REPUBLIC OF TURKEY

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

NETWORK ANOMALY DETECTION IN CLOUD AND IOT USING FUZZY LOGIC

Susan Ismael AbdulWahhab AL-HADEETHI

Master`s Thesis

Supervisor

Prof. Dr. Galip CANSEVER

Istanbul, 2023

NETWORK ANOMALY DETECTION IN CLOUD AND IOT USING FUZZY LOGIC

Susan Ismael AbdulWahhab AL-HADEETHI

Electrical and Computer Engineering

Master`s Thesis

ALTINBAŞ UNIVERSITY

2023

The thesis titled NETWORK ANOMALY DETECTION IN CLOUD AND IOT USING FUZZY LOGIC prepared by SUSAN ISMAEL ABDULWAHHAB AL-HADEETHI and submitted on 00 /00 /2022 has been **accepted unanimously** for the degree of Master of Science in Electrical and Computer Engineering.

Prof. Dr. Galip CANSEVER

Supervisor

Thesis Defense Committee Members:

Prof. Dr. Galip CANSEVER	Faculty of Engineering and Architecture, Altınbaş University	
Asst. Prof. Sefer KURNAZ	Faculty of Engineering and Architecture, Altınbaş University	
Asst. Prof. Yüksel BAL	Faculty of Engineering Computer Engineering, Topkapi University	

I hereby declare that this thesis meets all format and submission requirements of a Master`s Thesis.

Submission date of the thesis to the Graduate Education Institute: ____/____/____

I hereby declare that all information presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Susan Ismael AbdulWahhab AL-HADEETHI

Signature

DEDICATION

I dedicate my efforts to my father’s soul. I also dedicate it to my beloved mother. To my sisters, brother, and their families.

To my supportive husband who has never left my side and supported me throughout my study.

To my daughter and son.

All of you are my world.

PREFACE

I would like to thank my supervisor prof. Dr. Galip CANSEVER for his support during my study. I also would like to express my deepest gratitude to every person who helped me during my study time.

ABSTRACT

NETWORK ANOMALY DETECTION IN CLOUD AND IOT USING FUZZY LOGIC

AL-HADEETHI, Susan Ismael AbdulWahhab

M.Sc., Electrical and Computer Engineering, Altınbaş University,

Supervisor: Prof. Dr. Galip CANSEVER

Date: April / 2023

Pages: 90

The Internet of Things (IoT) is a new technology that seeks to give all physical objects a virtual presence on the Internet. The primary idea behind IoT is to give physical objects intelligence by incorporating electronic components, software, sensors, actuators, and network connectivity. Hence, the Internet of Things can be characterized as a network of these smart items that can collect and share data via the Internet. As a result, this thesis proposes a solution capable of detecting a wide range of intrusions on connected objects, employing powerful tracing techniques to collect precise information on the monitored systems and comparing various machine learning (ML) techniques to achieve superior detection capabilities.

Keywords: IOT, ML, IDS, Malware, DL.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

IDS	:	Intrusion Detection System
IOT	:	Internet of Things
NIST	:	National Institute for Standards and Technology
ML	:	Machine Learning
VDC	:	Virtual Data Centers
FL	:	Fuzzy Logic
CC	:	Cloud Computing

1. INTRODUCTION

1.1 RESEARCH BACKGROUND

The IoT intends to give all physical items a virtual presence on the Internet through the use of a new technology. IoT's core concept is to give physical items intelligence by integrating electronic parts, software, sensors, actuators, and network connectivity into them. As a result, the IoT may be described as a network of these intelligent items that can communicate data online[1] IoT applications are often used in industries such as oil and gas exploration, predicting equipment failure, controlling fluid flow, monitoring seismic waves at exploration sites and predicting potential drilling sites. It is common to use software-based systems to monitor and supervise the variables and devices of control systems connected through servers and specific communication drivers [2] In the context of the global trend for technology, the industry specializing in vertical business, characterized by manufacturing the product from the beginning until it is available on the shelf, without the need to partition production to other suppliers, has been losing its projection due to the demand of IoT devices to innovation-oriented industries, which target devices aimed at smart homes, autonomous cars, and startups, as pointed out [2]. Regarding IoT attacks, in the last decade laptops and servers have been the biggest targets of hacker attacks. However, in recent years there has been a growing increase in attacks targeting connected homes, such as lighting, thermostat, and smart locks, as indicated by the Arbor report (2018). The Internet of Things allowed a 60% increase in the number of attacks and the same report also states that the increase was not only in volume but also in frequency and complexity, among them the reflection attack. It is not unusual to find that homeowners frequently use the default passwords provided by manufacturers on their home automation devices. Kaspersky security experts point out that attacks targeting IoT devices more than increased between 2019 and 2022 [3]. The study was carried out in an environment prepared to monitor simulated attacks through a honeypot environment and the data obtained can be visualized in Figure 1. With the increase in attacks, IoT security is an area that needs new solutions and simulations of existing solutions in this new scenario, considering the restrictions of IoT devices [4].

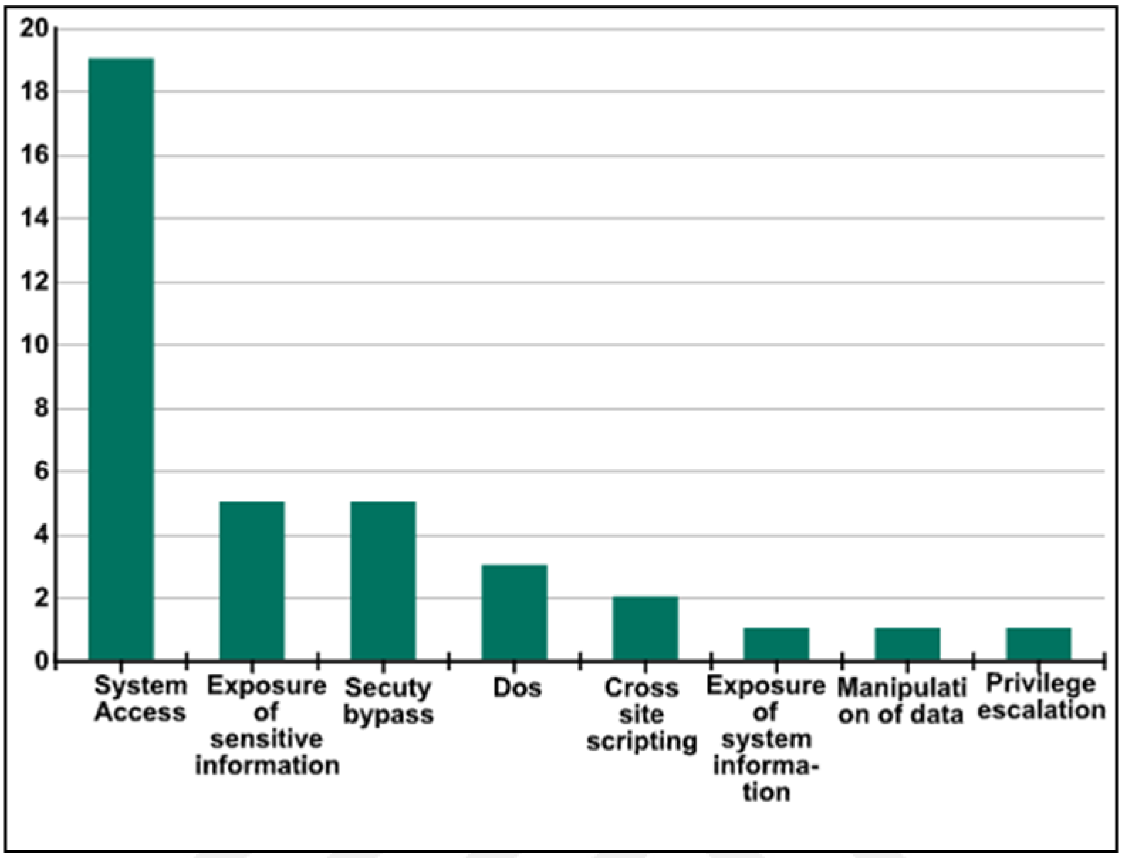

Figure 1.1: Kaspersky IOT Security Bulletin [3].

1.2 PROBLEM STATEMENT

Strong competition from manufacturers of connected objects has created a market where many technologies coexist. These objects can take various forms, ranging from alarm or light management systems to ovens or connected toys. They sometimes have extremely limited resources, making it impossible to implement traditional security solutions. The diversity of protocols existing in this ecosystem as well as the absence of an operating system or firmware common to these platforms make the work aimed at securing such objects very dependent on the systems studied. In addition, few manufacturers offer updates for their devices, making discovered vulnerabilities very dangerous for users. Thus, it becomes difficult to propose security solutions applicable to all objects, and the work carried out in terms of intrusion detection essentially concentrates on network analyses, since there is no need to consider the material resources of the objects nor even the applications which are executed on them. In addition, even while tracing techniques have been employed to identify abnormalities on many systems, particularly with the aid of system calls performed on the monitored computer, very few studies have concentrated on all the data included in the events[5] . Thus, the system call arguments or the various information at the level of the network packets or the processor have hardly been used to detect computer attacks, whereas they can prove invaluable to improve the quality of the detection. For example, an event related to the launching of a terminal can be benign or harmful for the system depending on the process which is at the origin of this system call. It is therefore appropriate to use all the information collected by the trace points to improve the quality of detection. In addition, few works have proposed tool suites for exploiting traces in a binary format to train ML algorithms. Indeed, to achieve such a result, several stages of transformation of the events collected with tracing techniques must be implemented [6]. The challenges to achieve this result are multiple. The first is to determine the event fields useful for training the ML algorithms, which can be very different between the various events that can be recorded by the tracer. Another challenge is to harvest these fields efficiently from the formats where the traces are stored, particularly binary formats, and to create other metrics useful for learning algorithms. It is also necessary to convert all these data into vectors of figures which will then be usable for learning the models. Finally, it is necessary to study the effectiveness of the different ML methods developed to improve the detection capabilities of the proposed solution. Indeed, each algorithm can give different results depending on the dataset it receives or the use of the model once its training is complete. For example, some algorithms may perform best at classifying sequences of text but may be much less optimal at classifying single words. It is therefore necessary to critically study the behavior of different learning techniques in the case of intrusion detection on connected objects, with traces relating to the system itself [7].

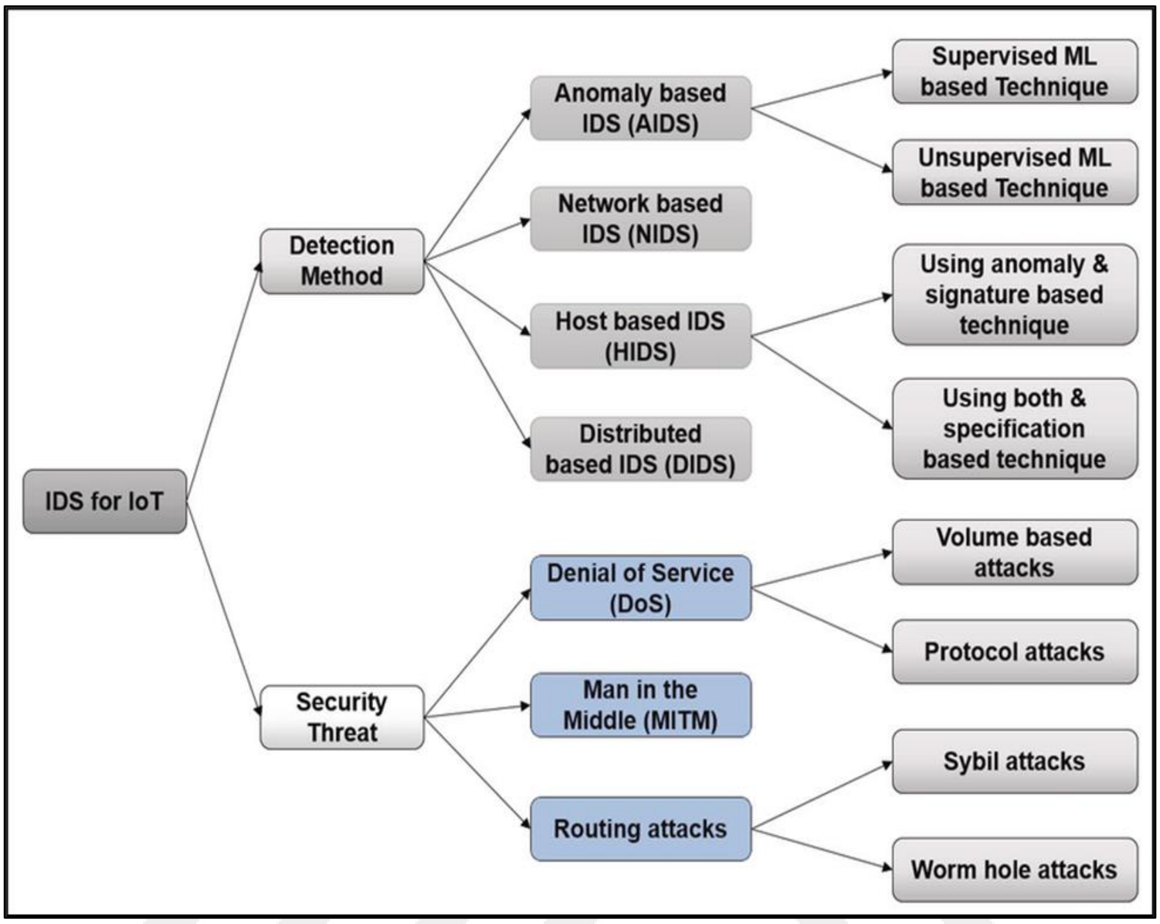

Figure 1.2: IDS With Security Threats In IOT and Clouds [5].

1.3 RESEARCH CONTRIBUTION

There is no impenetrable computer system, and attacks on these platforms continue to increase in number and intensity every day. At the same time, more and more systems are interconnected, and this phenomenon is accentuated with the advent of the Internet of Things, which is made up of innovative objects but very rarely integrates effective protections against cybercriminal attacks. They threaten their users, whether they are individuals or companies. It is therefore necessary to find solutions capable of effectively securing these objects and respecting their hardware and software constraints. One of the first steps to improving the security of such devices is to be able to detect attack attempts as soon as they occur. For this, it is necessary to take an interest in past attacks involving connected objects but also to be able to prevent new types of attacks. This thesis, therefore, proposes a solution capable of detecting many types of intrusions on connected objects, using powerful tracing techniques to collect precise information on the monitored systems and comparing various ML techniques to achieve excellent detection capabilities on information security attacks that attack any network.

1.4 RESEARCH MAJOR AIM AND MINOR OBJECTIVES

The main problem of this thesis is the following: Is it possible to implement an intrusion detector based on the behavior of connected objects that is effective and based on tracing and ML techniques? To be able to answer this problem, it is necessary to achieve the following objectives:

a. Understand the characteristics of the ecosystem of cloud-connected objects.
b. Develop an architecture allowing the tracking of intelligent objects considering their characteristics.
c. Develop a framework capable of transforming the information contained in the traces into precise information that can be used by ML algorithms.
d. Implement and optimize various ML algorithms capable of classifying recorded events.
e. Deploy a test home automation system by identifying classic smart home architectures.
f. Identify and implement attacks on the connected objects studied.
g. Test the performance of the solution and the different algorithms on this system.

1.5 RESEARCH ORGANIZATION

The dissertation is structured into four chapters:

Chapter 2 of the thesis presents a critical review of the literature on the various concepts useful during this work. The topics of the Internet of Things, IDSs, and different tracking techniques will be discussed. Then, chapter 3 will present the methodology adopted in our work to set up and validate the proposed solution. Chapter 4 presents the article in which the developed solution is presented as well as the criticism of its performance. Also evaluates the success of answering the research question. The conclusion of the work as well as the discussion of future work are presented in chapter 5.

2. LITERATURE REVIEW

2.1 INTRODUCTION

Daily, the number of items capable of connecting over the Internet grows, as does the number of computers linked to this network. This tendency is compounded further by the continuous addition of new items, which happens as the world's variety expands. As a result of increasing rivalry among object designers, manufacturers are under pressure to reduce the time required to produce their goods and, as a result, the time required to release their products on the market for security assessment. As a result, an increasing number of vulnerabilities in smart objects are being uncovered, and the resulting hazards to consumers are getting increasingly serious. To begin, it is vital to be able to detect and avoid possibly damaging activities in order to protect these countless goods. Numerous remedies have been presented, however, the majority of them are ineffectual or very dependent on the object in issue. Several tactics for detecting intrusions are addressed in further detail in the following critical evaluation of the literature. Additionally, a range of strategies for achieving improved results are discussed in depth.

2.2 RELATED WORKS

Researchers are examining the security flaws in Internet of Things communication protocols from a range of angles [8], one of which is the protocol's own weakness. In accordance with the IEEE description of IoT architecture and the generic IoT architecture, this assessment focuses on IDSs for IoT paradigms without regard for any particular protocol or standard. Internet of Things security issues arise as a consequence of the several levels of IoT security. At the physical layer, worries about physical damage, device failure, and power constraints emerge. Network layer problems include denial of service attacks, sniffer attacks, gateway assaults, and unauthorized access.

In 2013, Ganapathy et al. [9] presentation included an introduction to network intrusion detection strategies that use intelligent feature selection and categorization. Although the study focused primarily on internet security and quality of service (QoS). Additionally, these researchers identified previously unknown traits and methods for classifying data. It was observed that their experiment may make use of 19 flow-related properties. These characteristics comprised fundamental properties, packet content properties, and traffic properties.

In 2014, Mitchell and Chen [10] conducted a review of 60 articles on wireless IDSs. WLANs, WPNs, WSNs, CPSs, ad hoc networks, and cellular telephony were all researched, and the findings indicated the benefits and drawbacks of IDSs. As Mitchell and Chen demonstrated, the most effective option for mobile telecommunications networks is anomaly-based IDSs (IDSs). However, these IDS provide difficulties owing to their high false-positive rate and computational complexity [10]. Increased false positive and false negative rates have a deleterious effect on the overall quality of QoS. If any user packets are mistakenly lost, a billing error and a delay will occur [10]. Unlawful analytic techniques, such as packet-based approaches that violate user privacy, pose a threat to anomaly-based IDSs as well [10]. Participants in this poll identified detection latency as a significant trait that warrants further exploration.

Also in 2014, Butun et al. [11] The researchers examined IDS in WSNs by examining 18 MANET publications and 17 WSN papers. The authors of this study studied whether or not WSNs may benefit from MANET-specific technology. Security threats associated with passive and active WSNs were classified as passive and aggressive. The following passive threats were classified: According to these experts, in order to ensure the network's security, a WSN-specific IDS must meet certain criteria, such as low power consumption. This demonstrates that the efficacy of an IDS in a WSN is determined by its impact on the network's energy usage.

Butun et al. 2015 [12]. The adoption of a hierarchical IDS model in WSNs has been suggested to assist minimize energy usage. Butun and colleagues [12] It is advised that a distributed IDS, a centralized IDS, or a hierarchical IDS, depending on the needs of the application, be utilized for mobile apps, stationary applications, and cluster-based applications, respectively.

Mori et al.2016 [13] According to the authors, "principal component analysis (PCA) is a widely used descriptive multivariate approach for quantitative data that may be extended to cope with data having heterogeneous measurement levels." As a result, PCA is extensively used across a diverse range of industries.

Surendar and Umamakeswari 2016 [14] suggested a 6LoWPAN-based constraint-based IDS for IoT networks. Their solution maintains a high degree of efficiency in terms of quality-of-service measures even when sinkhole assaults are discovered. The method removes malicious nodes from the network and recreates it without them. Systems for detecting intrusions based on behavioral rule-based specifications and the protocol model approach.

Elrawy et al. 2017 [15] created an anomaly-based statistical and data mining IDS using PCA that divides principal components into the most and least significant principal components in order to detect anomalies. The major and minor main component scores of the algorithm define the detection stage when this approach is applied. Among other approaches, PCA has been utilized in IDSs that make use of payload modeling, statistical modeling, data mining, and ML.

Butun et al. 2019 [16] proposed with a downward-IDS and an upward-IDS, in accordance with the hierarchical structure of the WSN network, a WSN NIDS combines the statistical model and rule model techniques. Studying anomaly-producing behavior in cluster heads and specific nodes involves using the downward-IDS and the upward-IDS. The two most distinctive features of the system are its ability to adapt to hierarchical WSN and its reliance on wireless sensor network clustering.

Finally, Le et al. 2021 [17] proposed a Detection of a broad variety of topology-based attacks against IPv6 RPL, including sinkhole attacks, rank attacks, local repair attacks, neighbor attacks, and goal-directed acyclic graph information solicitation (DIS) assaults. Another common attack technique is to use malicious nodes to produce DODAG information objectives (DIOs) in order to send DIS messages to the attacker nodes' neighbors [18] in order to raise network overhead [18]. The suggested IDS utilizes trace files to analyze protocol behavior and to learn techniques for establishing and maintaining a stable topology, as well as to learn protocols that the proposed IDS does not support. The device's low power consumption and exceptional efficacy in detecting RPL topological assaults may be of considerable use to large-scale networks.

3. SECURITY OF CLOUD-CONNECTED OBJECTS

3.1 INTRODUCTION

This chapter introduces the scientific and technological context in which the work of this thesis takes place. The objective is to understand the issues and problems that our contributions propose to respond to thereafter. First, we present the two major areas associated with our work: IOT based cloud computing and computer security. Then, we are interested in the association of these two domains by describing the network security mechanisms in the cloud that we consider the problem. Finally, we introduce the topics of security analysis and evaluation, which are treated in more detail in our contributions.

3.2 CLOUD COMPUTING

Cloud computing is an emerging paradigm in the world of computing, born out of the evolution of distributed systems and grid computing. However, although it has several similarities with the latter, cloud computing is differentiated by certain characteristics and models of service and deployment. Thus, we first expose some definitions, then describe the characteristics, models of service, and deployment specific to this paradigm. We also present some new associated technologies that make the cloud model possible.

3.2.1 Definitions and Characteristics

The term cloud computing, or more simply cloud, is also commonly used. Several translations, little used, have been attempted: According to the National Institute for Standards and Technology (NIST), cloud computing is a model for enabling universal, simple, on-demand network access to a shared collection of reconfigurable computing resources (networks, servers, storage, applications, and services), which can be quickly provisioned and released with little management effort or contact with the service provider [19]. The definition of cloud computing provided by the dictionary, is listed as an expression for the word cloud: computer organization model allowing access to digital resources whose storage is outsourced to several servers. In computing, the origin of the use of the term cloud would come from the historical use of clouds to represent computer networks, and in particular the Internet. This representation aims to hide the internal complexity of the networks and abstract a simple representation. It was therefore used to hide the complexity of the various elements making up the cloud, and the lack of information on the location and processing of data hosted in the cloud. would come from the historical use of clouds to represent computer networks, and in particular the Internet. This representation aims to hide the internal complexity of the networks and to abstract a simple representation. It was therefore used to hide the complexity of the various elements making up the cloud, and the lack of information on the location and processing of data hosted in the cloud.

Beyond the proposed definitions, NIST has defined a cloud computing with five essential characteristics, three service models, and four deployment models, as shown in Figure This model of cloud has the following five essential characteristics:

a. Self-service on demand: The utilization of services and resources is totally automated, and setting up and managing the configuration remotely is done by the user using a command console.
b. Broad network access: By using conventional and varied equipment, whether light or heavy, the services may be accessed through the Internet network.
c. Resource pooling: In a co-residency model, provider computer resources are pooled and utilized to serve several clients, with resources being dynamically distributed based on demand. Although it may be able to identify the location of the resources delivered to it at a more abstract level (such as a nation, region, or data center), the client often is unaware of the specific location of those resources.
d. Rapid elasticity: Resources can be elastically and automatically provisioned and released to respond quickly and scalable to demand. For the client, the resources appear unlimited and can be allocated in any quantity at any time.
e. Usage-based billing: Cloud systems automatically monitor and optimize resource usage by means of measurement at certain levels of abstraction appropriate to the type of service. Resource usage can be monitored, controlled, and reported to provide transparency for the service provider and customer.

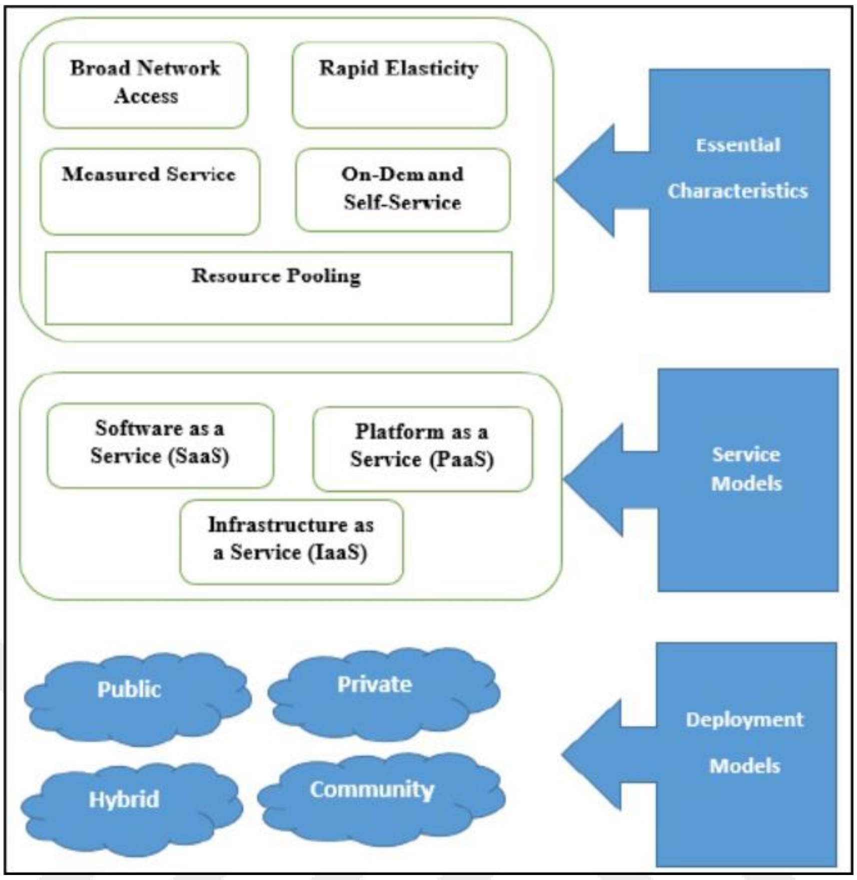

Figure 3.1: NIST Visual Model For Cloud Computing [20].

3.2.2 Service and Deployment Models

The hardware and software components of cloud infrastructure are what allow for the five fundamental aspects of cloud computing. Both a physical layer and an abstract layer can be viewed as being present. The servers, storage, and networking components that make up the physical layer are among the hardware resources required to enable cloud services that are offered. In terms of concept, the abstraction layer comes before the physical layer. It consists of software that is installed on the physical layer, presenting the fundamental elements of the cloud and enabling the service provider to offer various services. On the basis of this, many service and deployment models are available that enable the implementation of services on a cloud infrastructure.
There are 3 service models for the cloud [21], which define the type of service offered:

a. Software as a Service (SaaS): applications from the provider that are hosted in the cloud can be used by the client. Applications are accessible via a simple user interface, such a web browser or a software interface. The client only manages a small number of user configuration settings for the underlying cloud infrastructure, which does not include the m

b. Platform as a Service (PaaS): The customer can use the languages, libraries, services, and tools supplied by the supplier to deploy applications on a cloud infrastructure. The client has control over the installed programs and maybe the configuration options of the environment hosting the applications but not the underlying cloud infrastructure, which consists of the network, servers, operating systems, and storage.
c. Infrastructure as a Service (IaaS): Any sort of software, including operations and applications, may be deployed and run by the customer using computation, storage, networking, and other basic computer resources. The customer has limited control over some network components but manages the installed applications, operating systems, and storage of the cloud infrastructure.

More and more new service models not initially defined by the NIST appear following the acronyms XaaS [22], where X represents the service offered, such for example Storage, Network, and Security. These services are then generally included in the three models' reference services.

The cloud has four deployment models [23], which provide the management practices for the cloud infrastructure on which services are deployed.

a. Private cloud: the cloud infrastructure is used exclusively by a single organization. It can be managed by the organization itself, a third party, or a combination of both. The infrastructure can be placed physically on the premises of the organization or outside.
b. Community Cloud: The cloud infrastructure is used exclusively by a community of customers of organizations with common interests. It can be run by one or more community organizations, a third party, or a combination of both. The infrastructure can be placed physically on the premises of one or more organizations in the community or outside.
c. Public Cloud: The cloud infrastructure is used for the general public. It can be managed by an industrial, academic, governmental organization, or a combination of several organizations. The infrastructure is physically placed at the premises of the service provider.

d. Hybrid cloud: The cloud infrastructure is made up of two or more different cloud infrastructures (private, communal, or public) that are still separate legal companies but are connected by standardized or exclusive technologies to enable data and service portability.

3.2.3 The Technology Associated with Cloud Based IOT

To ensure the five essential characteristics of the cloud to deliver services, many technologies are used. Indeed, cloud computing is essentially an economic model, made possible by technological advances allowing the modulation of resources [24]. These advances are mainly articulated around virtualization, which has enabled new and better exploitation of physical computing resources.

3.2.3.1 Virtualization

Virtualization is a concept born in the 1960s, around the work of IBM on powerful computers called mainframes. It is a technique for segmenting physical, temporal, and spatial hardware resources between multiple virtual machines [25] . In [26], a virtual machine (VM) is defined as a copy of a real, efficient, and isolated machine. Virtualization makes it possible, among other things, to run several operating systems on the same physical machine. The segmentation and the scheduling of the virtual machines are carried out by an entity that comes to be placed between the material and the virtual machines: the manager of virtual machines or Virtual Machine Monitor (VMM), also called a hypervisor.

There are three main types of virtualization [27]: emulation, paravirtualization, and full virtualization. In emulation, the virtualized system is executed as an application in the user memory space. It is the kernel of the host operating system that controls system calls for access to hardware resources. In the case of para-virtualization, the code of the virtualized system is modified to be able to explicitly make calls to the underlying hypervisor. In full virtualization, the virtualized system is unaware of it and sends its instructions to the hardware. These instructions are then intercepted by the hypervisor which emulates the hardware resources.

Two types of hypervisors predominate type 1 (or bare metal) hypervisors and type 2 (or hosted) hypervisors. Type 1 hypervisors (Figure 3.2) [28] load immediately after running the firmware (BIOS or UEFI) of the machine and rely only on the services offered by the firmware. This one can pre-configure the events on which it is notified and must intervene. This significantly increases virtualization performance.

3.2.3.2 Virtualization of networks and virtual networks

Virtualization allows the creation of new virtual entities, in addition to virtual machines. Indeed, it is possible to constitute virtual infrastructures, which include virtual data centers (VDC), themselves containing virtual machines, virtual storage spaces, and virtual networks. The latter, products of network virtualization, make it possible to establish communications between virtual machines within virtual infrastructures.

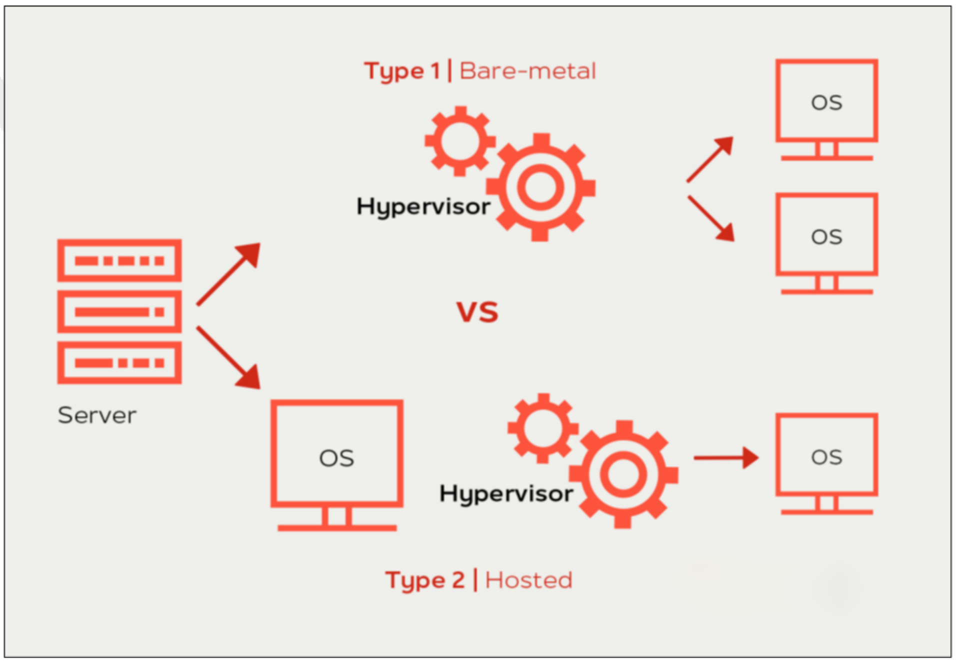

Figure 3.2: Types of Hypervisors [28].

In addition, the number and size of virtual networks prompted a consortium of network hardware and software vendors or publishers to create the Virtual eXtensible Local Area Network (VXLAN) standard [29] to meet these constraints.

a. External network virtualization

Network virtualization is a concept originally used in traditional physical network infrastructures. This is called external network virtualization [30]. Local Area Networks (LAN) are divided or combined into logical networks called virtual local area networks or Virtual Local Area Networks (VLAN). This makes it possible to create Broadcast Domains (broadcast domains) managed by the switches independently of the location of the equipment and terminals: these are logically managed broadcast domains. An identifier (VLAN ID) is used to mark the belonging of packets to a VLAN. It is conveyed in the links between switches and network cores/routers through links called trunks.

b. Internal network virtualization

Network virtualization in virtual infrastructures is called internal network virtualization [30]. This consists of emulating network components: interfaces, links, equipment, and protocols. The physical resources of the network are then shared between the virtual machines. It should be noted that external network virtualization can of course also be used in virtual networks. In addition, the recent Software-Defined Networking (SDN) model offers new possibilities for IaaS service providers [31]. This is a new way to design and manage network architectures, where the control plane is completely decoupled from the data plane. Indeed, the control plane is then not deployed on the same equipment which is in charge of transmitting the data. For instance, routing decisions on a switch are pushed to a dedicated controller, while the forwarding function remains on the switch. The network is then in a way "programmable".

In cloud infrastructures, one can potentially find a very large number of virtual machines, and therefore a very large number of virtual networks. VLANs can only assign 4096 different identifiers (VLAN IDs being coded on 12 bits). Additionally, VLANs do not allow logical networks to be distributed across boundaries established by OSI model Level 3 equipment (eg, a VLAN cannot be distributed across the Internet) and therefore restrict their use. This can be an obstacle to the migration of virtual machines requiring the crossing of level 3 borders. These problems of extensibility are solved by the concept of VXLAN, which consists of encapsulating level 2 frames (which can of course already belong to a VLAN and therefore contain a VLAN header) in level 4 UDP packets, capable of crossing broadcast domains. VXLAN segments are identified by an identifier: the VXLAN Network Identifier (VNI), coded on 24 bits and therefore making it possible to identify more than 16 million different networks. Tunnels are thus created on level 3 IP networks, making it possible to have VXLAN segments in different places. The entry points of such tunnels, called VXLAN Tunnel End Points (VTEP), are in charge of encapsulating the frames and transmitting them in UDP packets, or decapsulating the UDP packets and transmitting the frames to the correct recipients. VXLANs, therefore, solve the problem of limiting logical networks in the cloud and make it possible to cross layer 3 boundaries to distribute logical networks [29]. This is necessary for example when migrating a virtual machine between two physical machines separated by several level 3 boundaries, as shown in Figure 3.3 [32]. In this figure, the gray virtual machine assigned to a VXLAN can, unlike the red virtual machine assigned to a VLAN, migrate beyond the level 3 perimeter and is instantly connected to the correct IP network after migration.

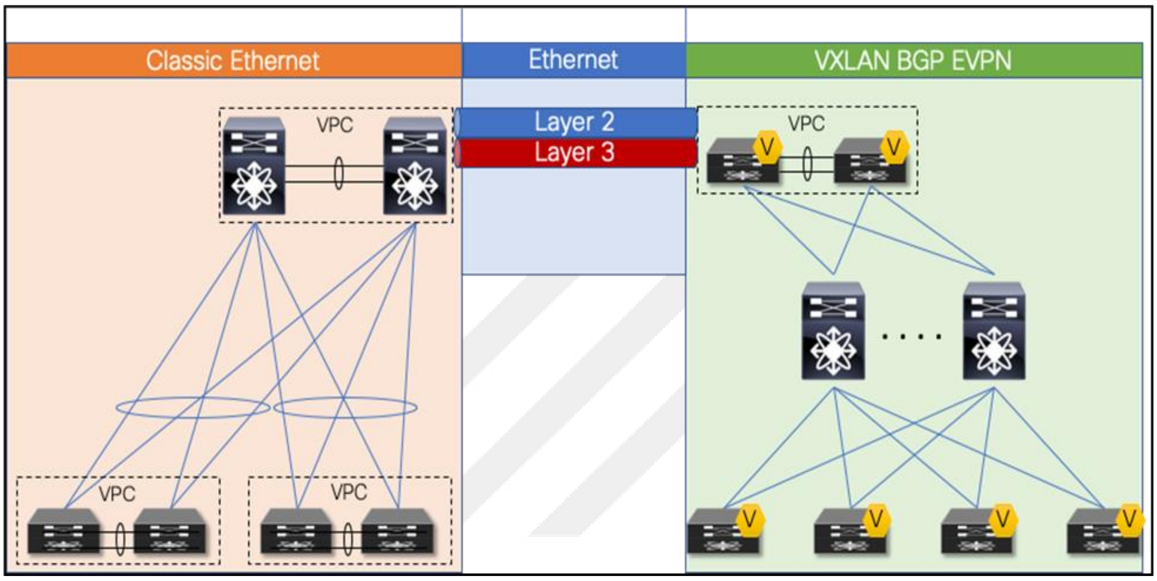

Figure 3.3: Interest of VXLANS for Migration [32].

3.3 SECURITY ISSUES

Cloud infrastructures are, like any distributed computing system, exposed to security issues. Indeed, the number and diversity of users of these infrastructures as well as the large quantity of hardware and software components imply the presence of security threats. Security is an area of dependability, which should first be introduced. To do this, we use the terms and definitions as presented in the Malicious and Accidental Fault Tolerance for Internet Applications (MAFTIA) project [33], [34]. Next, we look at security and specifically network security in cloud computing.

3.3.1 The Operational Safety

A computer system's dependability is what enables its users to have confidence in the service it provides for them. A system's behavior as seen by itself or its users is the service it provides; a user is any other system (human or otherwise) that interacts with the system under consideration. Non-dependability is when trust can no longer, or will no longer be able, to be placed in the service delivered. Dependability is articulated around three axes: the attributes characterizing it, the obstacles preventing its achievement, and the means to achieve it. Figure 3.4 presents these axes in the form of a tree.

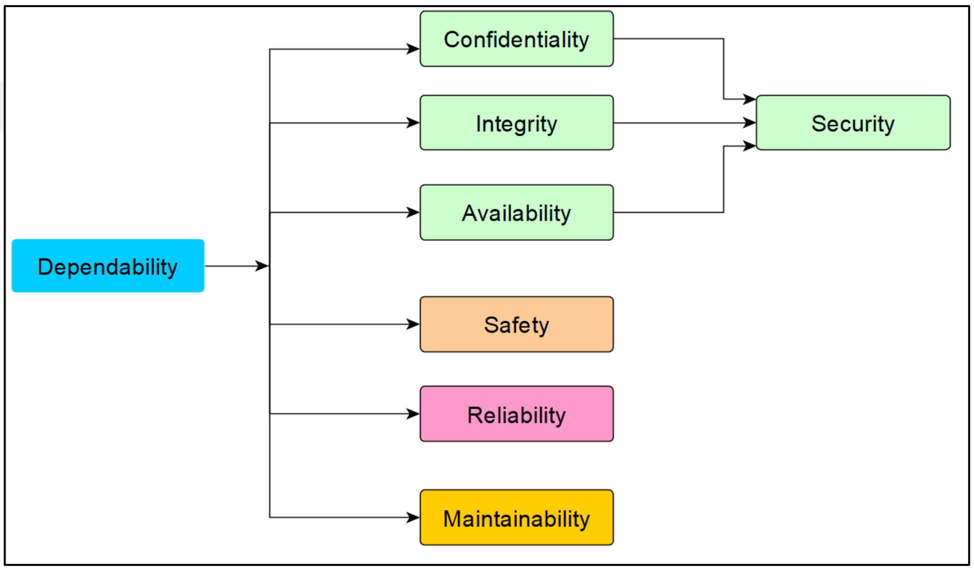

Figure 3.4: Dependability Tree.

3.3.1.1 Dependability attributes

The dependability attributes make it possible to express the properties expected of a system, and to assess the quality of the service delivered, as resulting from the obstacles and the means of opposing them. These are availability (ability to be ready for use), reliability (continuity of service), safety-harmlessness (absence of catastrophic consequences), confidentiality (absence of unauthorized disclosures of the information), integrity (absence of inappropriate alterations to the information), and maintainability (suitability for repairs and upgrades).

3.3.1.2 Impediments to operational safety

Impediments to dependability are the undesirable, but not expected circumstances, causes, or results of the non-dependability. These are faults, errors, and failures. There is a causal relationship between faults, errors, and failures. A fault is the adjudged or supposed cause of an error. Faults can be classified according to their phenomenological cause (physical or man-made), their nature (accidental or intentional), their phase of creation or occurrence (during development or in operation), their situation in relation to system boundaries (internal or external), and their persistence (permanent or temporary). An error is part of a system's state that can cause a failure.

3.3.1.3 Means for operational safety

The means for dependability are the methods and techniques making it possible to provide the system with the ability to deliver a service consistent with the performance of its function and to give confidence in this ability. These are fault prevention, fault tolerance, fault elimination, and fault prediction. Fault prevention consists of preventing, by construction, the occurrence or introduction of faults. It is mainly obtained by specification and development methods pertaining to systems engineering. Fault tolerance consists of providing a service capable of fulfilling the function of the system despite faults. It is implemented by error detection and system recovery. Fault elimination involves reducing the number and severity of faults. It can be carried out during the development phase of a system by verification, diagnosis, and correction, or during its operational phase by maintenance. Fault prediction consists of estimating the presence, creation, and consequences of faults. It is performed by evaluating the behavior of the system with respect to the occurrence of faults, their activation, and their consequences.

3.3.2 The Security

The area of dependability that interests us is that of safety-immunity, which we simply call safety. Security, which corresponds to the term security, is defined as the ability of the computer system to resist physical or logical external attacks. The Information Technology Security Evaluation Criteria (ITSEC) [35] Describe security as the intersection of three characteristics: secrecy, integrity, and information availability. We are therefore talking about the security of information, information systems, or data. Information is an element of knowledge translated by a set of signals according to a determined code, with a view to being stored, processed, or communicated. A datum is a representation of information in a conventional form adapted to its use [36]. It should be noted that there are also metadata, which corresponds to indirect information related to information or services. This is, for example, the time of creation, modification, or destruction of data, the identity of the person who performed an operation on data, the location of data, etc. Security has its own terms, the definitions of which we recall before detailing the three properties that compose it, then giving an overview of the means for security.

3.3.2.1 Terminology

Among the most used terms in the field of computer security, we find in particular vulnerability, attack, intrusion, threat, risk, and incident. The terms vulnerability, attack, and intrusion are defined in [33], [34].

a. Vulnerability: fault created during system development, or during operation, that can be exploited to create an intrusion.
b. Attack: fault of malicious interaction, through which an attacker deliberately seeks to violate one or more security properties. This is an intrusion attempt.
c. Intrusion: External malicious fault resulting from an attack that successfully exploited a vulnerability.
d. Threat: security attack possibilities and probabilities. A threat is defined by the attack process, the target, and by the outcome (consequences of a successful attack).
e. Risk: a result of the combination of threats and vulnerabilities.
f. Incident: an event which is not part of the standard operations of a service, and which causes or may cause an interruption of service or alter its quality.

3.3.2.2 Properties

Security is the combination of three properties (confidentiality, integrity, and availability), whose definitions we use here from [37].
Confidentiality is the property of information not to be revealed to users who are not authorized to know it. The now famous Heartbleed vulnerability [38], present in the OpenSSL cryptography software library, allows attacks violating data privacy. Indeed, by

exploiting this flaw, an attacker can recover information stored in memory on the targeted machines.

Integrity is the property of information not to be altered. The ILOVEYOU worm [39], which appeared in 2000, was propagated through Microsoft Outlook messaging. Once executed, it replaced many files present on the system. By altering the information in these files, he violated the data integrity property.

Availability is the property of information to be accessible when an authorized user needs it. For example, the online video game platforms of Sony and Microsoft have been the target of a denial of service attacks several times, rendering these platforms inaccessible and violating the property of data availability.

Other properties such as authenticity, auditability, intimacy, durability, or exclusivity are sometimes used. However, they remain expressible in terms of confidentiality, integrity, or availability. For example, the authenticity of a message corresponds to the integrity of its content (data) as well as its origin (metadata).

3.3.2.3 Means for security

Table 3.1, taken from [33], presents a classification of the methods available to guarantee security against attacks, vulnerabilities, and intrusions, according to the four means for dependability.

3.3.3 Threats, Vulnerabilities, and Attacks in the Cloud

Cloud computing is particularly exposed to threats and vulnerabilities that can be of various natures. We present the main characteristics of the attackers, threats, and vulnerabilities present in the cloud.

3.3.3.1 Nature of attackers

To understand who attackers in the cloud might be, we assume that any cloud actor is a potential attacker. An actor is a person who can fill one or more roles. There are many possible roles in the cloud. As this is also an economic model, certain roles are functions with a more economical tendency (linked to the notions of service, customer, and supplier). In addition, cloud infrastructures involve many technological components.

Table 3.1: Classification of Methods for Mecurity [33].

	Human attack	Technical attack	Vulnerability	Intrusion
Prevention	Deterrence, laws, social pressure, secret service...	Firewall, authentication, authorization...	Formal and semi-formal specifications, rigorous development, and management methods	Prevention and elimination of attacks and vulnerabilities
Tolerance	Prevention of vulnerabilities, elimination, and tolerance to intrusions		Prevention and elimination of attacks, tolerance to intrusions	Error detection and recovery, masking faults, intrusion detection, fault management
Elimination	Physical countermeasures, attacker capture	Preventive and corrective maintenance to remove malicious agents	Formal proof, model-checking, inspection, test, preventive, and corrective maintenance, including security patches	Elimination of attacks and vulnerabilities
Forecast	Intelligence gathering, threat assessment...	Analysis of latent malicious agents	Evaluation of vulnerabilities, difficulties in exploiting them, their potential consequences...	Prediction of vulnerabilities and attacks

Other roles are more technologically oriented (related to component checks). Table 3.2 lists the different roles as cited in the literature, depending on whether they are of an economic or technological orientation. In the rest of this document, we will focus on the economic roles of the customer and service provider.

In addition, the notions of an internal attacker (insider in) and an external attacker (outsider in) are difficult to define in this context. According to [40], an internal attacker is a person leading his attack from inside the attacked system, while an external attacker leads his attack from outside the system. The Computer Emergency Response Team (CERT) at Carnegie Mellon University has defined an insider attacker as a former or current employee, contractor, or another partner with access to an organization's network, system, or data and using that access. with the intention of harming the organization's information systems' availability, confidentiality, or integrity [41].

Two main types of an internal attacker that can exist in the cloud are differentiated in [41]: employee of the cloud provider (administrator who can target, for example, a client organization or his employer) and employee of the user organization of the cloud (who can target by cloud data or external employer data). The first type of internal attacker corresponds to a corrupt administrator of the cloud provider, carrying out the theft of sensitive information, the motivations of which are generally financial. It can also be an administrator employed by the customer organization, although this is not common in the cloud because the administrator is usually employed by the cloud provider. An employee who wants to harm his employer (the cloud provider) could do damage to a client organization to damage the reputation of the cloud provider.

Table 3.2: Roles in the IOT-Cloud.

Economic roles	Technology roles
Cloud service consumer cloud service provider Cloud services developer Distributor of cloud services	System administrator computer operator
Sellers Service provider Cloud consumers End users	Network administrator Storage Administrator
Service provider service user infrastructure provider	Database administrator Final user cloud architect
Customer Service provider infrastructure provider Service Provider Aggregator platform provider Consultant	code developer cloud administrator cloud operator
Manager cloud service manager cloud user Cloud Data Architect	Cloud Application Architect cloud developer

Table 3.2: Roles in the IOT-Cloud."Table Continued".

cloud storage administrator	

The four levels of administrators considered here are:

a. Application administrators.
b. System administrators.
c. Virtual machine administrators.
d. Accommodation administrators.

An administrator has the privileges of administrators of all levels below him (for example, a system administrator also has the privileges of the application administrator). The second type of insider attacker is one who seeks (potentially directed by an external source) to gain unauthorized access to their organization's data that uses the cloud. Also in [41], a third type of internal attacker is even proposed, close to the previous one, but with the difference that the employee of the organization uses cloud services to carry out an attack on his own employer, but not necessarily on data located in the cloud. In summary, [41]'s classification of internal cloud attackers, therefore, separates administrators of the cloud provider (with different levels targeting his employer or a client organization) and employee of the client organization (targeting his employer through the cloud). In other aspects, insider attackers are differentiated according to: service provider, an employee of the service provider, and employee (with authorized access privileges) in the cloud user's organization. Still, an external attacker simply corresponds to a user who does not belong to his victim's organization. Its range of action is relatively wide, being able to act from the Internet and generally being able to easily enter the cloud perimeter by subscribing to a subscription for example.

3.3.3.2 Classes of Attacks, Incidents, and Threats

The literature offers different ways of classifying computer attacks [42], according to different criteria: type of attacker, attack objective, attack vector, attack target, attack result, etc. Given the complexity of roles among cloud actors, diversity of targets, and potential objectives, it is more relevant to use the attack vector to obtain an orderly and easily understandable classification. Therefore, to present the main families of computer attacks, we have retained a classification based on the attack vector as a criterion, like that proposed in [43]. An attack vector is the means for an attack to reach its target. Table 3.3 presents a classification of computer attacks by attack vector. It gives an overview of the main categories of computer attacks but does not aim to list all the existing attacks. We have listed the main attacks found in cloud environments. These attacks include known vulnerabilities traditionally found in physical infrastructure. However, their exploitation and their visibility are facilitated by the current concentration of significant computing and storage capacities. In addition, there are also new forms of attack specific to virtual environments such as reconnaissance and side-channel attacks [44].

However, on the other hand, Alert Logic regularly carries out studies [45] on incidents in cloud infrastructures, based on the collection of large quantities of data. The incidents are sorted according to six classes, presented in Table 3.4. It should be noted that the terms attack and incident are confused here. These incidents are of the same nature as those found in traditional infrastructures (excluding the cloud). Studies are also conducted on incidents occurring in clouds and those occurring in traditional infrastructures, to make a comparison between the two types of incidents.

Web application attacks constitute a predominant proportion of cloud incidents, these are increasingly like those historically encountered in traditional infrastructures.

Table 3.3: Classes of Incidents In The Cloud.

Incident class	Definition
Application attacks	Exploit attempts on applications or services not running on the HTTP protocol.
Brute force attacks	Exploit attempts with many combinations to find a flaw.
Malware/botnet activity	Malicious activity of software installed on a host that can communicate using a command-and-control channel.
recon attack	Activity focused on scans and mapping of networks, applications, or services.
Vulnerability scan	Automatic discovery of vulnerabilities.
Web application attacks	Attacks targeting the interface, logic, or database of a web application.

For its part, the Cloud Security Alliance (CSA) [46], [47] has defined seven major classes of cloud threats:

a. Abuse and misuse of the cloud.
b. Insecure APIs and interfaces.
c. Internal malice.
d. Sharing technology issues.
e. Loss or leakage of data.
f. Account or Service Hijacking.
g. Unknown risk profile.

These classification efforts reflect the growing awareness of the dangers of cloud threats and the current importance of securing these environments.

3.4 NETWORK SECURITY MECHANISMS IN THE CLOUD

Network security mechanisms aim to ensure the security of the hosts and applications of the networks to which they belong. We are particularly interested in two network security mechanisms:

a. Network access control, carried out by firewalls.
b. Intrusion detection, carried out by ID systems or IDS.

3.4.1 Firewall Virtual

A firewall is a tool for controlling traffic flowing between the inside and outside of a security perimeter. The security perimeter constitutes the limit between the network that one considers safe or that one wishes to protect and the rest of the Internet [48]. Firewalls can provide different services, but let's focus on the basic service, namely access control at the network level. The main function of network access control is packet filtering, operated by filtering rules making it possible to prohibit or authorize certain types of traffic. These rules are established according to the parameters of the protocol headers located in the packets. They are applied to each packet passing through the firewall, respecting a certain order of priority in the application of the rules. Most firewalls are generally stateful or more simply stateful, i.e. they also check that packets belong to a current connection (such as a TCP connection) before authorizing them. They thus verify that each packet authorized by the rules initiates a new connection or is indeed part of an already existing connection.

Additionally, different levels of inspection can be performed on packets. The most common level is Deep Packet Inspection (DPI) which focuses on looking at the contents of a packet in depth to authorize access or not. Web application firewalls or Web Application Firewalls (WAF), dedicated to the protection of Web servers, are also increasingly widespread.

The complexity of cloud architectures, coupled with the complexity of today's applications, makes network access control requirements more important. The work of firewalls, in charge of these access controls, is therefore made more complicated. The security perimeter is more complex to draw than in a traditional environment, due to the multiplicity of organizations present in the cloud. Thus, the positioning of firewalls and their behavior are strategic aspects. To maintain defense in depth in virtual infrastructures, there are the following two types of virtual firewalls [49], shown in Figure 3.5 [50]:

a. Bridge mode firewall: virtual machine deployed as a virtual network gateway, capable of routing, filtering, and address translation of incoming and outgoing traffic. These firewalls are usually controlled by customers.
b. Firewall in hypervisor mode: software component embedded in the hypervisor that filters the traffic sent or received by the virtual machines without taking into account the network topology. It is usually controlled by the service provider, but some rules can be applied by customers only on certain virtual networks, which means giving customers partial control over this type of firewall.

3.4.2 Intrusion Detection Systems

According to [33], intrusion detection concerns all the practices and mechanisms used for the detection of errors that can lead to a security failure, and/or for the detection of attacks. Still, according to [33], an IDS is the implementation of intrusion detection practices and mechanisms. The role of IDS is therefore to detect and/or block (in this case we speak of Intrusion Prevention System) attacks occurring within a system or network. They can be deployed on the monitored host, which is called Host-Based IDS (HIDS); or on the monitored network, which is called Network-Based Intrusion Detection.

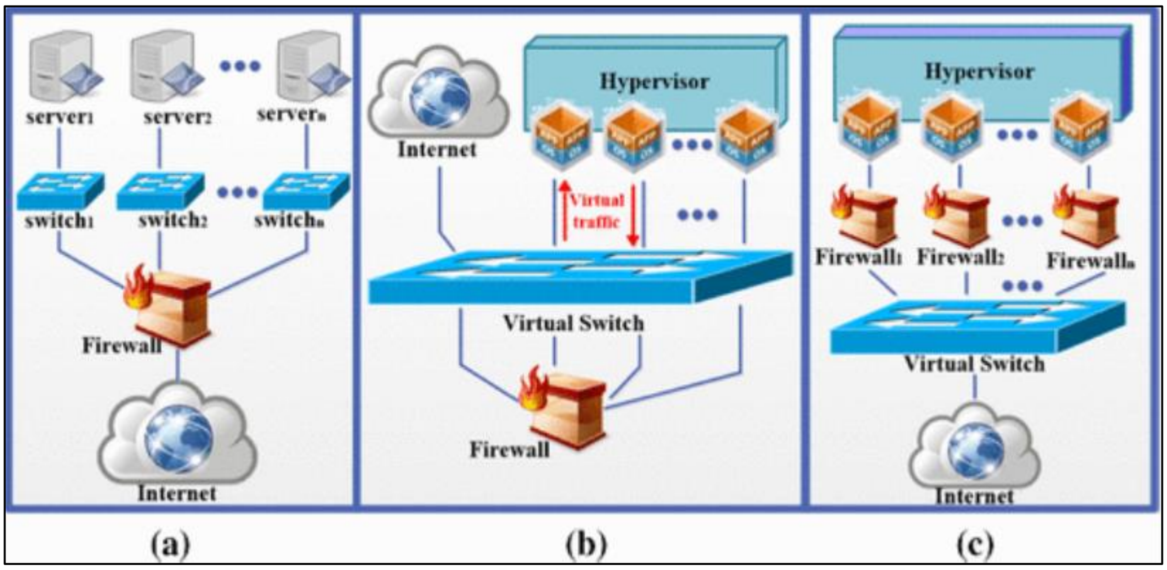

Figure 3.5: Firewalls in the Cloud [50].

As shown in Figure 3.6, there are two main detection techniques: by Misuse (signature-based detection or misuse detection) or by behaviors (anomaly detection). The signature-based approach consists of detecting attacks by checking whether the observations correspond to known attacks, while the behavior-based approach (or anomaly detection) consists of detecting an attack by checking that the observations do not correspond to known attacks. legitimate behaviors of reference. Some IDSs combine the two approaches in order to achieve better results.

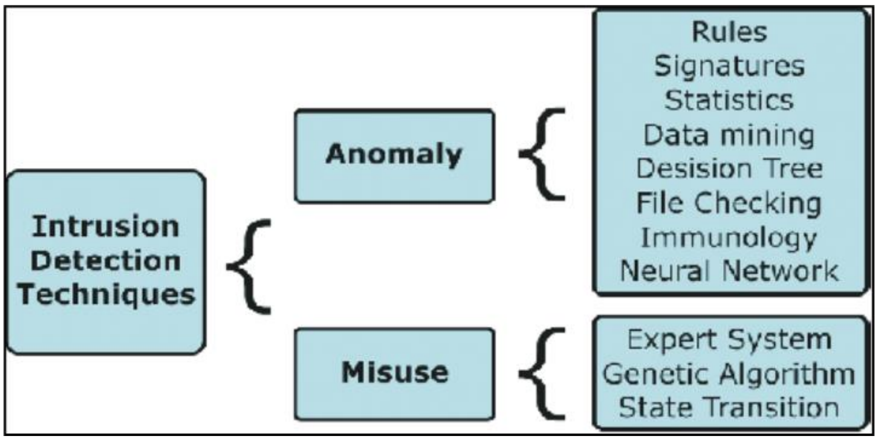

Figure 3.6: Intrusion Detection Techniques [51].

3.4.2.1 Cloud Probe Deployment Tiers

In a traditional physical infrastructure, the detection probes are generally placed at the entrance of the networks to be monitored, or in the operating systems to be monitored. In the cloud, the multitude of organizations sharing the infrastructure and the addition of virtualization makes the positioning strategy of the probes more difficult to establish. The Cloud Security Alliance (CSA) [52] has described several possible deployment levels of IDS/IPS probes in the cloud:

a. Internal Virtual: The probe is deployed on a virtual appliance that bridges the virtual switches and is attached to a physical network interface on the host.

b. External physical: the probe is deployed on dedicated physical equipment connected to the physical host which redirects traffic from its virtual switches using traffic mirroring or port mirroring, or directly forwards traffic to the probe which intercepts.

c. VMM: the probe is deployed within the virtual machine manager using the APIs offered.

d. Guest system: the probe is deployed within the operating system of virtual machines.

e. Application: the probe is deployed within the code of the monitored application.

f. Hybrid: The probe is deployed within the VMM to intercept traffic and send it to an internal virtual or external physical probe.

Figure 3.7 shows these different levels of deployment (identified by the same numbers in the figure) within a virtual infrastructure [53]. Table 3.5 presents the advantages and disadvantages of each of the possible deployment levels. This table shows that to deploy probes, a compromise has to be assessed between ease of deployment, scaling, performance, and exposure to flaws.

3.4.2.2 Features of Operating in the Cloud

New approaches to cloud intrusion detection provide operating characteristics specific to the environment in which these systems are expected to operate.
Most of the proposed solutions are distributed, we then speak of Distributed IDS (DIDS). Indeed, in a traditional environment, a single probe positioned at the entrance to a company's

network could be enough to monitor attacks coming from outside. As explained previously, the control perimeter is more porous in the cloud, and it is, therefore, necessary to position probes in a distributed manner to monitor all the different possible traffic and avoid overloads. One of the challenges is then the ability to synchronize probes and correlate detected events. The purpose of the correlation is to improve the detection of intrusions and to reduce the number of false alarms, and this function is carried out by a center.

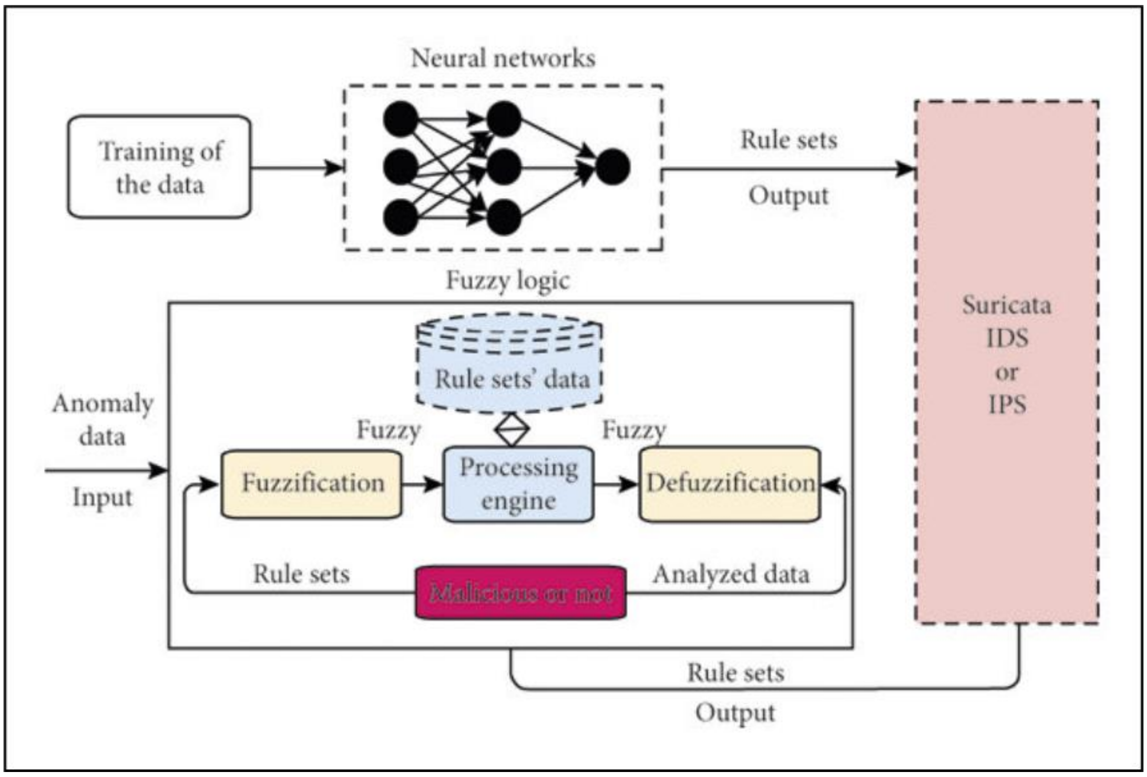

Figure 3.7: IDS/IPS Probe Deployment Levels [53].

management of security events and information, or Security Information and Event Management (SIEM). The role of such a unit is therefore to recover, analyze and offer a response to events occurring in cloud infrastructures. Although many different formats are used (syslog, CSV, JSON...), there is a standard data exchange format between probes and management systems, the Intrusion Detection Message Exchange Format (IDMEF), defined in RFC 4765 [54].

The second particularity of certain IDSs in the cloud is their cooperative character. This consists of informing all the other probes in the event of an alert generated by one of the probes. A majority vote can then be carried out to validate the alert and fill the filtering tables (used to discard malicious packets) of the other probes. It should be noted that the cooperation of the probes differs from the correlation of the alerts, in the sense that the cooperation helps in decision-making as to the validation of the raising of alerts.

Table 3.4: Advantages and disadvantages of cloud IDS/IPS probe deployment levels.

Deployment	Advantages	Disadvantages
Internal virtual	a. Easy to deploy. b. Easier encrypted traffic interception.	a. Loss of hardware support benefits (ASIC...). b. A probe overload quickly impacts virtual machines or the physical host. c. A hypervisor compromise compromises the probe. d. Risks of loss of connectivity in the case of an IPS due to the absence of hardware fail-open functionalities.
External physics	a. Independence from the hypervisor. b. Overloading the probe has less impact on virtual machines or the physical host. c. In the case of an IPS, a fail-close mechanism is available. d. Easier encrypted traffic interception	a. In the case of an IDS, not all virtual switches support port mirroring. b. Dedicated physical hardware is required. c. Possible scaling issues.
VMM	a. Easy to deploy. b. Interception of traffic between virtual machines. c. Interception of guest operations. - Possible inspection of offline virtual machines.	a. Loss of hardware support benefits (ASIC...). b. A probe overload quickly impacts virtual machines or the physical host. c. A hypervisor compromise compromises the probe.
Guest system	a. Easy to deploy. b. Interception of traffic between virtual machines. c. Interception of guest operations.	a. Loss of hardware support benefits (ASIC...). b. A probe overload quickly impacts virtual machines or the physical host. - A guest compromise compromises the probe.

Table 3.4: Advantages and disadvantages of cloud IDS/IPS probe deployment levels."TableContinued"

Application	a. Ability to use application logic to further define intrusion attempts. b. No deployment is necessary.	a. Requires application support. - Loss of hardware support benefits (ASIC...). b. A probe overload quickly impacts virtual machines or the physical host. - A guest compromise compromises the probe.
Hybrid	a. Interception of traffic between virtual machines. - Possibility to avoid overloads by using an external physical probe. b. The inspection policy can easily track moving virtual machines.	A compromise of the hypervisor compromises the probe.

3.5 SECURITY ANALYSIS AND ASSESSMENT

The objective of the evaluation and the analysis of a system is to reveal faults or to provide values making it possible to measure the effectiveness of the mechanisms of protection vis-à-vis malicious acts. Security analysis and evaluation contribute to the elimination and prediction of faults. This is an area of research that should be introduced because, in the following chapter, we detail the state of the art of methods and techniques used to evaluate and analyze the network security mechanisms that interest us to analyze.
Analyzing a system consists in examining or studying this system in order to identify and compare its components. As described in [55], the analysis comes from the field of verification, which reveals the existence of faults. Indeed, verification can be implemented by analysis techniques. We do not aim to conduct an analysis of codes or instances of software but to conduct an analysis of security mechanisms. However, the deployment of a security mechanism can be likened to the installation and configuration of security software.

3.6 NETWORK ACCESSIBILITY ANALYSIS

The accessibility analysis consists in discovering the different hosts of an information system and the network connectivity between them, taking into account the addressing plan, the rules of routing, filtering, and translation of packets. It can be carried out in a static way, a white box approach with prior knowledge of the analyzed network, or dynamically, a black box approach without prior knowledge. We present here these two aspects, as well as their advantages and disadvantages.

3.6.1 Static Analysis

The static analysis consists of analyzing routing, filtering, and packet transformation rules based on the configuration of network equipment. Algorithms are then executed to deduce the network accessibility while modeling this information in an accessibility or connectivity matrix, or even in an accessibility graph.

All these approaches are very interesting, whether they are centered on the calculation of accessibility, on the in-depth analysis of firewall configurations, or on the verification of security policies.

3.6.2 Dynamic Analysis

The objective of the dynamic analysis is to ascertain network accessibility. In contrast to static analysis, one aims to do a network mapping when little to no configuration information is available. To ascertain the current accesses, interrogations created by sending packets are conducted. Although there hasn't been much study done in the area of dynamic accessibility analysis, several tools and strategies that assist implement it are commonly utilized. These might be basic ICMP requests [56]. network port scans to get level 3 mapping, or level 4 mapping and above. It cannot account for changes in topology and only considers standard packet processing rules and connectivity. While port scanners [57] hey frequently cover a wide range of protocols for querying, but because of their aggressive character, they sometimes discover that security features prevent them from doing so. It should be noted that if there are too many hosts, completing a dynamic analysis of the entire network may take too long or perhaps be impossible. Additionally, it might occasionally be quite difficult to install scanners across all network segments in physical setups.

3.6.3 Comparison

Although having the same objective, static analysis and dynamic network accessibility analysis have advantages and disadvantages related to the different techniques and knowledge they use. The advantages and disadvantages of each method are summarized in Table 3.6.

Table 3.5: Advantages and Disadvantages of Network Reachability Scan Types.

	Advantages	Disadvantages
Static analysis	a. Fast. b. Can consider and anticipate changes in topology. c. Can be used outside of production, in the network design phase, prior to its deployment	a. Difficulty managing the consistency of the entire analyzed topology. – Assumes that the packet processing software mechanisms applied on the network are indeed those configured.
Dynamic analysis	a. The accesses found are confirmed.	a. Processing may be slow if the topology is complex. – Aggressive or intrusive techniques therefore inhibited.

Dynamic analysis is favored for audits employing minimal previous knowledge or if little trust is placed in the execution of the settings by the instruments that analyze the packets. Static analysis is often chosen in situations when one does not want to be too invasive on the network.

3.7 EVALUATION OF INTRUSION DETECTION SYSTEMS

The evaluation of the IDS makes it possible to validate the design and implementation of these systems. This also makes it possible to compare different IDSs to make the right choices in their integration into the system to be protected, in order to meet the needs in terms of security. IDSs have, like any system, specifications that must be met. In [58], [59] we differentiate between functional specifications and performance specifications. The main functional specifications expected of an IDS are:

i. Continuity of monitoring service and intrusion report.
ii. Low false positive rate for anomaly detection systems.
iii. Ability to efficiently process and merge data from multiple distributed sources.
iv. Production of sufficient information in the event of an intrusion.
v. A capacity to react against distributed and coordinated attacks.
vi. Fault tolerance: be designed in such a way as to be able to operate in a hostile environment, susceptible to attacks.
vii. Ease of configuration and modularity for each host or network segment in order to dissociate the tests.
viii. Easy scalability to learn from past experiences to improve detection.
ix. Interoperability with other security tools (network and host), forensic (post-incident analysis), and data processing.

The main performance specifications are:

i. Intrusion detection must be done in real-time.
ii. Scaling capability to support additional loads.
iii. The operation of the IDS should not interfere with the operation of the monitored system/network, i.e., the deployed agents should be aware of the resources consumed.

The main problem in the evaluation of IDS is the choice of the method, which is supposed to provide the most precise and complete result possible on the capacities of these systems. In order to evaluate SDIs, two main families of methods emerge:

a. evaluation by description and analysis, a rather white box approach, offline, based on models
b. experimental evaluation, a rather black box approach, online, conducted around the test. It should be noted that some approaches are hybrid.

We are interested in evaluating IDSs in production for which we do not necessarily have access to the specifications. Although we introduce evaluation by description and analysis, we mainly present the main experimental approaches.

3.7.1 Evaluation By Description and Analysis

This type of SDI evaluation is based on the modeling of the system to be evaluated whatever its stage of development. To date and to our knowledge, the most complete works on evaluation by description and analysis are undoubtedly analysis of IDS according to the classes of attacks to be covered, with the aim of allowing a better design of this type of system. Attacks are therefore classified mainly according to the parameters of the IDS and the expected alarms. An IDS is then evaluated according to its descriptions (specifications) and for each attack class and alarm raising conditions. The components identified in these studies are the probe, the preprocessor, the detector, and the alert generator.

3.7.2 Experimental Evaluation

Experimental evaluation relies on generating and injecting data into a deployed system and using metrics to measure the performance of that system. We first present some of the main approaches in the literature. The robustness of the experimental evaluation rests on the choice of the data used as well as on the evaluation metrics used, we then detail these two key elements.

3.7.2.1 Main Approaches

One of the very first works on the IDS experiment assessment was introduced in 1996 in [60]. The technique provided is a place to start when creating an SDI evaluation plan. There are three test procedures that are defined: one to gauge detection capacity, one to gauge resource utilization, and one to assess how well the IDS performs in the face of traffic congestion. MIT's Lincoln Laboratory and the Defense Advanced Research Projects Agency (DARPA) collaborate on the most well-known experimental IDS assessment study [61]. A large dataset (about 300 different attacks) was designed to be injected via scripts. However, the data has not been updated at all for several years, and these approaches were criticized from several angles in [62]. One can also cite the work supported by IBM Zurich [63] which aims to evaluate several IDSs comparatively using a platform equipped with several client hosts and servers controlled by a single machine. The available report unfortunately mentions very little information on the data used and the results obtained.

More recently, work on creating datasets has been provided in [64], where the authors propose a strategy to address the lack of datasets for IDS evaluation. Their approach allows the generation of malicious network traces on a virtual platform, and the replay of these traces thanks to a tool that verifies the detection capacity of the evaluated IDS. We can also cite [65], on the generation of traffic for NIDS with detection by signature from an application workload generation model. In addition, a packet processing capacity estimation method is presented. However, even if the methods proposed in these two works are interesting, the takeaway from this review is that for most tests, Snort matches or exceeds the performance of commercial IDSs. We can also cite the work from [66], [67], implementing attack scenarios targeting Web Applications, and used in a platform dedicated to the evaluation of IDS specialized in the protection of Web application servers.

3.7.2.2 Experimental Data

Experimental data is data that is used by test tools to generate activities that the evaluated IDS must process. The choice of these data is crucial for the calculated evaluation metrics to be relevant. Another important choice is how to collect and produce the data. This may involve the production of synthetic traffic reproducing a real environment or the capture and replay of traffic resulting from real observations. Examples of collecting data from real observations are honeypots [68]-[70]systems made voluntarily accessible to attackers and vulnerable in order to record their malicious activities. Recently, some works [71] - [73] have proposed honeypots deployed in public clouds, to profile attackers in the cloud. These characterizations could allow inferring experimental data for the cloud. Unfortunately, to date, these data remain similar to those observed in traditional environments and are not directly usable.

The types of experimental data used are associated with two categories of activities:

a. Legitimate Activities: Activities considered not to violate security rules or policies and are consistent with normal behavior. They are used to verify that an IDS does not react to these activities.
b. Malicious activities: activities considered to violate security rules or policies and correspond to abnormal behavior. They are used to verify that an IDS actually reacts to these activities, that is to say, to evaluate its false negative and true positive rates. It should be noted that numerous obfuscation techniques exist in order to make it more

difficult to detect malicious activities. This, therefore, makes it possible to evaluate the resistance of the IDS to attacks targeting the detection algorithms.

Many databases, tools, or distributions are available to generate legitimate or malicious activities from data. Protocol reverse engineering is one of the best-known associated methods. It was developed with the aim of analyzing activities in order to be able to reproduce them for evaluation purposes.

We first expose some techniques for obfuscating malicious activities, then come back in more detail on the reverse engineering of protocols.

c. Obfuscation techniques

The obfuscation of attacks consists in making legitimate the data handled by the IDS while preserving the negative interpretation of these data by the target. By using attack obfuscation techniques, an attacker can ensure that his attacks are not detected by the IDS, and this is called intrusion evasion [74]– [76]. In the field of experimental evaluation, this serves to particularly evaluate the false negative rate, ie the proportion of undetected attacks. These techniques are commonly embedded in malicious activity generation tools. The most common ones used to assess NIDS are:

a. IP fragmentation: consists of segmenting the attack on several packets so that the signature of the attack is no longer valid by a NIDS treating each packet independently.
b. Encryption: consists of encrypting the contents of attack packets so that only the target is able to read the contents to execute the attack. The NIDS is not able to decrypt the encrypted content and therefore detect the attack. The best-known example is the use of the HTTPS protocol to carry out web application attacks. We can also mention the very widespread virtual private networks (VPN) and in particular SSL VPNs.
c. Denial of service of the NIDS: consists in overloading the detection probe in order to alter
   the availability and continuity of the detection service. Making such an attack distributed (using multiple attacking sources) while spoofing source addresses, can make an investigation and service recovery even more complex.
d. Alert generation: consists of causing the NIDS to generate a large number of false positives in order to generate "noise" in order to hide the true positives.
e. Alteration of Time To Live (TTL): consists of inserting, between packets constituting an attack and which have a sufficiently large TTL, benign packets with a TTL small

enough to reach the NIDS but not the target of the attack because they will be rejected before reaching the target. In this way, the NIDS does not have the same perception as the recipient of the data exchanged.

The attack is then altered for the NIDS, which does not consider it as such (because the flow is composed of attack packets and benign packets) and does not detect it although it remains intact for the target (because it does not only receiving attack packets that have a large enough TTL).

f. Sending RST-like TCP packets with an invalid checksum may cause a NIDS to believe that communication with the target has ended (which actually refuses the RST-like packet), but the attacker may well continue to send subsequent packets.

g. Use of the Urgency Flag (URG) in a TCP packet is used to indicate the data to be processed directly by the recipient. Some NIDS do not process this parameter, it can be used to add useless data before the malicious pointed data so as to avoid detection of the malicious pattern when the set is read by the NIDS.

h. Shellcode polymorphism: allows to encrypt shellcode injected into the payload of a package. A decryption routine, different each time, decrypts the shellcode on the target machine. The NIDS is thus not able to find the signature of the shellcode that it sees encrypted.

There are also system attack obfuscation techniques used to evaluate system IDSs, (HIDS) Host-Based IDS, which we do not detail here since we are interested in network security mechanisms.

3.7.2.3 Protocol Reverses Engineering

protocol reverse engineering is the process of extracting application protocols without knowing the specification. We are interested in protocol reverse engineering because one of its possible applications is the evaluation of IDSs. Indeed, the inferred protocols can be used to replay traffic (which may correspond to legitimate or malicious activities) and to simulate the behavior of applications. There are two ways to infer protocols, either by analyzing the application's internal behavior or by analyzing its external behavior. We want to focus on network security, so we are interested in approaches and tools using network traces to infer and then replay protocols. Moreover, it should be noted that tools analyzing internal application behaviors also offer interesting formal representations, which propose a formal definition of the traffic replay problem, as well as a solution using dynamic binary analysis to replay application dialogs.

3.7.2.4 Evaluation Metrics

The basic metrics for the evaluation of IDS come from the field of learning (or ML), in which confusion matrices make it possible to evaluate the performance of classification algorithms by learning. Such matrices make it possible to compare expected results with obtained results. In the field of IDS evaluation, we are interested in comparing the activities carried out with the reactions of the IDS. There are four main metrics:

a. True negative or True Negative (TN): legitimate activity recognized as such.

b. True positive or True Positive (TP): malicious activity recognized as such.

c. False positive or False Positive (FP): legitimate activity recognized as malicious activity.

d. False negative or False Negative (FN): Malicious activity recognized as a legitimate activity.

Table 3.6: IDS Confusion Matrix.

		IDS reaction	
		No alert	Alert
Activities	Legitimate	True negative	False positive
	Malicious	False negative	True positive

Sometimes, the temporal notion is introduced in evaluations, as some authors propose the use of the rate of false alarms per day instead of the rate of false alarms over the entire campaign evaluation. Another, more concrete way to visualize the effectiveness of an IDS is the use of Receiver Operating Characteristic (ROC) type curves [77]. These curves are intended to synthesize, for a given IDS, the probability of recognizing malicious activities as such compared to the probability of considering legitimate activities as malicious. It is thus a matter of expressing, for a given activity, the probability of generating a true positive as a function of the probability of generating a false positive. The probability of true positives is equivalent to the detection rate (ability to recognize malicious activities as such), while the probability of false positives is equivalent to the rate of false alarms (ability to consider legitimate activities as malicious). These probabilities are of course dependent on the test environment, which makes it possible to plot different points establishing the correspondence between different values of one or the other of the two associated rates. In general, IDSs are often evaluated by comparing the measurements obtained with those of other IDS products. For example, in the case of the cloud, IDS solutions are generally evaluated by means of a comparison of their performance with the performance of a single IDS probe, on which the proposed system is. This is the case where the authors emphasize the low performance degradation for better resource optimization while avoiding having a single point of failure.

4. METHODOLOGY AND RESULTS

4.1 INTRODUCTION

In this chapter, we will describe our approach for detecting network anomalies in cloud and IoT using fuzzy logic. We will use two datasets, namely the UNSW dataset and KDD dataset. We will explain in detail the implementation of various ML techniques, such as Naive Bayes, J48 Classifier, XGBoost, KNN, and Random Forest, along with feature selection using J48. We will also discuss the rules used in the fuzzy type 1 model. Thus we will present the outcomes of our method for detecting network anomalies in Cloud and IoT environments, using various ML models such as RF, XGBoost, KNN, J-48 classifier, and Naive Bayes. We will demonstrate the performance of these models on two datasets, namely the UNSB dataset and the KDD dataset. Additionally, we will showcase the results of Fuzzy_types 1.

4.2 RESEARCH APPROACH

The research approach section is divided into two parts. The first part involves the use of classical ML algorithms. We will load the UNSW and KDD datasets and perform exploratory data analysis (EDA) to gain insights into the data. Then, we will train our models and move on to evaluation and comparison. The second part of the approach is called Fuzzy_type 1, which involves using J48 as feature selection. We will explain this in detail in the next subsections. Finally, we will evaluate our Fuzzy_type 1 model.

Figure 4.1 illustrates our proposed approach.

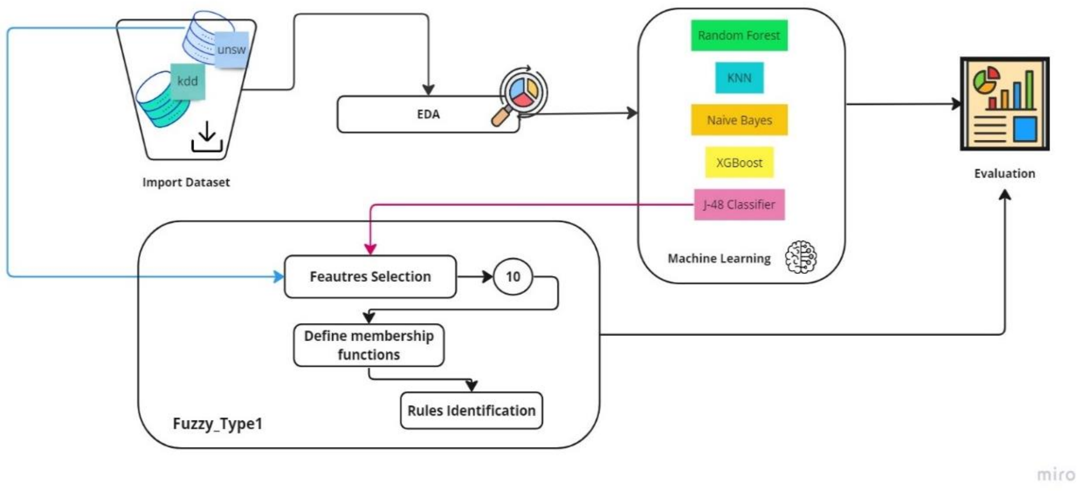

Figure 4.1: General Approach.

4.2.1 Dataset

4.2.1.1 UNSW-NB15

The Australian Centre for Cyber Security (ACCS) used the IXIA PerfectStorm tool in their Cyber Range Lab to build the UNSW-NB15 dataset. The technology was used to produce a blend of actual, current network traffic that is normal and synthetic, current attack behaviour. Using the Tcpdump utility, the network traffic was recorded, producing a 100 GB collection of raw packets in Pcap format.

There are nine distinct kinds of cyberattacks in the dataset, including fuzzers, analysis, backdoors, denial-of-service attacks, exploits, generic, reconnaissance, shellcode, and worms. Using the Argus and Bro-IDS tools, 49 features and class labels were created as part of the dataset analysis. Twelve algorithms were created to produce these attributes. The UNSW-NB15 dataset provides a valuable resource for researchers and practitioners to study and analyze modern cyber-attacks and normal network traffic.

4.2.1.2 NSL-KDD

The NSL-KDD dataset is an improvement over the KDD-CUP dataset, aimed at addressing some of the limitations of the latter. One of the key improvements in the NSL-KDD dataset is the removal of duplicate records in both the training and test sets, which helps to prevent overfitting of models and improve their generalizability. Additionally, the NSL-KDD dataset also increases the proportion of minority samples in the test set, making it more useful for evaluating the performance of intrusion detection models.

The NSL-KDD dataset consists of 2 main sets: the KDDTrain+ training set and the KDDTest+ test set. Each record in the dataset contains 42 attributes, with 41 attributes representing the characteristic attributes of the data and the remaining attribute indicating the type of attack. The dataset is widely used in intrusion detection research to evaluate the effectiveness of different ML models in detecting different types of attacks. Overall, the NSL-KDD dataset is an important resource for researchers and practitioners interested in studying and improving IDSs.

4.2.2 Exploratory Data Analysis

In this section, we will explain the exploratory data analysis (EDA) performed on both the UNSW and KDD datasets.

For the UNSW dataset, we first loaded the data and checked for any missing values. We then performed statistical analysis to gain an understanding of the dataset. We calculated the mean, standard deviation, and other descriptive statistics for each feature. We also plotted histograms to visualize the distribution of the data.

Next, we performed correlation analysis to check for any linear relationships between features. We used a heatmap to visualize the correlation matrix. We also used boxplots to compare the distribution of each feature between different attack types.

For the KDD dataset, we followed a similar process. We loaded the data and checked for missing values. We then performed statistical analysis and plotted histograms to visualize the distribution of the data.

The EDA helped us gain insights into the datasets and understand the characteristics of the data. This information was useful in selecting appropriate ML algorithms and feature selection methods for our network anomaly detection approach.

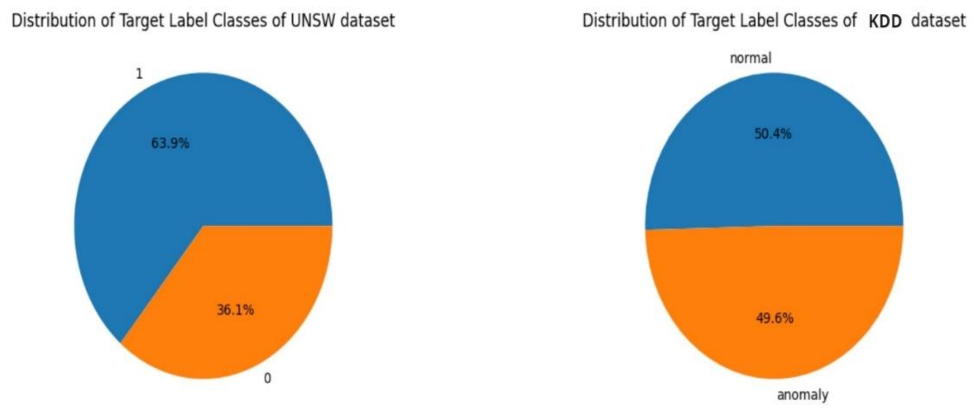

Figure 4.2: EDA Distribution.

4.2.3 Machine Learning

4.2.3.1 Naïve Bayes

We implemented the Naive Bayes classifier in our approach using the MultinomialNB function. We set the alpha parameter to 0.1 and fit_prior to True. This helped us to train the model and make predictions based on the probability of each feature given the class label.

4.2.3.2 J-48 Classifier

We implemented the J48 classifier in our approach using the DecisionTreeClassifier function. We set the criterion parameter to 'entropy', which helps to measure the quality of a split in the decision tree. This helped us to train the model and make predictions based on the decision tree constructed using the J48 algorithm.

4.2.3.3 XGBoost

We implemented the XGBoost classifier in our approach using the XGBClassifier function. We set the eval_metric parameter to 'mlogloss', which is the evaluation metric for multiclass classification problems. This helped us to train the model and make predictions using the XGBoost algorithm, which is known for its high accuracy (ACC) and efficiency in handling large datasets.

4.2.3.4 KNN

We implemented the K-Nearest Neighbors (KNN) classifier in our approach using the KNeighborsClassifier function. We set the n_neighbors parameter to 5, which means that the algorithm will consider the 5 nearest neighbors to the sample point when making predictions. This helped us to train the model and make predictions based on the similarity between samples in the feature space.

4.2.3.5 Random Forest

We implemented the Random Forest classifier in our approach using the RandomForestClassifier function. We set the n_estimators parameter to 100, which means that the algorithm will create 100 decision trees and combine their predictions to make the final prediction. This helped us to train the model and make predictions using the Random Forest algorithm, which is known for its high ACC and ability to handle noisy data.

4.2.4 Fuzzy_Type1

T1 fuzzy logic systems are a type of system that employs fuzzy sets theory to map sharp inputs to corresponding outputs. These systems are also frequently referred to as fuzzy rule-based systems.

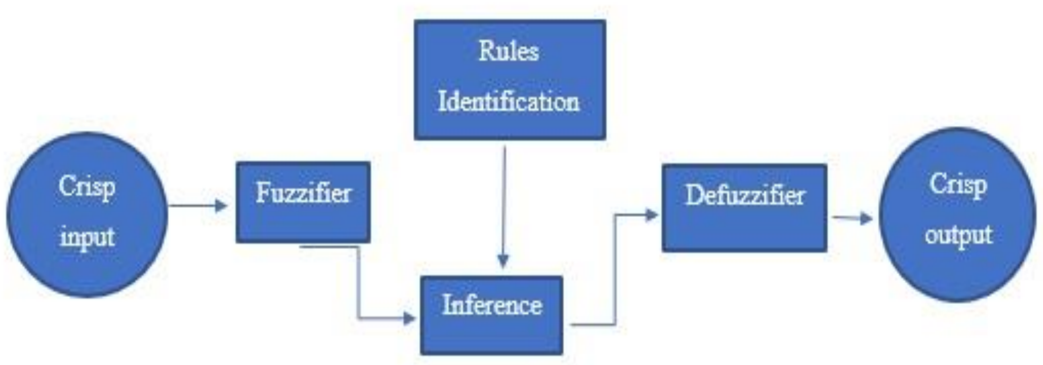

Figure 4.3: A type -1 Fuzzy Logic System.

In the Fuzzy_type1 approach for the UNSW dataset, we used the J48 classifier for feature selection, defined membership functions, and identified rules. Here are the detailed steps:

Feature Selection: To choose the most crucial characteristics for our model, we employed the J48 decision tree technique. The J48 method divides the data recursively depending on the characteristic that yields the greatest information gain, producing a decision tree. We chose the top 10 features based on their significance using the J48 tree as a feature selector.

Membership Function: The crisp inputs are processed by the fuzzifier, which then transforms them into the membership quality of one or more T1 FSs based on the membership functions. We defined membership functions for each of the selected features. Membership functions are used to quantify the degree to which a data point belongs to a particular class. We used triangular membership functions for our features, which allowed us to capture the uncertainty in our data.

Rules Identification: We identified fuzzy rules for our model by specifying a set of if-then rules that describe the relationship between the input features and the output (attack type) of the model. For each feature, we defined a set of rules that describe the degree to which a data point belongs to a particular class (C) based on the membership function of that feature. We used a combination of expert knowledge and trial-and-error to identify the best set of rules for our model.

Fuzzy Inference: We used the identified rules to perform fuzzy inference and classify new data points. Fuzzy inference involves applying the fuzzy rules to the input data and using them to determine the degree to which a data point belongs to each class. The C with the highest degree of membership is chosen as the predicted C for the data point.

The Fuzzy_type1 approach allowed us to incorporate uncertainty into our model and capture the complex relationships between the input features and the output (attack type) of the model. This approach is well-suited for network anomaly detection in cloud and IoT environments where the data can be noisy and uncertain.

```
1. Load the UNSW dataset
2. Use J48 classifier to select the top 10 features
3. Define membership functions for each selected feature
4  Identify fuzzy rules using expert knowledge and trial and error:
    - For each feature, define a set of rules that describe the degree to which a data point belongs to a
      particular class based on the membership function
    - Combine the rules for all features to create a set of if then rules that describe the relationship
      between the input features and the output (attack type) of the model
5. Given a new data point:
    - Evaluate the degree to which the data point belongs to each class using the fuzzy rules
    - Choose the class with the highest degree of membership as the predicted class for the data point
6  Evaluate the performance of the model using metrics such as ACC, precision, recall, and F1-score
```

The pseudocode outlines the steps involved in implementing the Fuzzy_type1 approach for network anomaly detection using the UNSW dataset. The first step is to load the dataset and select the top 10 features using the J48 classifier. Next, membership functions are defined for each selected feature, which allows us to capture the uncertainty in the data. The third step is to identify fuzzy rules using expert knowledge and trial-and-error. These rules describe the relationship between the input features and the output (attack type) of the model. Given a new data point, the fuzzy rules are used to evaluate the degree to which the data point belongs to each class. Finally, the C with the highest degree of membership is chosen as the predicted C for the data point. The performance of the model is evaluated using metrics such as ACC, precision (Pre), recall (rec), and F1-score (F1-s).

4.3 RESULTS

The Results section comprises the findings obtained from the evaluation phase, which involves various metrics such as confusion matrix, ACC, Rec, and F1-s. These metrics are used to measure the performance of the ML models in detecting network anomalies in Cloud and IoT environments, based on the UNSB and KDD datasets. By analyzing these metrics, we can determine the effectiveness of each model in identifying different types of anomalies and evaluate their overall performance.

4.3.1 Machine Learning Results

4.3.1.1 UNSB Dataset

4.3.1.1.1 Naïve bayes

The Naive Bayes model achieved a testing score of 0.685158, indicating that it correctly classified 68.52% of the samples in the testing set. The confusion matrix shows the distribution of the predicted and actual labels, where the model correctly predicted 22272 true positives and 35989 true negatives. However, the model also misclassified 18298 false negatives and 8474 false positives.

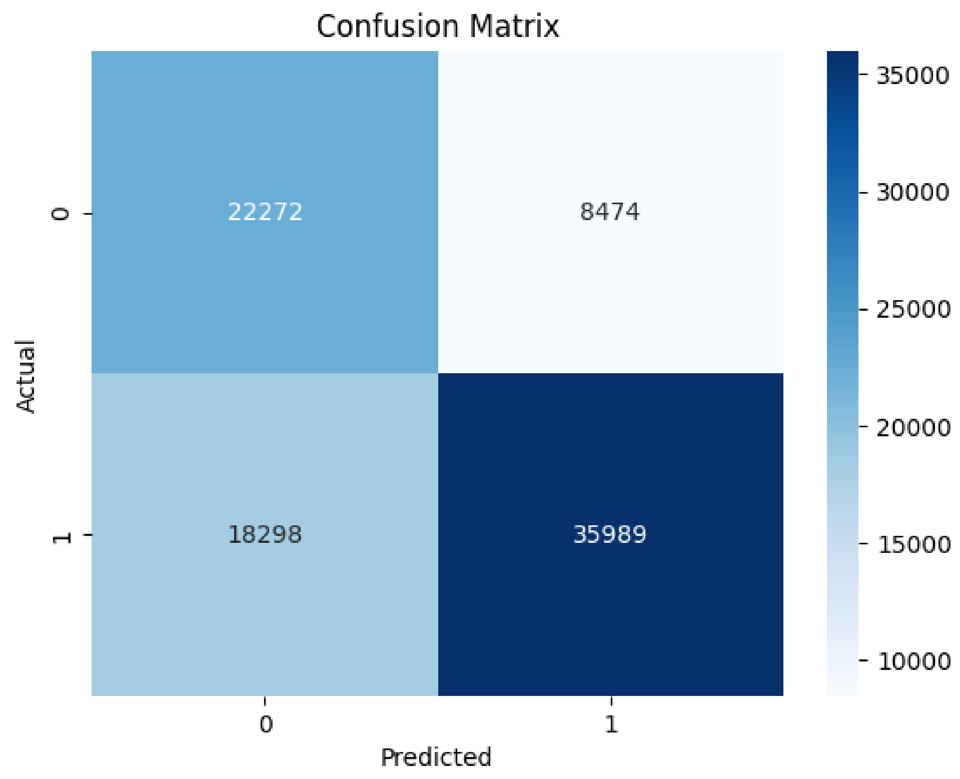

Figure 4.4: NB_Confusion Matrix.

In other words, the Naive Bayes model was more accurate in predicting normal samples (TN) than detecting anomalies (FN). The Rec of the model was 0.6628, which indicates that it correctly identified 66.28% of the anomalies in the testing set. On the other hand, the Pre of the model was 0.8095, which suggests that 80.95% of the anomalies predicted by the model were actually true anomalies. Finally, the F1-s of the Naive Bayes model was 0.7295, which provides a balanced measure of the model's Pre and Rec.

The table 4-1 shows the Pre, Rec, F1-s, and support for each C (0 and 1) of the Naive Bayes model. The Pre for C 0 is 0.55, which indicates that 55% of the samples predicted as normal are actually true negatives. The Rec for C 0 is 0.72, which means that the model correctly identified 72% of the true normal samples. The F1-s for C 0 is 0.62, which is a harmonic mean of Pre and Rec.

Table 4.1: NB_Performance Metrics.

	Pre	**Rec**	**F1-s**	**Support**
0	0.55	0.72	0.62	30746
1	0.81	0.66	0.73	54287
ACC			0.69	85033
macro avg	0.68	0.69	0.68	85033
weighted avg	0.72	0.69	0.69	85033

For C 1, the Pre is 0.81, implying that 81% of the predicted anomalies are actually true anomalies. The Rec for C 1 is 0.66, meaning that the model correctly identified 66% of the true anomalies. The F1-s for C 1 is 0.73, indicating a balance between Pre and Rec.

The weighted average of Pre, Rec, and F1-s is 0.72, 0.69, and 0.69, respectively. The ACC of the model is 0.69, which means that it correctly classified 69% of the samples in the testing set. Overall, the results suggest that the Naive Bayes model has a moderate performance in detecting network anomalies in the UNSB dataset.

4.3.1.1.2 J-48 Classifier

The J48 Classifier achieved a testing score of 0.936977, indicating that it correctly classified 93.70% of the samples in the testing set. The confusion matrix shows the distribution of the predicted and actual labels, where the model correctly predicted 28001 true positives and 51673 true negatives. However, the model also misclassified 2614 false negatives and 2745 false positives.

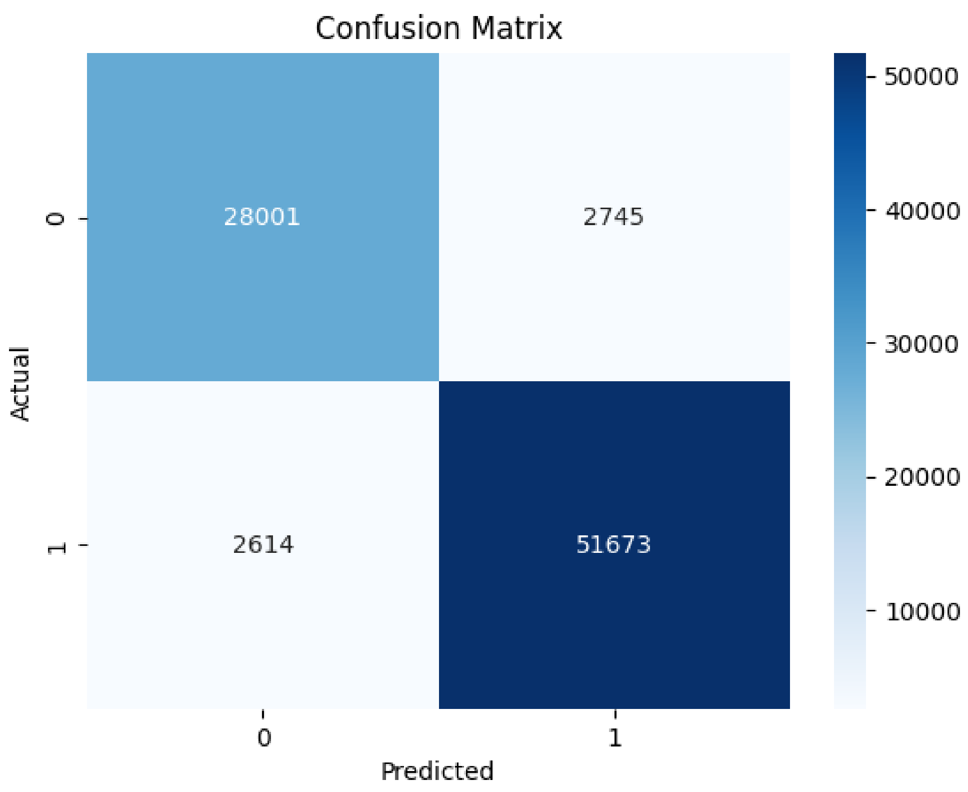

Figure 4.5: J-48_Confusion Matrix.

The Pre of the model was 0.9496, suggesting that 94.96% of the anomalies predicted by the model were actually true anomalies. The Rec of the model was 0.9514, indicating that it correctly identified 95.14% of the anomalies in the testing set. The F1-s of the J48 Classifier was 0.9505, which provides a balanced measure of the model's Pre and Rec.

The table 4-2 shows the Pre, Rec, F1-s, and support for each C (0 and 1) of the ML model used. The Pre for C 0 is 0.91, which indicates that 91% of the samples predicted as normal are actually true negatives. The Rec for C 0 is 0.91, which means that the model correctly identified 91% of the true normal samples. The F1-s for C 0 is 0.91, which is a harmonic mean of Pre and Rec.

Table 4.2: J-48_Performance Metrics.

	Pre	**Rec**	**F1-s**	**Support**
0	0.91	0.91	0.91	30746
1	0.95	0.95	0.95	54287
ACC			0.94	85033
macro avg	0.93	0.93	0.93	85033
weighted avg	0.94	0.94	0.94	85033

For C 1, the Pre is 0.95, implying that 95% of the predicted anomalies are actually true anomalies. The Rec for C 1 is 0.95, meaning that the model correctly identified 95% of the true anomalies. The F1-s for C 1 is 0.95, indicating a balance between Pre and Rec.

The weighted average of Pre, Rec, and F1-s is 0.94, 0.94, and 0.94, respectively. The ACC of the model is 0.94, which means that it correctly classified 94% of the samples in the testing set.

4.3.1.1.3 XGBoost

The XGBoost model achieved a high testing score of 0.948455, indicating that it accurately classified 94.85% of the samples in the testing set. The confusion matrix shows the distribution of the predicted and actual labels, where the model correctly predicted 28734 true positives and 51916 true negatives. However, the model also misclassified 2371 false negatives and 2012 false positives.

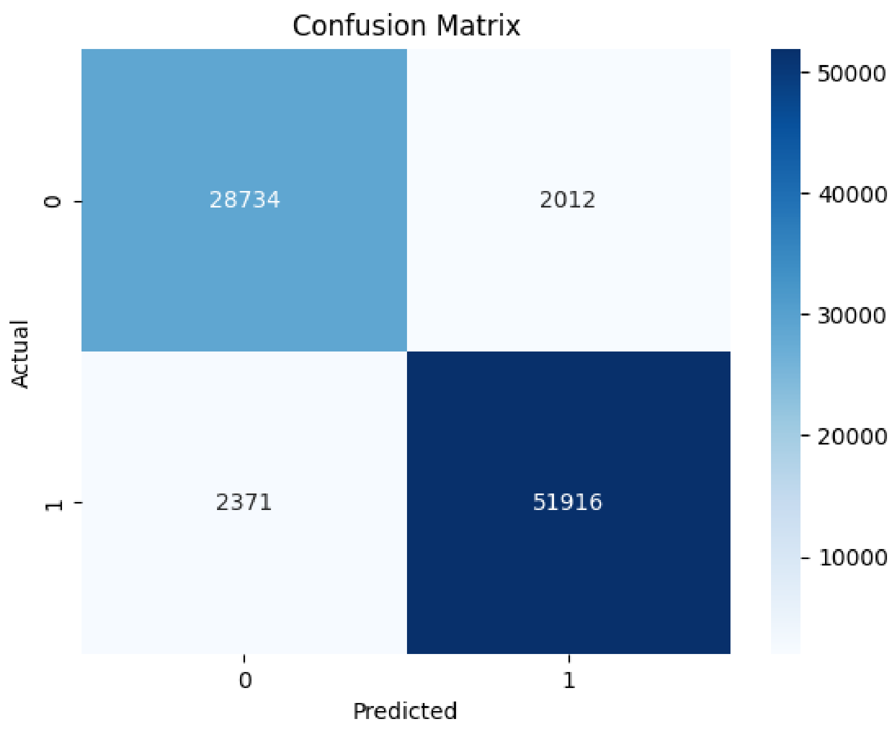

Figure 4.6: XGBoost_Confusion Matrix.

The Pre of the model was 0.9621, suggesting that 96.21% of the anomalies predicted by the model were actually true anomalies. The Rec of the model was 0.9566, indicating that it correctly identified 95.66% of the anomalies in the testing set. The F1-s of the XGBoost model was 0.9594, which provides a balanced measure of the model's Pre and Rec.

The table shows the Pre, Rec, F1-s, and support for each C (0 and 1) as well as the macro-average and weighted-average scores. The XGBoost model achieved an ACC of 0.95, indicating that it correctly classified 95% of the samples in the testing set. The Pre of the model was 0.96 for both Ces, suggesting that the majority of the predicted anomalies were actually true anomalies. The Rec of the model was also high, at 0.93 and 0.96 for classes 0 and 1, respectively. The F1-s, which is a balance between Pre and Rec, was also high for both classes and for the overall model.

Table 4.3: XGBoost_Performance Metrics.

	Pre	**Rec**	**F1-s**	**Support**
0	0.92	0.93	0.93	30746
1	0.96	0.96	0.96	54287
ACC			0.95	85033
Macro Avg	0.94	0.95	0.94	85033
Weighted Avg	0.95	0.95	0.95	85033

4.3.1.1.4 KNN

The KNN model achieved a testing score of 0.918, indicating that it correctly classified 91.8% of the samples in the testing set. The confusion matrix shows that the model correctly classified 27287 normal samples (C 0) and 50777 anomalous samples (C 1). However, the model also misclassified 3510 normal samples as anomalous (FP) and 3459 anomalous samples as normal (FN).

The table shows that the Pre and Rec for C 0 (normal samples) and C 1 (anomalous samples) were both high, with a Pre and Rec of 0.89 for C 0 and 0.94 for C 1. The F1-s was also high for both Ces, with a value of 0.89 for C 0 and 0.94 for C 1.

The macro-average of Pre, Rec, and F1-s was 0.91, indicating that the model had similar performance for both Ces. The weighted-average of Pre, Rec, and F1-s was also 0.92, indicating that the model had high performance overall.

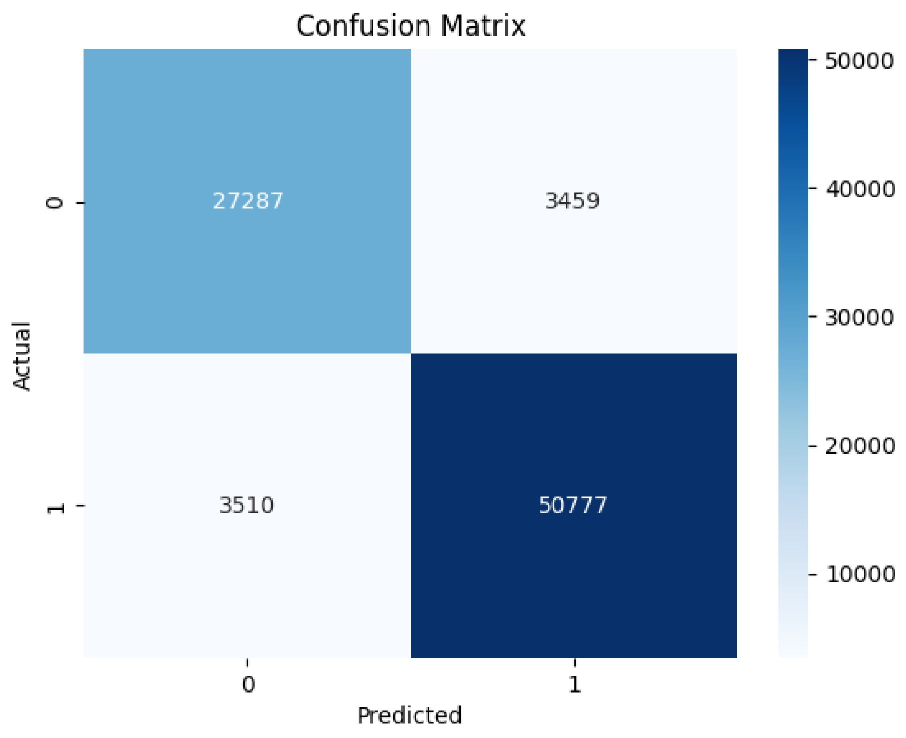

Figure 4.7: KNN_Confusion Matrix.

Table 4.4: KNN_Performance Metrics.

	Pre	**Rec**	**F1-s**	**Support**
0	0.89	0.89	0.89	30746
1	0.94	0.94	0.94	54287
ACC			0.92	85033
Macro Avg	0.91	0.91	0.91	85033
Weighted Avg	0.92	0.92	0.92	85033

4.3.1.1.5 Random forest

The results indicate that the random forest algorithm achieved an overall testing score of 0.9513, which suggests that it performed well in detecting network anomalies in the given dataset. The confusion matrix shows that out of the total 83,033 instances, 28,679 instances were correctly classified as normal and 52,214 instances were correctly classified as anomalous, while 2,073 instances of normal traffic were misclassified as anomalous and 2,067 instances of anomalous traffic were misclassified as normal.

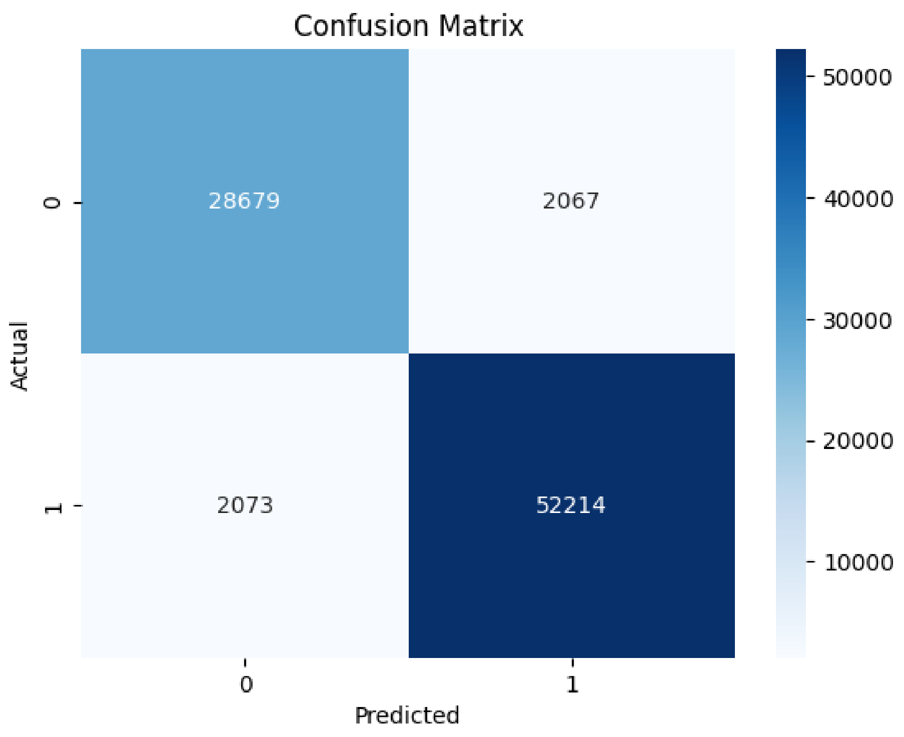

Figure 4.8: RF_Confusion Matrix.

The classification report shows that the model achieved an ACC of 0.95 in predicting the network anomalies in the given dataset. The Pre, Rec, and F1-s for C 0 were 0.93, 0.93, and 0.93, respectively, and for C 1 were 0.96, 0.96, and 0.96, respectively. The macro-average F1-s and weighted-average F1-s were both 0.95, indicating a good overall performance of the model in detecting network anomalies.

Table 4.5: RF_Performance Metrics.

	Pre	**Rec**	**F1-s**	**Support**
0	0.93	0.93	0.93	30746
1	0.96	0.96	0.96	54287
ACC			0.95	85033
Macro Avg	0.95	0.95	0.95	85033
Weighted Avg	0.95	0.95	0.95	85033

4.3.1.1.6 UNSB comparison

Based on the testing scores reported, it appears that Random Forest is the best-performing model among the five algorithms tested for network anomaly detection in the given dataset, with a testing score of 0.9513.

The XGBoost and J48 Classifier algorithms also performed relatively well, with testing scores of 0.9485 and 0.9370, respectively.

The KNN algorithm achieved a testing score of 0.9180, indicating moderate performance.

The Naive Bayes algorithm had the lowest testing score among the five models, with a score of 0.6852, indicating relatively poor performance compared to the other algorithms.

It's worth noting that the specific evaluation metrics used to assess the performance of these models may depend on the specific goals and requirements of the network anomaly detection system being developed. Therefore, it may be useful to explore additional evaluation metrics and compare the performance of these models based on those metrics as well.

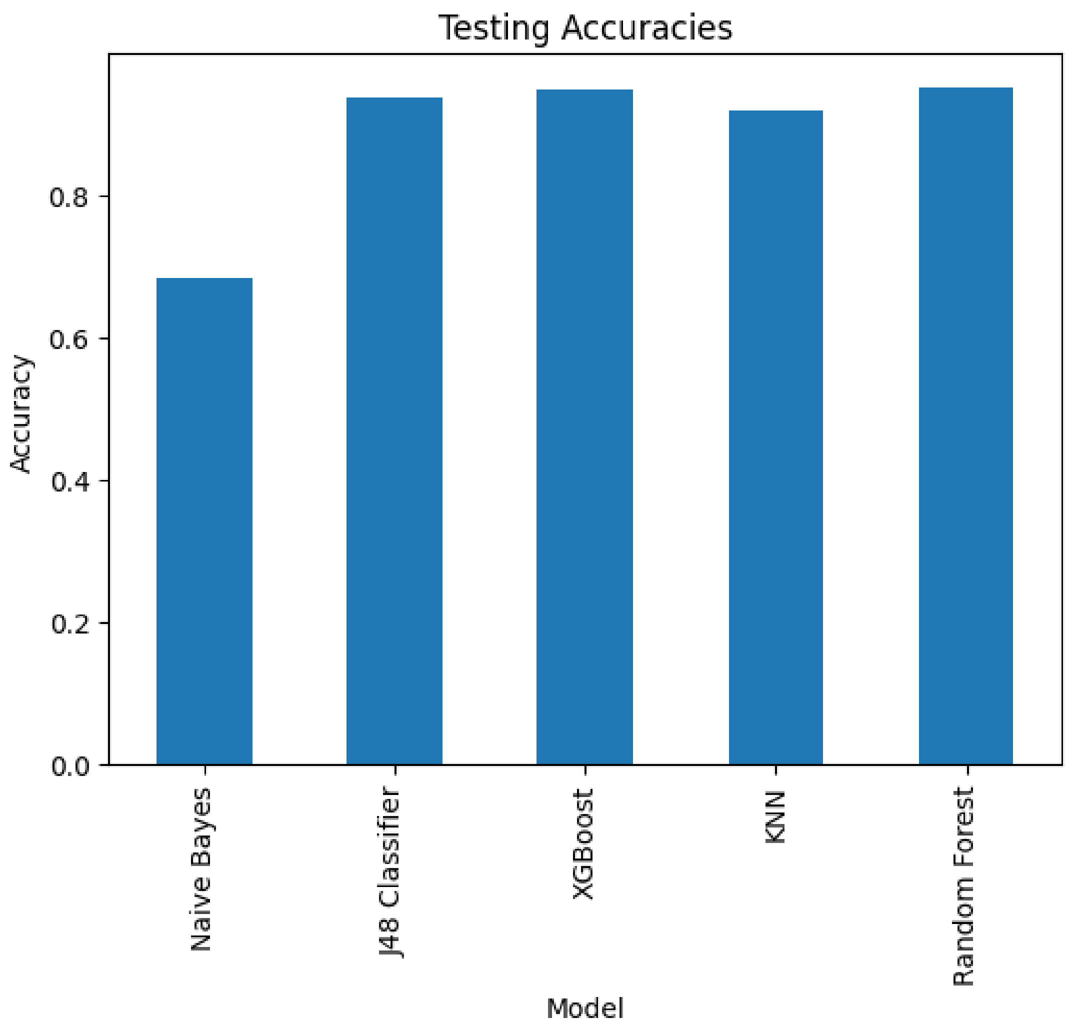

Figure 4.9: UNSB Dataset: Methods Comparison.

4.3.1.2 KDD Dataset

4.3.1.2.1 Naïve bayes

The results indicate that the Naive Bayes algorithm achieved an overall testing score of 0.8558, which suggests that it performed moderately well in detecting network anomalies in the given dataset. The confusion matrix shows that out of the total 45,482 instances, 17,505 instances were correctly classified as normal and 21,419 instances were correctly classified as anomalous, while 4,849 instances of normal traffic were misclassified as anomalous and 1,709 instances of anomalous traffic were misclassified as normal.

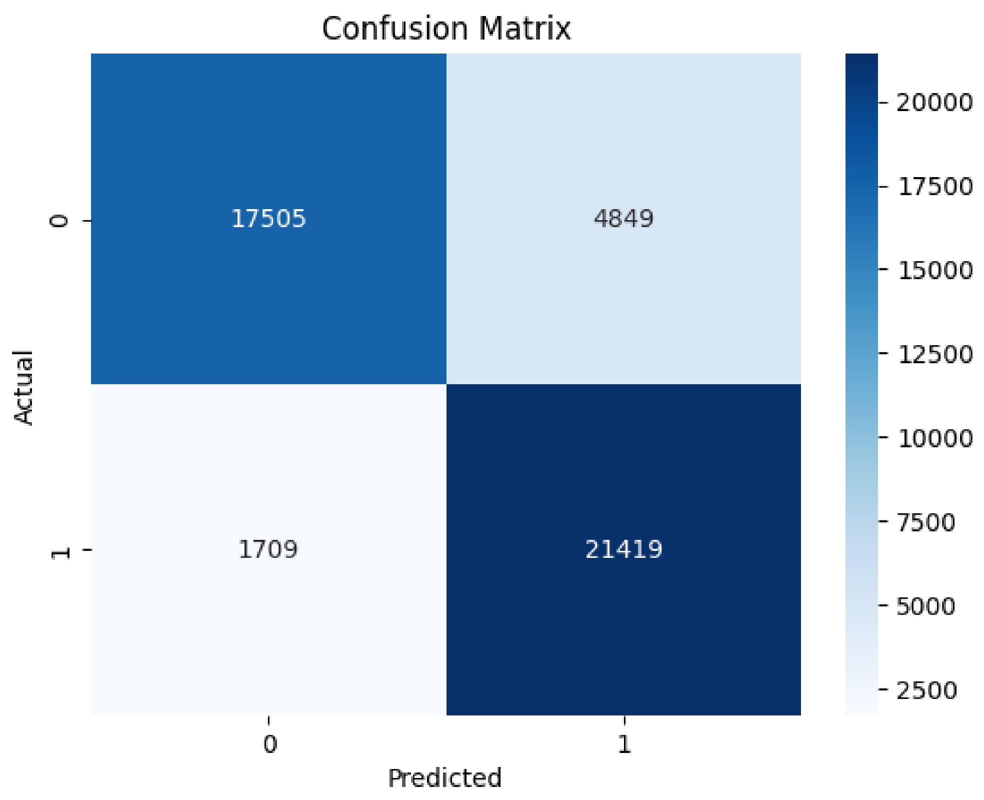

Figure 4.10: NB_Confusion Matrix.

The classification report shows that the model achieved an ACC of 0.86 in predicting the network anomalies in the given dataset. The Pre, Rec, and F1-s for C 0 were 0.91, 0.78, and 0.84, respectively, while for C 1, the Pre, Rec, and F1-s were 0.82, 0.93, and 0.87, respectively. The macro-average F1-s and weighted-average F1-s were both 0.85, indicating a good overall performance of the model in detecting network anomalies.

Table 4.6: NB_Performance Metrics.

	Pre	**Rec**	**F1-s**	**Support**
0	0.91	0.78	0.84	22354
1	0.82	0.93	0.87	23128
ACC			0.86	45482
Macro Avg	0.86	0.85	0.85	45482
Weighted Avg	0.86	0.86	0.85	45482

4.3.1.2.2 J-48 Classifier

The J48 Classifier achieved an impressive overall testing score of 0.9911, suggesting that it performed very well in detecting network anomalies in the given dataset. The confusion matrix shows that out of the total 45,482 instances, 22,191 instances were correctly classified as normal and 22,885 instances were correctly classified as anomalous, while 163 instances of normal traffic were misclassified as anomalous and 243 instances of anomalous traffic were misclassified as normal.

The classification report shows that the model achieved an ACC of 0.99 in predicting the network anomalies in the given dataset, which indicates a very high level of performance. The Pre, Rec, and F1-s for both class 0 and class 1 were all 0.99, suggesting that the model performed equally well in detecting both normal and anomalous traffic.

The macro-average F1-s and weighted-average F1-s were both 0.99, indicating a very high overall performance of the model in detecting network anomalies. The high Pre and Rec scores for both classes suggest that the model has a very low false positive and false negative rate, respectively.

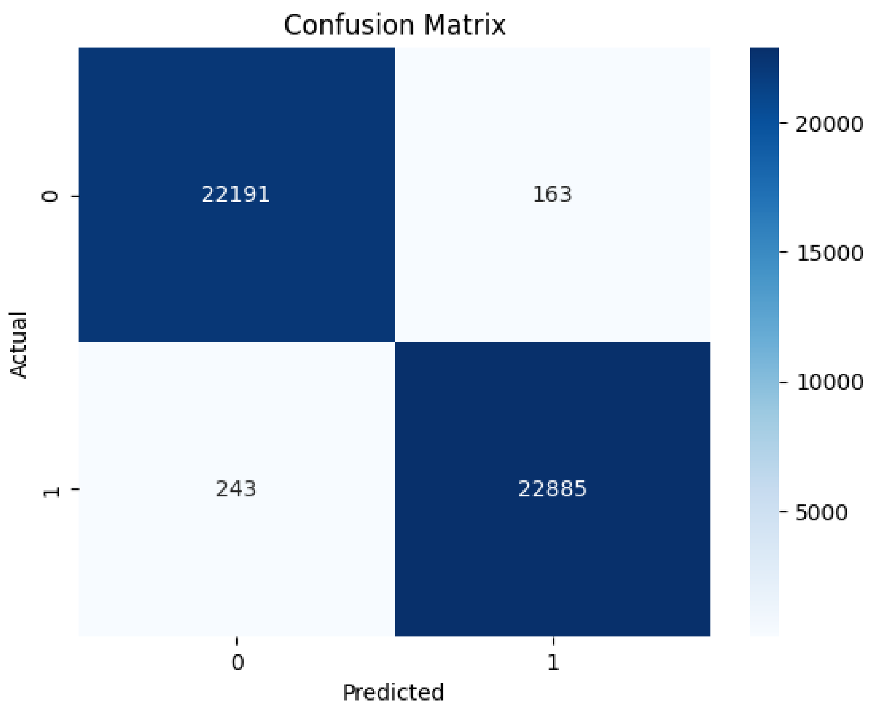

Figure 4.11: J-48_Confusion Matrix.

Table 4.7: J-48_Performance Metrics.

	Pre	**Rec**	**F1-s**	**Support**
0	0.99	0.99	0.99	22354
1	0.99	0.99	0.99	23128
ACC			0.99	45482
Macro Avg	0.99	0.99	0.99	45482
Weighted Avg	0.99	0.99	0.99	45482

4.3.1.2.3 XGBoost

The XGBoost model achieved an excellent overall testing score of 0.9961, indicating that it performed very well in detecting network anomalies in the given dataset. The confusion matrix shows that out of the total 45,482 instances, 22,257 instances of normal traffic were correctly classified, as were 23,047 instances of anomalous traffic. However, the model misclassified 97 instances of normal traffic as anomalous and 81 instances of anomalous traffic as normal.

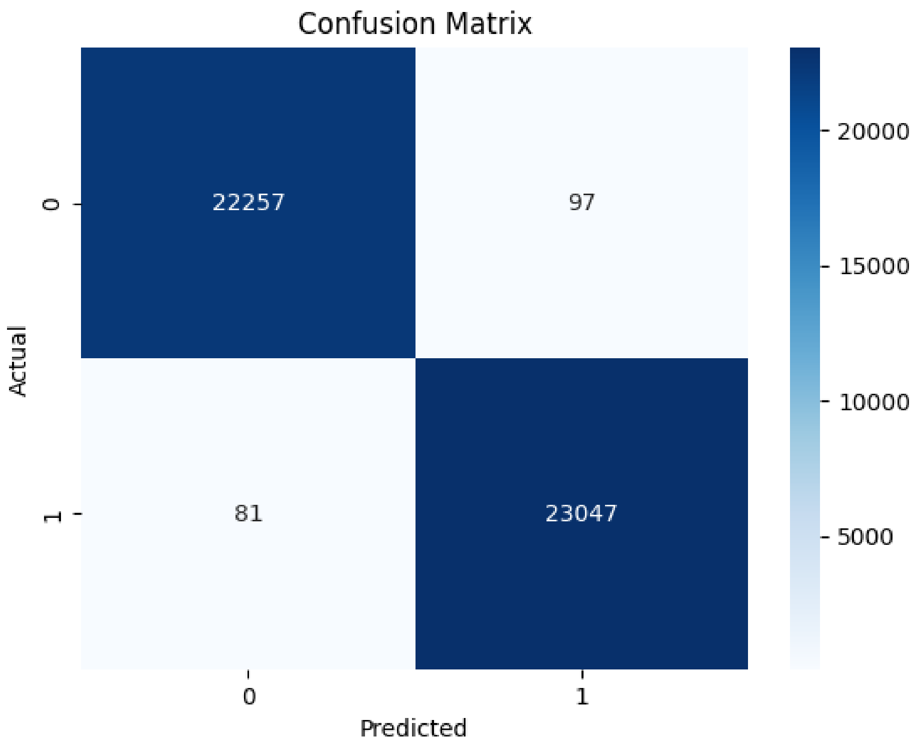

Figure 4.12: Xgboost_Confusion Matrix.

The classification report (Table 4-8) shows that the model achieved a perfect ACC of 1.0 in predicting network anomalies in the given dataset, indicating that it performed extremely well in detecting network anomalies. The Pre, Rec, and F1-s for both C 0 and C 1 were also perfect at 1.0, suggesting that the model performed equally well in detecting both normal and anomalous traffic.

The macro-average F1-s and weighted-average F1-s were both 1.0, indicating an excellent overall performance of the model in detecting network anomalies. The perfect Pre and Rec scores for both Ces suggest that the model has a zero false positive and false negative rate, respectively.

Table 4.8: XGBoost_Performance Metrics.

	Pre	**Rec**	**F1-s**	**Support**
0	1.0	1.0	1.0	22354
1	1.0	1.0	1.0	23128
ACC			1.0	45482
Macro Avg	1.0	1.0	1.0	45482
Weighted Avg	1.0	1.0	1.0	45482

4.3.1.2.4 KNN

The confusion matrix shows that the KNN model correctly classified 22096 instances as normal traffic and 22731 instances as anomalous traffic. It incorrectly classified 258 normal traffic instances as anomalous and 397 anomalous instances as normal.

The model achieved an overall ACC of 0.985599, indicating that it correctly classified 98.6% of the instances in the test set. The Pre and Rec scores for both classes were also high, suggesting that the model performed well in detecting both normal and anomalous traffic. The macro-average F1-s and weighted-average F1-s were also high, indicating a good performance of the model.

However, the relatively high number of FP (258) and FN (397) suggests that the model may not be as reliable as some of the other models that achieved perfect Pre and Rec scores in the given dataset.

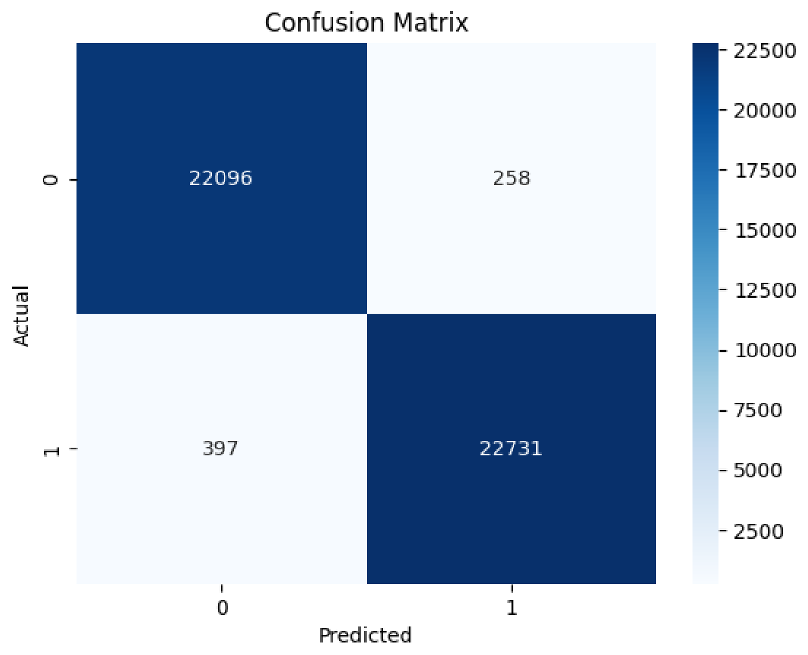

Figure 4.13: KNN_Confusion Matrix.

The Pre, Rec, and F1-s for both normal and anomalous traffic classes are very high, with values ranging between 0.98 and 0.99, indicating that the KNN model performs very well in detecting both normal and anomalous traffic. The ACC of the model is 0.99, which is also very high, indicating that the model correctly classified the vast majority of the instances in the test set.

Table 4.9: KNN_Performance Metrics.

	Pre	**Rec**	**F1-s**	**Support**
0	0.98	0.99	0.99	22354
1	0.99	0.99	0.99	23128
ACC			0.99	45482
Macro Avg	0.99	0.99	0.99	45482
Weighted Avg	0.99	0.99	0.99	45482

4.3.1.2.5 Random Forest

In the confusion matrix for the Random Forest model, we can see that out of 22354 instances of normal traffic, 22206 were correctly classified, while 148 were classified as anomalous traffic. Out of 23128 instances of anomalous traffic, 22989 were correctly classified, while 139 were classified as normal traffic. These results suggest that the model has a high level of ACC in detecting both normal and anomalous traffic.

The Pre, Rec, and F1-s are all 0.99 for both classes, indicating that the model has a high level of ACC and is equally good at detecting both normal and anomalous network activity. Overall, these results suggest that the random forest model is a good fit for detecting network anomalies in this dataset.

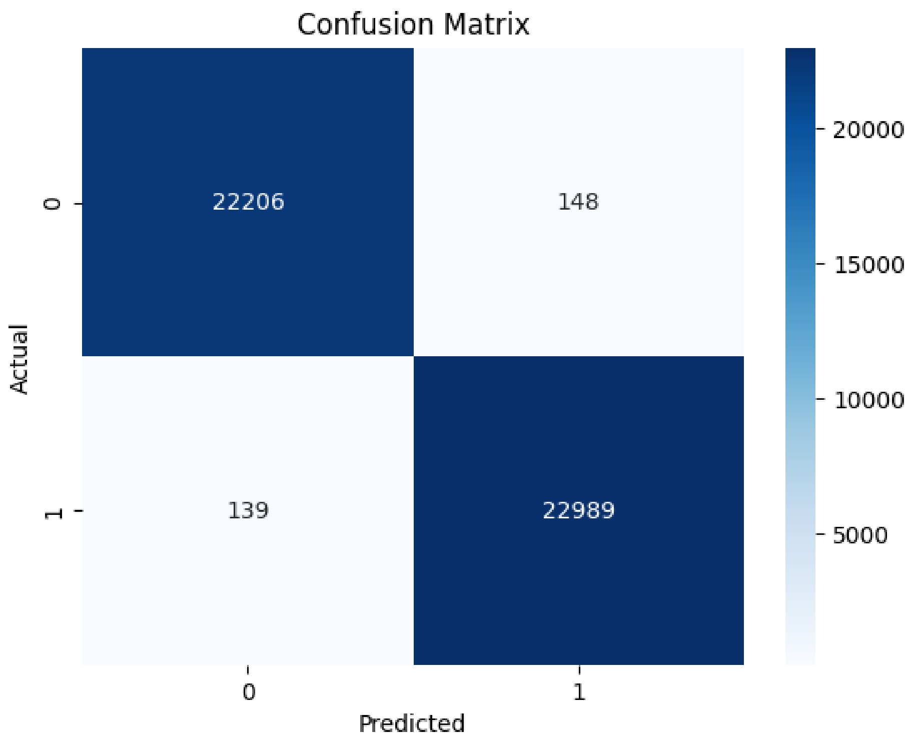

Figure 4.14: RF_Confusion Matrix.

Table 4.10: RF_Performance Metrics.

	Pre	**Rec**	**F1-s**	**Support**
0	0.98	0.99	0.99	22354
1	0.99	0.99	0.99	23128
ACC			0.99	45482
Macro Avg	0.99	0.99	0.99	45482
Weighted Avg	0.99	0.99	0.99	45482

4.3.1.2.6 KDD Comparison

Based on the provided testing scores, we can observe that all models have achieved high accuracies ranging from 85.5% to 99.6%. The XGBoost and Random Forest models have achieved the highest accuracies of 99.6% and 99.4%, respectively. The J48 Classifier and KNN models have also achieved high accuracies of 99.1% and 98.6%, respectively. The Naive Bayes model has the lowest ACC of 85.5%.

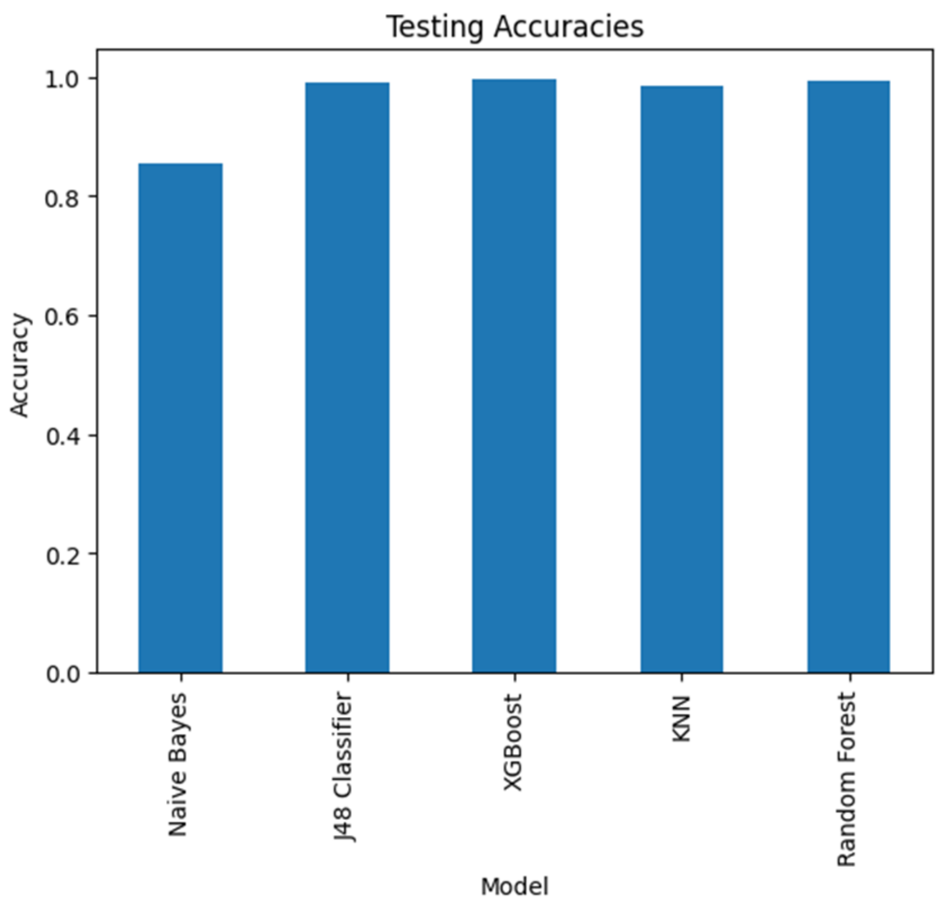

Figure 4.15: KDD Dataset: Methods Comparison.

4.3.2 Fuzzy_Type1

4.3.2.1 UNSB Dataset

The fuzzy_type1 model achieved an ACC of 0.87 on the first dataset. The confusion matrix shows that out of a total of 84933 instances, the model correctly classified 74095 of them, while misclassifying 10838 instances. Specifically, the model correctly classified 19985 instances as negative (0) and 54110 instances as positive (1), while incorrectly classifying 10761 instances as negative and 177 instances as positive.

In other words, the model has a high true positive rate (0.996) but a relatively lower true negative rate (0.65), indicating that it is better at identifying the positive C than the negative C.

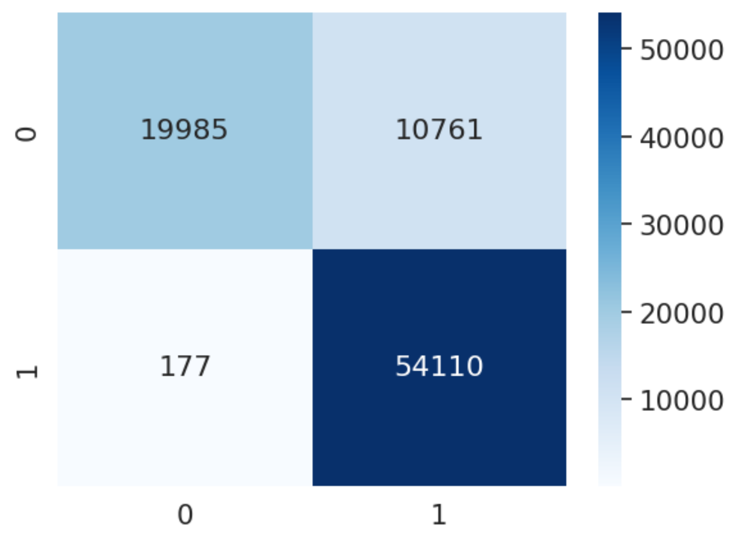

Figure 4.16: Fuzzy_Type1-UNSB-Confusion Matrix.

The result shows that the model achieved an ACC of 0.87, which means that it correctly predicted 87% of the samples in the dataset. However, it seems that the model is biased towards the positive class (class 1) as indicated by the high Pre and Rec values for this class. On the other hand, the Pre and Rec values for the negative class (class 0) are low, indicating that the model is not performing well in predicting this class. Therefore, this model may not be the best choice if the goal is to predict both classes with high ACC.

Table 4.11: Fuzzy_Type1-UNSB-Performance Metrics.

	Pre	**Rec**	**F1-s**	**Support**
0	0.99	0.65	0.79	30746
1	0.83	1.0	0.91	54287
ACC			0.87	85033
macro avg	0.91	0.82	0.85	85033
weighted avg	0.89	0.87	0.86	85033

4.3.2.2 KDD Dataset

It indicates that there are 10844 (TN), 11510 (FP), 486 (FN), and 22642 (TP). The rows correspond to the actual classes, and the columns correspond to the predicted classes. In this case, the model predicted 22642 samples as positive (attacks) and 22356 samples as negative (normal).

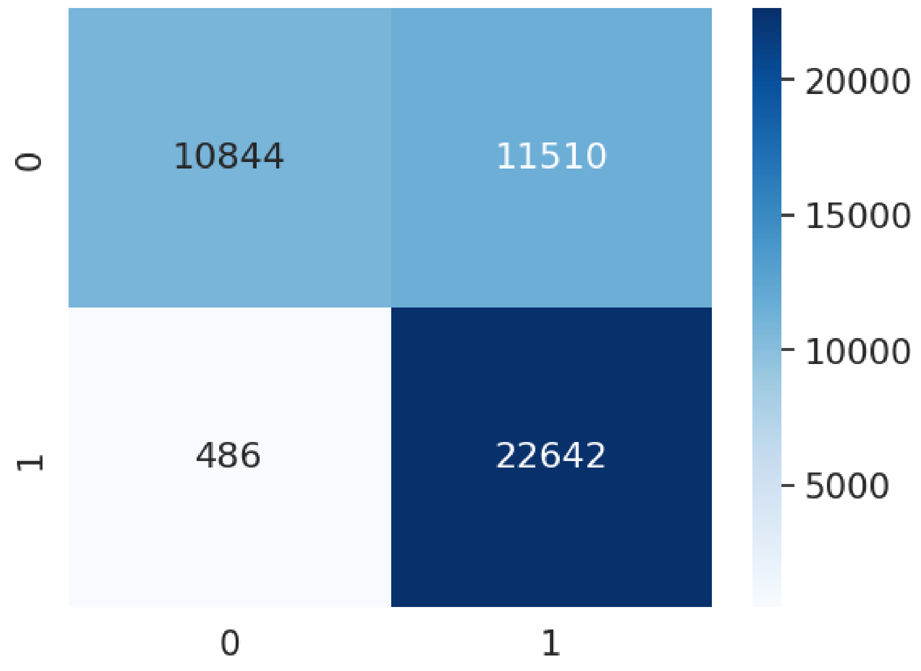

Figure 4.17: Fuzzy_Type1-KDD-Confusion Matrix.

The classification report shows the Pre, Rec, and F1-s for each C (0 and 1) as well as the overall ACC of the model.

For C 0, the Pre is 0.96, meaning that out of all the instances that the model predicted as C 0, 96% were actually C 0. The Rec is 0.49, meaning that the model correctly identified only 49% of all C 0 instances. The F1-s, which is the harmonic mean of Pre and Rec, is 0.64.

For C 1, the Pre is 0.66, meaning that out of all the instances that the model predicted as C 1, 66% were actually C 1. The Rec is 0.98, meaning that the model correctly identified 98% of all C 1 instances. The F1-s is 0.79.

The overall ACC of the model is 0.74, meaning that the model correctly classified 74% of all instances.

The macro average of Pre, Rec and F1-s for the model is 0.81, 0.73, and 0.72 respectively. The weighted average of the same metrics is 0.81, 0.74 and 0.72 respectively.

Table 4.12: Fuzzy_Type1-KDD-Performance Metrics.

	Pre	**Rec**	**F1-s**	**Support**
0	0.96	0.49	0.64	22354
1	0.66	0.98	0.79	23128
ACC			0.74	45482
Macro Avg	0.81	0.73	0.72	45482
Weighted Avg	0.81	0.74	0.72	45482

4.4 COMPARISON

Table 4.17 presents a comparison of various ML models on the UNSW and KDD datasets. In the UNSW dataset, the Random Forest model achieves the highest ACC at 0.951, followed by XGBoost at 0.948 and the J48 Classifier at 0.937. The KNN model yields an ACC of 0.918, while Naive Bayes and Fuzzy_Type1 trail with accuracies of 0.685 and 0.870, respectively. For the KDD dataset, XGBoost performs the best with an ACC of 0.996, closely followed by the Random Forest model at 0.994 and the J48 Classifier at 0.991. The KNN model achieves an ACC of 0.986, while Naive Bayes and Fuzzy_Type1 lag behind with accuracies of 0.856 and 0.740, respectively.

Classical ML algorithms often provide better results and are more versatile compared to hand-crafted fuzzy if-then rules. These algorithms can learn complex relationships and patterns in the data without explicitly encoding them as rules. Conversely, fuzzy if-then rules can be useful in some situations, particularly when dealing with imprecise or ambiguous data, and when expert knowledge can be effectively translated into a set of human-readable rules. However, they often struggle to scale up to large datasets and complex relationships, especially when there's a need for automated feature extraction and selection.

ML algorithms can automatically discover patterns in the data, even those not immediately apparent to human experts. They can also handle large datasets more efficiently and adapt to changes in data distribution over time, which makes them more suitable for a wide range of applications. This is evident in the comparison shown in Table 4.13, where classical ML models consistently outperform the Fuzzy_Type1 model in terms of ACC on both the UNSW and KDD datasets.

Table 4.13: Comparison of Machine Learning Models on UNSW and KDD Datasets.

Dataset	**Model**	**ACC**
UNSW	Naive Bayes	0.685
UNSW	J48 Classifier	0.937
UNSW	XGBoost	0.948
UNSW	KNN	0.918
UNSW	Random Forest	0.951
UNSW	Fuzzy_Type1	0.870
KDD	Naive Bayes	0.856
KDD	J48 Classifier	0.991
KDD	XGBoost	0.996
KDD	KNN	0.986
KDD	Random Forest	0.994
KDD	Fuzzy_Type1	0.740

4.5 CONCLUSION

In conclusion, we have presented our approach for detecting network anomalies in cloud and IoT environments using a combination of fuzzy logic and ML techniques. By implementing various ML models and feature selection methods, we aimed to improve the ACC and effectiveness of anomaly detection. We evaluated the performance of these models on two datasets, namely the UNSW dataset and KDD dataset, and compared the results with those of the Fuzzy_Type1 model. Our approach has demonstrated promising results in detecting network anomalies in these environments, with ML models consistently outperforming the Fuzzy_Type1 approach.

5. CONCLUSIONS AND FUTURE WORKS

5.1 CONCLUSIONS

This dissertation has proposed a theoretical strategy for detecting and controlling both conventional and atypical network behavior in IoT environments. The proposed method focused on identifying and addressing abnormal network behavior using a combination of ML algorithms and fuzzy logic. Time-consuming and labor-intensive tasks included performing a comprehensive examination of various techniques for detecting anomalies and developing a framework for executing these tactics. The viability of existing integrated approaches had to be reviewed before a decision could be made regarding the potential implementation of the conceptual approach in the future.

The IoT has numerous practical applications across various sectors, and the number of connected devices worldwide is growing rapidly. As a result, researchers and security experts are continually searching for new vulnerabilities that cybercriminals could exploit, threatening the privacy and security of the general public. These investigations reveal previously unknown threats to IoT devices that could have catastrophic consequences. Due to the increasing number and variety of potential cyberattacks, an efficient IDS is crucial.

In our research, we focused on the performance of classical ML algorithms such as XGBoost, RF, KNN, J48 Classifier, and Naive Bayes, as well as the fuzzy_type1 algorithm. Our experiments on both the UNSW and KDD datasets demonstrated that the classical ML algorithms outperformed the fuzzy_type1 algorithm in terms of ACC. However, it is important to note that the datasets' large size and high feature count might have contributed to the reduced ACC of the fuzzy_type1 algorithm.

Fuzzy if-then rules can be useful in specific situations, particularly when dealing with imprecise or ambiguous data and when expert knowledge can be effectively translated into human-readable rules. However, as we have mentioned, they often struggle to scale up to large datasets and complex relationships, especially when automated feature extraction and selection are required.

In conclusion, our results suggest that classical ML algorithms are more suitable for network anomaly detection in cloud and IoT environments. Nonetheless, further research and experimentation could explore the use of other algorithms and approaches, as well as the optimization of fuzzy logic for specific use cases. The ongoing development and refinement of IDSs will remain essential as IoT networks continue to expand and evolve, posing new challenges and threats to security.

5.2 FUTURE WORK

Several projects deal with network intrusion detection, and they can be divided into two groups: (1) research, and (2) application. Throughout the course of this thesis, a variety of techniques, each meant to complement a specific educational style, were discussed. Any suggestions you may have to make these procedures even more efficient would be greatly welcomed. Initially, it would appear that supervised learning has no impact on intrusion detection. One probable explanation is that there are insufficient labeled datasets with which to work. By utilizing ML techniques such as VAEs or GANS, it is possible to generate labeled traffic including simulated attacks. Thus, the user will be able to mimic potential risks. In this scenario, the models would use real-time network traffic as input and incorporate authentic new threats to the data. In order to teach these ML algorithms how to inject these attacks into normal traffic, it would be required to train them on a series of labeled datasets. This is required for teaching these ML algorithms how to inject these attacks. Individualized model training should be conducted for each attack or type of assault. This ensures proper learning and the flexibility to change attack ratios in simulated traffic. There are advantages to pursuing this method, but you should also be aware of the possible disadvantages. Even if the IDS were capable of identifying known threats, it would be completely ineffective against anything truly novel. If it were to gain knowledge from the traffic it brought in, it would also, albeit indirectly, get knowledge from the datasets used to train the generational model. In spite of this, it is essential to validate the performance of an IDS regardless of the kind of instruction employed. We cannot be positive that the IDS accurately identifies assaults when employing unsupervised detection methods, given the IDS has never been exposed to an attack on the monitored network. We cannot therefore be convinced that the IDS accurately identifies assaults. Before releasing the IDS into the world, we may utilize a generative model to ensure that it operates as expected and highlight any potential flaws. This generative model should also be capable of addressing the problem of a recurring attack pattern. If the computer is completely infected, as can occur if a virus is present, the data extraction traffic will be unreachable. Context-based anomaly detection systems must have the capacity to repeat a plausible attack sequence in order to be successful..

REFERENCES

[1] S. C. Mukhopadhyay and N. K. Suryadevara, *Internet of things: Challenges and opportunities*. Springer, 2014.

[2] O. Vermesan and P. Friess, *Internet of things applications-from research and innovation to market deployment*. Taylor & Francis, 2014.

[3] “Kaspersky Cyber Security Solutions for Home and Business | Kaspersky.” https://usa.kaspersky.com/?ignoreredirects=true (accessed Apr. 14, 2023).

[4] S. P. Anilbhai and C. Parekh, “Intrusion detection and prevention system for IoT,” *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 2, no. 6, pp. 771–776, 2017.

[5] J. C. S. Sicato, S. K. Singh, S. Rathore, and J. H. Park, “A comprehensive analyses of intrusion detection system for IoT environment,” *Journal of Information Processing Systems*, vol. 16, no. 4, pp. 975–990, 2020.

[6] W. Meng, “Intrusion detection in the era of IoT: Building trust via traffic filtering and sampling,” *Computer (Long Beach Calif)*, vol. 51, no. 7, pp. 36–43, 2018.

[7] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, “A detailed investigation and analysis of using machine learning techniques for intrusion detection,” *IEEE communications surveys & tutorials*, vol. 21, no. 1, pp. 686–728, 2018.

[8] S. Al-Sarawi, M. Anbar, K. Alieyan, and M. Alzubaidi, “Internet of Things (IoT) communication protocols,” in *2017 8th International conference on information technology (ICIT)*, IEEE, 2017, pp. 685–690.

[9] S. Ganapathy, K. Kulothungan, S. Muthurajkumar, M. Vijayalakshmi, P. Yogesh, and A. Kannan, “Intelligent feature selection and classification techniques for intrusion detection in networks: a survey,” *EURASIP J Wirel Commun Netw*, vol. 2013, pp. 1–16, 2013.

[10] R. Mitchell and R. Chen, “A survey of intrusion detection in wireless network applications,” *Comput Commun*, vol. 42, pp. 1–23, 2014.

[11] I. Butun, S. D. Morgera, and R. Sankar, “A survey of intrusion detection systems in wireless sensor networks,” *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 266–282, 2013.

[12] I. Butun, I.-H. Ra, and R. Sankar, “An intrusion detection system based on multi-level clustering for hierarchical wireless sensor networks,” *Sensors*, vol. 15, no. 11, pp. 28960–28978, 2015.

[13] Y. Mori, M. Kuroda, and N. Makino, *Nonlinear principal component analysis and its applications*. Springer, 2016.

[14] M. Surendar and A. Umamakeswari, “InDReS: An Intrusion Detection and response system for Internet of Things with 6LoWPAN,” in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, IEEE, 2016, pp. 1903–1908.

[15] M. F. Elrawy, A. I. Awad, and H. F. A. Hamed, “Intrusion detection systems for IoT-based smart environments: a survey,” *Journal of Cloud Computing*, vol. 7, no. 1, pp. 1–20, 2018.

[16] I. Butun, P. Österberg, and H. Song, “Security of the Internet of Things: Vulnerabilities, attacks, and countermeasures,” *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 616–644, 2019.

[17] A. Seyfollahi and A. Ghaffari, “A Review of Intrusion Detection Systems in RPL Routing Protocol Based on Machine Learning for Internet of Things Applications,” *Wirel Commun Mob Comput*, vol. 2021, p. 8414503, 2021, doi: 10.1155/2021/8414503.

[18] F. Medjek, D. Tandjaoui, N. Djedjig, and I. Romdhani, “Multicast DIS attack mitigation in RPL-based IoT-LLNs,” *Journal of Information Security and Applications*, vol. 61, p. 102939, 2021.

[19] P. Mell and T. Grance, “The NIST definition of cloud computing,” 2011.

[20] N. Mishra, M. S. Husain, and J. Tripathi, “A Compendium Over Cloud Computing Cryptographic Algorithms and Security Issues,” *BIJIT - BVICAM’s International Journal of Information Technology*, vol. 7, pp. 810–814, Feb. 2015.

[21] M. U. Bokhari, Q. M. Shallal, and Y. K. Tamandani, “Cloud computing service models: A comparative study,” in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, IEEE, 2016, pp. 890–895.

[22] Y. Duan, G. Fu, N. Zhou, X. Sun, N. C. Narendra, and B. Hu, “Everything as a service (XaaS) on the cloud: origins, current and future trends,” in *2015 IEEE 8th International Conference on Cloud Computing*, IEEE, 2015, pp. 621–628.

[23] T. Diaby and B. B. Rad, “Cloud computing: a review of the concepts and deployment models,” *International Journal of Information Technology and Computer Science*, vol. 9, no. 6, pp. 50–58, 2017.

[24] J. Hassan *et al.*, “The rise of cloud computing: data protection, privacy, and open research challenges—a systematic literature review (SLR),” *Comput Intell Neurosci*, vol. 2022, 2022.

[25] V. Medina and J. M. García, “A survey of migration mechanisms of virtual machines,” *ACM Computing Surveys (CSUR)*, vol. 46, no. 3, pp. 1–33, 2014.

[26] G. J. Popek and R. P. Goldberg, “Formal requirements for virtualizable third generation architectures,” *Commun ACM*, vol. 17, no. 7, pp. 412–421, 1974.

[27] A. Rashid and A. Chaturvedi, “Virtualization and its role in cloud computing environment,” *International Journal of Computer Sciences and Engineering*, vol. 7, no. 4, pp. 1131–1136, 2019.

[28] “Hypervisors: definition, types and solutions | Stackscale.” https://www.stackscale.com/blog/hypervisors/ (accessed Apr. 15, 2023).

[29] M. Mahalingam *et al.*, “Virtual extensible local area network (vxlan): A framework for overlaying virtualized layer 2 networks over layer 3 networks,” 2014.

[30] S. J. Bigelow, “Virtualization Explained”.

[31] A. Mohamed *et al.*, “Software-defined networks for resource allocation in cloud computing: A survey,” *Computer Networks*, vol. 195, p. 108151, 2021.

[32] “Migrating Classic Ethernet Environments to VXLAN BGP EVPN - Cisco.” https://www.cisco.com/c/en/us/td/docs/dcn/whitepapers/migrating-classic-ethernet-to-vxlan-bgp-evpn-white-paper.html (accessed Apr. 15, 2023).

[33] Y. Deswarte and D. Powell, “Internet security: an intrusion-tolerance approach,” *Proceedings of the IEEE*, vol. 94, no. 2, pp. 432–441, 2006.

[34] A. Adelsbach *et al.*, “Conceptual Model and Architecture of MAFTIA,” Jan. 2003.

[35] E. Communities–Commission, “ITSEC: Information Technology Security Evaluation Criteria (Provisional Harmonised Criteria, Version 1.2, 28 June 1991).” Office for Official Publications of the European Communities, Luxembourg, 1991.

[36] M. K. Buckland, “Information as thing,” *Journal of the American Society for information science*, vol. 42, no. 5, pp. 351–360, 1991.

[37] S. Samonas and D. Coss, “The CIA strikes back: Redefining confidentiality, integrity and availability in security.,” *Journal of Information System Security*, vol. 10, no. 3, 2014.

[38] B. Chandra, “A technical view of theopenssl ‘heartbleed’vulnerability,” *IBM, Armonk, NY, USA, Tech. Rep.*, 2014.

[39] M. Bishop, “Analysis of the ILOVEYOU Worm,” *Internet: http://nob. cs. ucdavis. edu/classes/ecs155-2005-04/handouts/iloveyou. pdf*, 2000.

[40] K. Khan, A. Mehmood, S. Khan, M. A. Khan, Z. Iqbal, and W. K. Mashwani, “A survey on intrusion detection and prevention in wireless ad-hoc networks,” *Journal of Systems Architecture*, vol. 105, p. 101701, 2020.

[41] G. J. Silowash, D. M. Cappelli, A. P. Moore, R. F. Trzeciak, T. Shimeall, and L. Flynn, “Common sense guide to mitigating insider threats,” 2012.

[42] V. L. L. Thing and J. Wu, “Autonomous vehicle security: A taxonomy of attacks and defences,” in *2016 ieee international conference on internet of things (ithings) and ieee green computing and communications (greencom) and ieee cyber, physical and social computing (cpscom) and ieee smart data (smartdata)*, IEEE, 2016, pp. 164–170.

[43] S. Hansman and R. Hunt, “A taxonomy of network and computer attacks,” *Comput Secur*, vol. 24, no. 1, pp. 31–43, 2005.

[44] J. Lindemann and M. Fischer, “A memory-deduplication side-channel attack to detect applications in co-resident virtual machines,” in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018, pp. 183–192.

[45] S. N. Shirazi, A. Gouglidis, A. Farshad, and D. Hutchison, “The extended cloud: Review and analysis of mobile edge computing and fog from a security and resilience perspective,” *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2586–2595, 2017.

[46] C. M. R. da Silva, J. L. C. da Silva, R. B. Rodrigues, L. M. do Nascimento, and V. C. Garcia, “Systematic mapping study on security threats in cloud computing,” *arXiv preprint arXiv:1303.6782*, 2013.

[47] C. S. Alliance, “Top threats to cloud computing v1. 0,” *White Paper*, vol. 23, 2010.

[48] M. G. Gouda, A. X. Liu, and M. Jafry, “Verification of distributed firewalls,” in *IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference*, IEEE, 2008, pp. 1–5.

[49] V. Tkachov, M. Hunko, and V. Volotka, “Scenarios for implementation of nested virtualization technology in task of improving cloud firewall fault tolerance,” in *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)*, IEEE, 2019, pp. 759–763.

[50] N. Dezhabad and S. Sharifian, “Learning-based dynamic scalable load-balanced firewall as a service in network function-virtualized cloud computing environments,” *J Supercomput*, vol. 74, pp. 3329–3358, 2018.

[51] L. Vokorokos, M. Ennert, M. > Čajkovský, and J. Radušovský, “A survey of parallel intrusion detection on graphical processors,” *Open Computer Science*, vol. 4, no. 4, pp. 222–230, 2014.

[52] “Security Guidance for Critical Areas of Focus in Cloud Computing | CSA.” https://cloudsecurityalliance.org/artifacts/security-guidance-v4/ (accessed Apr. 16, 2023).

[53] S. Einy, C. Oz, and Y. D. Navaei, “The anomaly-and signature-based IDS for network security using hybrid inference systems,” *Math Probl Eng*, vol. 2021, pp. 1–10, 2021.

[54] H. Debar, D. Curry, and B. Feinstein, “RFC 4765: The intrusion detection message exchange format (IDMEF).” RFC Editor, 2007.

[55] J. Bau and J. C. Mitchell, “Security modeling and analysis,” *IEEE Secur Priv*, vol. 9, no. 3, pp. 18–25, 2011.

[56] J. Treurniet, “A network activity classification schema and its application to scan detection,” *IEEE/ACM Transactions on Networking*, vol. 19, no. 5, pp. 1396–1404, 2011.

[57] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, “Surveying port scans and their detection methodologies,” *Comput J*, vol. 54, no. 10, pp. 1565–1581, 2011.

[58] A. A. Cárdenas, J. S. Baras, and K. Seamon, “A framework for the evaluation of intrusion detection systems,” in *2006 IEEE Symposium on Security and Privacy (S&P’06)*, IEEE, 2006, pp. 15-pp.

[59] A. Milenkoski, M. Vieira, S. Kounev, A. Avritzer, and B. D. Payne, “Evaluating computer intrusion detection systems: A survey of common practices,” *ACM Computing Surveys (CSUR)*, vol. 48, no. 1, pp. 1–41, 2015.

[60] N. J. Puketza, K. Zhang, M. Chung, B. Mukherjee, and R. A. Olsson, “A methodology for testing intrusion detection systems,” *IEEE Transactions on Software Engineering*, vol. 22, no. 10, pp. 719–729, 1996.

[61] R. P. Lippmann *et al.*, “Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation,” in *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX’00*, 2000, pp. 12–26 vol.2. doi: 10.1109/DISCEX.2000.821506.

[62] C. Thomas, V. Sharma, and N. Balakrishnan, “Usefulness of DARPA dataset for intrusion detection system evaluation,” in *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2008*, SPIE, 2008, pp. 164–171.

[63] K. A. Jackson, “Intrusion detection system (IDS) product survey,” *Los Alamos National Laboratory*, 1999.

[64] F. Massicotte, F. Gagnon, Y. Labiche, L. Briand, and M. Couture, “Automatic evaluation of intrusion detection systems,” in *2006 22nd Annual Computer Security Applications Conference (ACSAC’06)*, IEEE, 2006, pp. 361–370.

[65] S. Antonatos, K. G. Anagnostakis, and E. P. Markatos, “Generating realistic workloads for network intrusion detection systems,” in *Proceedings of the 4th International Workshop on Software and Performance*, 2004, pp. 207–215.

[66] E. Alata, M. Kaâniche, V. Nicomette, and R. Akrout, “An automated approach to generate web applications attack scenarios,” in *2013 Sixth Latin-American Symposium on Dependable Computing*, IEEE, 2013, pp. 78–85.

[67] R. Akrout, E. Alata, M. Kaaniche, and V. Nicomette, “An automated black box approach for web vulnerability identification and attack scenario generation,” *Journal of the Brazilian Computer Society*, vol. 20, pp. 1–16, 2014.

[68] I. Mokube and M. Adams, “Honeypots: concepts, approaches, and challenges,” in *Proceedings of the 45th annual southeast regional conference*, 2007, pp. 321–326.

[69] N. Kambow and L. K. Passi, “Honeypots: The need of network security,” *International Journal of Computer Science and Information Technologies*, vol. 5, no. 5, pp. 6098–6101, 2014.

[70] S. Sharma and A. Kaul, “A survey on Intrusion Detection Systems and Honeypot based proactive security mechanisms in VANETs and VANET Cloud,” *Vehicular communications*, vol. 12, pp. 138–164, 2018.

[71] D. Bove, “Using Honeypots to Detect and Analyze Attack Patterns on Cloud Infrastructures,” *Security Research Group Department of Computer Science Friedrich-Alexander University Erlangen-Nürnberg*, 2018.

[72] A. R. Jayakrishnan and V. Vasanthi, “Attack Patterns on IoT devices using Honey Net Cloud”.

[73] C. Kelly, N. Pitropakis, A. Mylonas, S. McKeown, and W. J. Buchanan, “A comparative analysis of honeypots on different cloud platforms,” *Sensors*, vol. 21, no. 7, p. 2433, 2021.

[74] J. A. P. Marpaung, M. Sain, and H.-J. Lee, “Survey on malware evasion techniques: State of the art and challenges,” in *2012 14th International Conference on Advanced Communication Technology (ICACT)*, IEEE, 2012, pp. 744–749.

[75] M. Särelä, T. Kyöstilä, T. Kiravuo, and J. Manner, “Evaluating intrusion prevention systems with evasions,” *International Journal of Communication Systems*, vol. 30, no. 16, p. e3339, 2017.

[76] I. Homoliak, M. Teknos, M. Ochoa, D. Breitenbacher, S. Hosseini, and P. Hanacek, “Improving network intrusion detection classifiers by non-payload-based exploit-independent obfuscations: An adversarial approach,” *arXiv preprint arXiv:1805.02684*, 2018.

[77] J. Luo, S. Chai, B. Zhang, Y. Xia, J. Gao, and G. Zeng, “A novel intrusion detection method based on threshold modification using receiver operating characteristic curve,” *Concurr Comput*, vol. 32, no. 14, p. e5690, 2020.