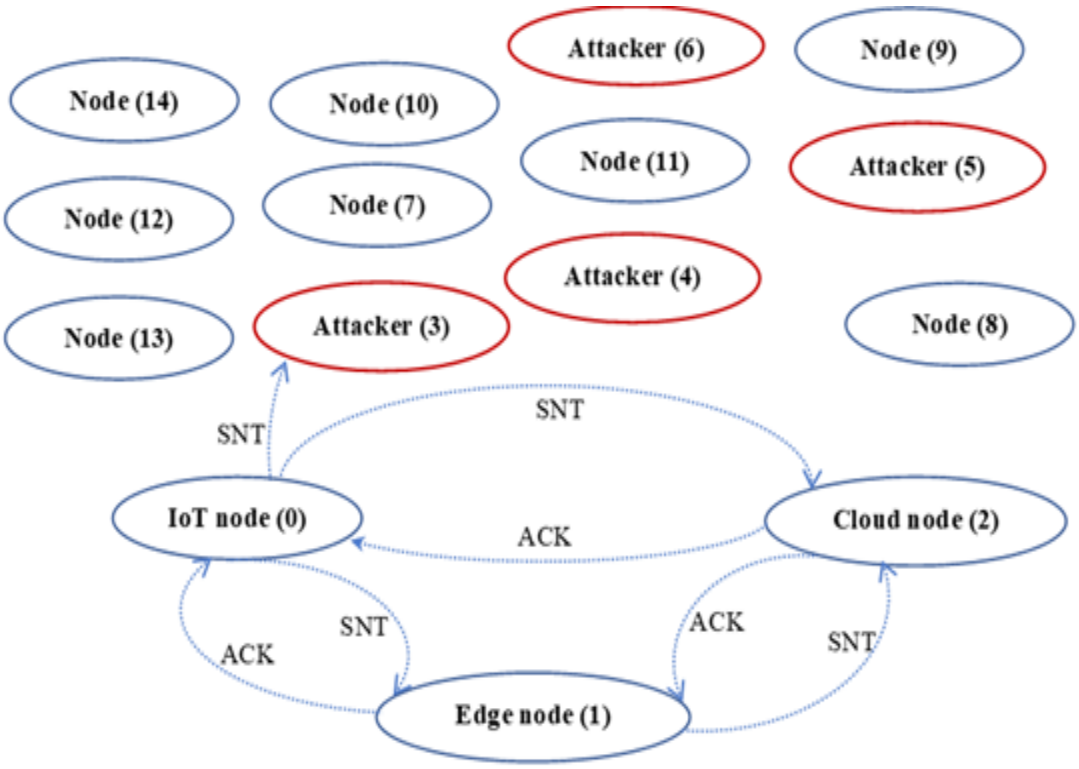



**Figure 29:** DDoS Attack in the Simulated IoT Network

Suppose at some time $t_1$, one data packet '**d**' retrieved by the source node (**0**), an IoT device through its sensor, and from the surrounding environment. After the retrieval, this data packet is released from the IoT device for its destination node (**2**). However, an IoT node generally does not push its data directly to the cloud server. It must go to the Edge server node right after leaving the IoT node for better processing of the data, as per the resource capacity of the edge server. The outcome of this processing is mostly transferred to the destination cloud server node (**2**). Now, assume that after some time interval and at time $t_2$, there comes an attacker node (**3**), which is an outsider and tends to either disrupt the network communication or wants to steal sensitive information from the network. As the source IoT node (**0**) is unaware of these sudden network changes, it may transfer the data packet to the attacker node (**3**). Since node (**3**) is an outsider node and is not bound by our MDRL security method, it will not reply with a response ACK message. Once the sender node (**0**) does not receive the ACK message within a stipulated time interval, then it will request to update the network security policy, where node (**3**) will be marked as an illegal node, and it will be discarded for all future data communication. After making this request, the Sender node (**0**) will

retransmit the same data packet after taking it from its memory buffer. The memory buffer will be used to save a copy of the data packet before transmitting it to a node. Once the data packet is successfully transferred to the node and acknowledged with a response message (**ACK**) by the source node within the stipulated time interval, the entry of the source node is cleared from the memory buffer. Once the data packet arrives at a benign intermediate node, then the intermediate node becomes the source node for the data packet, and again it follows the MDRL approach to transfer the data packet securely to the next intermediate node and continues until it reaches the destination cloud server node (**2**).

### 4.2.2 Mathematical Representation of Message Driven Reinforcement Learning (MDRL) Security Method

Suppose a large public or private distributed network has a total of '**i**' number of IoT devices connected to a centralized Edge Server. The edge server will do the required computation on behalf of an IoT device and upload the processed data to the cloud server. Our proposed novel reinforcement learning-based security method will prevent and detect malware attacks in the underlying IoT network to secure Edge computation. It includes the following components.

**A. State Space**

Our proposed RL-based security in IoT-Edge computing uses the following state space 'S' defined below by using equation **(1)**.

$S_1 = \{X_i\}$, where, $0 <= \mathbf{i} < n$. **(1)**

Here, **n** is the total number of IoT devices in the network.

And, $X_i$ denotes the states when the data packet arrives at the IoT node (**i**).

$S_2 = \{E\}$, Here, 'E' denotes the state when the data packet arrives at the Edge device for preprocessing.

$S_3 = \{C\}$, Here, C denotes the state when the data packet arrives at the cloud server for permanent storage.

Now, 'S' is defined as below by using equation **(2)**.

$S = S_1 \cup S_2 \cup S_3$. **(2)**

**B. Action Space**

As already shown in **Figure 22**, Our proposed research model uses the Action space 'A' that has the following action items ($A_i$) as defined below.

$A_1$ -> Detect and Prevent Malware Attacks

$A_2$ -> Data Traffic Maintenance

$A_3$ -> Dynamic Configuration of IoT-Edge-Cloud Nodes

$A_4$ -> Initiate updating the RL-based security policy

Now, 'A' is defined as below by using equation **(3)**.

$A = \{A_1, A_2, A_3, A_4\}$ **(3)**

**C. State Transitions**

We defined state transition $\Delta S$ in our RL-based security policy as the movement of the data packet from one state to another. Mathematically, it is defined as below by using equation **(4)**.

$\Delta S = \{X_i, X_j\}, \Delta S \in S.$ **(4)**

Here, $X_i$ is the current node, and $X_j$ is the destination node for the data packet.

**D. Reward Function**

We defined the Reward Function in terms of three network parameters, namely, **1)** Average Throughput, **2)** End-to-End (E2E) delay, and **3)** Packet Delivery Ratio (PDR). If both source and destination nodes are benign, the agent is rewarded with a positive value calculated by taking the inputs of all three network parameters. The data packet will be lost if the destination node is malicious. In this case, the agent is rewarded with a negative value calculated again with the inputs of these three network parameters.

Suppose that $\Delta t$ is the time duration for a data packet to travel from one node to another. This movement of the data packet is termed an action ($A_i$) over the time duration $\Delta t$. This action will either give a positive reward or a negative reward to the Agent. Hence, we will have the following two cases to consider.

**Case 1: Positive Reward**

In this case, both source and destination nodes are benign. In this case, the data packet will safely reach the destination node and will result in a Positive Reward. The positive reward will give the following benefits to the system.

- Increase in Average Throughput (ΔTP)
- Decrease in E2E delay (ΔE)
- Increase in Packet Delivery Ratio (ΔD)

**Case 2: Negative Reward**

In this case, the source node is benign, and the destination node is malicious. In this case, the data packet will be lost and will result in a negative reward. The negative reward will have the following negative impacts on the system.

- Decrease in Average Throughput (ΔTP)
- Increase in E2E delay (ΔE)
- Decrease in Packet Delivery Ratio (ΔD)

Moving ahead, we calculate the reward function for the Agent from a data packet by its movement from the same Source to the Destination nodes, as shown in equation **(5)**. These values lie between 1 to 100.

$$R(X_t, A_t) = \left(\frac{\Delta TP}{\overline{TP}}\right) \times \left(\frac{\Delta E}{\bar{E}}\right) \times \left(\frac{\Delta D}{\bar{D}}\right) \times 100 \qquad \textbf{(5)}$$

Here, the value of R ($X_t$, $A_t$) lies between 0 and 100. i.e., R($X_t$, $A_t$) € [0, 100].

- $\overline{TP}$ is the Average Throughput
- $\bar{E}$ is the Average End-to-End delay
- $\bar{D}$ is the Packet Delivery Ratio

## 4.3 Real-Time Data Communication-Based Experiment

After successfully implementing the proposed novel reinforcement learning-based security mechanism designed to protect IoT devices within an edge computing framework using the NS-2.35 network simulator [32] [33], and leveraging the Ad hoc On-Demand Distance Vector (AODV) routing protocol for communication [37]. Further enhancements are made to refine and validate the model's robustness.

To strengthen and evaluate the proposed security method under realistic conditions, three types of datasets as below were utilized:

**1)** Real-Time Public Domain Dataset-Based Experiment
**2)** Real-Time Personal Computer Temperature Sensor Data-Based Experiment
**3)** Real-Time In-house IoT Device Communication-Based Experiment

### 4.3.1 Proposed Model Validation Using Real-Time Public Domain Data-Based Communication

We have also used datasets available in the public domain, specifically the IoT-23 public dataset [29], to validate the proposed model. This IoT network traffic was captured in the Stratosphere Laboratory, AIC group, FEL, CTU University, Czech Republic. The primary objective of this dataset is to provide the research community with a comprehensive and labelled collection of real-world IoT traffic, including both benign activity and traffic generated by malware infections. This facilitates the development and evaluation of advanced machine learning algorithms for IoT security applications. It is a widely recognised, publicly available, and labelled dataset used for analysing malicious and benign network traffic associated with Internet of Things (IoT) devices. It comprises twenty-three distinct scenarios, each representing network traffic captured under varying conditions. Specifically, twenty of these scenarios are derived from IoT devices that have been deliberately infected with various forms of malware, providing insights into malicious behaviours and attack patterns. The remaining three scenarios capture network traffic generated by legitimate, non-infected IoT devices operating under normal conditions. Each scenario is provided as a packet capture (PCAP) file, making the dataset highly valuable for security researchers and machine learning practitioners working on intrusion detection, anomaly detection, and other cybersecurity-related tasks in IoT environments.

In the IoT-23 dataset [29], each IP address is carefully labelled to capture its behavioural role within the communication network, providing a clear distinction between benign and malicious activities. Benign IP addresses are typically associated with normal IoT device operations, producing bidirectional traffic patterns that follow expected communication flows, such as sensor reporting, device updates, or cloud service

interactions. In contrast, malicious IP addresses are flagged due to abnormal behaviours linked to cybersecurity threats, including botnet activity, command-and-control (C2) signalling, malware propagation, distributed denial-of-service (DDoS) attacks, and unauthorized data exfiltration. These malicious behaviours often differ significantly from benign traffic in terms of packet frequency, payload characteristics, connection persistence, and flow directionality, making them detectable through network monitoring and machine learning techniques. By providing a mix of both benign and malicious communication traces, the IoT-23 dataset serves as a valuable benchmark for researchers and practitioners in developing and validating intrusion detection systems (IDS), anomaly detection algorithms, and threat intelligence frameworks tailored for IoT environments. These include asymmetries in data flow, such as one-sided traffic without corresponding acknowledgements, abnormally high packet frequencies, irregular payload sizes, or timing anomalies between transmissions. Such deviations from expected network behaviour can serve as critical indicators of compromised networks, enabling intrusion detection systems (IDS) and machine learning-based security models to differentiate between legitimate and adversarial traffic patterns [29] [34] [35].

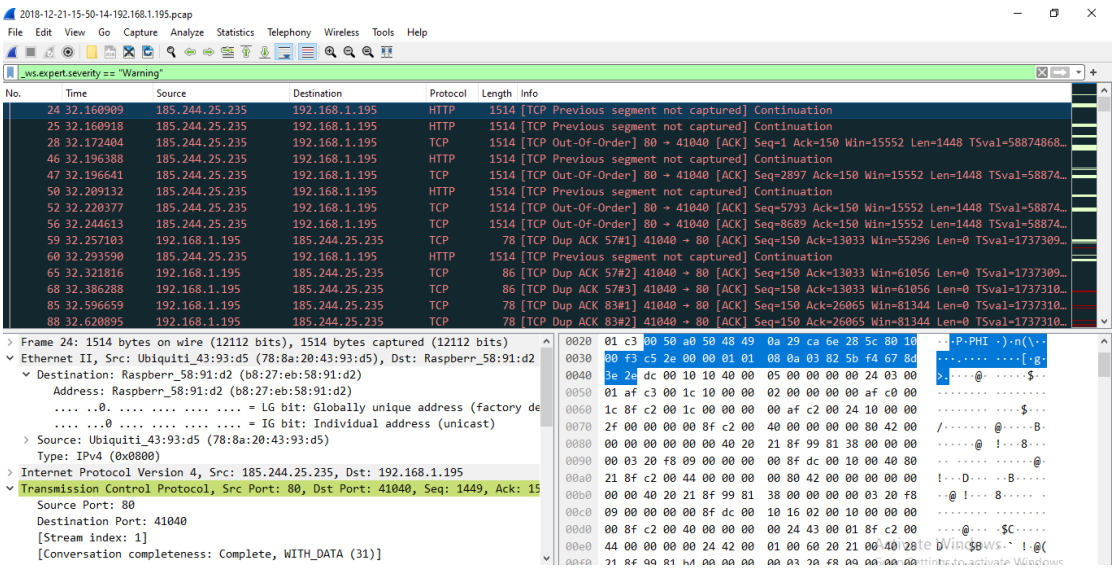



**Figure 30:** Malicious data packets on IoT-23-Dataset

Above **Figure 30** illustrates malware-induced traffic packets extracted from the IoT-23 dataset, highlighting anomalies in their structure that were used as input for our security framework.

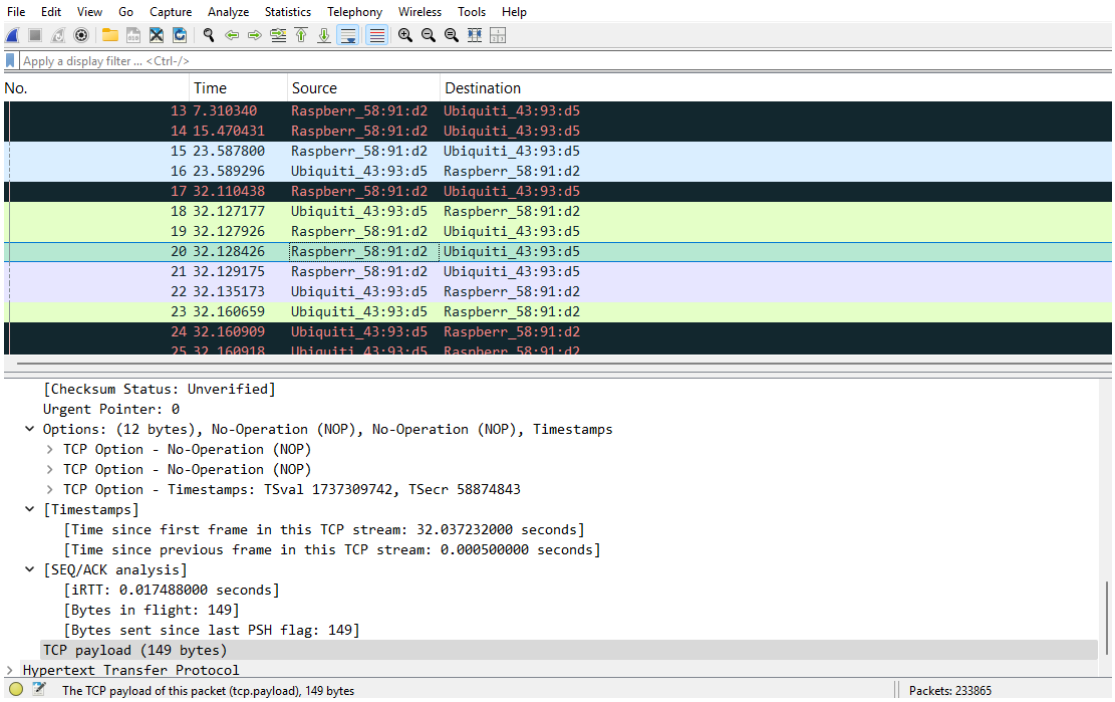



**Figure 31:** Non-Malicious Data Packets into IoT-23-Dataset

Above **Figure 31** illustrates the non-malicious data packets extracted from the IoT-23 dataset. The inclusion of such nuanced features in the dataset provides a rich testbed for developing and validating intelligent network security mechanisms, particularly those based on machine learning and reinforcement learning, that must adaptively learn to detect anomalies in real-time under dynamic conditions. This inherent contrast in communication behaviour enables effective training and testing of our proposed security model, which dynamically adapts to malicious behaviours in real time.

We again extended the simulation-based novel reinforcement learning-based security mechanism tailored for IoT-Edge computing environments by using this public dataset. In this approach, a dedicated RL agent is deployed at the edge server to continuously monitor and analyse incoming network traffic from IoT devices. The agent performs real-time feature extraction from each data packet to determine the behavioural characteristics of the sender device. Based on these features, the agent classifies the packet as either malicious or benign. A key component of our system is a temporary memory buffer, designed to serve dual purposes. If a data packet is found malicious, it will be stored in a separate memory buffer and labelled as MALICIOUS. These flagged instances serve as training data for the agent, enabling it to adaptively learn and improve

its ability to detect and mitigate similar threats in the future. On the other hand, if a data packet is deemed benign but cannot be immediately forwarded due to bandwidth limitations, it is temporarily stored and transmitted once sufficient bandwidth becomes available. The Edge server deployed agent operates as a continuously running thread within the server environment, allowing for real-time response and learning. The agent's reward function is strategically designed to encourage performance enhancements by maximizing Packet Delivery Ratio (PDR) and average throughput, while minimizing End-to-End (E2E) delay [16]. The proposed security framework is composed of several interconnected modules that collectively form a robust security layer, effectively shielding the IoT-Edge architecture from a wide range of cyber threats.

### 4.3.2 Proposed Model Validation Using Real-Time Personal Computer Temperature Sensor Data

Our proposed security model is capable of detecting and mitigating DDoS attacks by dynamically observing packet behaviour and flagging anomalies based on learned patterns. To support this, we have performed an extensive analysis on a widely used various sensor data from our personal computer by using the tool "HWiNFO" [75] and also using the Wireshark network analysis tool [45]. The extraction of sensor data from our personal computer is depicted in **Figure 32**.

The Wireshark tool [45] played a crucial role in our research by enabling the extraction of relevant features and traffic patterns associated with malicious activities. As a widely recognized network protocol analyzer, Wireshark provides deep packet inspection capabilities, allowing us to capture, filter, and analyze network traffic in real time. By examining packet headers, payloads, and communication flows, we were able to identify abnormal behaviors such as unusual port access, irregular packet sizes, suspicious IP addresses, and anomalous communication frequencies. These extracted features were essential for distinguishing between benign and malicious traffic, thereby serving as a foundation for training and validating our reinforcement learning based security model. The use of Wireshark not only enhanced the reliability of our dataset but also ensured that the analysis reflected realistic network-layer attack scenarios. The analysis results were used to categorize malicious packets, as summarized in **Table 7**.

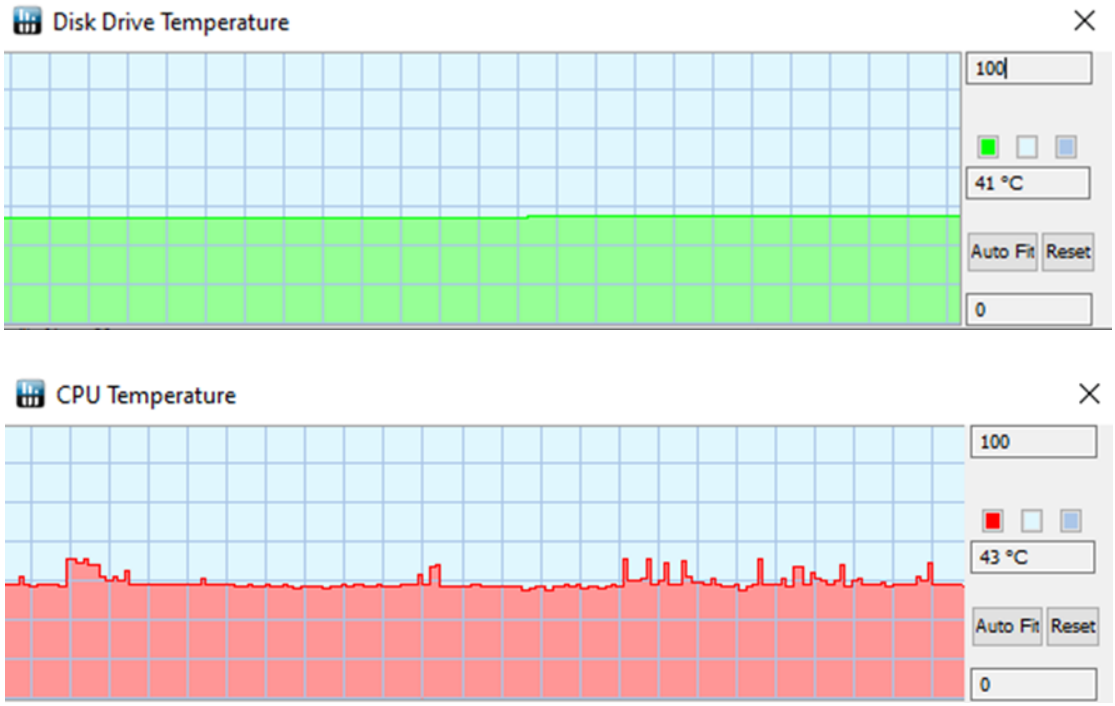



**Figure 32**: Personal Computer Temperature Sensor Data

**Table 7**: Network Data Packet Attributes

| | |
|---|---|
| **Header** | Sender's IP address |
| | Receiver IP address |
| | Protocol |
| | Packet sequence number |
| **Payload** | Actual data to send |
| **Trailer** | Data to show the end of the packet |
| | Error correction bits |

In our proposed methodology, as already illustrated in **Figure 20**, the Reinforcement Learning (RL) Agent makes decisions based on the Markov Decision Process (MDP) framework [44], which models the environment in terms of state transitions and rewards. However, the actual selection of actions from the available action space is guided by the Epsilon-Greedy algorithm [35] [107] [108], a widely adopted strategy

that balances exploration (trying new actions) and exploitation (choosing the best-known action). This hybrid mechanism allows the agent to optimize its learning process while adapting to the dynamic threat landscape in real-time. In our proposed methodology, as shown in **Figure 33**, the Agent makes decisions based on an MDP. But the decision to apply the actions is taken by using the Epsilon-Greedy Algorithm [107] [108]. Our reinforcement agent is installed on the Edge server, which continuously monitors each data packet arriving from the IoT device(s); hence, we claim that our proposed approach can detect and defend against all types of malware attacks. We used the most popular basic reinforcement learning algorithm [113] to derive Q-values, which are initialized to zero. The following equation **(6)** is used to calculate the same Q-values.

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r - Q(s, a)] \qquad \textbf{(6)}$$

Here, 'α' represents the discount factor. The RL agent state (s), action (a), and the corresponding reward (r) predominantly depend on the value of α.

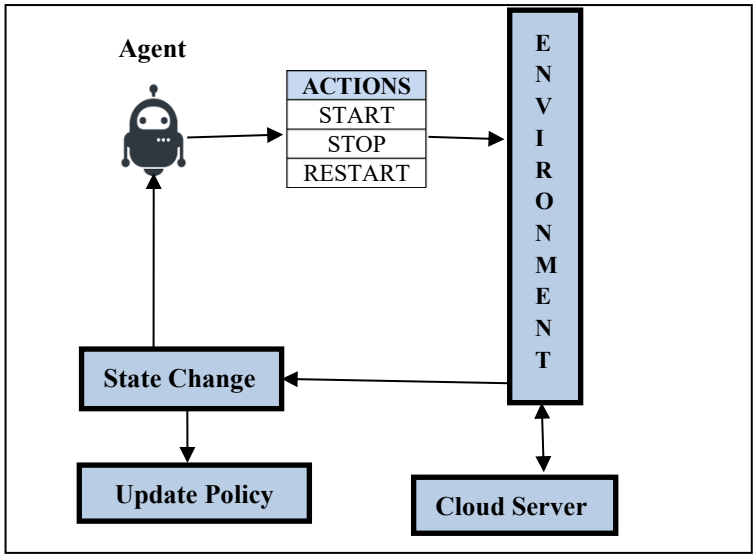



**Figure 33**: Proposed RL-based security method

Each action taken by the agent yields an integer reward (**R**), either -1 or 1. Rewards are given to the agent based on the basis of performance metrics PDR, average throughput, and E2E delay. Increase in PDR and Average throughput are contributing to reward function value, but an increase in E2E delay should be penalized, not rewarded [111]

[113]. These performance metrics are calculated once the action is completed. A malware attack deviates the PDR, average throughput, and E2E delay values with respect to benign data communication. We further classify the actions as positive or negative. We have defined a reward threshold limit constant (**λ**) on which if the calculated reward function value ($R_{cal}$) at time 't' by equation **(7)** is found to be greater, then the agent will be rewarded with '1', otherwise will be rewarded with '-1' as shown by equation **(8)**. The positive or negative action optimises the knowledge base of the agent and updates the malware detection policy, but either increases or decreases the cumulative reward by 1, respectively. Our model is trained to maximise reward while maintaining consistency in PDR, average throughput, and E2E delay [2].

$$R_{cal}(t) = \frac{\Delta \text{PDR (t)}}{PDR\ (max)} + \frac{\Delta \text{Average Throughput (t)}}{Average\ Throughput\ (max)} - \frac{\Delta \text{E2E delay (t)}}{E2E\ delay\ (max)} \qquad \textbf{(7)}$$

$$\mathbf{R} = \begin{cases} 1, & \text{if } R_{cal}(t) > \lambda \\ -1, & \text{if } R_{cal}(t) < \lambda \end{cases} \qquad \textbf{(8)}$$

### 4.3.3 Proposed Model Validation Using Real-Time In-house IoT Device Communication

We have also validated our proposed model using real-time data collected from an actual Internet of Things (IoT) deployment. The core of the setup was based on the ESP-WROOM-32 microcontroller [41] [42], a cost-effective and high-performance device equipped with integrated Wi-Fi and Bluetooth functionalities, making it well-suited for Edge computing and IoT environments. This microcontroller was interfaced with a DHT22 sensor module [43], which was responsible for continuously monitoring and recording ambient temperature and humidity data. The setup enabled seamless transmission of sensor readings to the Edge server for further processing and analysis, thereby supporting the real-time evaluation of network behaviour under dynamic environmental conditions.

Data communication between the In-house IoT device ESP-WROOM-32 and the local Edge server was established by using the home WiFi connection. The temperature and

humidity data of the surrounding environment are captured by the DHT22 sensor and transferred to the wired-connected ESP-WROOM-32 microcontroller. As soon as the data is received at the microcontroller, it is further forwarded to the local Edge server for preprocessing. We are also transmitting the malformed, malicious data packets to the microcontroller at certain intervals. The novel RL-based security agent is deployed on the Edge server, which is responsible for filtering and discarding malicious data packets arriving at the Edge server at discrete times. It allows only non-malicious data packets for preprocessing at the Edge server. The pre-processed data at the Edge server is again further transferred to the Cloud server for permanent storage and later use.

### 4.3.4 Adaptive Epsilon Greedy Reinforcement Learning (AEGRL)

Once we completed the design and implementation of the proposed security model of IoT-Edge computation through the NS-2.35 simulator [32] [33] and on simulation datasets, we started working on an Adaptive Epsilon Greedy Reinforcement Learning security method termed "AEGRL", which is to update and improve our security model to work perfectly for real-world IoT network communication. The proposed model, AEGRL, was validated using the Personal Computer Sensor Data (**Tool**: HWiNFO) [75], and the In-house IoT dataset. For this, we first used a public dataset [29] to improve and validate the proposed security model. After this, we used in-house real-time datasets from an ESP-WROOM-32 IoT device [41] [42] connected with a DHT22 Temperature and Humidity sensor module [43]. Apart from temperature and humidity data collected through the DHT22 sensor device, we also collected data from the built-in touch sensor module in the ESP-WROOM-32 device. In this model, an agent is deployed on the edge server to monitor network traffic originating from IoT devices across the network. We have used the MQTT protocol [31] to implement this security method to protect the IoT-Edge computation. MQTT is a well-known and widely used lightweight protocol for carrying out information using a “publish or subscribe” model. As already discussed, Node-RED [47] is used to enable the MQTT broker and runs on the Edge server. It enables the data communication between IoT devices and the Cloud server. The agent uses network data packet feature extraction to assess whether the sender device is transmitting malicious or benign data. In our proposed epsilon adaptive approach, the epsilon value depends on the volume of the incoming network traffic

data. If the volume of the traffic is comparatively lesser, then the agent does the exploration of the IoT-Edge computing network environment with a small value of Ɛ and increases the familiarity knowledge database of the environment, whereas when the volume of the network traffic data is comparatively higher then with the higher value of Ɛ, then the agent exploits its current knowledge database and select the most optimal action to trace malicious data packets. To enhance the agent's learning process, we created a temporary memory buffer for various purposes. If a malicious data packet is detected based on the criteria outlined in **Table 8**, then it is stored in this buffer and labelled as MALICIOUS. This enables the agent to learn from the data, improving its ability to identify and prevent similar attacks in the future. Conversely, if the data packet is determined to be BENIGN, it is stored in a separate data buffer for transmission when network bandwidth allows. Since bandwidth availability varies with network traffic, this ensures efficient use of resources. The agent is implemented as a continuous-thread program, installed on the edge server, and actively monitors the IoT environment. Rewards are given to the agent based on improvements in Packet Delivery Rate (PDR), average throughput, and reductions in End-to-End (E2E) delay. The proposed AEGRL method is described in **Figure 34**. RL-enabled Machine Learning Classifiers to detect anomalies in the data packets. Only non-malicious data packets are filtered through the proposed security layer, and malicious data packets are discarded. The action selection criteria based on Ɛ-value are depicted in **Figure 35**.

**Table 8:** Remarks on Malicious Data Packets

| **Malicious Data Packet Identifier** |
|---|
| Out-of-order in the packet sequence |
| Lost/Dropped packet segment |
| Retransmission of the same packet |
| Duplicate ACK received multiple times |
| Reception of dummy packets with zero payload |

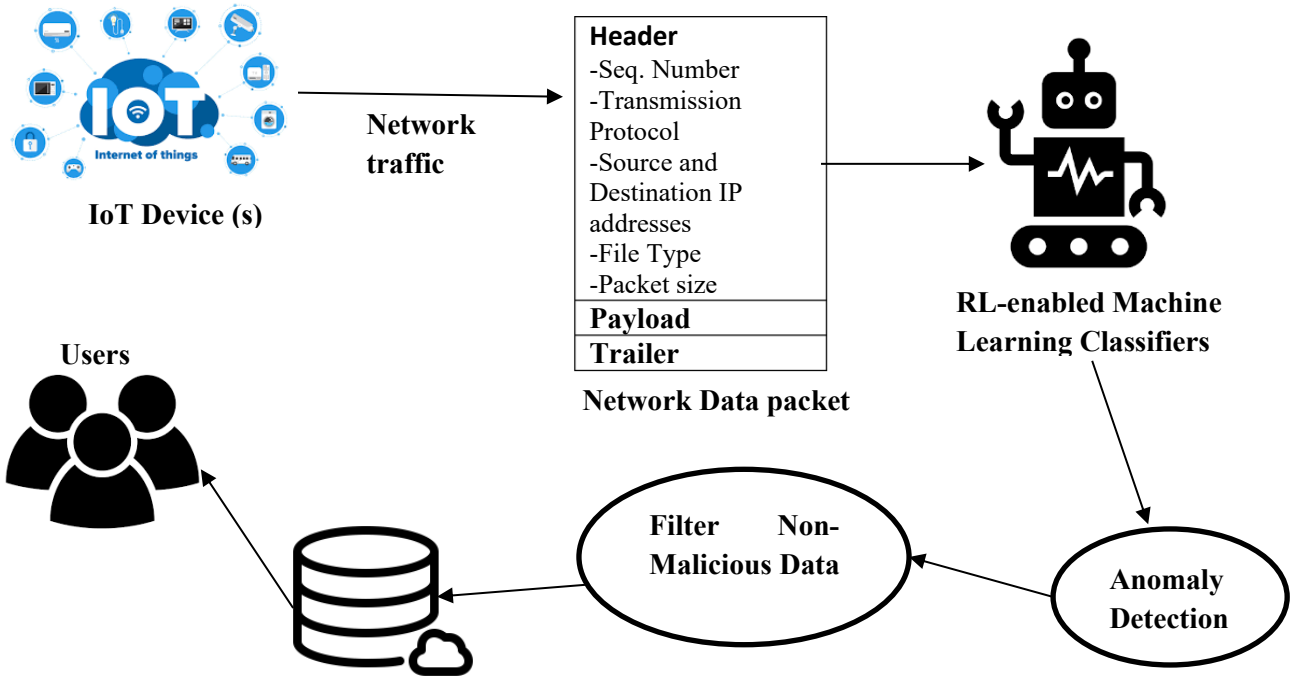



**Figure 34:** Proposed AEGRL Security Model for IoT Device

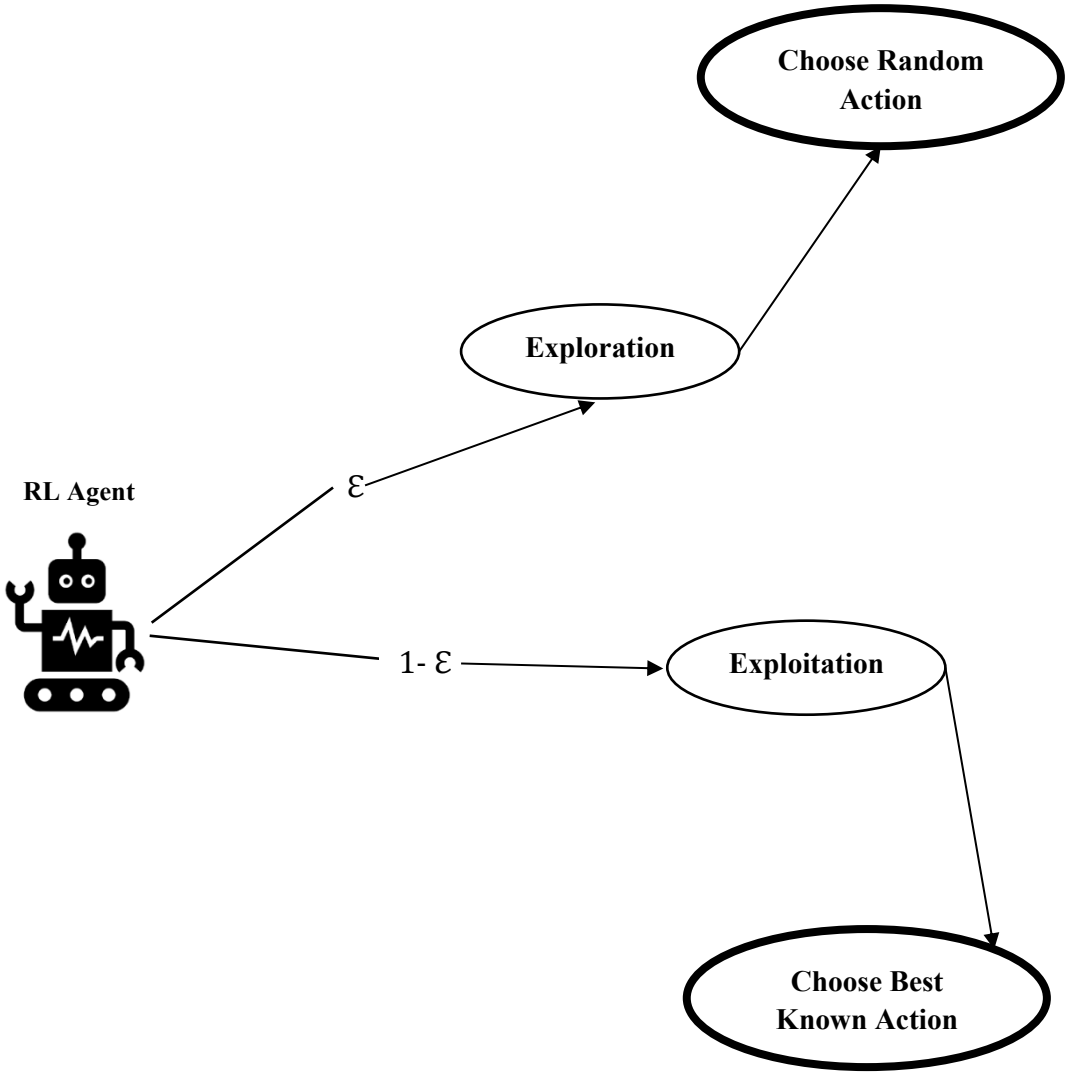



**Figure 35:** Action Selection from Epsilon-Greedy Learning Policy

### 4.3.5 Mathematical Representation of Adaptive Epsilon Greedy Reinforcement Learning Method (AEGRL) Security Method

The proposed model, AEGRL, is designed to protect sensitive data and maintain the integrity of IoT networks as they grow and evolve. Our research introduces an adaptive reinforcement learning approach tailored to enhance the security of IoT-Edge computing environments. This method enhances accuracy, robustness, and scalability by dynamically balancing exploration and exploitation based on real-time attack volumes. The adaptive reinforcement learning approach allows the model to fine-tune its actions based on the immediate security environment, ensuring it can effectively respond to varying levels of threat intensity. This model is well-equipped to handle the complexities of modern networks and has the following components.

**A. Action Space:**

The following function defines the actions taken by the agent at time t by using equation **(9)**.

$$A(t) = \mu \int_0^t d\omega \int_0^t d\sigma \qquad (9)$$

Here, **μ** represents the action frequency of the agent, which specifies the number of trace attempts the agent makes per second to identify and track suspicious data packets. This parameter is dynamic and directly influenced by malicious activity targeting the Edge server. When no cyber-attacks are detected, **μ** is set to zero, meaning the agent remains idle to conserve computational resources. However, as the frequency of cyber-attacks increases, **μ** correspondingly escalates, prompting the agent to increase its tracing and monitoring activity. **ω** represents the traffic volume measured at time **t**, and **σ** represents the computing resource available at time **t**.

**B. State Space:**

The action frequency (μ) also defines the state of the agent space (S). Each State transition of the agent is triggered based on the specific action taken by the agent. If the network traffic suffers no malware attacks for a long time, then the state of the agent becomes IDLE. If the agent observes certain changes in the behaviour of the network, then its state changes to RUNNABLE. When the agent executes to trace the suspicious data packets, the state changes to ACTIVE. Now, if the agent

waits for any external inputs to proceed further, then its state transitions to the WAIT state. Finally, when the agent finishes the execution and provides the suspicious data packet information, its state reaches the STOP state. The action frequency (**μ**) not only determines the number of trace attempts made by the agent but also defines the agent's state within its operational state space. The states of the agent in the proposed security model are represented by using equation **(10)**.

$$S = \{IDLE, RUNNABLE, ACTIVE, WAIT, STOP\} \quad \textbf{(10)}$$

Each state transition of the agent is triggered by a specific action based on network conditions and the agent's observations. The states of the agent are defined as follows:

1) **IDLE**: The agent remains in the IDLE state when the network traffic shows no signs of malware attacks over an extended period. In this state, the agent conserves resources and remains inactive.
2) **RUNNABLE**: If the agent detects unusual or suspicious patterns in network behaviour, such as a sudden increase in traffic or deviations from normal data packet characteristics, it transitions to the RUNNABLE state. This indicates that the agent is prepared to conduct further analysis.
3) **ACTIVE**: When the agent begins executing its tracing mechanism to identify and track suspicious data packets, it enters the ACTIVE state. In this state, the agent actively monitors, classifies, and mitigates potential threats.
4) **WAIT**: The agent transitions to the WAIT state if it requires external inputs or additional data to proceed with its operation. For instance, the agent may pause to synchronise with other components of the IoT-Edge framework or await further instructions.
5) **STOP**: Once the agent completes its execution, identifies suspicious data packets, and reports the findings, it transitions to the STOP state. This state marks the conclusion of a tracing cycle.

**C. Reward Space:**

Each action taken by the agent in the surrounding environment yields an integer reward value ranging between 0 and 100. This reward is instrumental in refining the Reinforcement Learning (RL) policy, enabling the agent to optimize its decision-making process over time. In our approach, we employed the Adaptive Epsilon Greedy Reinforcement Learning (AEGRL) technique. This method ensures that, with a small probability (Ɛ), the agent explores the environment to discover optimal actions. During the exploration phase, the agent actively learns about its environment, identifying which actions lead to the highest rewards [13]. Once sufficient information is gathered and the agent reaches the saturation phase, it transitions to an exploitation mode. In this mode, the agent primarily selects actions that maximize the reward, thus enhancing the overall efficiency and effectiveness of its operations.

In our experiment, the epsilon (Ɛ) value was dynamically varied between 0.1 and 0.9. At smaller Ɛ-values, the agent predominantly explores the environment by taking random actions. This initial randomness facilitates better identification of malicious data packet features and improves the accuracy of reward calculations for each action. As the agent begins to exhibit optimal behaviour, the Ɛ-value is gradually increased, ensuring that only actions yielding maximum rewards are prioritised. The action taken by the agent at time 't' is defined as below by using equation **(11)**.

$$a_t = \begin{cases} Q_t(a), \text{ with probability } 1 - \varepsilon \\ \text{Any action 'a', with probability } \varepsilon \end{cases} \quad \textbf{(11)}$$

#### 4.3.5.1 Adaptive Epsilon Value Function in the Proposed Greedy Reinforcement Learning

In reinforcement learning (RL), the value of Epsilon (Ɛ) in the Ɛ-greedy strategy plays a pivotal role in balancing exploration and exploitation. Selecting an appropriate Ɛ is highly dependent on the environment's level of uncertainty and the sparsity of reward signals. Failure to account for these dynamics can lead to inconsistent learning behavior and suboptimal agent performance [53]. To address this, it becomes essential to define

an evaluation function that dynamically quantifies the agent's performance regarding its current Ɛ-value. This function allows for adaptive tuning, enabling the agent to respond effectively to environmental changes, particularly in security-sensitive IoT-edge scenarios. In this research work, we proposed a dynamic epsilon schedule, inspired by traditional epsilon-greedy strategies [54], but enhanced for resilience under adversarial conditions. Unlike static schedules, our model incrementally increases the Ɛ-value in response to the observed volume of malicious or anomalous data in the network. This adaptive mechanism prioritises exploration during high-threat scenarios, allowing the agent to discover new strategies and pathways for threat mitigation, while reducing Ɛ in more stable conditions to refine policy convergence. Such a strategy ensures that the learning agent remains responsive and proactive, particularly in edge environments where security threats may evolve rapidly and unpredictably. By correlating exploration rates with real-time threat levels, the proposed dynamic Ɛ-schedule enhances both the robustness and adaptability of the RL model. Our proposed methodologies have integrated adaptive epsilon strategies, especially in noisy or non-stationary environments. Using this evaluation metric, we aim to derive a dynamic update rule for Epsilon that maintains or enhances the agent's effectiveness over time. Our proposed RL security policy is robust to model uncertainty and disturbances, aligning with dynamically adapting Ɛ-value. In our proposed security method, the calculation of the Ɛ-value at time 't' depends on the traffic volume of the malicious data.

In our proposed security method for IoT-Edge communication, the adaptivity depends on the ratio of the volume of malicious data packets to the total volume of data packets arrived on the Edge device.

We have defined the adaptive learning factor for the proposed novel reinforcement learning-based agent as follows by using equation **(12)**.

$$\alpha_t = \alpha_{max} . \frac{V_{malicious}(t)}{V_{total}(t)} \qquad \textbf{(12)}$$

Here,

- $V_{malicious}(t)$ is the malicious data packets received in the time duration (0, t).

- $V_{total}(t)$ is the total number of data packets received in the time duration (0, t).

The Ɛ-value at time 't' is calculated using equation **(13)** as shown below.

$$\mathcal{E}(t) = \min\left(1, \mathcal{E}_0 + \alpha_t \cdot \frac{V_{malicious}}{V_{total}}\right) \qquad \textbf{(13)}$$

Here,

- $\mathcal{E}(t)$ is the Epsilon-value at time 't'.
- $\mathcal{E}_0$ is the initial epsilon value at time t = 0.
- $\alpha_t$ is the learning rate at time 't'.

### 4.4 Performance Metrics of the Proposed Security Method

The following performance metrics are used to calculate the efficiency and accuracy of the proposed Security Method.

1) **Packet Delivery Ratio (PDR)**: Packet Delivery Ratio (PDR) is a fundamental network performance metric that measures the efficiency and reliability of data transmission. It is defined as the ratio of the number of data packets successfully received by the destination to the total number of packets sent by the source. A higher PDR indicates better network performance, as it reflects minimal packet loss and more reliable communication [121].
2) **Average Throughput:** Average Throughput in network communication refers to the mean rate at which data is successfully transmitted from a source to a destination over a given time interval. It measures the amount of useful data received (excluding retransmissions and errors) rather than the total data sent. Throughput is commonly expressed in kilobits per second (kbps) or megabits per second (Mbps), and serves as a critical indicator of network efficiency and capacity [122].
3) **End-to-End Delay:** End-to-End Delay in network communication refers to the total elapsed time required for a data packet to travel from the source node to the destination node. This metric encompasses all forms of delay encountered

during the transmission process, including propagation delay (signal travel time across the medium), processing delay (time taken to process headers and perform routing decisions), queuing delay (time spent waiting in buffers due to congestion), and serialization delay (time required to place packet bits onto the transmission medium). End-to-end delay is typically measured in milliseconds (**ms**) and serves as a critical quality of service (**QoS**) indicator, particularly in latency-sensitive applications [123].

## 4.5 Summary

This chapter presents the design, implementation, and evaluation of reinforcement learning-based security methods for IoT-Edge computing systems, using both network simulations and real-time experimental setups. First of all, it introduces a network simulator-based approach where a Message Driven Reinforcement Learning (MDRL) method is proposed to secure IoT-Edge communications. It explains how MDRL enhances decision-making in threat detection and mitigation, supported by its mathematical representation to establish a formal foundation. After the simulation-based experiment is completed, it shifts toward real-time data communication experiments. It discusses methods based on public domain datasets and validates the proposed model using live data from a personal computer temperature sensor, collecting Disk drive and CPU temperatures over time, and a real-time In-house IoT device dataset. To improve adaptability in dynamic environments, an Adaptive Epsilon Greedy Reinforcement Learning (AEGRL) method is developed. Also, we discussed the mathematical formulation of AEGRL, including the adaptive epsilon value function, which balances exploration and exploitation in security decisions. Finally, we discussed the performance metrics used to evaluate the proposed security methods. These metrics provide a quantitative basis for measuring efficiency, accuracy, adaptability, and robustness of the MDRL and AEGRL approaches in both simulated and real-world IoT-Edge scenarios.

# CHAPTER 5

# OVERVIEW OF THE EXPERIMENT

We conducted our experiment in two phases. Each phase has the following components, as shown in **Figure 36**.

1) **IoT Sensor**: The IoT sensor unit collects data from the surrounding environments, like room temperature, humidity, etc, or collects some event data like touch data.
2) **Communication Network**: The communication network enables data communication between two parties. It can be either wired, wireless, or both.
3) **Edge Processing Unit**: This unit remains on the Edge device, mostly near the IoT device, to preprocess the data received from it.
4) **Security Layer**: This layer provides the security layer and is implemented in the Edge device to allow only benign data to go for preprocessing on the Edge layer.
5) **Data Aggregation Unit**: This unit remains either on the Edge device itself or on a separate data storage device and stores the resultant processed data from the Edge layer.
6) **Cloud Server**: This unit stores the resultant processed data for global access.
7) **User**: The user accesses the resultant processed data of the IoT device for their specific needs.

In the first phase, we used the Ad-hoc On-Demand Distance Vector (AODV) Routing Protocol [37] to transfer data among network simulation nodes using the NS-2.35 simulator [32] [33]. In this phase, we validated our proposed model using the simulation and public datasets [29]. We referred to this phase as the "Simulation Phase." Finally, in the second phase, we validated our proposed model using the public dataset again, primarily with real-time Personal Computer Sensor Data (Tool: HWiNFO) [74] and an In-house IoT device, ESP-WROOM-32 [41] [42] dataset. We termed this second phase the "Real-time Device Communication Phase." Both phases are described in the following sections.

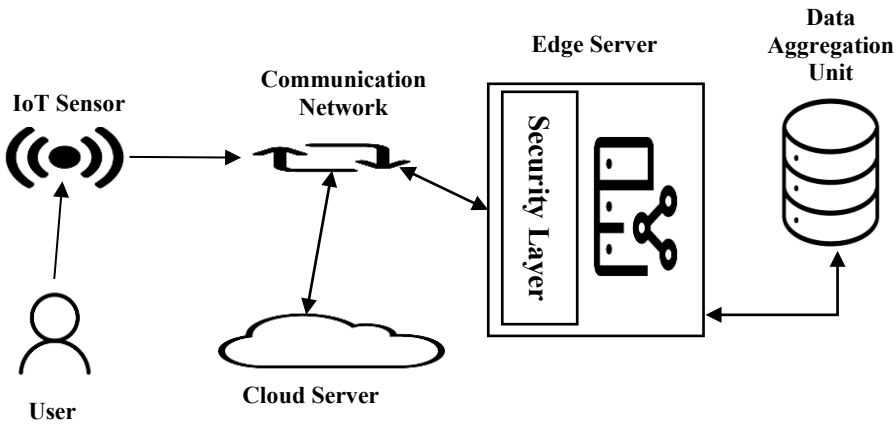



**Figure 36**: Components of the Experiment Setup

## 5.1 Simulation Phase

We worked with the NS-2.35 simulator [32] [33] in the simulation phase, and the performance metrics such as PDR, average throughput, and E2E delay of the proposed model were validated by using the simulation trace (*.tr) file. The research work done by us in this phase is published in the journal paper [4]. There are three different types of nodes created for the data communication in the simulated network, which are **1)** IoT node, **2)** Edge node, and **3)** Cloud node. The experiment was carried out over an NS-2.35 simulator to simulate the network model. The simulation of the experiment for non-malicious and malicious networks was carried out by using the same network settings with an equal number of total nodes in both. The only difference is that the malicious network is having minimum one node acting as an attacker node instead of acting as an IoT node. Hence, we classify an IoT node as an attacker (malicious) node or a benign (non-Malicious) node. The proposed methodology provides security against the most prevalent IoT attacks, which is the DDOS attack [79]. The number of nodes and positions of each node are configured by using a network setting configuration file. Similarly, transmission protocol, packet size, data packet transmission rate, etc., are defined in a separate configuration file. We carried out our experiment in three steps. In the first step, we simulated a non-malicious network with the number of nodes equal to 25, 50, 75, and 100 nodes one after another. In this non-malicious network, all the nodes are benign, and there is no external self-beneficial intent in the network. In this non-malicious network, we calculated the average packet delivery ratio (PDR), average throughput, and average end-to-end delay. In the second step, we repeated this round of

experiment with the malicious network configuration and with the same series of nodes. Network data configuration files remain the same in both non-malicious and malicious networks. The only differences between non-malicious and malicious networks are with the nodes, namely, Node (**3**), Node (**4**), Node (**5**), and Node (**6**), as shown in **Figure 37**. In malicious networks, Node (**3**), Node (**4**), Node (**5**), and Node (**6**) are malicious, and they are dropping/flooding data packets into the network. Otherwise, in the non-malicious network, they act like other benign nodes. Now again, we calculated the average packet delivery ratio (PDR), average throughput, and average end-to-end delay on this malicious network. In the third and final step, we turn **ON** the proposed RL security policy on the malicious network and then analyze and continuously improve our RL security policy to meet our security expectations and maintain the network performance during DDoS attacks in the network.

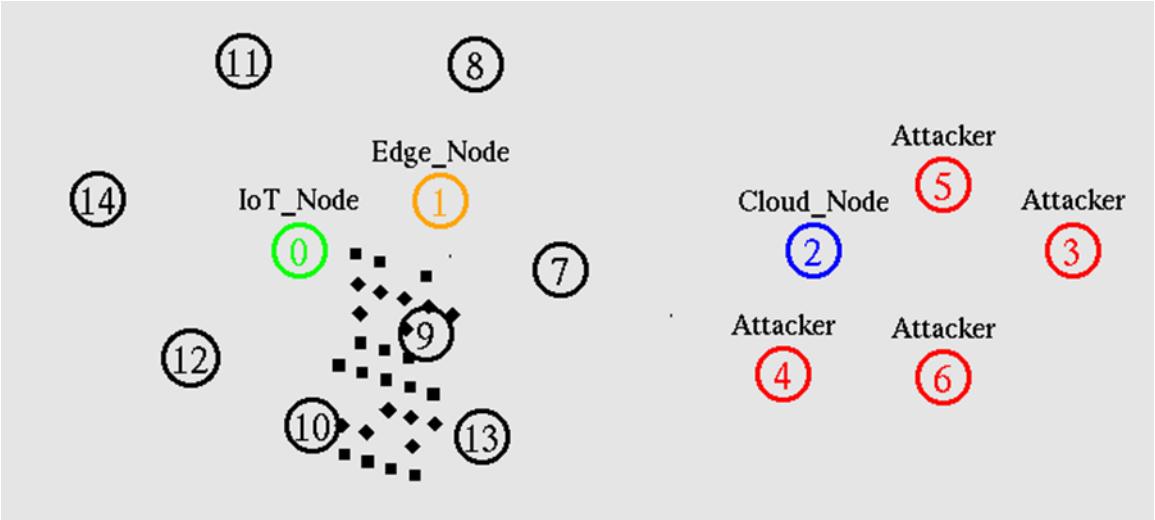



**Figure 37:** Network Simulation of Malicious communication of IoT, Edge, and Cloud nodes

### 5.2 Real-time Device Communication Phase

Once the experiment on the simulator was completed and the expected results were achieved, we moved to real-time validation of our proposed security model. The goal is to validate our security model under real-time constraints with a real IoT device, making it ready for real-time end users. In this phase, we designed and implemented a security layer on the network comprising four nodes: **1)** IoT node, **2**) Edge server node, **3)** Cloud server node, and **4)** Attacker node. This security layer is first verified and evaluated on the public datasets [29] and then finally on the real-time data

communication using the IoT-23 public dataset [29] and an In-house IoT device ESP-WROOM-32 [41] [42]. MQTT protocol [30] is used in this phase to implement the proposed security method to protect the IoT-Edge computation, and Node-RED [47] is used to enable the MQTT broker that runs on the Edge server. It enables data communication among all four nodes. We assumed that the Attacker node had unauthorised access to the IoT communication network. In this setup, data flows from IoT devices to the Cloud server via the Edge server. The RL agent deployed on the Edge server plays a pivotal role in managing the incoming data traffic from IoT devices, with its specific responsibilities depicted in **Figure 38**. **Figure 39** provides a screenshot of the communication network we created using the MQTT protocol. In this proposed model, IoT devices transmit data to the Edge server for preprocessing. Once the results and reports are generated, they are forwarded to the Cloud server for permanent storage and global access. Simultaneously, the Attacker node continuously sends malicious data packets to the Edge server, attempting to distort the results and reports. However, after deploying our novel reinforcement learning (RL) policy on the Edge server, the IoT communication channel is fully secured. The RL policy effectively eliminates malicious data packets, ensuring uninterrupted and accurate processing and transmission of data.

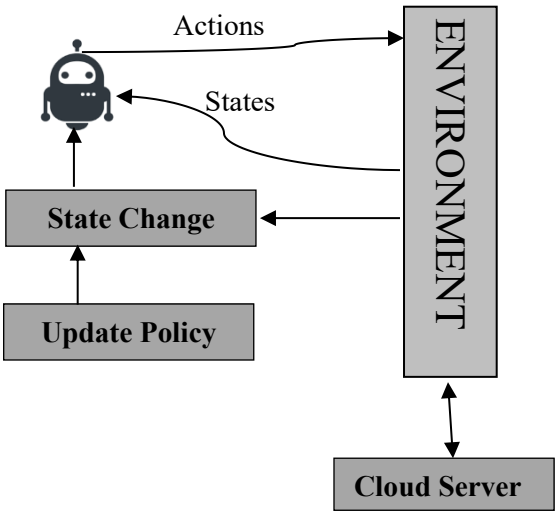



**Figure 38:** Agent Roles in Proposed RL Model

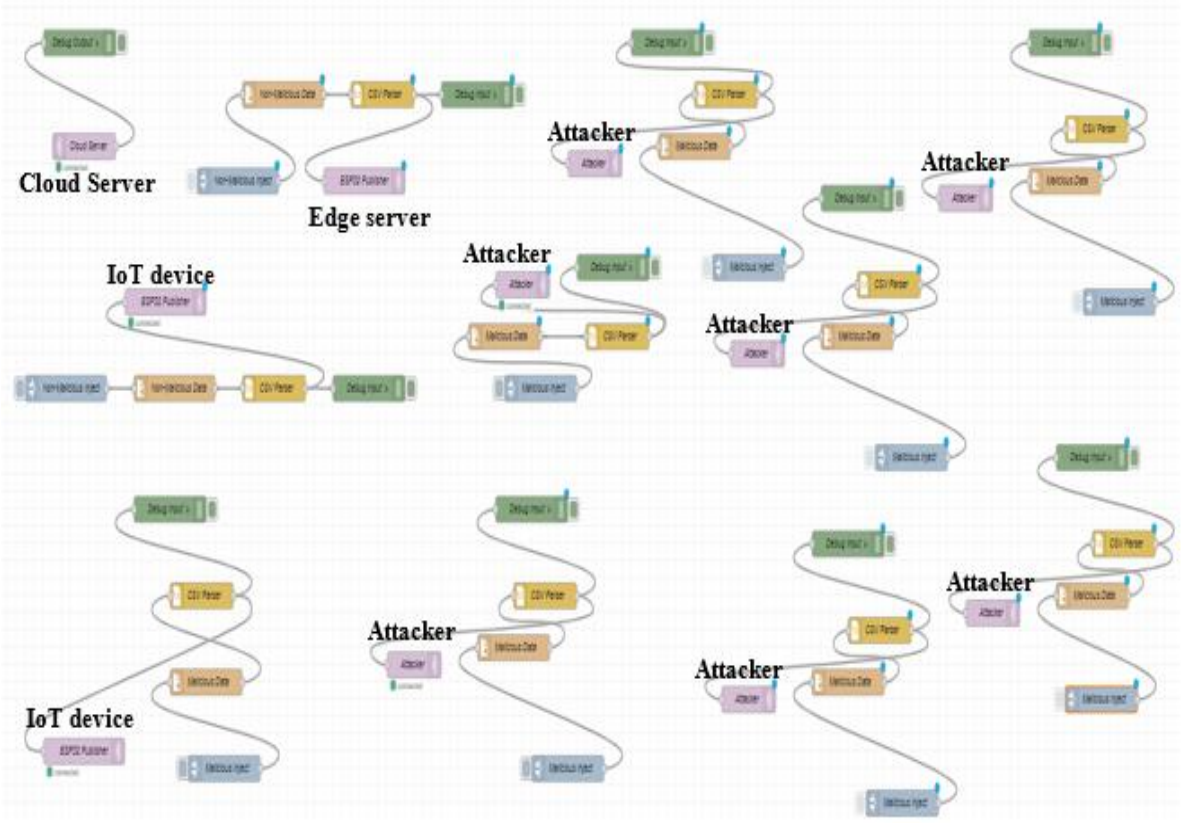



**Figure 39:** Data Communication into the MQTT protocol

## 5.3 Performance Matrices Evaluation

The following standard evaluation metrics are used to evaluate the performance of our proposed novel reinforcement learning-based security method [90].

1) **True Positive (TP)**: The number of actual malicious data packets correctly classified as malicious.
2) **True Negative (TN)**: The number of actual benign data packets correctly classified as benign.
3) **False Positive (FP)**: The number of benign data packets incorrectly classified as malicious.
4) **False Negative (FN)**: The number of malicious data packets incorrectly classified as benign.

Using the standard metrics explained above, Accuracy (Acc), Recall (Rec), Detection Rate (DR), and F1-score (F1) are calculated using equations **(14)**, **(15)**, **(16)**, and **(17)**, respectively, as follows.

$$Acc = \frac{TP+TN}{TP+TN+FP+FN} \quad \textbf{(14)}$$

$$Rec = \frac{TP}{TP+FN} \quad \textbf{(15)}$$

$$DR = \frac{TP}{TP+FP} \quad \textbf{(16)}$$

$$\mathrm{F1} = \frac{2*\mathrm{TP}}{2*\mathrm{TP}+\mathrm{FP}+\mathrm{FN}} \quad \textbf{(17)}$$

We classified the network data of the packets arriving on the edge server from the IoT device based on the selection of network packet features. These features are already shown in **Table 8**. We created two classes for the data packets: **1)** Malicious, and **2)** Benign. Each class has specific limit values of the feature attributes. Such classification and limitations are used to determine the Accuracy ($A_{cc}$), True Positive Rate (TPR), and False Positive Rate (FPR) by using equations **(18)** and **(19)**, respectively.

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \quad \textbf{(18)}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{TN}+\mathrm{FP}} \quad \textbf{(19)}$$

When we started the training at time t = 0, the DR value remained 0. But once the model is trained with the threshold amount of training datasets, then the value of TPR, FPR, and DR becomes near 1 at the threshold time $\mathbf{t_{\mu}}$ as shown in equations **(20)** and **(21)**.

$$\mathrm{TPR},\ \mathrm{DR} = 1,\ t \geq \mathbf{t_{\mu}} \quad \textbf{(20)}$$

$$\mathrm{FPR} = 0,\ t \geq \mathbf{t_{\mu}} \quad \textbf{(21)}$$

## 5.4 Summary

In this chapter, we have presented a comprehensive overview of the implementation phases of our proposed reinforcement learning-based security model for IoT devices in Edge computing environments. The development and validation of the model were carried out in two major phases. In the first phase, the core functionality of the model was designed and implemented using the NS-2.35 network simulator [32] [33], which allowed for controlled experimentation and reproducibility. This phase utilised a simulation dataset to emulate network behaviour under various conditions, including normal operations and malicious intrusions. The simulation-based approach enabled rapid prototyping and fine-tuning of key parameters, such as agent learning rates and reward functions. The use of the AODV routing protocol [37] further helped simulate mobile ad-hoc network conditions, reflecting real-world routing behaviour in distributed IoT scenarios. Upon achieving the expected security and performance benchmarks, this phase concluded with a stable and validated prototype of the model.

In the second and final phase, the model was extended to assess its performance under realistic operational settings involving real-time data communication. This involved integrating the model with two distinct types of datasets: **1)** Publicly available IoT-23 dataset [29], which contains labelled real-world network traffic including benign and malicious behaviours, and **2)** Real-Time In-house Datasets generated using the Personal Computer Sensor Data (**Tool**: HWiNFO) [75], and the ESP-WROOM-32 microcontroller platform [41] [42]. The In-house dataset was instrumental in evaluating the model's behaviour in a real-time environment, with direct communication between Sensors and the Edge server. By transitioning from simulation-based validation to real-world testing, this dual-phase implementation approach not only ensured the robustness and scalability of the proposed model but also demonstrated its practical applicability for securing IoT-edge infrastructures under both synthetic and live network conditions.

# CHAPTER 6

## RESULTS AND DISCUSSION

When the probability distribution of a state variable is known, the Q-learning algorithm serves as a robust reinforcement learning method to determine the optimal action in a given environment. Q-learning is a model-free technique that relies on estimating a Q-value function, which represents the expected cumulative reward for each state-action pair under a particular policy. These Q-values are iteratively updated, allowing the agent to converge toward an optimal policy even in environments with stochastic dynamics [55] [87] [113].

In our proposed novel reinforcement learning-based security framework, we leverage Q-learning to adaptively protect IoT devices in an Edge computing environment. The actions in this model are defined based on behavioural anomalies observed in the communication channel, as well as irregularities detected during data packet feature extraction. By mapping these observations into discrete states and assigning context-sensitive responses, the model dynamically refines its policy to handle evolving threats [88] [89]. Specifically, we define three distinct states in our approach:

1) **Non-Malicious**: It represents the data packet as normal or benign.
2) **Malicious**: It represents abnormal or intrusive data packets
3) **Unknown**: – It represents those data packets that exhibit uncertain or ambiguous behaviour that may require further inspection or isolation.

The probability distribution over these three states is derived from the frequency of occurrence of each state over time, reflecting the changing landscape of the operational environment. This probabilistic representation enhances the agent's ability to make context-aware decisions, especially in edge settings where real-time adaptability is crucial. By continuously updating the Q-values based on these state transitions and feedback from environment interactions, the agent becomes increasingly proficient in identifying and mitigating security threats, even in the face of previously unseen or zero-day attacks.

The performance of the proposed learning base reinforcement learning based security method to secure IoT-Edge computation is validated by using four datasets: the simulation dataset, a public domain dataset, Personal Computer Sensor Data, and

finally, an In-house IoT device-generated dataset. The parameter details of these three datasets are provided in **Table 9**.

**Table 9:** Test Datasets for Experiment

| Test Datasets | Source |
|---|---|
| Simulation Dataset | NS-2.35 simulator [32] [33] |
| Public Dataset | IoT-23 public dataset [29] |
| Personal Computer Sensor Data | HWiNFO [75] |
| In-House IoT Dataset | ESP-WROOM-32 [41] [42] connected with a DHT22 Temperature and Humidity sensor module [43] |

To emphasize the validation of the proposed security method, we performed repeated simulations on typical operating points to verify the effectiveness of the technique. Specifically, a set of at least 10 independent simulations is first performed on each of the three phases. Every experiment consists of up to 10000 trials, during which the Q-learning agent seeks attack target sequences to reach the optimal convergence of the security policy to neutralize the external attack objective setup on the system. At the beginning of the first trial in each experiment, we initialize the learning agent with a very low learning rate of 0.005. At the end of each episode, we increase the learning rate by a factor of 1.05. The proposed novel reinforcement learning security policy is built by following the steps below.

**Step 1**: START

**Step 2**: Define the number of Q-states, which includes "**Malicious**", "**Non-Malicious**", "**Blocked**", "**Quarantined**", and "**Unknown**". Which comes as the total number of Q-states (s) equals **5**.

**Step 3**: Define the number of actions (a) that are taken by the agent during the execution thread. Possible actions are "**Idle**", "**Trace**", "**Throttle**", "**Discard**", "**Forward**", and "**Policy_Update**". It comes as the total number of actions (a) equals **6**.

**Step 4**: Initialize and Set the Environment for the Agent.

**Step 5**: Initialize an initial Q-matrix ($Q_0$) of size (s, a) with all initial values "0".

**Step 6**: Define a destination Q-matrix ($Q_d$) of size (s', a'), when s' = goal_state.

**Step 7**: Initialize reinforcement learning hyperparameters, learning rate ($\alpha = 0.001$), initial epsilon value ($\varepsilon_0 = 0.05$), and discount factor ($\gamma = 0.95$) with a value between 0 and 1.

**Step 8**: Set num_episodes = 10000.

**Step 9**: Execute each episode until next_state = goal_state.

**Step 10**: At the end of each episode, update the RL-based security policy.

**Step 11**: At the end of each episode, increase the reinforcement learning rate ($\alpha$) by the factor 1.05.

**Step 12**: At the end of each episode, set the goal_state for which gives the maximum cumulative reward ($R_c$) to derive an optimal policy.

**Step 13**: Calculate the Accuracy of the proposed RL-based security model when the optimal policy is obtained.

**Step 14**: END.

Setting such hyperparameter values is aimed to achieve following goals [12] - [16].

1) $\boldsymbol{\varepsilon_0 = 0.05}$: This initial epsilon value ensures minimal exploration even in the stable conditions and facilitates preventing the following.
    a) Overfitting to a fixed policy.
    b) Failures under sudden environmental change.
    c) Energy consumption should be low.
    d) Decision stability should be high.
2) $\boldsymbol{\alpha = 0.001}$: Taking this initial learning rate value and gradually increasing it by a factor of 1.05 helps in avoiding sudden excessive exploration and results in controlled adaptation to attacks.
3) $\boldsymbol{\gamma = 0.95}$: This discount factor value gives strong importance to future rewards while still considering immediate outcomes, and ensures decisions are not too greedy or too delayed, resulting in optimal policy convergence faster with long-term awareness.

The proposed reinforcement learning-based security method demonstrated excellent performance, consistently outperforming comparable approaches reported in recent

literature. To comprehensively validate its effectiveness, we conducted experiments in two distinct phases:

1) **Simulation Phase**: In this phase, the proposed model was developed and tested using the NS-2.35 simulator on a simulated IoT-Edge communication network with synthetic attack scenarios.
2) **Real-time In-house Device Communication Phase**: In this phase, the proposed model was deployed on a physical setup using the ESP-WROOM-32 microcontroller to evaluate its efficiency and robustness under actual network conditions.

## 6.1 Simulation Phase

We started with the simulation phase and derived a novel reinforcement learning method to secure IoT-Edge computation. We termed this security method "Message Driven Reinforcement Learning" (MDRL). We have executed the network simulation with both non-malicious and malicious network configuration data, as already shown in **Figure 28**. Firstly, we set the **reinforcePolicyEnabled** flag to **false** for non-malicious simulation. With the trace file created with this non-malicious simulation, we calculate three network parameters: **1)** Packet Delivery Ratio (PDR), **2)** Average throughput, and **3)** Average End-to-End Delay (E2E delay). In the next step, we set this variable to **true**, which means our secure reinforcement policy is applied to the network simulation for evaluation and efficiency. Thus, we ran the network simulation with the same network configuration again for both non-malicious and malicious network configuration data and calculated these three network parameters again: **1)** Packet Delivery Ratio (PDR), **2)** Average throughput, and **3)** Average End-to-End Delay with the help of the trace files created for this non-malicious and malicious simulation. Increasing/decreasing the number of nodes in the simulated environment helps to capture more data to evaluate the efficiency of our reinforcement policy over the simulated network. We varied the total number of nodes to 25, 50, 75, and 100 in our network simulation. Our experimental results prove that our secure reinforcement policy can completely shut down DDOS attacks and make the network consistent even in the case of malware attacks.

We employed the AWK programming language to compute the three primary performance metrics: **1)** Packet Delivery Ratio (PDR), **2)** Average Throughput, and **3)** End-to-End Delay. AWK, named after its creators Aho, Weinberger, and Kernighan, is a versatile scripting language designed for efficient data extraction, manipulation, and report generation. Unlike compiled languages, AWK operates as an interpreted language, requiring no prior compilation. It supports the use of variables, numeric and string functions, conditional statements, and logical operators, making it particularly effective for processing large text-based datasets such as network simulation trace files. In this work, AWK was utilized to parse and analyze simulation output files, enabling the automated computation of the selected network performance metrics with high precision [124].

The NS-2.35 simulator-generated trace file (.tr) is used to compute the three primary performance metrics: **1)** Packet Delivery Ratio (PDR), **2)** Average Throughput, and **3)** End-to-End Delay. The following is the AWK Linux command used to calculate each individual.

awk -f **<awk-file-name>.awk <trace-file-name>.tr**

We created three separate "<awk-file-name>.awk" files to compute Packet Delivery Ratio (PDR), Average Throughput, and End-to-End Delay.

**Figure 40**, **Figure 42**, and **Figure 41** show the improvements in the network parameter values when a secure RL policy is applied between non-malicious and malicious simulations. We achieved the expected outcomes when we turned "**ON**" the **reinforcePolicyEnabled** flag in our simulations, and the calculated network parameter values were up to 98.23% accurate even on the malicious simulation.

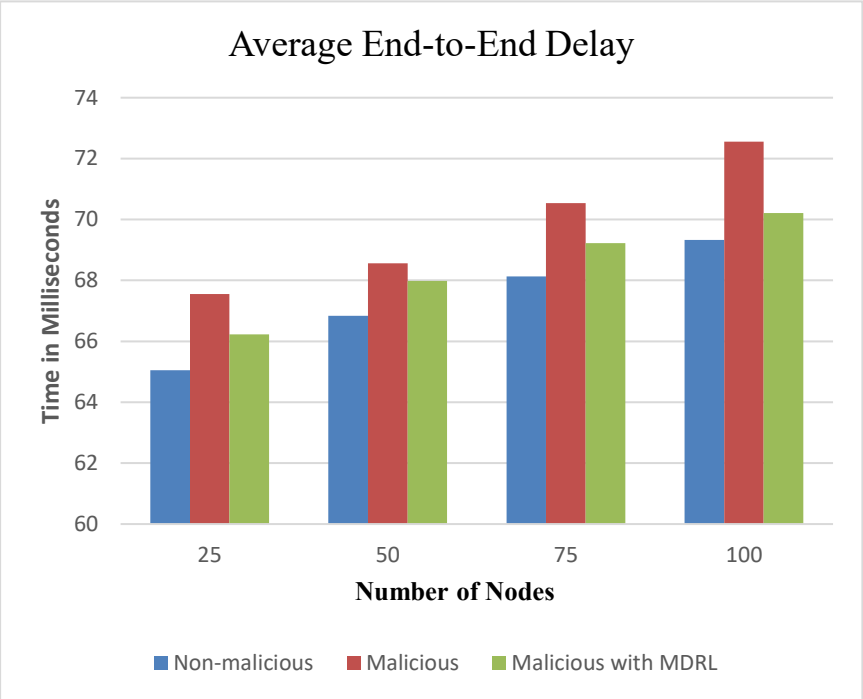



**Figure 40**: Average End-to-End Vs. Number of Nodes

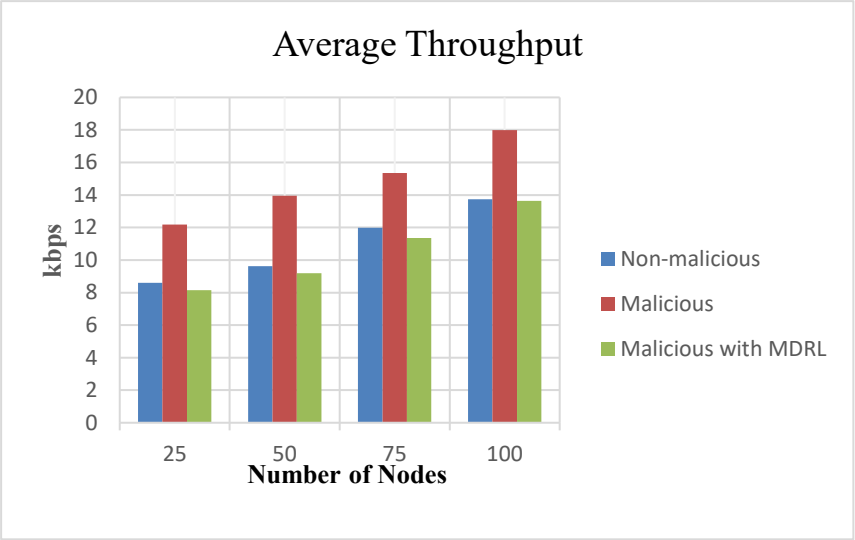



**Figure 42**: Average Throughput Vs. Number of Nodes

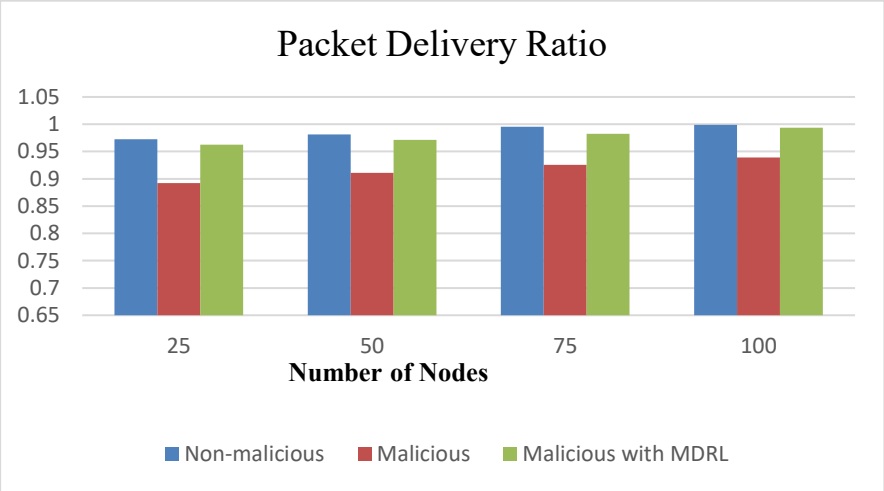



**Figure 41**: Packet Delivery Ratio Vs. Number of Nodes

### 6.1.1 Comparative Analysis of Experimental Results

Our proposed RL-based security model can provide the desired protection to the Edge computation for the data arriving on the Edge server from the IoT device. It also protects the IoT data while transferring from the Edge server to the Cloud server for permanent storage. It works comparatively better than similar security methods. Our work aligns with the approach proposed by [18], [33], [34], and [35] in its common goal of securing IoT data from the hands of intruders. Similar to us, they also discussed the heterogeneity among IoT devices, Edge servers, and Cloud servers on data communication. Our approach also addresses the problems and challenges they discussed, offering cost efficiency and performance advantages. Our approach is cost-effective as it does not require any extra hardware integration to add more to the network bandwidth, memory, and storage. Our proposed method is a software-based approach that models a security layer on top of the AODV communication protocol. The comparative performance evaluation result data are shown in **Table 10**. A graphical presentation of the comparison of our proposed security method, MDRL, with other available security methods is shown in **Figure 43**. We achieved 98.23% accuracy in detecting and preventing DDoS attacks in IoT-Edge computing and having a shallow error rate, which we strongly believe will keep the continuity and integrity of normal business flow to a great extent and would have a negligible impact when malware attacks happen during these times.

**Table 10:** Performance Evaluation Comparison

| **Researcher (s)** | **Security Model** | **Machine Learning method** | **Accuracy** |
|---|---|---|---|
| Proposed Methodology | MDRL | Reinforcement Learning | 98.23 % |
| Mishra et al. (2022) [40] | Blockchain and machine learning techniques | Machine learning classifiers | 96% |

| | | | |
|---|---|---|---|
| Bhunia et al. (2014) [39] | Honeynet-based defense mechanism | Honeynet-based Model | 95.8% |
| Xiao et al. (2016) [19] | Q-learning and Dyna-Q | Reinforcement Learning | 95% |
| Chaudhary et al. (2019) [36] | Support Vector Machine (SVM) learning | Supervised Learning | 90% |
| Ioannou et al. (2019) [34] | Support Vector Machine (SVM) learning | Supervised Learning | 81% |
| Khatun et al. (2019) [35] | Artificial Neural Network (ANN) | Deep Learning | 77.51% |

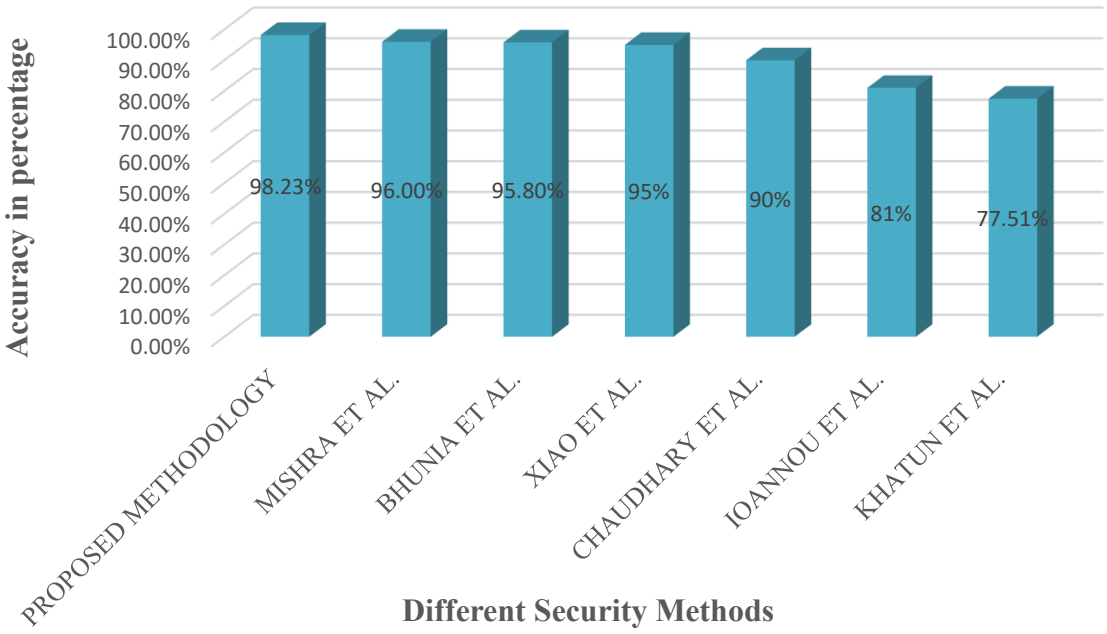



**Figure 43**: Comparative Analysis of Different Security Methods

## 6.2 Real-Time Communication Phase

We extended our research work to enable our proposed security model to support real-time in-house device communication. The proposed model in this phase is named the Adaptive Epsilon-Greedy Reinforcement Learning-based Security (AEGRL) method. As already discussed, in this phase, we started to validate our proposed model to provide the desired outcomes using public datasets [29] and then finally designed and implemented the model using real-time data communication using the Personal Computer Sensor Data (**Tool**: HWiNFO) [75], and an In-house IoT device ESP-WROOM-32 [41] [42]. In this phase, we extended the proposed method as an innovative reinforcement learning-based security approach to enhance the protection of edge computing in IoT networks. In this phase, again, the performance of our method was evaluated using three key metrics: 1) End-to-End (E2E) delay, 2) Packet Delivery Ratio (PDR), and 3) Average Throughput. Based on our experimental analysis and evaluation, we confirm that our approach meets expectations and outperforms existing methods in similar domains. For experimentation, we utilized sample input data of

varying sizes (50, 100, 150, and 200), with 25-30% of the data designated as the training dataset for our AEGRL security model. Each episode configuration is listed in **Table 11**. In **Figure 44**, we have shown the obtained accuracy variation of the proposed model on each episode.

**Table 11**: Data Packets Configuration

| **Episode #** | **Amount of malware data packets** |
|---|---|
| Episode 1 | 5-10% |
| Episode 2 | 10-20% |
| Episode 3 | 20-30% |
| Episode 4 | 30-40% |

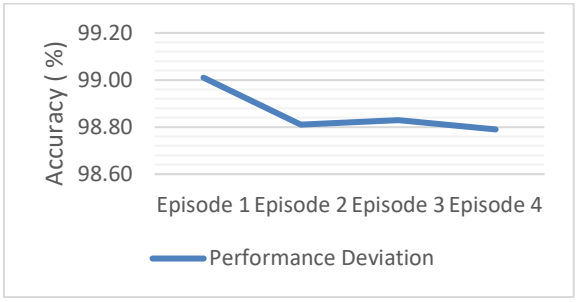



**Figure 44**: Performance deviation in measuring accuracy

Below **Figure 45** shows the variation of the performance metrics, including response time, detection rate, scalability, robustness, and consistency in a regular time interval of 1 second (1000 ms).

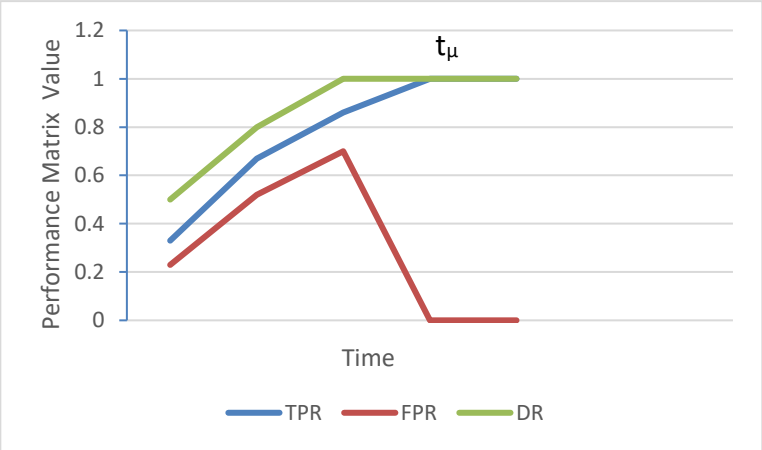



**Figure 45:** Performance Matrices Vs. Time Graph

### 6.2.1 Confusion Matrix

Machine learning models are widely employed for classification tasks across various domains. Evaluating their performance is essential to ensure accuracy and reliability. A key tool in this assessment is the confusion matrix, which visually represents the model's predictions against actual outcomes. It is a key evaluation metric in machine learning, particularly for classification problems, which helps visualise the performance of a classification algorithm by comparing actual versus predicted classifications. It categorises predictions into four categories: correct predictions for both classes (**true positives** and **true negatives**) and incorrect predictions (**false positives** and **false negatives**). By analysing these metrics, the confusion matrix helps identify areas where the model misclassifies data, providing insights for improvement [103] [104] [105]. We have used binary classification for detecting data among classes: **1)** Malicious (Positive class), and **2)** Benign (Negative class). We drew the confusion matrix for a total of 10,000 data packets, and the classification of these data packets resulting from the experiment is shown in **Table 12**. The visual representation of the confusion matrix is shown in **Figure 46**.

**Table 12:** Data Packets Count for the Confusion Matrix Diagram

| | **Predicted Benign (No. of data packets)** | **Predicted Malicious (No. of data packets)** |
|---|---|---|
| **Actual Benign (No. of data packets)** | True Negative (7962) | False Positive (38) |
| **Actual Malicious (No. of data packets)** | False Negative (25) | True Positive (1975) |

By using the confusion matrix, we performed the following operations.

1) Adjusted the reward function based on False Positive (FP) and False Negative (FN) rates.
2) Controlling and limiting the False Negatives more heavily if security sensitivity is high.
3) Train the agent to minimize costly errors, improving learning efficiency.

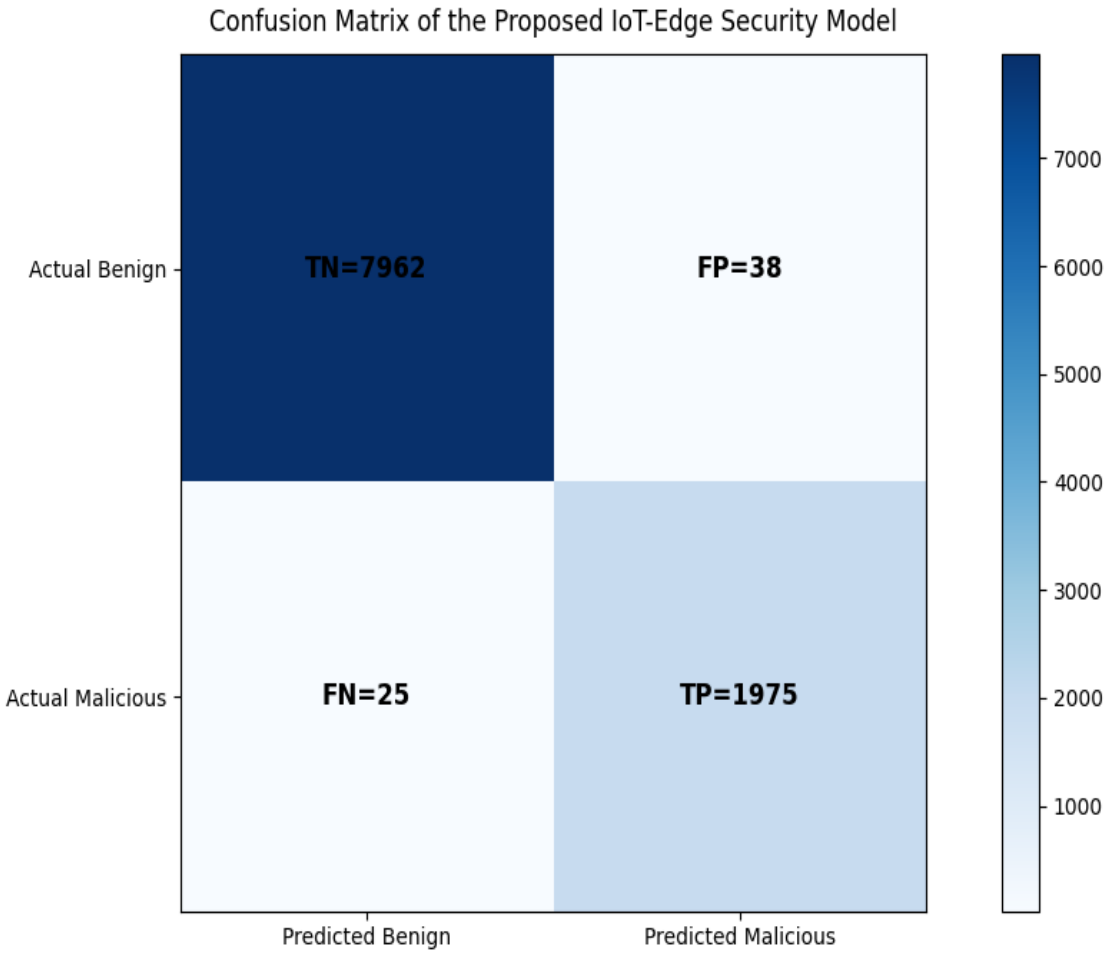



**Figure 46:** Confusion Matrix Diagram of the Proposed Security Model

Already discussed, Equations (10) and (12) calculate the Accuracy ($A_{cc}$) and Detection Rate (DR) of our proposed IoT-Edge security model as follows.

$$Acc = \frac{1975 + 7962}{1975 + 7962 + 38 + 25} = 0.9937$$

$$DR = \frac{1975}{1975 + 25} = 0.9875$$

Hence, the proposed IoT-Edge security model achieves 99.37% accuracy and 98.75% detection rate, respectively.

## 6.3 Performance Evaluation Based on Hyperparameters

Our proposed novel reinforcement learning security approach is based on the SARSA algorithm [54], which is as follows, by using equation **(22)**.

$$Q(s,a) \leftarrow Q(s,a) + \alpha\,[\mathrm{r} + \gamma\,\mathrm{Q}(\mathrm{s}',\mathrm{a}') - \mathrm{Q}(\mathrm{s},\mathrm{a})] \qquad \textbf{(22)}$$

Here, α represents the learning rate, and γ is the discount factor. It updates the value of taking action (a) in state (s). After executing this action, the environment returns a reward (r), transitions to a new state (s'), and selects a new action in the state (s').

The Reinforcement Learning (RL) agent should be given the same task, starting state, and environment consistently for periodic evaluation. During training, the evaluation score fluctuates, indicating that the Q-function has been significantly updated due to changes in network parameters. When a new action in a given state attains the highest value, it suggests that learning is ongoing and has not stagnated. This implies that the agent is actively refining its policy, whether improving or deteriorating, based on the learned experiences [72]. In network security, machine learning (ML) models are frequently employed for tasks such as intrusion detection, malware classification, anomaly detection, and threat prediction. Accuracy ($A_{cc}$) is one of the key metrics used to evaluate how well these models perform [102]. We have fine-tuned the proposed security method with the following hyperparameters and by calculating the Accuracy of the model.

a) Learning rate (α)
b) Discount factor (γ)

### 6.3.1 Learning Rate (α)

The optimal learning rate for an RL agent depends on several factors, including the network's size, the complexity of the task, and the amount of available training data. However, fine-tuning the learning rate through experimentation is often necessary to achieve optimal performance for a specific model. A learning rate that is too high may lead to overfitting, where the model becomes overly reliant on the training data and fails to generalize to unseen data. Conversely, a learning rate that is too low can result in underfitting, preventing the model from effectively capturing patterns within the data

and making accurate predictions. The learning rate has a significant impact on the RL agent's performance as a crucial hyperparameter. Proper tuning of this parameter can enhance training stability and enable the model to achieve state-of-the-art results [102]. The periodic evaluation scores will exhibit significant fluctuations if the learning rate is too high. This occurs because the Q-function, approximated by the RL agent, undergoes drastic changes after each optimization step. As a result, the RL agent may struggle to converge to an optimal solution, leading to unstable learning dynamics and suboptimal policy updates. The impact of the Learning Factor on the Accuracy of the Proposed Method is shown in **Figure 47**. We fine-tuned the learning rate using the following steps:

**Step 1**: Initialize with a very low learning rate with a value of 0.001.

**Step 2**: Perform the training steps with 200 iterations on each episode.

**Step 3**: Monitor fluctuations in the evaluation score during these steps. If no significant fluctuations are observed, increase the learning rate by a factor of 1.05 and repeat Step 2.

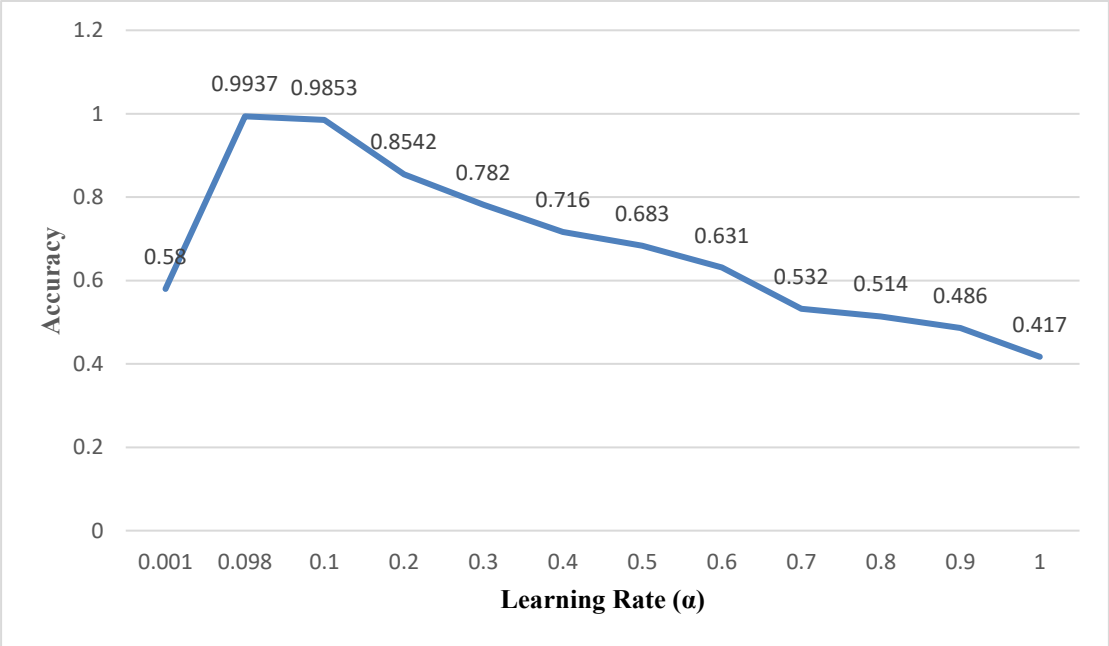



**Figure 47**: Impact of the Learning Factor on the Accuracy of the Proposed Method

### 6.3.2 Discount Factor (γ)

In reinforcement learning (RL), the choice of the discount factor (γ) plays a crucial role in shaping the trade-off between short-term and long-term rewards during policy optimization. A large discount factor emphasizes the value of future rewards, encouraging long-term planning. However, in environments with high stochasticity or non-stationary dynamics, this can lead to significant challenges. Future rewards in such settings are difficult to estimate accurately, and overemphasis on them may cause overestimation bias, oscillations in policy updates, or unstable convergence of the value function. On the other hand, a small discount factor favours immediate rewards, often leading to faster but myopic learning, where the agent optimizes for short-term gains at the expense of overall performance. While this can improve stability and generalization in volatile environments, it may also produce suboptimal policies in tasks where a long-term strategy is essential. This dynamic perspective on discount factor selection represents a promising avenue for advancing RL in complex, real-world applications such as autonomous driving, financial trading, and large-scale IoT network optimization. This motivates the development of adaptive mechanisms that update the discount factor in response to environmental feedback and agent performance. The best range of the discount factor in reinforcement learning depends on the nature of the problem intended to solve, and how much importance should be placed on future rewards versus immediate rewards. In our proposed adaptive reinforcement learning security method, we have varied the discount factor (γ) within the range 0.6 to 0.99. The accuracy of the proposed model, depending upon the adjusted discount factor, is shown in **Figure 48**. We achieved the highest accuracy of 99.37% at the discount factor value 0.90. This adjustment of the discount factor triggers balanced outcomes but still leans towards short-term goals. This range is suitable for real-time tasks where quick reactions matter [106].

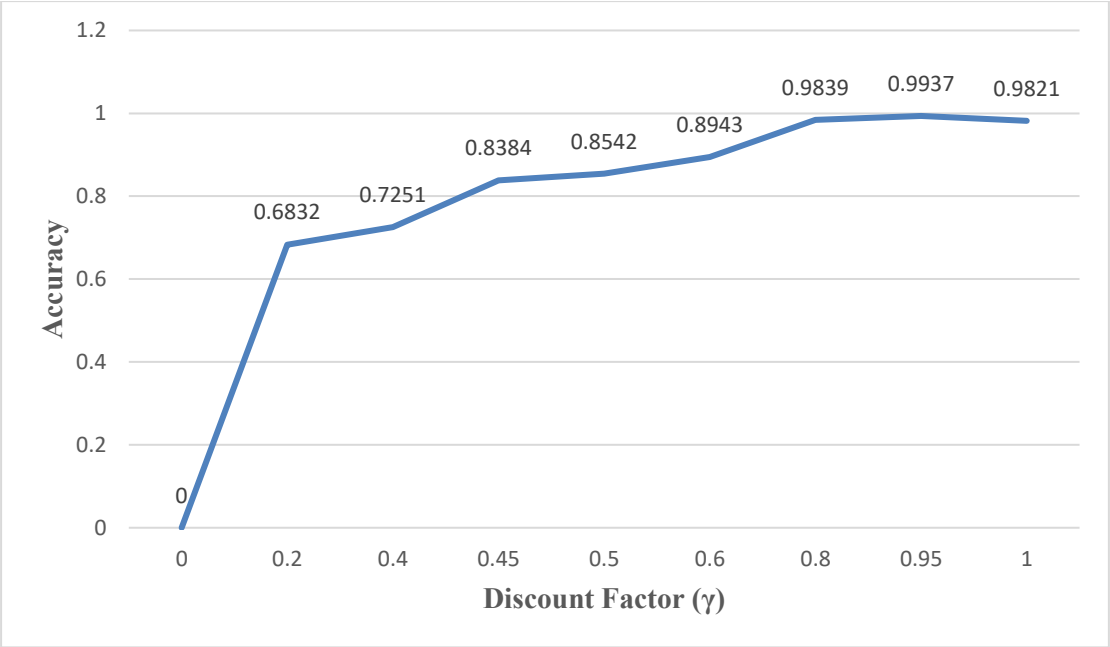



**Figure 48:** Impact of the Discount Factor on the Accuracy of the Proposed Method

## 6.4 Determination of Epsilon-Greedy Learning Policy

The epsilon-greedy policy is a commonly used method in reinforcement learning for balancing between exploring new actions and exploiting known ones. It mostly selects the action that currently seems to give the highest reward, which is known as Exploitation, while occasionally trying out random actions, which is known as Exploration [66]. This behaviour is governed by a parameter, epsilon (Ɛ), which ranges between 0 and 1. A high (Ɛ) value encourages greater exploration, which is beneficial during the early stages of learning when the environment is largely unknown, while a low ε value favors exploitation once the agent has accumulated sufficient knowledge. To improve efficiency, (Ɛ) is often decayed over time, allowing the agent to explore extensively at first and gradually shift toward exploiting learned policies as it converges. Although simple and computationally inexpensive, the epsilon-greedy policy can be inefficient in environments with large or continuous action spaces, where random exploration may rarely lead to meaningful discoveries. For instance, an Ɛ-value of 0.1 implies that the agent will choose a random action 10% of the time, and the optimal known action 90% of the time. This strategy enables the agent to avoid getting

stuck in suboptimal strategies by continually exploring better possibilities as learning progresses [107].

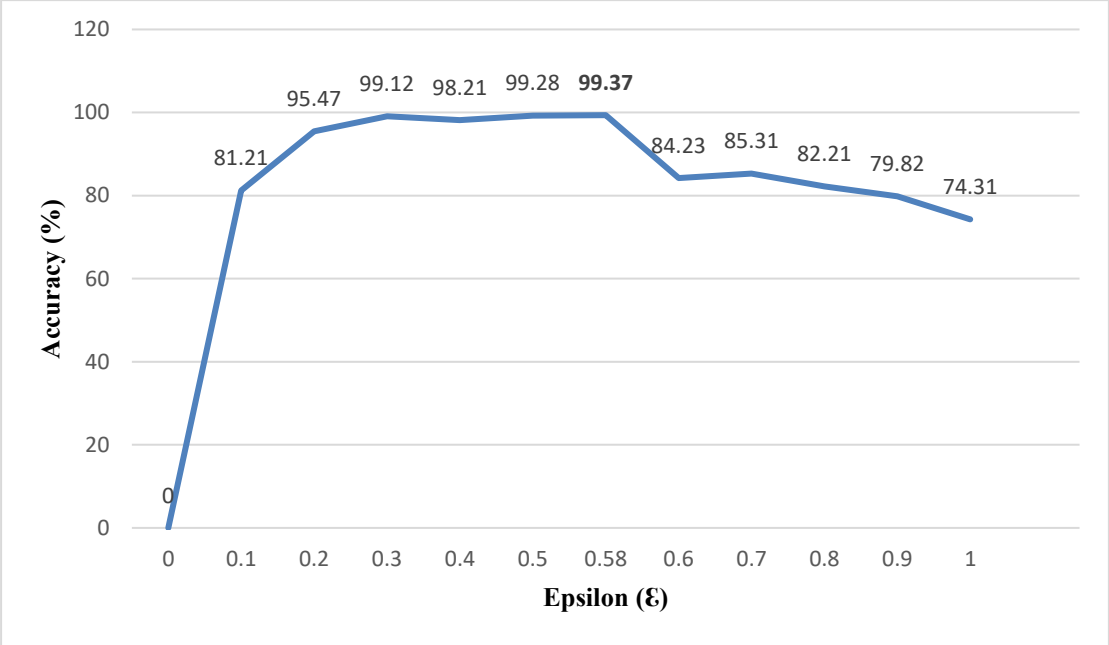



**Figure 49:** Impact of Epsilon (Ɛ)-value on the Accuracy of the Proposed Model

The proposed model has the epsilon (Ɛ) variance that depends on the network traffic of malicious data. The model achieves the highest accuracy of 99.37% at the Ɛ-value of 0.098, which is also depicted in **Figure 49**. At the moment of the start of the training steps, we kept the Ɛ-value equal to 1. It signifies that the Agent needs much exploration at the beginning to understand the environment and initial policy formation. By the time, there is less exploration and more exploitation needed to select the best action out of the possible actions derived from the policy updated time-to-time. Hence, Ɛ-value decreases with the increase of the Training steps, which is as depicted in **Figure 50**.

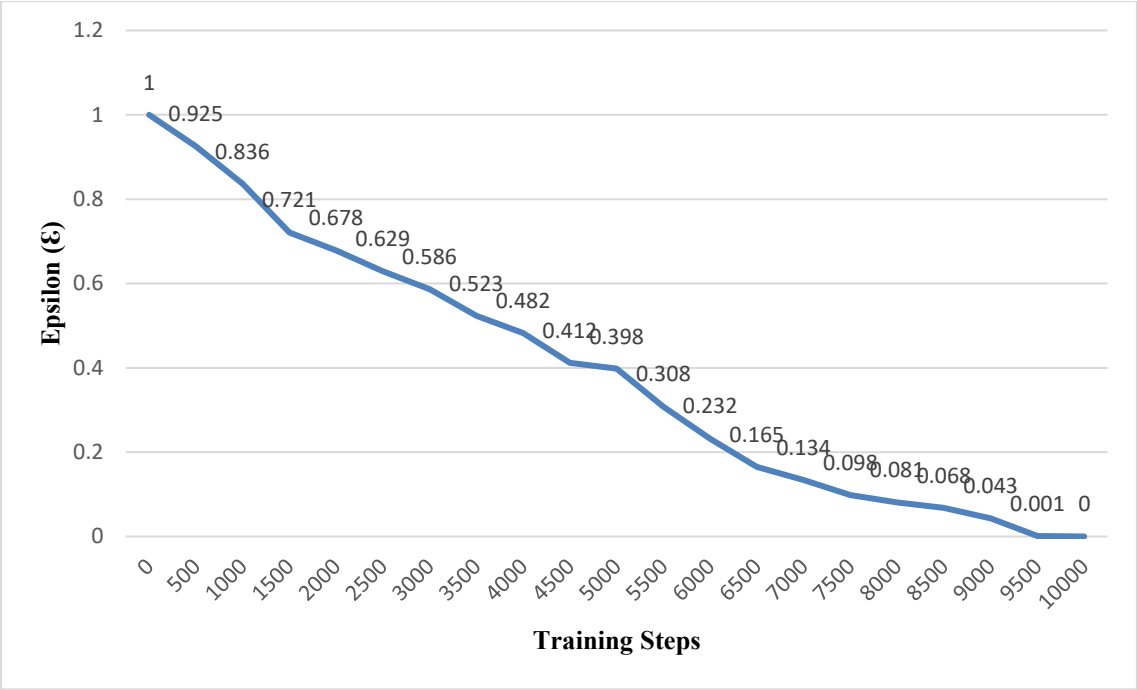



**Figure 50:** Impact of Epsilon (Ɛ) on the Training Steps of the RL Agent

### 6.4.1 Determination of Novel Adaptive Epsilon Value in Q-learning

The proposed AEGRL method has a very low learning rate (α) in the range of (0, 1) with a median slightly variant discount factor (γ) in the range of (0.4, 0.6). With this discount factor value, the proposed algorithm can balance the current set of rewards and long-term future rewards and benefit them simultaneously. With the initial requirements of exploration of the proposed algorithm, we set the epsilon (Ɛ) value to 0.001, and once the iteration steps reach up to 10,000 and within time steps of 100 milliseconds, the Ɛ-value reaches near zero. Until now, the proposed algorithm has achieved the optimal policy (π), which can be exploited for the rest of the data communication time. The AEGRL is found to work excellently with this novel epsilon (Ɛ) value in the range of (0, 0.1) regarding scalability, consistency, robustness, latency, and detection rate. The scalability is evaluated up to 100 IoT nodes, whereas the detection rate is up to 99.37%. The maximum latency of the proposed security method is measured under 600 microseconds. The consistency and robustness are evaluated with the attacker node count of 10%-50% of the total node count in the network.

## 6.5 Comparative Analysis of Experimental Results

The results of our proposed method, AEGRL, are presented in **Table 13**, **Table 14**, and **Table 15** provide the E2E delay, PDR, and Average throughput values, respectively, observed between non-malicious and malicious networks.

**Table 13:** E2E delay b/w non-malicious and malicious communication using the proposed AEGRL security method

| **NON-MALICIOUS** | | **MALICIOUS** | |
|---|---|---|---|
| **DATA SIZE** | **AVERAGE E2E DELAY IN MILLISECONDS** | **DATA SIZE** | **AVERAGE E2E DELAY IN MILLISECONDS** |
| 500 | 987 | 50 | 1233 |
| 1000 | 1220 | 100 | 1418 |
| 1500 | 1529 | 150 | 1813 |
| 2000 | 1932 | 200 | 2438 |

**Table 14:** PDR b/w non-malicious and malicious communication using the proposed AEGRL security method

| NON-MALICIOUS | | MALICIOUS | |
|---|---|---|---|
| **DATA SIZE** | **PACKET DELIVERY RATIO** | **DATA SIZE** | **PACKET DELIVERY RATIO** |
| 500 | 1 | 50 | 0.691 |
| 1000 | 1 | 100 | 0.732 |
| 1500 | 1 | 150 | 0.813 |
| 2000 | 1 | 200 | 0.891 |

**Table 15:** Average Throughput b/w non-malicious and malicious communication using the proposed RL security method

| NON-MALICIOUS | | MALICIOUS | |
|---|---|---|---|
| **DATA SIZE** | **AVERAGE THROUGHPUT (KBPS)** | **DATA SIZE** | **AVERAGE THROUGHPUT (KBPS)** |
| 500 | 4.339 | 50 | 4.312 |
| 1000 | 6.127 | 100 | 6.233 |
| 1500 | 9.631 | 150 | 9.804 |
| 2000 | 12.383 | 200 | 12.892 |

The proposed AEGRL security approach performed exceptionally well regarding the Packet Delivery Ratio (PDR), ensuring no data packets were lost during communication. As a result, the average throughput of data packets remained unaffected even during malicious attacks. However, the primary challenge lies in

managing end-to-end (E2E) delay, as an influx of malicious data packets at the edge server can increase latency, leading to slower responsiveness for users. **Figure 51** illustrates the E2E delay observed in non-malicious and malicious communication networks under our proposed reinforcement learning (RL) security layer. **Figure 52** highlights the PDR achieved in these networks, demonstrating the robustness of our approach. Similarly, **Figure 53** depicts the average throughput comparisons between non-malicious and malicious communication networks under the RL security layer. The node setup for the IoT Data Communication Network is already shown in **Figure 39**. It includes scenarios where multiple attacks generate malicious data packets targeting the IoT-Edge-Cloud communication channel. This setup effectively validates the resilience and efficiency of our proposed security framework.

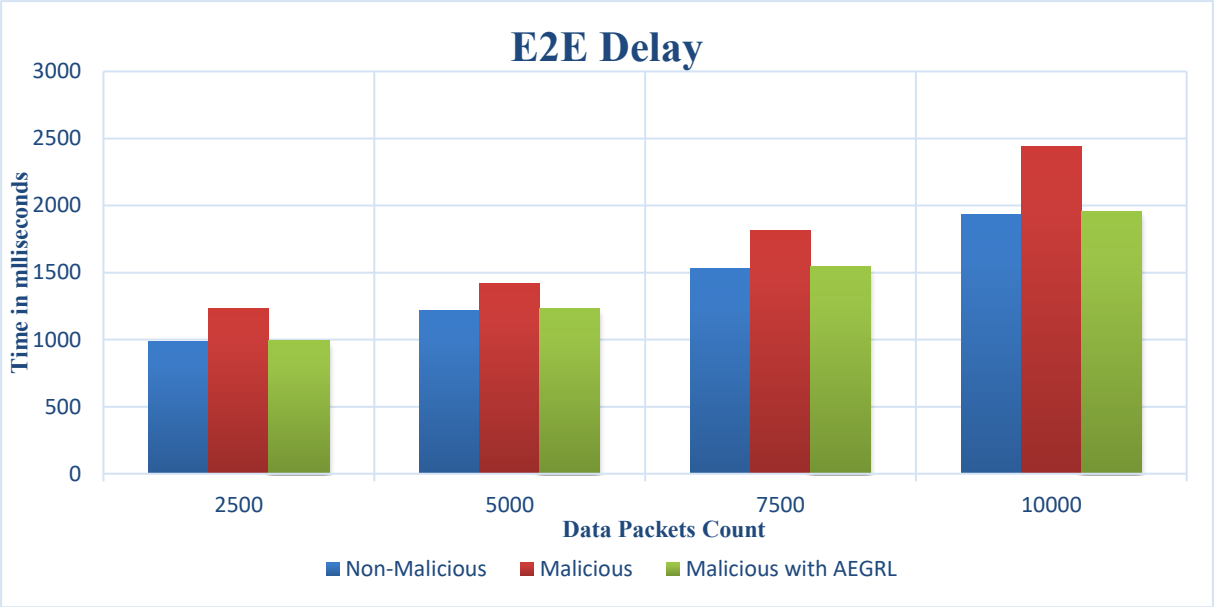



**Figure 51:** E2E Delay b/w Non-malicious and Malicious Communication with the Proposed Security Layer

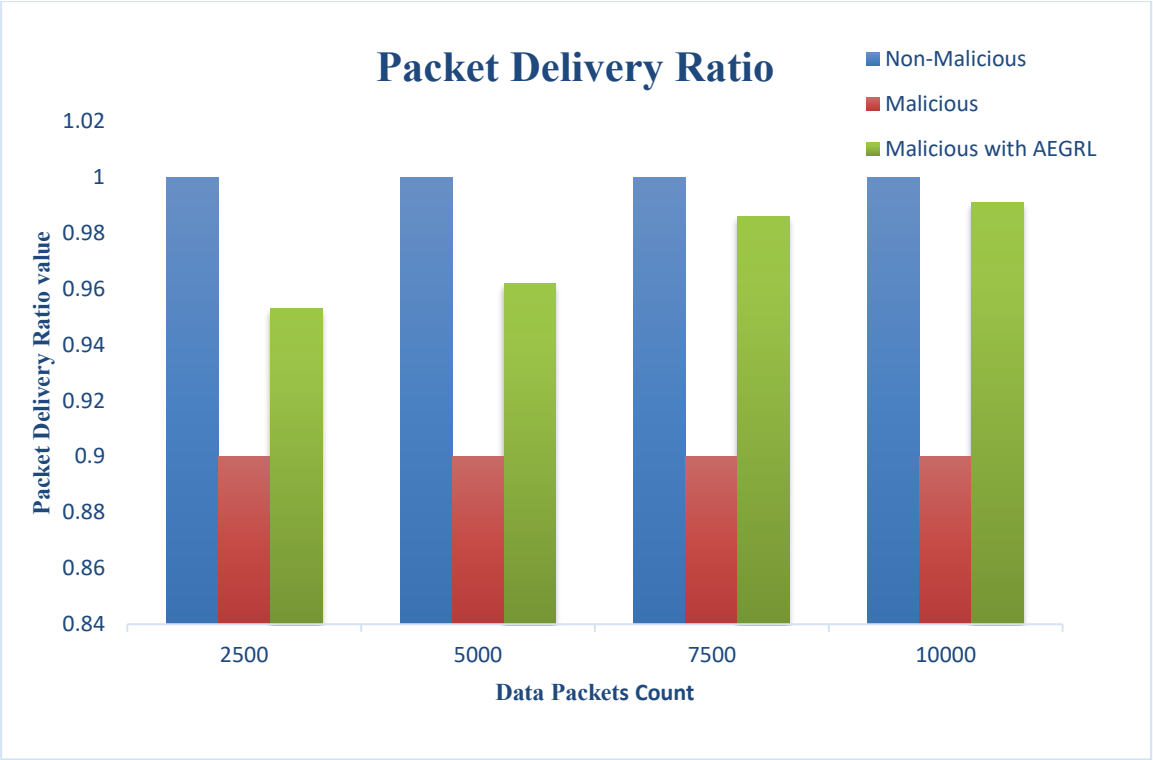



**Figure 52:** PDR b/w Non-malicious and Malicious Communication with the Proposed Security Layer

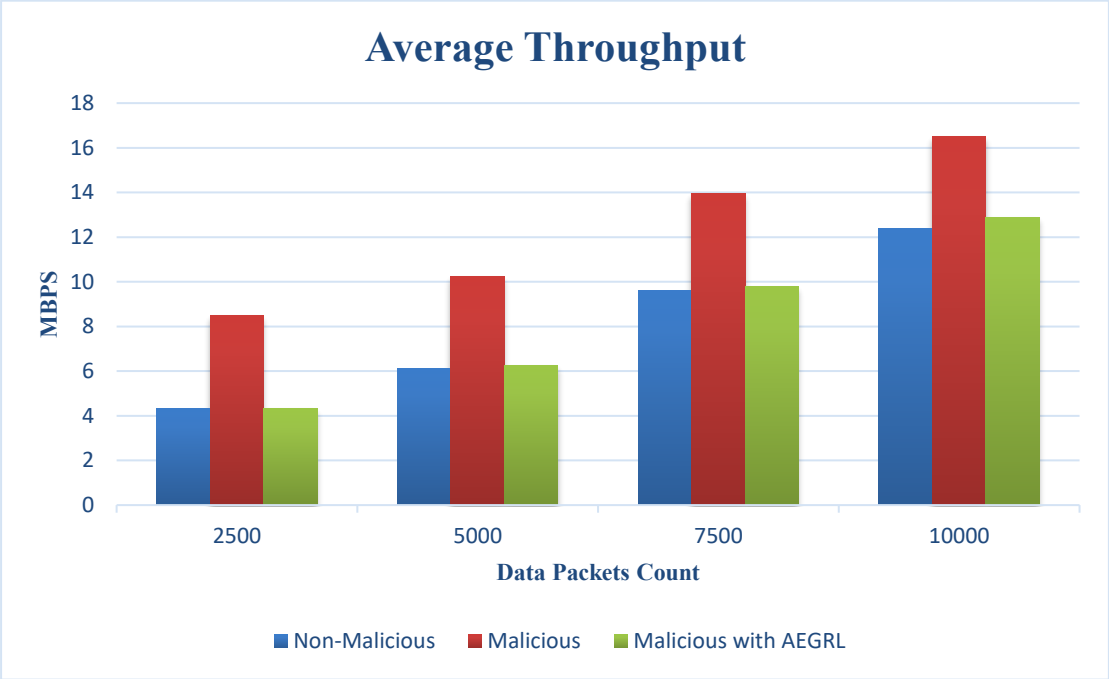



**Figure 53:** Average Throughput b/w Non-malicious and Malicious Communication with the Proposed Security Layer

**Table 16:** Performance Evaluation of AEGRL Comparison with Existing Security Methods

| **Researcher (s)** | **Research Parameter (s)** | **Performance Metrics** | **Type of Attack (s) Detected** | **Proposed Methodology** | **Accuracy (%)** |
|---|---|---|---|---|---|
| Proposed Method | E2E Delay, Packet Delivery Ratio, and Average Throughput | Network Security, Scalability | DoS/DDoS | Novel Reinforcement Learning | 99.37 |
| Geetha et al. (2024) [130] | Packet Delivery Ratio | Network Security | DoS/DDoS | Support Vector Machine (SVM) | 96.4 |
| Louai A. Maghrabi (2024) [129] | Packet Delivery Ratio | Network Security | DoS/DDoS | Random Forest (RF) | 90.17 |
| Mishra et al. (2022) [40] | Scalability, Packet Delivery Ratio | Network Security | Data poisoning attacks, Data breaching attacks | Blockchain and machine learning techniques | 96 |
| Ullah et al. (2021) [136] | Packet Delivery Ratio | Network Security | DoS/DDoS | Convolutional Neural Network (CNN) | 99.03 |
| Sudheera et al. (2021) [135] | Packet Delivery Ratio | Network Security | DoS/DDoS | Machine Learning-based Classifiers | 99 |

| | | | | | |
|---|---|---|---|---|---|
| Bhunia et al. (2014) [39] | Packet Delivery Ratio, E2E Delay | Network Security | Jamming | Honeynet-based defense mechanism | 95.8 |
| Xiao et al. (2016) [19] | Radio channel utilization | Network Security | Spoofing | Reinforcement Learning | 95 |
| Zhang et al. (2023) [38] | E2E Delay | Network Security | DoS/DDoS | Deep Reinforcement Learning | 90 |
| Malialis et al. (2015) [16] | Packet Delivery Ratio | Network Security, Scalability | DoS/DDoS | Reinforcement Learning | 88 |
| Ioannou et al. (2019) [34] | Packet Delivery Ratio | Network Security, Scalability | Routing | Support Vector Machine (SVM) | 81 |
| Khatun et al. (2019) [35] | Packet Delivery Ratio, E2E Delay | Network Security | DoS/DDoS | Deep Reinforcement Learning | 77.51 |

Additionally, the performance of our proposed method relative to other similar security approaches is detailed in **Table 16**. A graphical representation of the comparative analysis of the proposed model in terms of accuracy is depicted in **Figure 54**, highlighting the superior performance of our approach. These results demonstrate the efficacy and reliability of our method in securing IoT-edge computing environments. The proposed security model performs as expected and is capable of bringing the desired output in terms of E2E delay, PDR, and Average throughput values. The calculated values on the performance metrics are shown below, which prove the efficiency of our proposed security model. The results are based on the public and in-house IoT datasets. It also outperforms the baseline and the state-of-the-art existing security mechanisms to protect IoT-Edge computations.

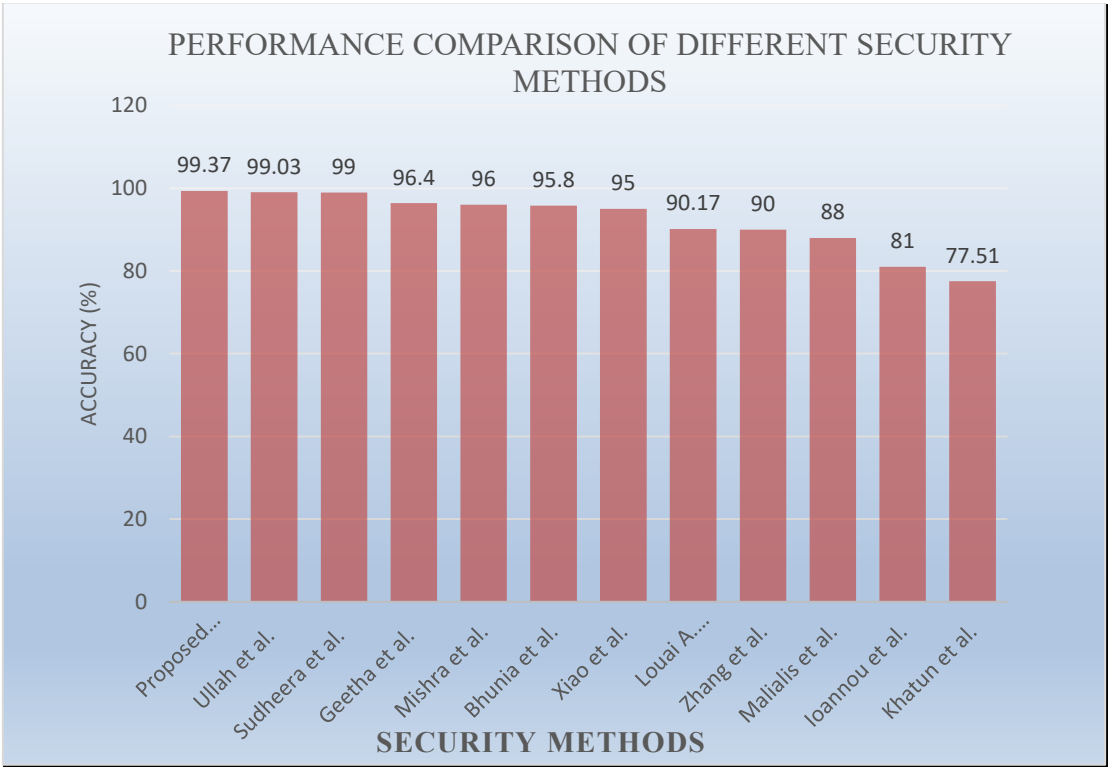



**Figure 54:** Performance Evaluation Comparison of AEGRL with Other Security Methods

## 6.6 Research Summary

We designed and implemented a novel reinforcement learning (RL) driven security framework tailored for safeguarding IoT-Edge computational environments against network-level threats, particularly Distributed Denial-of-Service (DDoS) attacks. The core idea of the model revolves around dynamically evaluating environmental conditions in the vicinity of the potential attack region, enabling an intelligent agent to make adaptive security decisions based on continuous feedback. To simulate communication and threat scenarios, we leveraged the Ad hoc On-Demand Distance Vector (AODV) routing protocol [37], which was selected due to its reactive nature, making it highly suitable for resource-constrained IoT environments where route discovery is initiated only when required.

The experimentation was conducted in two distinct phases, starting with a pure simulation-based evaluation. In this first phase, we utilized NS-2.35, a well-established and extensible network simulation tool [32] [33], capable of simulating large-scale,

multi-hop wireless communication scenarios. This platform allowed us to create custom modules for RL agent integration, malicious traffic generation, and Edge node interaction. We developed scenario files to define network topology, including node positions, communication flows, and attack vectors. These scenario files consisted of IoT nodes, Edge nodes, and malicious attacker nodes, each assigned specific coordinate locations to emulate real-world IoT device behaviour.

Again, we extended our research work to enable the proposed novel reinforcement learning security model to function optimally in real-time IoT network communication using the MQTT communication protocol. For this, we started with public datasets [29] to improve and validate our security model, and then finally design and implement the proposed security model using real-time data communication with the in-house IoT device ESP-WROOM-32 [41] [42]. We used the Node-RED server [47], which is ideal to run at the edge of the network on low-cost hardware. Our approach can efficiently detect DDoS attacks on IoT-Edge computation. The experimental results show that our proposed method outperforms other similar security methods efficiently.

# Chapter 7

## Conclusion and Future Works

In this research work, we worked to show and prove how the reinforcement learning method can be used to detect and prevent DDOS Attacks on Edge Computing of IoT Devices. Again, we proposed a novel reinforcement learning methodology that is much more efficient and able to almost shut down the malware attacks completely on the underlying communication network. Our proposed novel reinforcement learning security model for protecting IoT-Edge computation was efficient in both the simulation phase and the real-time in-house device communication phase. We emphasized the need for IoT-Edge computing and, at the same time, discussed the security issues and challenges of the IoT computation mechanism, which is a paramount concern for organizations. Our proposed methodology defines and establishes a much better security framework as compared with other similar approaches. Our security model does not require any extra hardware integration among the IoT device, Edge server, and Cloud server communication channels. Hence, it is also a cost-effective method to provide IoT device security in the case of Edge computing with a Cloud server. The proposed security approach worked excellently in terms of Packet Delivery Ratio (PDR), Average throughput, and End-to-End (E2E) delay.

However, modern IoT networks are becoming increasingly complex and dynamic, requiring more advanced communication strategies. Therefore, in our future work, we aim to enhance the security with a more robust, resilient, and scalable security protocol for IoT-Edge computation, which will support a wide range of IoT devices with current industry demands and can be easily integrated with any existing IoT communication protocols. We intend to focus on current demand industry security standards, compliance frameworks, and protocols for interoperability across vendors and ecosystems. We also intend to work to enhance the proposed model to implement efficient security measures between heterogeneous IoT-Edge-Cloud layers, and also work on enhancing the self-healing and fault-tolerant capability of the proposed security model, which allows it to continue functioning under diverse security attacks.

# REFERENCES


[1] Anit Kumar, and Dhanpratap Singh, “Securing IoT Devices in Edge Computing Through Reinforcement Learning”, Computers & Security, 2025, 104474, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2025.104474

[2] Anit Kumar, and Dhanpratap Singh, "Reinforcement learning based security method for IoT-Edge computing.", Wireless Networks, 2026, https://doi.org/10.1007/s11276-026-04123-5

[3] Anit Kumar, and Dhanpratap Singh, “Adaptive epsilon greedy reinforcement learning method in securing IoT devices in edge computing.”, Discover Internet of Things 4, no. 1 (2024): 27.

[4] Anit Kumar, and Dhanpratap Singh, “Detection and prevention of DDoS attacks on edge computing of IoT devices through reinforcement learning.”, International Journal of Information Technology 16.3 (2024): 1365-1376.

[5] Anit Kumar, and Priyanka Chawla, “A systematic literature review on load balancing algorithms of virtual machines in a Cloud computing environment.”, In Proceedings of the International Conference on Innovative Computing & Communications (ICICC). 2020.

[6] Anit Kumar, and Dhanpratap Singh, “Detection of Security Attacks on Edge Computing of IoT Devices through NS2 Simulation.”, In Journal of Physics: Conference Series, vol. 2327, no. 1, p. 012016. IOP Publishing, 2022.

[7] Anit Kumar, and Dhanpratap Singh, “A Comprehensive Survey on Security Attacks to Edge Server of IoT Devices through Reinforcement Learning.”, In Proceedings of the AICTE-sponsored National Conference on Multidisciplinary Research and Innovation-21 (NCMRAI-21). 2022.

[8] Anit Kumar, and Dhanpratap Singh, “Resource-Aware Dynamic Load Balancing System in Heterogeneous Distributed Computing Systems through Swarm Intelligence.”, Available at SSRN 3734997 (2020).

[9] Fox, Garett, and Rajendra V. Boppana, “Detection of malicious network flows with low preprocessing overhead.”, Network 2, no. 4 (2022): 628-642.

[10] Uprety, Aashma, and Danda B. Rawat, “Reinforcement Learning for IoT Security: A Comprehensive Survey.”, IEEE Internet of Things Journal 8.11 (2020): 8693–8706.

[11] Albishry, Nabeel, Rayed AlGhamdi, Abdulmohsen Almalawi, Asif Irshad Khan, and Pravin R. Kshirsagar, “An attribute extraction for automated malware attack classification and detection using soft computing techniques.”, Computational Intelligence and Neuroscience 2022 (2022).

[12] Accessed on 21 March 2025, “https://www.geeksforgeeks.org/multi-armed-bandit-problem-in-reinforcement-learning/”

[13] dos Santos Mignon, Alexandre, and Ricardo Luis de Azevedo da Rocha, “An adaptive implementation of ε-greedy in reinforcement learning.”, Procedia Computer Science 109 (2017): 1146-1151.

[14] Zhang, Tengyue, Hong Wen, Yixin Jiang, and Jie Tang, “Deep Reinforcement Learning Based IRS for Cooperative Jamming Networks under Edge Computing.”, IEEE Internet of Things Journal (2023).

[15] Elgendy, Ibrahim A., Wei-Zhe Zhang, Yiming Zeng, Hui He, Yu-Chu Tian, and Yuanyuan Yang, “Efficient and secure multi-user multi-task computation offloading for mobile-edge computing in mobile IoT networks.”, IEEE Transactions on Network and Service Management 17, no. 4 (2020): 2410-2422.

[16] Malialis, Kleanthis, and Daniel Kudenko, “Distributed response to network intrusions using multiagent reinforcement learning.” Engineering Applications of Artificial Intelligence 41 (2015): 270-284.

[17] Khoda, Mahbub E., Tasadduq Imam, Joarder Kamruzzaman, Iqbal Gondal, and Ashfaqur Rahman, “Robust malware defense in industrial IoT applications using machine learning with selective adversarial samples”, IEEE Transactions on Industry Applications 56, no. 4 (2019): 4415-4424.

[18] Gottipati, Sai Krishna, Yashaswi Pathak, Rohan Nuttall, Raviteja Chunduru, Ahmed Touati, Sriram Ganapathi Subramanian, Matthew E. Taylor, and Sarath Chandar, “Maximum reward formulation in reinforcement learning.”, arXiv preprint arXiv:2010.03744 (2020).

[19] Xiao, Liang, Yan Li, Guoan Han, Guolong Liu, and Weihua Zhuang, “PHY-layer spoofing detection with reinforcement learning in wireless networks”, IEEE Transactions on Vehicular Technology 65, no. 12 (2016): 10037-10047.

[20] Li, Xiong, Maged Hamada Ibrahim, Saru Kumari, and Rahul Kumar, “Secure and efficient anonymous authentication scheme for three-tier mobile healthcare systems with wearable sensors”, Telecommunication Systems 67 (2018): 323-348.

[21] Lin, Fuhong, Yutong Zhou, Xingsuo An, Ilsun You, and Kim-Kwang Raymond Choo, “Fair resource allocation in an intrusion-detection system for edge computing: Ensuring the security of Internet of Things devices”, IEEE Consumer Electronics Magazine 7, no. 6 (2018): 45-50.

[22] Hsu, Ruei-Hau, Jemin Lee, Tony QS Quek, and Jyh-Cheng Chen., “Reconfigurable security: Edge-computing-based framework for IoT.” IEEE Network 32, no. 5 (2018): 92-99.

[23] Yuan, Jie, and Xiaoyong Li., “A reliable and lightweight trust computing mechanism for IoT edge devices based on multi-source feedback information fusion”, IEEE Access 6 (2018): 23626-23638.

[24] Arfaoui, Amel, Ali Kribeche, and Sidi-Mohammed Senouci. “Context-aware anonymous authentication protocols in the internet of things dedicated to e-health applications.”, Computer Networks 159 (2019): 23-36.

[25] Li, Bingcong, Tianyi Chen, and Georgios B. Giannakis, “Secure mobile edge computing in IoT via collaborative online learning.” IEEE Transactions on Signal Processing 67, no. 23 (2019): 5922-5935.

[26] F. Qiao, J. Wu, J. Li, A. K. Bashir, S. Mumtaz and U. Tariq (2020), “Trustworthy Edge Storage Orchestration in Intelligent Transportation Systems Using Reinforcement Learning.” in IEEE Transactions on Intelligent Transportation Systems, doi: 10.1109/TITS.2020.3003211.

[27] Huang, Yunhan, Linan Huang, and Quanyan Zhu, “Reinforcement learning for feedback-enabled cyber resilience”, Annual reviews in control 53 (2022): 273-295.

[28] A.A. Diro, and N. Chilamkurti (2017), “Distributed Attack Detection Scheme using Deep Learning Approach for Internet of Things”, Future Generation Computer Systems, http://dx.doi.org/10.1016/j.future.2017.08.043

[29] Accessed on 22 March 2025, https://mcfp.felk.cvut.cz/publicDatasets/IoT-23-Dataset/IndividualScenarios/CTU-IoT-Malware-Capture-34-1/

[30] Accessed on 22 March 2025, https://mosquitto.org/"

[31] Accessed on 22 March 2025, https://cookbook.nodered.org/mqtt/connect-to-broker

[32] Accessed on 22 March 2025, https://www.isi.edu/nsnam/ns/tutorial/

[33] Accessed on 22 March 2025, https://www.nsnam.com/2020/06/installation-of-ns2-ns-235-in-ubuntu.html

[34] Ioannou, Christiana, and Vasos Vassiliou, "Classifying security attacks in IoT networks using supervised learning." in 15th International Conference on Distributed Computing in Sensor Systems (DCOSS). IEEE, 2019.

[35] Khatun, Mirza Akhi, Niaz Chowdhury, and Mohammed Nasir Uddin, "Malicious nodes detection based on artificial neural network in IoT environments" in 22nd International Conference on Computer and Information Technology (ICCIT). IEEE, 2019.

[36] Chaudhary, Pooja, and Brij B. Gupta. "DDoS detection framework in resource constrained internet of things domain." in IEEE 8th Global Conference on Consumer Electronics (GCCE). IEEE, 2019.

[37] Accessed on 28 May 2025, "https://medium.com/nerd-for-tech/aodv-routing-protocol-network-simulation-53f3a23918aa"

[38] Zhang, Haodi, Jianye Hao, and Xiaohong Li, "A method for deploying distributed denial of service attack defense strategies on edge servers using reinforcement learning." IEEE Access 8 (2020): 78482-78491.

[39] Bhunia, Suman, Shamik Sengupta, and Felisa Vázquez-Abad, "Cr-honeynet: A learning & decoy based sustenance mechanism against jamming attack in crn." in IEEE Military Communications Conference, pp. 1173-1180. IEEE, 2014.

[40] Mishra, Kamta Nath, Vandana Bhattacharjee, Shashwat Saket, and Shivam Prakash Mishra, "Security provisions in smart edge computing devices using blockchain and machine learning algorithms: a novel approach.", Cluster Computing (2022): 1-26.

[41] Accessed on 22 March 2025, "https://www.espressif.com/en/products/socs/esp32"

[42] Accessed on 22 March 2025, https://en.wikipedia.org/wiki/ESP32

[43] Accessed on 22 March 2025, "https://www.waveshare.com/wiki/DHT22_Temperature-Humidity_Sensor"

[44] Accessed on 23 July 2025, https://builtin.com/machine-learning/markov-decision-process

[45] Accessed on 23 July 2025, https://www.wireshark.org/

[46] Accessed on 23 July 2025, "https://www.oracle.com/in/java/technologies/"

[47] Accessed on 23 July 2025, https://nodered.org/

[48] Accessed on 23 July 2025, https://www.python.org/

[49] Accessed on 23 July 2025, https://www.arduino.cc/en/software

[50] Accessed on 23 July 2025, "https://www.eclipse.org/ide/"

[51] Accessed on 23 July 2025, https://www.oracle.com/java/

[52] Accessed on 23 July 2025, "https://www.gatevidyalay.com/checksum-checksum-example-error-detection/"

[53] Accessed on 23 July 2025, "https://www.geeksforgeeks.org/implementing-checksum-using-java/"

[54] Sutton, Richard S., and Andrew G. Barto, "Introduction to reinforcement learning", MIT press.", Cambridge, MA (1998).

[55] Accessed on 13 April, 2025, https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc/

[56] F. Lin, Y. Zhou, X. An, I. You and K. K. R. Choo, "Fair Resource Allocation in an Intrusion-Detection System for Edge Computing: Ensuring the Security of Internet of Things Devices" in IEEE Consumer Electronics Magazine, vol. 7, no. 6, pp. 45-50, Nov. 2018, doi: 10.1109/MCE.2018.2851723.

[57] C. Aggarwal and K. Srivastava, "Securing IOT devices using SDN and edge computing," in 2nd International Conference on Next Generation Computing Technologies (NGCT), 2016, pp. 877-882, doi: 10.1109/NGCT.2016.7877534.

[58] Sufyan Almajali, Haythem B. Salameh, Moussa Ayyash, and Hany Elgala (2018), "A Framework for Efficient and Secured Mobility of IoT Devices in Mobile Edge Computing", Third IEEE International Conference on Fog and Mobile Edge Computing (FMEC 2018) at Spain, doi: 10.1109/FMEC.2018.8364045.

[59] Accessed on 02 June 2025, https://www.airtel.in/blog/business/role-of-mqtt-protocol-in-iot-messaging/

[60] S. Rathore, P. K. Sharma, and J. H. Park, "Xssclassifier: An efficient xss attack detection approach based on machine learning classifier on snss.", Journal of Information Processing Systems, vol. 13, no. 4, 2017.

[61] Moh, Melody, and Robinson Raju, "Machine learning techniques for security of Internet of Things (IoT) and fog computing systems." in International Conference on High Performance Computing & Simulation (HPCS), pp. 709-715. IEEE, 2018.

[62] Mahdavinejad, Mohammad Saeid, Mohammadreza Rezvan, Mohammadamin Barekatain, Peyman Adibi, Payam Barnaghi, and Amit P. Sheth, "Machine learning for Internet of Things data analysis: A survey.", Digital Communications and Networks 4, no. 3 (2018): 161-175.

[63] Waqas, Muhammad, Shanshan Tu, Jialin Wan, Talha Mir, Hisham Alasmary, and Ghulam Abbas, "Defense scheme against advanced persistent threats in mobile fog computing security.", Computer Networks 221 (2023): 109519.

[64] Dawood, Suroor M., Alireza Hatami, and Raad Z. Homod, "Trade-off decisions in a novel deep reinforcement learning for energy savings in HVAC systems.", Journal of Building Performance Simulation 15, no. 6 (2022): 809-831.

[65] Goudarzi, Mohammad, Marimuthu S. Palaniswami, and Rajkumar Buyya, "A distributed deep reinforcement learning technique for application placement in edge and fog computing environments.", IEEE Transactions on Mobile Computing (2021).

[66] Yu, Zhihao, Wanda Chen, Jie Wang, and Kaiwen Ye, "Deep Reinforcement Learning Based Access Control Strategy for Edge Computing in IoT System.", in IEEE International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI), pp. 699-702. IEEE, 2021.

[67] Xinghua Qu, Zhu Sun, Yew Soon Ong, Abhishek Gupta, and Pengfei Wei (2020), "Minimalistic Attacks: How Little it Takes to Fool Deep Reinforcement Learning Policies", IEEE Transactions on Cognitive and Developmental Systems

[68] Rui Zhang, Hui Xia, Chao Liu, Ruo-bing Jiang, and Xiang-guo Cheng, "Anti-Attack Scheme for Edge Devices Based on Deep Reinforcement Learning", Hindawi, Wireless Communications and Mobile Computing Volume 2021, Article ID 6619715, 9 pages https://doi.org/10.1155/2021/6619715

[69] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li (2018), "Adversarial Examples: Attacks and Defenses for Deep Learning", IEEE Transactions on Neural Networks and Learning Systems

[70] Jaehyoung Park, Dong Seong Kim, and Hyuk Lim (2020), "Privacy-Preserving Reinforcement Learning Using Homomorphic Encryption in Cloud Computing Infrastructures", IEEE Open Access Journal

[71] Amin Azmoodeh, Ali Dehghantanha, and Kim-Kwang Raymond Choo (2018), "Robust Malware Detection for Internet Of (Battlefield) Things Devices Using Deep Eigenspace Learning", IEEE Transactions on Sustainable Computing

[72] Ying-Feng Hsu and Morito Matsuoka (2020), "A Deep Reinforcement Learning Approach for Anomaly Network Intrusion Detection System", IEEE 9th International Conference on Cloud Networking (CloudNet), 2020, pp. 1-6, doi: 10.1109/CloudNet51028.2020.9335796.

[73] P. Mohamed Shakeel, S. Baskar, V. R. Sarma Dhulipala, Sukumar Mishra, and Mustafa Musa Jaber (2018), "Maintaining Security and Privacy in Health Care System Using Learning Based Deep-Q-Networks", Springer, Journal of Medical Systems (2018) 42:186, https://doi.org/10.1007/s10916-018-1045-z

[74] Rohan Doshi, Noah Apthorpe, and Nick Feamster (2018), "Machine Learning DDoS Detection for Consumer Internet of Things Devices", IEEE Symposium on Security and Privacy Workshops

[75] Windows OS Sensors data collecting tool - HWiNFO," https://www.hwinfo.com/

[76] Zhang, P., Jiang, C., Pang, X., & Qian, Y. (2020), "STEC-IoT: A security tactic by virtualizing edge computing on IoT.", IEEE Internet of Things Journal, 8(4), 2459-2467.

[77] Almutairi, J., and Aldossary, "M. A novel approach for IoT tasks offloading in edge-cloud environments.", J Cloud Comp 10, 28 (2021). https://doi.org/10.1186/s13677-021-00243-9

[78] Fatima Alwahedi, Alyazia Aldhaheri, Mohamed Amine Ferrag, Ammar Battah, and Norbert Tihanyi, "Machine learning techniques for IoT security: Current research and future vision with generative AI and large language models, Internet of Things and Cyber-Physical Systems", Volume 4, 2024, Pages 167-185, ISSN 2667-3452, https://doi.org/10.1016/j.iotcps.2023.12.003.

[79] Accessed on 28 September 2025, https://www.comparitech.com/blog/information-security/ddos-statistics-facts/

[80] Kizza, Joseph Migga, "Internet of things (iot): growth, challenges, and security.", Guide to Computer Network Security. Cham: Springer International Publishing, 2024. 557-573.

[81] Wu, Y., Wang, K., Sun, Y., & Zhang, M. (2019), "A Survey of Physical Layer Security Techniques for IoT.", IEEE Communications Surveys & Tutorials, 21(4), 3685-3701. doi:10.1109/COMST.2019.2938397.

[82] Shahin, M., Maghanaki, M., Hosseinzadeh, A., & Chen, FF (2024), "Advancing Network Security in Industrial IoT: A Deep Dive into AI-Enabled Intrusion Detection Systems.", Advanced Engineering Informatics, 62, 102685.

[83] K. Akkaya, A. Mohamed, and M. F. Younis (2020), "A survey on routing protocols for wireless sensor networks.", Ad Hoc Networks, 9(3), pp. 305-332. doi:10.1016/j.adhoc.2020.04.002.

[84] A. Tewari and B. Gupta (2017), "Security, privacy, and trust of different layers in Internet-of-Things (IoTs) framework.", Future Generation Computer Systems, 108(4), pp. 1121-1132. doi:10.1016/j.future.2017.04.028.

[85] D. Evans, J. Wallgren, and A. P. L. Massey (2012), "The Internet of Things: Security challenges.", IEEE Security & Privacy, 10(5), pp. 26-34. doi:10.1109/MSP.2012.145.

[86] Accessed on 02 October 2025, https://www.hivemq.com/blog/the-history-of-mqtt-part-2-mqtt-3-standardization/

[87] Accessed on 24 November 2025, https://www.datacamp.com/tutorial/bellman-equation-reinforcement-learning

[88] Karimipour, Hadis, Ali Dehghantanha, Reza M. Parizi, Kim-Kwang Raymond Choo, and Henry Leung, "A deep and scalable unsupervised machine learning system for cyber-attack detection in large-scale smart grids.", IEEE Access 7 (2019): 80778-80788.

[89] M. Eskandari, Z. H. Janjua, M. Vecchio and F. Antonelli, "Passban IDS: An Intelligent Anomaly-Based Intrusion Detection System for IoT Edge Devices," in IEEE Internet of Things Journal, vol. 7, no. 8, pp. 6882-6897, Aug. 2020, doi: 10.1109/JIOT.2020.2970501.

[90] Yazdinejad, Abbas, Behrouz Zolfaghari, Ali Dehghantanha, Hadis Karimipour, Gautam Srivastava, and Reza M. Parizi., "Accurate threat hunting in industrial internet of things edge devices.", Digital Communications and Networks 9, no. 5 (2023): 1123-1130.

[91] Shen, Tingda, Lijiao Ding, Jinze Sun, Changqiang Jing, Feng Guo, and Chuankun Wu, "Edge computing for IoT security: Integrating machine learning with key agreement.", in 3rd International Conference on Consumer Electronics and Computer Engineering (ICCECE), pp. 474-483. IEEE, 2023.

[92] Mhaisen, Naram, Noora Fetais, Aiman Erbad, Amr Mohamed, and Mohsen Guizani, "To chain or not to chain: A reinforcement learning approach for blockchain-enabled IoT monitoring applications.", Future Generation Computer Systems 111 (2020): 39-51.

[93] S. S. Khan and M. G. Madden, "A Survey of Recent Trends in One Class Classification." in Artificial Intelligence and Cognitive Science, M. G. Madden, Ed. Springer, 2010, pp. 188–197.

[94] J. R. Quinlan, "Induction of decision trees.", Machine Learning, vol. 1, no. 1, pp. 81–106, 1986. [Online]. Available: https://doi.org/10.1023/A:1022643204877.

[95] C. Cortes and V. N. Vapnik, "Support-Vector Networks." Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.

[96] Y. A. Malkov and D. A. Yashunin, “Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs.” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 4, pp. 824–836, Apr. 2020.

[97] Ige, Tosin, Christopher Kiekintveld, Aritran Piplai, Amy Waggler, Olukunle Kolade, and Bolanle Hafiz Matti, “An investigation into the performances of the Current state-of-the-art Naive Bayes, Non-Bayesian and Deep Learning Based Classifier for Phishing Detection: A Survey.”, arXiv preprint arXiv:2411.16751 (2024).

[98] Khoudi, Zakaria, Nasereddine Hafidi, Mourad Nachaoui, and Soufiane Lyaqini, “New Approach to Enhancing Student Performance Prediction Using Machine Learning Techniques and Clickstream Data in Virtual Learning Environments.”, SN Computer Science 6, no. 2 (2025): 139.

[99] J. Weng and M. D. Luciw, “Brain-Inspired Concept Networks: Learning Concepts from Cluttered Scenes.”, IEEE Intelligent Systems, vol. 29, no. 6, pp. 56-63, Nov.-Dec. 2014.

[100] Mukhtar, Basem Ibrahim, Mahmoud Said Elsayed, Anca D. Jurcut, and Marianne A. Azer, “IoT vulnerabilities and attacks: SILEX malware case study.”, Symmetry 15, no. 11 (2023): 1978.

[101] N. Windpassinger, “The layered OSI reference model in an ocean of IoT protocols”, Nicolas Windpassinger Blog, Aug. 22, 2018. [Online]. Available: “https://nicolaswindpassinger.com/osi-reference-model”, [Accessed: Mar. 16, 2025].

[102] Wu, Yanzhao, Ling Liu, Juhyun Bae, Ka-Ho Chow, Arun Iyengar, Calton Pu, Wenqi Wei, Lei Yu, and Qi Zhang, “Demystifying learning rate policies for high accuracy training of deep neural networks.” in IEEE International Conference on Big Data, pp. 1971-1980. IEEE, 2019.

[103] Accessed on 29 March 2025, https://developers.google.com/machine-learning/crash-course/classification/thresholding

[104] Accessed on 29 March 2025, "https://www.geeksforgeeks.org/confusion-matrix-machine-learning/"

[105] Accessed on 29 March 2025, https://spotintelligence.com/2024/09/06/confusion-matrix-a-beginners-guide-how-to-tutorial-in-python/

[106] Kim, MyeongSeop, Jung-Su Kim, Myoung-Su Choi, and Jae-Han Park. 2022, "Adaptive Discount Factor for Deep Reinforcement Learning in Continuing Tasks with Uncertainty", Sensors 22, no. 19: 7266. https://doi.org/10.3390/s22197266

[107] Accessed on 09 April 2025, https://milvus.io/ai-quick-reference/what-is-an-epsilongreedy-policy

[108] Tokic, Michel, "Adaptive ε-greedy exploration in reinforcement learning based on value differences.", in Annual conference on artificial intelligence, pp. 203-210. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[109] Li, Donghao, Ruiquan Huang, Cong Shen, and Jing Yang, "Near-optimal conservative exploration in reinforcement learning under episode-wise constraints.", In International Conference on Machine Learning, pp. 19527-19564. PMLR, 2023.

[110] Oh, Sang Ho, Min Ki Jeong, Hyung Chan Kim, and Jongyoul Park, "Applying reinforcement learning for enhanced cybersecurity against adversarial simulation.", Sensors 23, no. 6 (2023): 3000.

[111] Yinhao Xiao, Yizhen Jia, Chunchi Liu, et al, "Edge Computing Security: State-of-The-Art and Challenges," in Proceedings of the IEEE, 2019.

[112] Nguyen, Thanh Thi, and Vijay Janapa Reddi, "Deep reinforcement learning for cyber security.", IEEE Transactions on Neural Networks and Learning Systems 34, no. 8 (2021): 3779-3795.

[113] Yan, Jun, Haibo He, Xiangnan Zhong, and Yufei Tang, "Q-learning-based vulnerability analysis of smart grid against sequential topology attacks.", IEEE Transactions on Information Forensics and Security 12, no. 1 (2016): 200-210.

[114] Wang, Jing., "Multi agent system based smart grid anomaly detection using blockchain machine learning model in mobile edge computing network.", Computers and Electrical Engineering 121 (2025): 109825.

[115] Oh, Sang Ho, Jeongyoon Kim, Jae Hoon Nah, and Jongyoul Park, “Employing deep reinforcement learning to Cyber-Attack simulation for enhancing cybersecurity”, Electronics 13, no. 3 (2024): 555.

[116] Mahmoodi Khaniabadi, Shadi, Amir Javadpour, Mehdi Gheisari, Weizhe Zhang, Yang Liu, and Arun Kumar Sangaiah, “An intelligent sustainable efficient transmission internet protocol to switch between User Datagram Protocol and Transmission Control Protocol in IoT computing”, Expert Systems 40, no. 5 (2023): e13129.

[117] Malik, Jahanzaib, Adnan Akhunzada, Ahmad Sami Al-Shamayleh, Sherali Zeadally, and Ahmad Almogren, “Hybrid deep learning based threat intelligence framework for industrial iot systems”, Journal of Industrial Information Integration 45 (2025): 100846.

[118] Bikila, Dawit Dejene, and Jan Čapek, “Machine learning-based attack detection for the Internet of Things”, Future Generation Computer Systems 166 (2025): 107630.

[119] Vadisetty, Rahul, “Adaptive Machine Learning-Based Intrusion Detection Systems for IoT Era”, In International Ethical Hacking Conference, pp. 251-273. Singapore: Springer Nature Singapore, 2024.

[120] Arif, Fahim, Nauman Ali Khan, Javed Iqbal, Faten Khalid Karim, Nisreen Innab, and Samih M. Mostafa, “DQQS: Deep Reinforcement Learning-Based Technique for Enhancing Security and Performance in SDN-IoT Environments”, IEEE Access 12 (2024): 60568-60587.

[121] Accessed on 11 April 2025, “https://www.sciencedirect.com/topics/computer-science/packet-delivery-ratio”

[122] Accessed on 11 April 2025, https://www.techtarget.com/searchnetworking/definition/throughput

[123] Accessed on 11 April 2025, https://www.sciencedirect.com/topics/computer-science/end-to-end-delay

[124] Accessed on 11 April 2025, https://www.nsnam.com/2023/10/awk-and-ns2-tracefile-ns2-tutorial-3.html

[125] Accessed on 11 April 2025, https://www.unb.ca/cic/datasets/ids-2017.html

[126] Accessed on 11 April 2025, https://research.unsw.edu.au/projects/unsw-nb15-dataset

[127] Ahmad, Rasheed, Izzat Alsmadi, Wasim Alhamdani, and Tawalbeh, L.A., "A comprehensive deep learning benchmark for IoT IDS.", Computers & Security 114 (2022): 102588.

[128] Ren, Bin, Yunlong Tang, Huan Wang, Yichuan Wang, Jianxiong Liu, Ge Gao, and Wei Wei, "A Multiagent Deep Reinforcement Learning Autonomous Security Management Approach for Internet of Things.", IEEE Internet of Things Journal 11, no. 15 (2024): 25600-25612

[129] Maghrabi, Louai A., "Automated network intrusion detection for Internet of Things: Security enhancements.", IEEE Access 12 (2024): 30839-30851.

[130] Geetha, C., Shiny Duela Johnson, A. Sheryl Oliver, and D. Lekha, "Adaptive weighted kernel support vector machine-based circle search approach for intrusion detection in IoT environments.", Signal, Image and Video Processing 18, no. 5 (2024): 4479-4490.

[131] Accessed on 22 August 2025, "https://www.redwolfsecurity.com/simulating-ddos-attacks-many-attackers-enough/"

[132] Accessed on 22 August 2025, https://blog.cloudflare.com/ddos-threat-report-for-2024-q2/

[133] Mahajan, Ratul, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker, "Controlling high bandwidth aggregates in the network.", ACM SIGCOMM Computer Communication Review 32, no. 3 (2002): 62-73.

[134] Vitorino, João, Rui Andrade, Isabel Praça, Orlando Sousa, and Eva Maia, "A comparative analysis of machine learning techniques for IoT intrusion detection", in International Symposium on Foundations and Practice of Security, pp. 191-207. Cham: Springer International Publishing, 2021.

[135] Sudheera, Kalupahana Liyanage Kushan, Dinil Mon Divakaran, Rhishi Pratap Singh, and Mohan Gurusamy, "ADEPT: Detection and identification of correlated attack stages in IoT networks", IEEE Internet of Things Journal 8, no. 8 (2021): 6591-6607.

[136] Ullah, Imtiaz, and Qusay H. Mahmoud, “Design and development of a deep learning-based model for anomaly detection in IoT networks”, IEEE Access 9 (2021): 103906-103926.

[137] Accessed on 22 August 2025, https://www.electronifyindia.com/blogs/news/esp32-wroom-32-vs-other-esp-modules

[138] A. A. Khedher and A. B. Hamida, “Design and implementation of a smart monitoring system for temperature and humidity using DHT22 sensor”, in International Journal of Advanced Computer Science and Applications (IJACSA), vol. 12, no. 5, pp. 524–529, 2021.

[139] Accessed on 22 August 2025, https://www.kaggle.com/datasets/toobajamal/kdd99-dataset

[140] Shah, A., & Patel, H. (2021), “Comparative analysis of Raspberry Pi and ESP32 for IoT applications”, International Journal of Scientific Research in Engineering and Management (IJSREM), 5(3), 1–5.

## List of Publications

Following **Table 17** is the list of publications we published during our PhD research work.

**Table 17**: List of Publications on Proposed Research Work

| References | Title of the paper | Publication Date, Journal Name, Publisher | ISSN/ISBN Number |
|---|---|---|---|
| [1] | Securing IoT devices in edge computing through reinforcement learning | 08/04/2025, Computers & Security, Elsevier | 1872-6208 |
| [2] | Reinforcement learning based security method for IoT-Edge computing | 30/03/2026, Wireless Networks, Springer Nature Publishing | 1572-8196 |
| [3] | Adaptive epsilon greedy reinforcement learning method in securing IoT devices in edge computing | 20/11/2024, Discover Internet of Things, Springer International Publishing | 2730-7239 |
| [4] | Detection and prevention of DDoS attacks on edge computing of IoT devices through reinforcement learning | 28/12/2023, International Journal of Information Technology, Springer Nature Singapore | 2511-2112 |
| [5] | A systematic literature review on load balancing algorithms of virtual machines in a cloud computing environment | 31/03/2020, Proceedings of the International Conference on Innovative Computing & Communications (ICICC) | ICICC-2020 |

| | | | |
|---|---|---|---|
| [6] | Detection of Security Attacks on Edge Computing of IoT Devices through NS2 Simulation | 01/08/2022, Journal of Physics: Conference Series, IOP Publishing | 1742-6596 |
| [7] | A Comprehensive Survey on Security Attacks to Edge Server of IoT Devices through Reinforcement Learning | 01/03/2022, AICTE Sponsored National Conference on Multidisciplinary Research and Innovation - 21 (NCMRAI - 21) | 978-93-91331-24-5 |

## List of Workshops

Following **Table 18** is the list of workshops attended by us during our Ph.D. research work.

**Table 18:** List of Workshops Attended

| Sr. No. | Workshop title | Duration | Organizing Institute |
|---|---|---|---|
| 1 | INNOVATION, CREATIVITY AND ENTREPRENEURSHIP – ICE2020: A COOL WORKSHOP | July 6, 2020 to July 10, 2020 | Satya Institute of Technology and Management, Gajularega, Vizianagaram-535003, Andhra Pradesh, India |

## List of Patents

Following **Table 19** is the list of patents obtained by us during our Ph.D. research work.

**Table 19:** List of Patents Received

| Sr. No. | Patent # | Application # | Date of Grant | Patent Title | Organizing Institute/ Issuing Authority |
|---|---|---|---|---|---|
| 1 | 444313 | 202111055536 | 10/08/2023 | SECURING IoT DEVICES IN EDGE COMPUTING THROUGH REINFORCEMENT LEARNING | The Patent Office, Government of India |