University of Nottingham

Doctoral Thesis

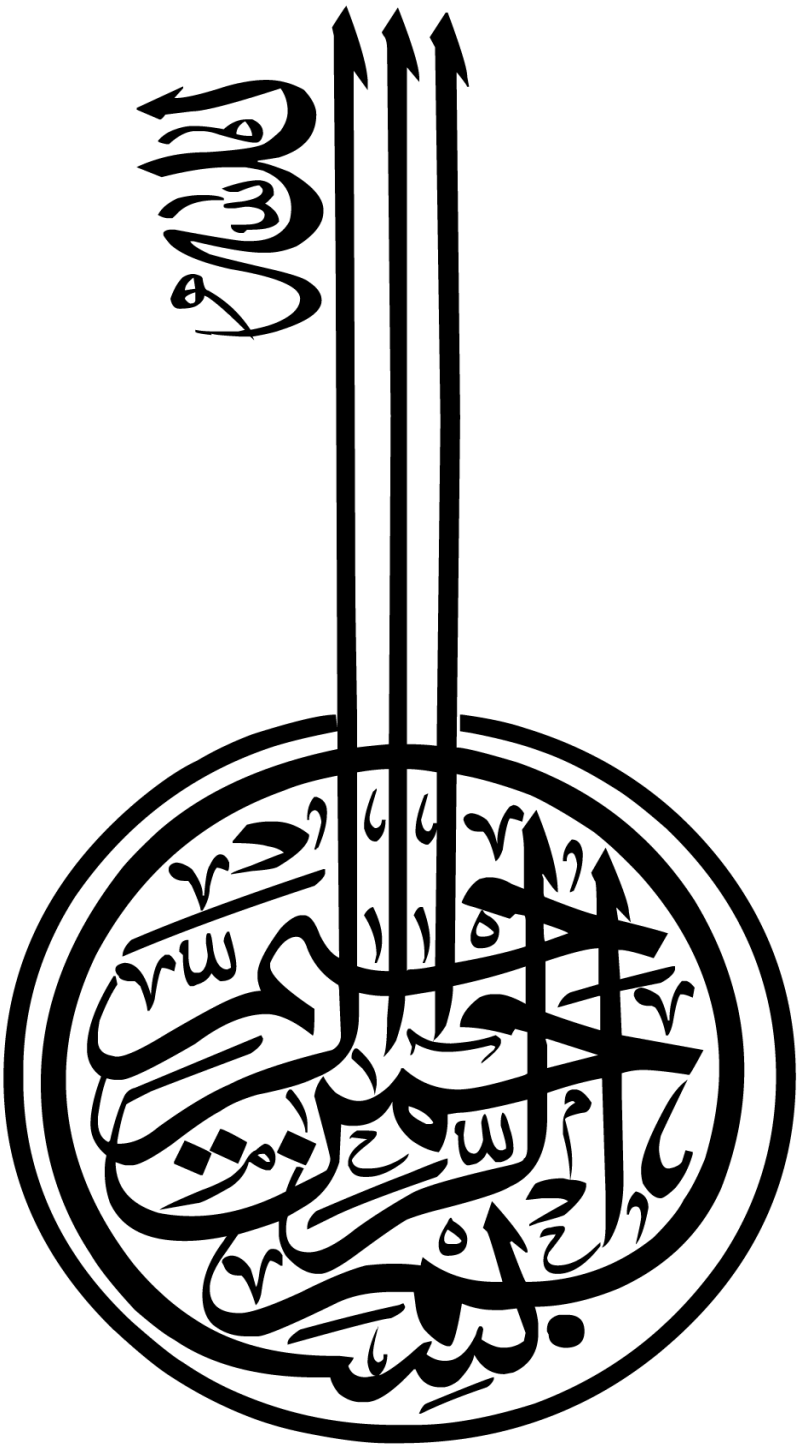
Cloud Intrusion Detection Systems: Fuzzy Logic and Classifications

Author:
Saeed Alqahtani

Supervisor:
Prof. Robert John

Thesis submitted to the University of Nottingham
for the degree of Doctor of Philosophy

September 2017



In the Name of God, Most Gracious, Most Merciful

"The crux of the problem, really, is the excessively wide gap between the precision of classical logic and the imprecision of the real work."

Lotfi A. Zadah,
Coping with the Imprecision of the Real World: An Interview, 1984

Abstract

Cloud Computing (CC), as defined by national Institute of Standards and Technology (NIST), is a new technology model for enabling convenient, on-demand network access to a shared pool of configurable computing resources such as networks, servers, storage, applications, and services that can be rapidly provisioned and released with minimal management effort or service-provider interaction. CC is a fast growing field; yet, there are major concerns regarding the detection of security threats, which in turn have urged experts to explore solutions to improve its security performance through conventional approaches, such as, Intrusion Detection System (IDS). In the literature, there are two most successful current IDS tools that are used worldwide: Snort and Suricata; however, these tools are not flexible to the uncertainty of intrusions. The aim of this study is to explore novel approaches to uplift the CC security performance using Type-1 fuzzy logic (T1FL) technique with IDS when compared to IDS alone. All experiments in this thesis were performed within a virtual cloud that was built within an experimental environment. By combining fuzzy logic technique (FL System) with IDSs, namely SnortIDS and SuricataIDS, SnortIDS and SuricataIDS for detection systems were used twice (with and without FL) to create four detection systems (FL-SnortIDS, FL-SuricataIDS, SnortIDS, and SuricataIDS) using Intrusion Detection Evaluation Dataset (namely ISCX). ISCX comprised two types of traffic (normal and threats); the latter was classified into four classes including Denial of Service, User-to-Root, Root-to-Local, and Probing. Sensitivity, specificity, accuracy, false alarms and detection rate were compared among the four detection systems. Then, Fuzzy Intrusion Detection System model was designed (namely FIDSCC) in CC based on the results of the aforementioned four detection systems. The FIDSCC model comprised of two individual systems pre-and-post threat detecting systems (pre-TDS and post-TDS). The pre-TDS was designed based on the number of threats in the aforementioned classes to assess the detection rate (DR). Based on the output of this DR and false positives of the four detection systems, the post-TDS was designed in order to assess CC security performance. To assure the validity of the results, classifier algorithms (CAs) were introduced to each of the four detection systems and four threat classes for further comparison. The classifier algorithms were OneR, Naive Bayes, Decision Tree (DT), and K-nearest neighbour. The comparison was made based on specific measures including accuracy, incorrect classified instances, mean absolute error, false positive rate, precision, recall, and ROC area. The empirical results showed that FL-SnortIDS was superior to FL-SuricataIDS, SnortIDS, and SuricataIDS in terms of sensitivity. However, insignificant difference was found in specificity,

false alarms and accuracy among the four detection systems. Furthermore, among the four CAs, the combination of FL-SnortIDS and DT was shown to be the best detection method. The results of these studies showed that FIDSCC model can provide a better alternative to detecting threats and reducing the false positive rates more than the other conventional approaches.

Acknowledgements

All praise is due to the Almighty God for the countless blessings He showered upon me throughout my journey in life, and for all the physical and mental strength. He bestowed on me during the activities of preparing this thesis and conducting the associated research.

First and foremost, my deepest gratitude and thanks go to my principal supervisor Prof. Rober John, who guided me through every single step along the way for the last three years, and helped me become the person who I am today. Without his great support, advice and above all the unparalleled patience of Prof. John, this thesis would not have seen the light.

Contents

List of Figures

List of Tables

List of Abbreviations

Symbol	Meaning
AD	**A**nomaly **D**etection
CA	**C**lassifier **A**lgorithm
CC	**C**loud **C**omputing
DLO	**D**ata **L**ocation **O**ptions
DoS	**D**enial of **S**ervice
FIDSCC	**F**uzzy **I**ntrusion **D**etection **S**ystem in **C**loud **C**omputing
FL	**F**uzzy **L**ogic
HIDS	**H**osts **B**ased on **I**ntrusion **D**etection **S**ystem
IaaS	**I**nfrastructure **as a S**ervice
IDL	**I**ntrusion **D**etector **L**earning
IDS	**I**ntrusion **D**etection **S**ystem
ILO	**I**nfrastructure **L**ocation **O**ptions
IPS	**I**ntrusion **P**revention **S**ystem
NIDS	**N**etworks **B**ased on **I**ntrusion **D**etection **S**ystem
PaaS	**P**latform **as a S**ervice
Probe	**P**robing: **S**urveillance and **P**robing for **I**nformation
R2L	**R**emote-to-**L**ocal: **U**nauthorised **A**ccess from a **R**emote **M**achine
SaaS	**S**oftware **as a S**ervice
SD	**S**ignature **D**etection
T1FL	**T**ype-1 **F**uzzy **L**ogic
T2FL	**T**ype-2 **F**uzzy **L**ogic
U2R	**U**ser-to-**R**oot: **U**nauthorized **A**ccess to **G**ain **L**ocal **R**oot

To my parents, Zenah, Jana, Majed and each member of my lovely family for their endless love, support and encouragement

Chapter 1

Introduction

1.1 Background

The use of the Internet has been increasing day by day and Internet traffic is exponentially increasing. The service providers (SPs) such as web SPs, email SPs, cloud SPs have to deal with millions of users per second; yet the level of threats to these services is also very high. To deal with this much number of users is a big challenge but, detection and prevention of such kinds of threats is even more challenging and also vital due to the fact that those threats might cause a severe loss to the SPs in terms of privacy leakage or unavailability of the services to the users. Therefore, IT industry attempts to provide very safe services to the users. For example, since the technology of cloud computing (CC) has been become the most demanding service to consumers, IT industry incorporated the cloud security issues by providing the technology of CC alongside Intrusion Detections Systems (IDS) or Intrusion Prevention Systems (IPS) [Mell and Grance, 2009].

CC has been described by Iyoub et al. (2013) as a commodity where it is similar to electricity once connected to the devices [Iyoob et al., 2013]. This is due to the fact that CC is not only a specific technology; but also it is a step in making IT a commodity which is enabled by technological advances. This description shows the value of CC from the following facts;

1. There is a huge prospect for savings in the CC market as it has recently grown to about a billion U.S. dollars a year [Marston et al., 2011, Nair, 2014].

2. The usage of data in CC is often automatically and constantly collected in order to make better decisions [Leavitt, 2009, Nair, 2014].

3. The migration to CC technology has been so active and fast[Jacobson, 2010, Nair, 2014].

As a consequence, there is no doubt that the use of CC has a huge impact on business strategy towards information sharing but some issues regarding its detection performance still remained such as false alarms. Thus, such issues have urged experts to discover solutions in order to improve security performance. One of these solutions is IDS; yet, such a technique is no longer sufficient to specify and assign the threats within CC accurately; and therefore, Nair [2014] has recommended to incorporate these techniques with some intelligent approaches such as embedded programming approach, agent based approach, software engineering approach, artificial intelligence system and soft computing (SC)[1]. In this research, Fuzzy Logic (FL) has been used as it is flexible to the uncertainties of intrusions. In this research, Fuzzy Logic (FL) has been used as a SC technique alone.

FL is an effective approach to tackle real-life problems and vagueness in human thinking with the uncertainty in real life [Hosmer, 1993, Zadeh, 2008, 1984, Zimmermann, 1987]. This thesis explored new novel approaches that may uplift CC security performance using Type-1 fuzzy logic (T1FL) technique with IDS when compared to IDS alone. This is because of the fact that T1FL provides a better alternative to detecting threats and reducing false positive ratios than other conventional approaches. All experiments in this thesis were performed within a virtual cloud that was built within an experimental environment. By combining fuzzy logic technique with IDSs, namely SnortIDS and SuricataIDS, SnortIDS and SuricataIDS for detection systems were used twice (with and without FL) to create four detection systems (FL-SnortIDS, FL-SuricataIDS, SnortIDS, and SuricataIDS) using Intrusion Detection Evaluation Dataset (namely ISCX)[2]. ISCX comprised two types of traffic (normal and threats); the latter was classified into four classes including Denial of Service, User-to-Root, Root-to-Local, and Probing. Sensitivity, specificity, accuracy, false alarms and detection rate were compared among the four detection systems. Then, Fuzzy Intrusion Detection System model was designed (namely FIDSCC) in CC based on the results of the aforementioned four detection systems. The

[1]Soft computing techniques are Fuzzy Logic (FL), Neural Network (NN), and Genetic Algorithms (GAs).

[2]This dataset is found under the website of university of New Brunswick (http://www.unb.ca/cic/research/datasets/index.html)

FIDSCC model comprised two individual systems pre-and-post threat detecting systems (pre-TDS and post-TDS). The pre-TDS was designed based on the number of threats in the aforementioned threat classes to assess the detection rate (DR). Based on the output of this DR and the false positive of the four detection systems, the post-TDS was designed in order to assess CC security performance. To assure the validity of the results, classifier algorithms (CAs) were introduced to each of the four detection systems and the four threat classes for further comparison. The classifier algorithms were OneR, Naive Bayes, Decision Tree (DT), and K-nearest neighbour. The comparison was made based on specific measures including accuracy, incorrect classified instances, mean absolute error, false positive rate, precision, recall, and ROC area.

1.2 Research Motivation and Hypothesis

The motivation behind this study was the fact that there has been a lack of extensive research in the field of cloud security. For example, based on the literature review, the research in the area of the techniques of cloud security such as IDS/IPS, has not tackled the uncertainties in CC. This is because of the following reasons;

1. majority of the scholars in the industry and publications focused on the short descriptive and practical guidance presented by the perspectives of decision-making [Foster et al., 2009, Jacobson, 2010, Menascé and Ngo, 2009, Truong, 2014, Yinglei and Lei, 2011].

2. majority of the studies have been directed towards analysing if the IDS/IPS models should be adopted by the organisation or not, and if yes, then which model would be viable for the organisation?.

3. majority of studies have been focused on the following areas;

   (a) Cloud security definition [Iyoob et al., 2013, Lindner et al., 2010, Paulitsch, 2009].

   (b) analysing and presenting the challenges and benefits of business solutions in CC [Demirkan et al., 2010, Durowoju et al., 2011, Li et al., 2012a, Singh, 2010, Tsao et al., 2010, YiPeng, 2011].

   (c) the adoption of IDS/IPS models into CC [Casey G., 2012, Wu et al., 2013].

   (d) cloud environment's application and architecture [Ferguson and Hadar, 2011, Yinglei and Lei, 2011].

Consequently, there is a need to focus on cloud security concerns regarding its detection performance. In this thesis, there are three main aspects: CC, IDS, and FL. In order to connect each aspect to others, all experiments were performed within a virtual cloud that was built in an experimental environment. Then we used the combination of FL and IDS as detection systems in order to maximise the accuracy, specificity, detection rate and minimise the sensitivity and false alarms ratios. To assure the validity of the results, some classifier algorithms (CAs), such as Decision Tree (DT), Naive Bayes (NB), OneR and K-Nearest Neighbour (KNN) as stated in [Chauhan et al., 2013], were introduced to each of the four detection systems and the four threat classes: DoS, U2R, R2L, and Probe for further comparison. Therefore, we hypothesised that a better alternative in detecting threats and reducing false positive ratios may be provided by a model of Fuzzy IDS within CC more than conventional approaches.

1.3 Aim and Objectives

The main aim is to explore new novel approaches that can uplift the CC security performance using Type-1 fuzzy logic with IDS when compared to IDS alone. In order to achieve that target, the objectives of this study including the following:

1. to propose a Fuzzy-Logic engine based on IDSs (FL-SnortIDS and FL-SuricataIDS) in order to define the uncertainties of intrusions in CC and evaluate the security performance.
2. to compare the four detection systems (SnortIDS, SuricataIDS, FL-SnortIDS, and FL-SuricataIDS) and investigate which of these four detection systems outperforms others.
3. to assure the validity of the results through four classifier algorithms (CAs) so-called OneR, NB, DT, and KNN.
4. to compare these four CAs to each of the four detection systems and the four threat classes (DoS,U2l, R2L and Probe).

1.4 Research Question

The research question for this study was that "does T1FL system uplift CC security performance compared to the IDS alone?".

The criteria to answer such a question were done on the basis of the following metrics:

1. **Accuracy:** The numbers of threats detected
2. **False Alarms Ratio:** The false positives and false negatives ratio per each detection system.
3. **Sensitivity Ratio:** The true positives and false negatives ratio per each detection system.
4. **Specificity Ratio:** The false positives and true negatives ratio per each detection system.
5. **Threat Detection Rate:** The threat categories for a scanning attack of each detection systems

1.5 Research Contributions

This thesis provides the following three contributions;

1. **A Comparative Analysis for The Alerts of IDS Fuzzy Classifiers Approaches within Cloud Computing**. This contribution was is reported in chapter 4 in order to enhance the security that IDS provides using ISCX dataset. Two IDSs: SnortIDS and SuricataIDS have been developed in which they differ in their detection capabilities, performance and accuracy. SnortIDS and SuricataIDS detection systems were used twice (with/without FL) to create four detection systems (FL-SnortIDS, FL-SuricataIDS, SnortIDS, and SuricataIDS) using Intrusion Detection Evaluation Dataset (namely ISCX).
2. **A Comparative Analysis of Different ClassificationTechniques for Cloud Intrusion Detection Systems' Alerts and Fuzzy Classifiers**. This contribution is reported in chapter 5, WEKA was used as a data miner to analyse the data of each detection system from chapter 4. In chapter 5, the most common classification algorithms are utilised: Decision Tree (DT), Naive Bayes, OneR, and K-Nearest Neighbour (K-NN). These algorithms were chosen after investigating the most effective classification algorithms that are widely used. The aim of this study was to present a comparative study for the performance of each detection system: SnortIDS, SuricataIDS, FL-SnortIDS, and FL-SuricataIDS in order to test which classifier algorithm is the best and presents significant results.
3. **Security Performance Evaluation Using Type-1 Fuzzy Logic Approach (FIDSCC System)**. This contribution was conducted in chapter 6 that produces

a model using fuzzy inference engine based IDS to evaluate the security performance by tracing intruders on CC. This model called FIDSCC system. There were two fuzzy inference systems were created: first is to evaluate the attack classes of ISCX within CC and another is to assess security performance. Certain tests have been made for the comparison purposes amongst IDSs' systems and Fuzzy classifiers. These were satisfactory tests where the combining results of detection performances of all type of threats based on detection rate and false positive ratio showed that fuzzy classifiers do better than IDSs' systems within CC.

1.6 Academic Publications Produced

The following publications were produced as a direct result of the work undertaken during the course of conducting this research:

- Alqahtani, S.M., Balushi, M.A., John, R., 2014a. An intelligent intrusion detection system for cloud computing (SIDSCC), in: Computational Science and Computational Intelligence (CSCI), 2014 International Conference on, IEEE. pp. 135-141.
- Alqahtani, S.M., Balushi, M.A., John, R., 2014b. An intelligent intrusion prevention system for cloud computing (SIPSCC), in: Computational Science and Computational Intelligence (CSCI), 2014 International Conference on, IEEE. pp. 152-158.
- Saeed M. Alqahtani and Robert John, A Comparative Study of Different Fuzzy Classifiers for Cloud Intrusion Detection Systems' Alerts. *published in The 2016 IEEE Symposium Series on Computational Intelligence (IEEE SSCI 2016).*
- Saeed M. Alqahtani and Robert John, A Comparative Analysis of Different Classification Techniques for Cloud Intrusion Detection Systems' Alerts and Fuzzy Classifiers. *accepted by Computing Conference 2017 (IEEE SAI2017).*

1.7 Thesis Structure

The first chapter introduces the thesis topic and relevant concepts. It also provides a list of the aims of the thesis, as well as the scientific contributions and papers published during the study. The literature review in the second chapter was presented, highlighting

the recent developments in IDSs' systems and fuzzy logic based on IDS systems in CC. The third chapter describes the research problem and all the experimental design materials and proposed methods. In the fourth chapter, a comparative analysis of ranking IDS based on fuzzy classifiers' approaches was conducted to analytically compare the functionality, working and the capability of the four detection systems (FL-SnortIDS, FL-SuricataIDS, SnortIDS, and SuricataIDS in order to boost the security performance. Then, a comparative study was conducted in the fifth chapter against the results of aforementioned detection systems through classifier algorithms (CAs) that were introduced to each of the four detection systems and the four threat classes (DoS, U2L, R2L, Probe) for further comparison. In the sixth chapter, FIDSCC model was designed in order to compare the detection rate and false positive ratio through the results of the aforementioned four detection systems. Finally, the last chapter presents a summary of the work undertaken in this research along with recommendations for future work.

Chapter 2

Literature Review

In this modern age, digital communication is no more a big task and thus every single internet user can have an access to on-line information pool or can interact with anyone without worrying about the distance between them. According to the statistics reported in [Stats, 2016], the total estimated internet users in the year of 2016 are almost 3,424,971,237, which is 46.1% of the world population. Thus, it cloud be concluded that internet has now become a part of modern age human.

Computer networks are attacks prone, unreliable and unsafe, which means that the users may experience malicious activities and lose their privacy, personal data or any other important information that is available on-line, depending on the nature of attacks. Although, domestic users may ignore this insecurity issue, yet it would be a vital problem for most of the people who have privacy issues or who may not be able to recover from any kind of personal data loss. Similarly, corporate offices, banks, hospitals, law enforcement organisations, emails services providers and millions of other organisations take extreme care of their privacy and availability of their services on-line [Talwar, 2015].

This chapter provides an overview of intrusion detection system, fuzzy logic, cloud computing and its security techniques such as anomaly detection and signature detection. This chapter also covers some approaches for evaluating cloud security performance, together with related scientific literature review. Moreover, related work on IDS techniques, IDS based fuzzy logic, IDS based on data mining , and data mining and classification are given.

2.1 Background

Cyber-attacks have become very crucial and it is now considered that the cyber war has now over taken the nuclear war in this modern age [Shackelford, 2009]. Thus, many international rules have been created by the law enforcement agencies including USA's [Burkadze, 2016, Graham, 2010, O. A. Hathaway and Spiegel, 2012]. It has also attracted the attentions of many researchers and a great deal of work has been done in literature to protect the systems from cyber-attacks such as, inventory of authorised and unauthorised software and devices, to make configurations of hardware and software secure, to install intelligent firewalls, to install anti-malware software, to develop intrusion detection systems and to develop malware defensive systems. The most followed strategy to prevent such kind of cyber-attacks is the development of intrusion detection system (IDS) [Google Patents, 2009].

IDS systems are basically hardware or software systems that are deployed along with the main systems to monitor all the digital activities and the incoming as well as outgoing network traffic. These systems are made intelligent enough to detect the malware or suspicious activities by monitoring the whole system; and therefore, they produce alarms or reports against such activities. IDS system acts as a firewall and keeps the main system safe from the malwares. Hence, it is deployed along with almost every critical system that is exposed to threats, making the organisation reliable and trustworthy.

The capability of IDS systems to detect the suspicious activities depends on how they have been developed. Stronger the IDS system would be, safer the main system would be, leading the organisation to win the trust of its clients. Moreover, IDS systems are consistently upgraded due to the fact that the cyber-attacks are becoming crucial and stronger day by day. A great deal of work has been done in literature in making intelligent and strong IDS systems. For example, Reputation Services have been added in the IDS systems. These services gather information about the suspicious protocols, IP addresses, domains and finally make a decision that either the traffic is malicious or not [Hwang et al., 2009]. Transforming the wired IDS systems to wireless systems has also increased the safety level of the critical systems [Brutch and Ko, 2013]. With the fast growing HTTPS traffic, the SSL traffic inspection feature has also been added in the IDS systems to stay up to date [Augustin and Baláž, 2011].

There is a big challenge to detect and prevent attacks with a huge number of users. This is due to the fact that the threats might cause a severe loss to the service providers in terms of privacy leakage or unavailability of the services to the users. To incorporate this

issue, several Intrusion Detections Systems (IDSs) such as Snort and Suricata have been developed that differ in their detection capabilities, performance and accuracy along in order to enhances the performance and accuracy of these developed IDSs in terms of increased accuracy, specificity, sensitivity and reduced false alarms. Other different tools are also available that performs Intrusion Detection. For instance, Security Onion system has the capability to monitor vLANs and virtualised networks but it cannot be used as an intrusion prevention system [Burks, 2012]. OSSEC system can generate real-time alarms and has the capability of monitoring the files integrity [Bray et al., 2009]. OpenWIPS-NG system is dependent on network interfaces, devices, servers and other infrastructures [Bezborodov et al., 2016]. BRO system is an alternative to Security Onion but has more defined rules to detect the malicious activities [Paxson et al., 2013]. Among all IDS systems, Snort and Suricata are considered to be the most efficient tool that performs real-time protection, real-time traffic analysis, protocol analysis, content matching, packets logging on IP networks and possesses many kinds of attacks detection capability [Roesch et al., 2009].

Although there are a number of advantages of intrusion prevention systems (IPS), but there is a need for the leak of alarm, the false rate of alarm and delivery in real time [Xin and Yun-jie, 2010]. The IPS classification system depends on the platform of technology and detection system. The operation platform has the general classification of IPS into Network based IPS (NIPS) and Host based IPS (HIPS). Salah et al. [2010] claimed that the NIPS helps the detection of traffic by use of barriers and also deals with issues in detection although there is a relativity of HIPS on the installation of the network hosts and protection against the malwares [Salah et al., 2010].

The role of IDS and IPS is to detect and prevent the network topology against abnormal activities. IDS can detect abnormal activities, which are inadequate, erroneous or irregular in a network, and report them to the administrator. In addition to that, IDS also can work in a way that detects an unauthorised intrusion if a network or a server is experienced to do so. IPS is an extension of IDS system where IPS can also block connections or drop abnormal packets if they consist of unauthorised data. The categories of IDS/IPS can be classified in two levels; Network-Based Intrusion Detection/Prevention System (NIDS/NIPS) and Host-Based Intrusion/Prevention Detection System (HIDS/HIPS) [Alqahtani et al., 2014a,b].

NIDS verifies from the network traffic and then observing multiple hosts to identify intrusions. Also, NIDS works on sensors where they capture and analyse each packet of the network traffic and then identify abnormal content. HIDS is circulated in host

machines or a server where its role is to analyse data and identify unusual behaviour. Also, HIDS works as a machine that compares the usual profile of the host with the captured activities to identify potential anomalies [Alqahtani et al., 2014a,b]. NIPS is a system that observes the whole network traffic in order to capture the malicious activities. HIPS is a system that monitors the installed software package of a single host. There are others levels such as NBA[1] and WIPS[2] but they can be considered under the main categories of IPS [Alqahtani et al., 2014a] and [Alqahtani et al., 2014b].

2.2 Fuzzy Logic

2.2.1 Fuzzy Systems and Uncertainty

in 1965, fuzzy set theory was first introduced by Zadeh [1965]. The field of FSs has evolved over the last 50 years and FSs have been accepted as an adequate methodology for developing systems that are able to deliver adequate performance in the face of uncertainty and imprecision [Feng, 2006, Hagras and Wagner, 2012, Lee, 1990, Sugeno, 1985]. Hence, FLSs have been successfully implemented in many real world applications and are used in many areas. In addition, FS theory provides a simple and efficient method of designing FLSs that is close to human thinking and perception [Zadeh, 1994].

In particular, fuzzy logic control (FLC), as one of the earliest applications of FLSs, has become one of the most successful applications. The first FLC was developed in 1975 by Mamdani and Assilian to control a steam engine in a laboratory [Mamdani and Assilian, 1975]. The first industrial application of fuzzy logic was developed in 1976 by Blue Circle Cement and SIRA in Denmark for a cement kiln controller that went into operation in 1982 [Holmblad, 1982]. In 1987, the Sendai subway system in Japan operated an automatic train controller based on fuzzy logic systems. Since then, FLSs have been applied with great success to many applications. Recently, FLSs have been used in many real-world applications used in people's daily lives such as washing machines, air conditioners, video cameras, medical diagnosis systems, car braking systems, etc. [Wang, 1999],[Lee, 1990],[Karray and De Silva, 2004].

FLSs are widely accepted for their ability to model and handle uncertainty. Several efforts have been made to define the uncertainties in FLSs and their underlying sets. A

[1]NBA is a Network Behaviour Analysis system that tests the flow of network traffic and then detects and prevents the malicious traffic

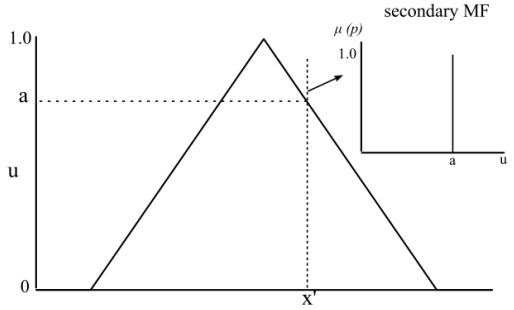
[2]WIPS is a Wireless Intrusion Prevention System that monitors and analyses the abnormal traffic in wireless networks

general discussion about uncertainty was presented by Klir and Wierman [1999] showing three types of uncertainty: fuzziness, strife and non-specificity. Fuzziness (vagueness) results from the imprecise boundaries of FSs [Mendel, 2001]. Non-specificity (imprecision) is linked to information-based imprecision. Strife (discord) expresses conflicts among the various sets of alternatives [Mendel, 2001].

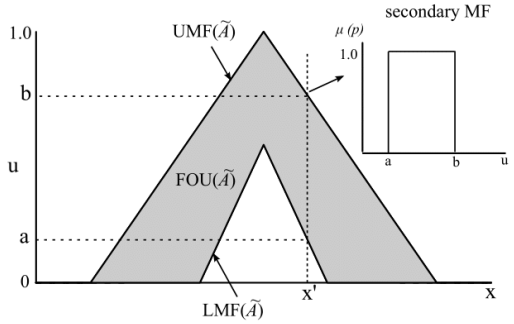
The T1 FLS is the most common and widely used FLS. T1 FLSs are based on T1 FSs and have been successfully applied in many applications, such as control systems (especially for the control of complex non-linear systems that are difficult to model analytically [Zadeh, 1973],[King and Mamdani, 1977]), decision making, classification problems, system modelling and others [Ross, 2009]. However, it has been shown that there are limitations in the ability of T1 FSs to directly model and minimise the effects of uncertainty [Hagras, 2004], [Hagras, 2007], [Mendel, 2001]. This is because T1 MFs are precise, as stated by Mendel [2001] and their membership grade is a crisp value (see Figure 2.1(a)).

Recently, a significant increase in research has been devoted to more complex forms of fuzzy logic such as IT2 FLSs and, more recently, GT2 FLSs. This advancement is due to the realisation that T1 FSs can only handle a limited range of uncertainty whereas T2 FSs allow for better modelling of uncertainty as they encompass an Footprint of uncertainty (FOU), which, associated with their third dimension, gives more degrees of freedom to the use of T2 FSs in comparison to T1 FSs (i.e., T2 FSs are described by MFs that are characterized by more parameters than are MFs for T1 FSs [Mendel, 2007a]).

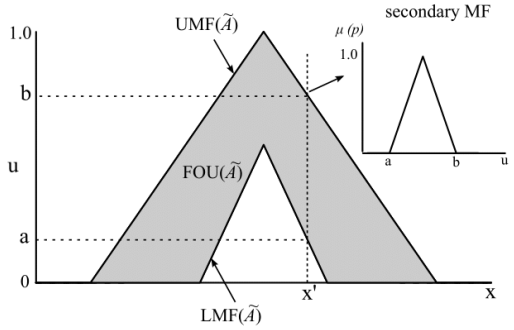
In 1975, T2 FSs was introduced by Zadeh as an extension of the concept of T1 FSs [Zadeh, 1975], characterised by MFs that are themselves fuzzy for which the membership degrees are expressed as FSs on [0,1], have been widely accepted as more capable of modelling higher orders of uncertainty than T1 FSs [John and Coupland, 2007],[John, 1998, Mendel, 2007a,b,c, Wu and Mendel, 2009]. Mendel Mendel [2001] argued that T1 FSs are inadequate to model many types of uncertainty that can be present in an FLS, whereas T2 FSs are able to handle them including the three types of uncertainty : fuzziness, strife, and non-specificity. IT2 FSs Mendel [2001] are a special case of GT2 FSs and currently the most widely used due to their great reduction in computational cost. It has been shown in the literature many types of FSs such as interval-valued FSs, gray set, intuitionistic FSs, etc. [Sebastian and John, 2016],[Yang et al., 2014]. Bustince et al. [2016] presents more details about the definitions and the relationships between different FSs.



(a) Type-1 fuzzy set



(b) Interval type-2 fuzzy set



(c) General type-2 fuzzy set

FIGURE 2.1: An example of the three types of FSs with view of the secondary MFs with same input x' (a) type-1 fuzzy set, (b) interval type-2 fuzzy set and (c) general type-2 fuzzy set.

Figure 2.1 shows the three different types of fuzzy sets. A T1 FS shown in Figure 2.1(a), an IT2 FS shown in Figure 2.1(b) and a GT2 FS shown in Figure 2.1(c). These figures also show the secondary MFs (third dimension) of the three fuzzy sets using the same input $x = x'$. It is important to consider the uncertainty models provided by these three types of FSs. In the T1 FS, the degrees of membership are specified as crisp numbers taking values within the interval [0,1]. In the GT2 FS, the degrees of membership are

themselves fuzzy and each is specified as a T1 FS (a secondary membership function). In a case when the secondary MF is equal to 1, then GT2 FS will reduce to an IT2 FS. The T1 FS is depicted in Figure 2.1(a), showing the value of the primary domain (membership) at $x = x'$ that takes only one value, a, at which the secondary grade equals 1. So, in the case of the T1 FS, there is no uncertainty associated with the primary membership value at each x value [Mendel, 2001]. For an IT2 FS (as shown in Figure 2.1(b)), the primary domain (membership) at $x = x'$ takes values within the interval [a,b] and each point in this interval has a secondary membership equal to 1. Hence, an IT2 FS contains a maximum amount of uncertainty that is equally distributed over the interval [a,b] [Mendel et al., 2014]. For a GT2 FS (as shown in Figure 2.1(c)), the primary (domain) membership at $x = x'$ also takes values within the interval [a,b], but is different from the IT2 FS since each point in this interval has a different secondary membership. Overall, it can be observed that the uncertainty that is associated with a GT2 FS is located between the uncertainty of a T1 FS and an IT2 FS [Mendel et al., 2014].

2.2.2 Basic Concepts of Type-1 Fuzzy Sets

A fuzzy set was defined originally by Zadeh [1965] as an extension of a (classical) crisp set. In crisp sets, the membership of elements is a binary value, which allows an element to either belong (full membership) or not belong to the set (no membership at all). In contrast, FSs permit partial membership so that an element may partially belong to a set. In a crisp set, the membership or non-membership of an element x in the crisp set A is described by the membership function (MF) (also called characteristic function) μ_A of A, where [Mendel, 2001],[Zimmermann, 2011],[Klir and Yuan, 1995]

$$\mu_{_A}(x) = \begin{cases} 1, & \text{if } \ x \in A \\ 0, & \text{if } \ x \notin A \end{cases} \tag{2.1}$$

Fuzzy set theory extends this concept by defining partial memberships, which can take values in the interval [0,1]. In a T1 FS (see Figure 2.1(a)) also called simply a FS, A is defined on a universe of discourse X and the MF for A is $\mu_{_A}$: $X \rightarrow$ [0,1] [Mendel, 2001]. For each element in the set, $x \in X$, the value of $\mu_{_A}(x)$ is the degree of membership of x in A between zero and one. A fuzzy set A in universe of discourse X can be represented

in pairs of x and the value of its MF, $\mu_A(x)$ as [Mendel, 2001]:

$$A = \{(x, \mu_A(x)) \mid \forall x \in X\} \tag{2.2}$$

If the fuzzy set A has a continuous universe of discourse X, it can be written as [Zadeh, 1965],

$$A = \int_X \mu_A(x)/x, \tag{2.3}$$

where the integral sign denotes union, and the division sign represents association (the collection of all points $x \in X$ with associated MF $\mu_A(x)$). If the fuzzy set A has a discrete universe of discourse X, it can be written as [Zadeh, 1965],

$$A = \sum_X \mu_A(x)/x, \tag{2.4}$$

where the summation sign denotes the set-theoretic operation of union and the division (slash) sign represents association (the collection of all points $x \in X$ with associated MF $\mu_A(x)$).

2.2.2.1 Properties of Type-1 Fuzzy Sets

To describe T1 FSs more specifically, we will define some of their properties [Zadeh, 1975],[Jang et al., 1997],[Klir, 2005].

The 'support' of a T1 FS A defined on X is a crisp set that contains all the elements of X ($x \in X$) that have non-zero membership grades in A (i.e., $\mu_A(x) > 0$) and is defined as:

$$\text{support}(A) = \{x \in X \mid \mu_A(x) > 0\} \tag{2.5}$$

A T1 FS that has a single point in X as the support with $\mu_A(x) = 1$ is called a 'singleton' T1 fuzzy set (see Figure 2.3(a)).

The 'core' of a fuzzy set A is the set of all points $x \in X$ such that $\mu_A(x) = 1$:

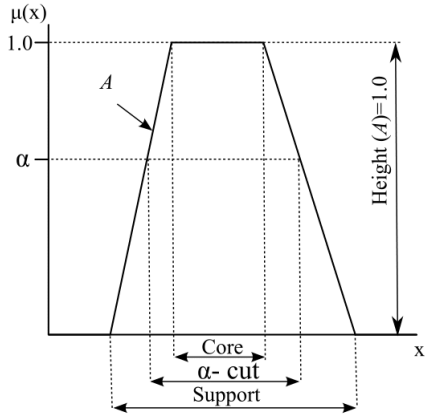


FIGURE 2.2: An example of a trapezoidal membership function and some of its properties

$$\text{core}(A) = \{x \in X \mid \mu_A(x) = 1\} \tag{2.6}$$

The 'height' of T1 FS A is the largest membership value, and it is defined as follows:

$$\text{height}(A) = \text{supp}_{x \in X}(\mu_A(x)) \tag{2.7}$$

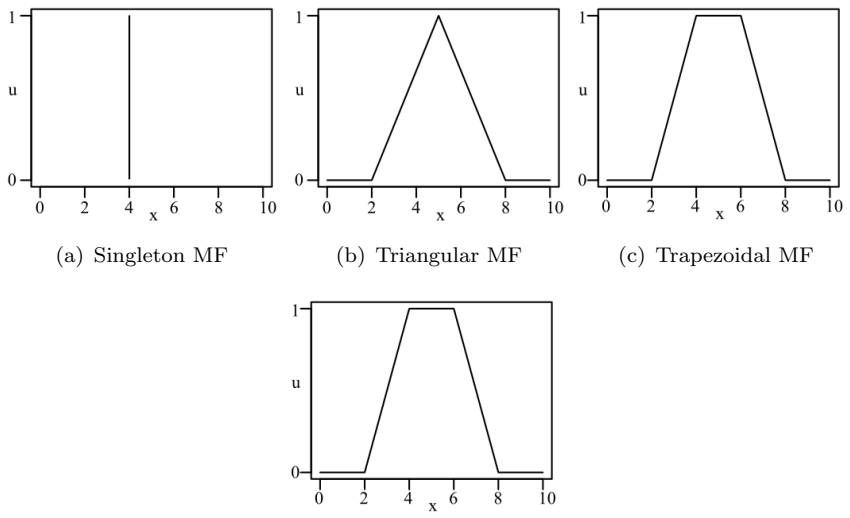
A fuzzy set A is called 'normal' when $\text{height}(A) = 1$ and it is called 'subnormal' when $\text{height}(A)$ ¡ 1.

One of the most important concepts of fuzzy sets is the concept of an α-cut. The α-cut of an FS A is the 'crisp' set of all elements that have a membership value greater than or equal to α. A T1 FS A is defined on X and α is a number in [0,1], an α-cut, A_α is defined by [Klir, 2005]:

$$A_\alpha = \{x \in X \mid \mu_A(x) \geq \alpha\} \tag{2.8}$$

These properties are illustrated by the trapezoidal membership function A in Figure 2.2

The membership functions commonly used in practice are singleton, triangular, trapezoidal, Gaussian, and bell-shaped [Mendel, 2001]. Generally, MFs can either be chosen by experts (users) or they can be created using optimisation methods [Jang, 1992],[Wang and Mendel, 1992b],[Wang and Mendel, 1992a]. Examples of singleton, triangle, trapezoidal and Gaussian MFs are depicted in Figure 2.3.



(a) Singleton MF (b) Triangular MF (c) Trapezoidal MF

(d) Gaussian MF

FIGURE 2.3: Examples of four T1 MFs (a) Singleton MF, (b) triangular MF, (c) trapezoidal MF and (d) Gaussian MF.

2.2.2.2 Type-1 Fuzzy Set Operations

Corresponding to the crisp set operations of union, intersection and complement, fuzzy sets have the same operations and are called Zadeh's operations as they were initially defined in Zadeh's paper [Zadeh, 1965].

Assume T1 FSs A and B are two subsets of X and are defined by their MFs $\mu_A(x)$ and $\mu_B(x)$. The union of A and B is defined by $\mu_{A\cup B}(x)$:

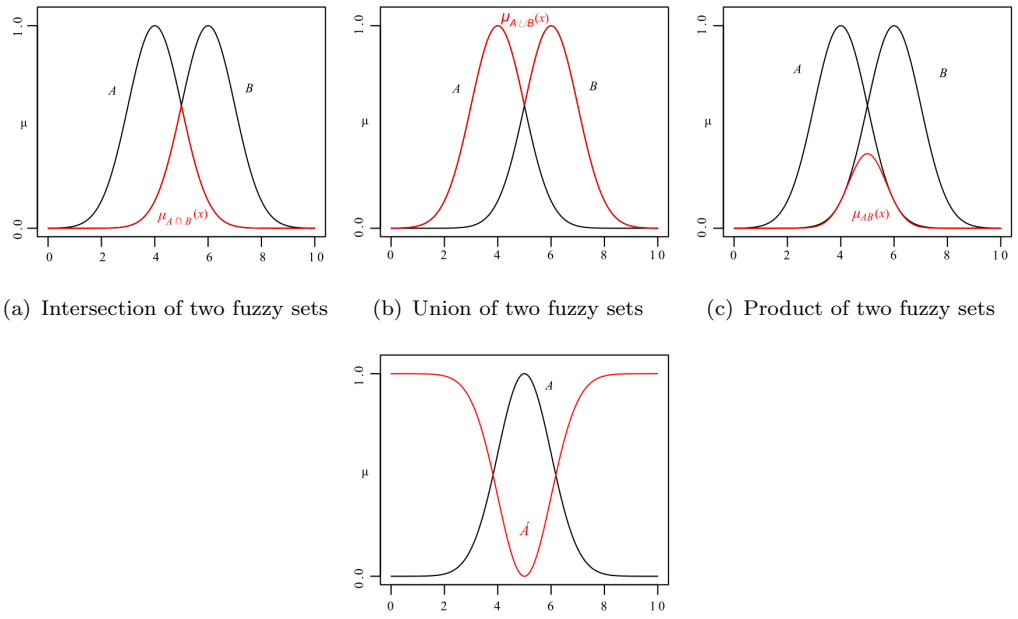
$$\mu_{A\cup B}(x) = \max[\mu_A(x), \mu_B(x)], \quad \forall x \in X \tag{2.9}$$

The intersection of A and B is defined by $\mu_{A\cap B}(x)$:

$$\mu_{A\cap B}(x) = \min[\mu_A(x), \mu_B(x)], \quad \forall x \in X \tag{2.10}$$

The product of A and B is defined by $\mu_{A*B}(x)$:

$$\mu_{A*B}(x) = \text{prod}[\mu_A(x), \mu_B(x)], \quad \forall x \in X \tag{2.11}$$



(a) Intersection of two fuzzy sets (b) Union of two fuzzy sets (c) Product of two fuzzy sets

(d) Complement of a fuzzy set

FIGURE 2.4: Examples of fuzzy operations using two T1 FSs A and B. (a) Intersection of two FSs, (b) union of two FSs,(c) product of two FSs and (d) complement of an FS.

The complement of A is defined by $\mu_{A'}(x)$:

$$\mu_{A'}(x) = 1 - \mu_A(x), \quad \forall x \in X \tag{2.12}$$

Thus the intersection, product and union operations are used to combine T1 FSs. They are equivalent to the operators 'AND' and 'OR' used in classical logic. Examples of the union, intersection, product and complement of the two T1 FSs A and B using (2.9), (2.10), (2.11) and (2.12) are depicted in Figure 2.4. While, different t-norms and t-conorms have appeared in the literature and were developed to generally define the operations, $\mu_{A\cup B}(x)$ and $\mu_{A\cap B}(x)$, in this thesis only product t-norm (2.11) that is used by the inference engine to combine the firing strengths from multiple antecedents and the maximum t-conorm (2.9) that is used to combined output fuzzy subset by taking the maximum (union) over all of the fuzzy subsets.

2.2.3 Brief Review of Type-1 Fuzzy Logic Systems

T1 FLSs are also known as fuzzy rule-based systems that can be considered as systems that map crisp inputs into outputs by utilising fuzzy sets theory [Negnevitsky, 2005].

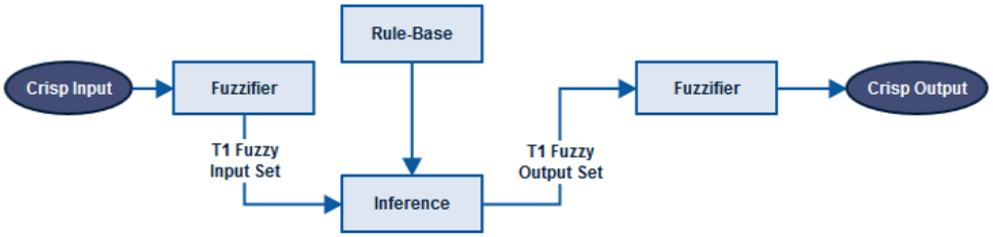
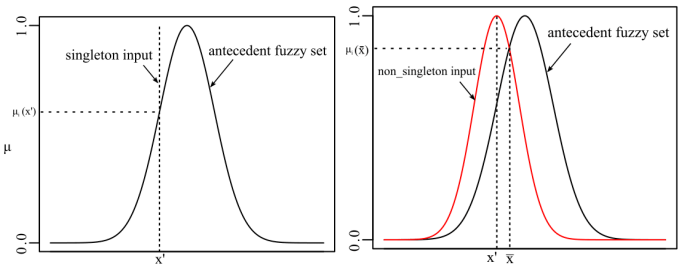


FIGURE 2.5: A type-1 fuzzy logic system.

The main types of FLSs are Mamdani **?** and Takagi & Sugeno [Mamdani, 1974, Takagi and Sugeno, 1985]. In this thesis, the Mamdani method is used because the core part of our investigations has been trying to capture the linguistic uncertainty, thus, using FSs in the whole FLS (including in the output modelling) is intuitive and valuable. In the also popular Takagi & Sugeno FLSs, the output is characterised by a function, rather than a FS, thus it is not couched in the context of using linguistic terms. The well-known Mamdani fuzzy model contains four components: fuzzifier, rule base, inference engine and defuzzifier [Mendel, 2001]. Figure 2.5 shows these components. As shown, crisp inputs are first fuzzified into type-1 fuzzy input sets. These activate the inference engine and the rule base to produce output T1 FSs, which are then combined to produce an aggregated T1 output FS. Finally, a defuzzifier produces a crisp output from the fuzzy output set(s) resulting from the inference engine. Further details on T1 FLSs can be found in [Mendel, 2001],[Cox, 1992]. The background and description of each of these components in the context of T1 FLSs are summarised below.

2.2.3.1 Fuzzifier

The fuzzifier maps crisp inputs into a membership grade of one or more T1 FSs, based on the given membership functions. This is achieved by evaluating the crisp inputs and assigning each input a membership degree $\mu_A(x)$ in its input FS. According to the type of fuzzification in [Mendel, 2001], T1 FLSs can be divided into singleton fuzzy logic systems (SFLSs) and non-singleton fuzzy logic systems (NSFLSs). A non-singleton T1 FLS has the same structure as a singleton T1 FLS, and they share the same type of rules; the only difference is the type of fuzzification. The majority of FLSs are using SFLS because singleton fuzzification is simpler and faster to compute. In singleton fuzzification, inputs are considered to be singleton FSs (see Figure 2.3), while non-singleton fuzzification models the FLS inputs as FSs. Non-singleton fuzzification allows better modelling of input uncertainties by modelling inputs as fuzzy sets and modelling linguistic uncertainty



(a) Singleton fuzzification (b) Non-singleton fuzzification

FIGURE 2.6: Examples of the fuzzification of a crisp input using (a) singleton and (b) non-singleton fuzzification.

using antecedent FSs in two steps [Wagner and Hagras, 2010]. To better cope with noisy and imprecise input measurements, the non-singleton fuzzifier is an efficient choice [Wang, 1994]. Figure 2.6 shows singleton and non-singleton fuzzification. More details on non-singleton fuzzification can be found in [Mendel, 2001],[Mouzouris and Mendel, 1994, 1997, Sahab and Hagras, 2011]. A rich discussion on fuzzification, membership function creation and desired output can be found in [Sinha and Dougherty, 1993].

2.2.3.2 Rule Base

Fuzzy rule base (set of fuzzy rules) is the core part of an FLS that is used in the inference process. The most commonly used are: Mamdani [Mamdani, 1974], where the rule consequents are FSs, and Takagi & Sugeno [Sugeno, 1985], where the rule consequents are crisp functions of the inputs. The rules can be expressed as a collection of conditional statements in the form IF–THEN statements. Without a loss of generality, the multiple inputs single output (MISO) system with the output variable y is considered here. In Mamdani-type fuzzy rules (first used by Mamdani in 1977), each rule is in the form of (2.13) with n inputs $x_1 \in X_1, \ldots, x_n \in X_n$ and one output $y \in Y$.

$$\text{IF } x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } A_2 \ \ldots \text{ and } x_n \text{ is } A_n \text{ THEN } y \text{ is } B, \tag{2.13}$$

where $x_1, x_2, \ldots, x_n$ are the input variables to the FLSs and $A_1, A_2, \ldots, A_n$ are the input FSs in the antecedent part and y is the output variable and B is the output FSs.

A fuzzy rule contains two parts, the IF part called the antecedent part and the THEN part called the consequent part. To generate these rules, many methods can be used such as deriving them from experts or from given numerical data sets as shown by [Wang and

Mendel, 1992c],[Nozaki et al., 1997],[Hong and Lee, 1996] where they generated fuzzy rules from training data. FSs are associated with the terms shown in the antecedent or consequent parts of rules and MFs are used to describe these FSs [Mendel, 2001].

2.2.3.3 Inference Engine

The inference engine in the Mamdani type (Equation (2.14)) provides a mapping of input FSs to output FSs using the rules (see (2.13)) from the rule base and the connecting operators such as the union (OR in classical logic) or the intersection (AND in classical logic). The result is an output fuzzy set by using (2.14), then the defuzzifier converts them to a crisp output. In this thesis, the inference system uses the AND operator for connecting inputs. The Mamdani implication (inference) [Mendel, 2001]:

$$\mu_{A^* \to B}(x^{**}, y) \equiv \mu_{A^*}(x^{**}) \bigstar \mu_B(y), \tag{2.14}$$

where $\bigstar$ is product or minimum operation, $A^* \in A_1, A_2, \ldots, A_n$, $x^{**} \in x_1, x_2, \ldots, x_n$, $\mu_{A^*}(x^{**})$ is the input MF and $\mu_B(y)$ is the consequent MF.

2.2.3.4 Defuzzifier

Once the inference system has produced output fuzzy sets, the defuzzifier in the Mamdani model converts them into crisp values. In general, there are many methods (defuzzifiers) that have been proposed in the literature for the defuzzification of type-1 fuzzy sets such as centroid (also known as centre of gravity (COG)), centre of sets (COS), height and centre of sums [Mendel, 2001]. The centroid method is one of the most commonly used defuzzification strategies. It returns the COG value and it is the one used in this thesis. The process to find the crisp output using $y_{COG}(X)$ is defined using (2.15) [Mendel, 2001]:

$$y_{COG}(X) = \frac{\sum_{i=1}^{N} y_i \mu_B(y_i)}{\sum_{i=1}^{N} \mu_B(y_i)}, \tag{2.15}$$

where $\mu_B(y_i)$ is the aggregated value of $\mu_B(y)$, N is the number of discretized points that are used to find the COG of B and y_i is the output variable. For more complete details, sources such as [Mendel, 2001],[Hellendoorn and Thomas, 1993],[Van Leekwijck and Kerre, 1999] have provided comparisons of different methods of defuzzification.

2.3 Cloud Computing

In recent years, the investment demand in the IT sector has increased and become necessary in an organisation's daily work. In such a manner, networks have expanded and become responsible for (1) the decrement of distance in the organisations, and (2) the requesting response time. Hence, the IT sector considers these two points to provide a very safe network management for their customers. For example, the IT industry invests a great deal of money to tackle the major concerns in cloud computing (CC), e.g easiness of breach, through Intrusion Detections Systems (IDS) or Intrusion Prevention Systems (IPS) [Mell and Grance, 2009]

> According to the National Institute of Standards and Technology (NIST), *they stated that the definition of network management is to involve the deployment, integration and coordination of all the hardware, software and human elements to monitor, test, poll, configure, analyse, evaluate, and control the network and element resources to meet the real-time, operational performance and quality-of-service (QoS) requirements at reasonable cost* [Cloud, 2011].

In the 20^{th} century, the idea of cloud computing was taken by J.C.R Licklider and John McCarthy in 1960 [Sumter, 2010]. They stated that CC is not a new idea as the the types of CC services have been taken off from the past and the internet was able to support high bandwidth transmissions [Anonymous, 2012, Mohamed, 2009]. At present, the technology of CC has begun to get the attention of organisation as CC plays an important role to minimise costs and to increase an organisation operational efficiency [Anonymous, 2012, Chen and Deng, 2009]. Nunez et al. [2011] claimed that CC is a technological solution that can be used to increase computer services in the age of different networks, information, and communication revolution [Nunez et al., 2011]. See fig. 2.7.

However, Cai-dong et al. [2009] argues that CC has not been standardised yet, though many researchers have conducted a number of studies in most of CC's fields [Cai-dong et al., 2009]. Although there is no doubt that CC can provide significant advantages for computers, some of these services are inconsistent with each other as this has been defined by [Popovic and Hocenski, 2010]. Also, CC is still a relatively new and emerging technology for organisations which may not help reduce the cost effectively.

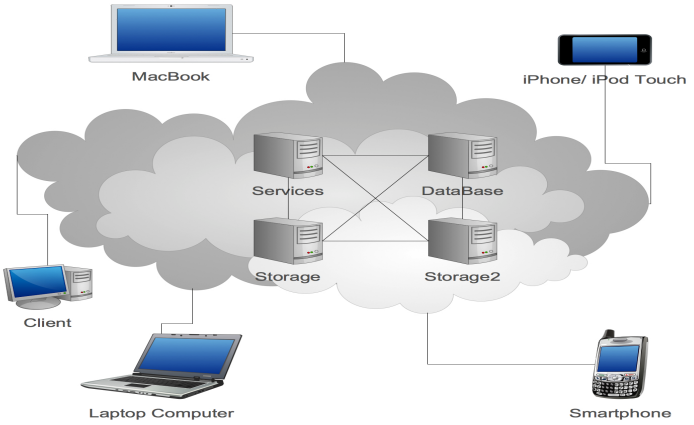


FIGURE 2.7: How Cloud Computing Works

CC has three main services that are divided into three models. These models are Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). As it can be seen in fig. 2.8, Hwang et al. [2009]) presented a map of CC models to different levels of CC's operations [Hwang et al., 2009].

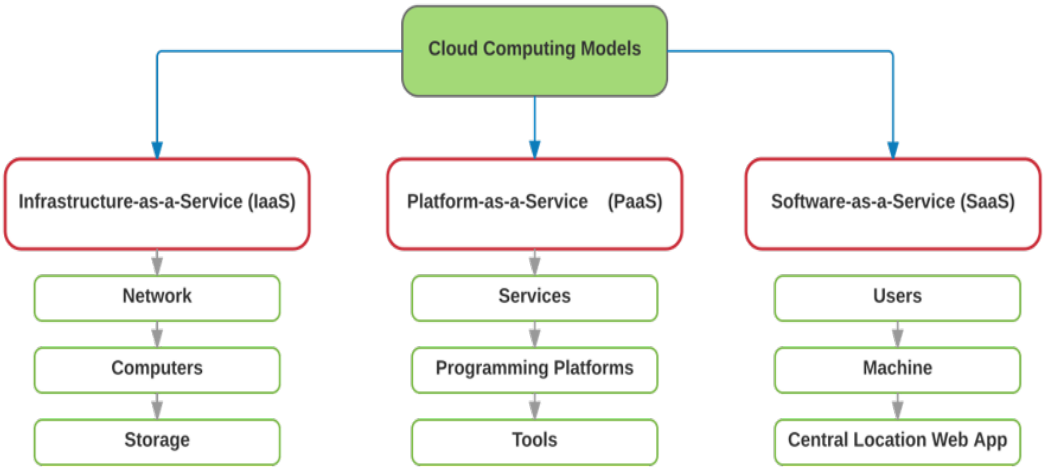


FIGURE 2.8: Cloud Computing Models
Hwang et al. [2009].

1. **Infrastructure-as-a-Service (IaaS)**:
   IaaS is considered as fundamental services provided through CC model. IaaS is the ability which provides the fundamental computing resources such as processing,

storage, network to the customers where they eligible to deploy and run applications and operating systems [Dawoud et al., 2010].

2. **Platform-as-a-Service (PaaS)**:
   PaaS is the ability which deploys fundamental computing resources onto the infrastructure of CC to the customers, where the provider supports the customer-created or acquired applications created using programming languages, libraries, services, and tools. Rake-Revelant et al. [2010] said that the typical example of PaaS is Microsoft Azure and Google App as they proposed a common model of PaaS from multiple analysis companies which were based on specific business environmental elements [Rake-Revelant et al., 2010].

3. **Software-as-a-Service (SaaS)**:
   SaaS is the ability that utilises the applications of provider, which are running on the infrastructure of CC and supplied to the customer. This service is located on servers, and it is supplied as a service. According to Gagnon et al. [2011], SaaS service is provided through the Internet which allows customer to access via any device without having unparalleled software installed and supplies a device-independent web applications with the extensions of web services [Gagnon et al., 2011].

2.3.1 Decision Making in Cloud Computing

Decision making is an important matter in the management of CC as there are current decisions that come by every now and then. For example, there are various reasons to help motivate people, enterprises and organisations to use CC. One of the most attractive factors for enterprises is elasticity which permits enterprises to scale its IT resources up, and down in a short interval of time. A number of options can be obtained once CC is used by the users, yet some key decisions may be encountered. IT outsourcing level is the 1st key decision for any user. The options are;

1. CC Infrastructure Outsourcing which is known as IaaS from Rackspace, GoGrid, Amazon etc.

2. Development platform and Infrastructure Outsourcing known as PaaS from Google App Engine, Force.com and Microsoft Azure.

3. Entire Software Outsourcing including the Infrastructure and Platform known as SaaS.

On the basis of the above options of service delivery, the consumer has various choices for practicality, preparation, information and infrastructure location as shown in table 2.1. There might be a task to optimise models to specify the simple combination of choices which can run better for the user under budget and satisfying the user. There is a set of services which are compatible to users with respect to their business. These types are Public, Private and Hybrid Cloud. The aim of offering these cloud's ownership is that the user should have some choices according to their business.

1. **Public Cloud:** As shown in fig. 2.9, public cloud literally means exchange of information publicly, yet its ownership is limited and it does not permit the users to interact with other consumers. This type of CC might be owned, managed, and operated by a business, academic, organisation or government, or some combination of them.

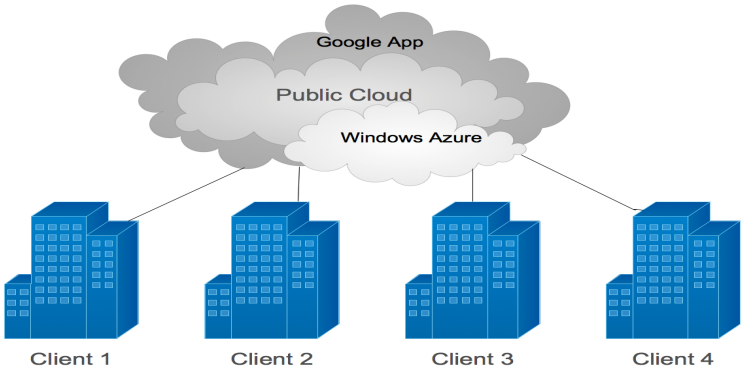


FIGURE 2.9: Public Cloud

2. **Private Cloud:** As it can be seen in fig. 2.10, private cloud is operated by the users but it is created for big firms for resource-sharing between departments of their industry. This type of CC might be owned, managed, operated by the organisation or may be run by a third party.

3. **Hybrid Cloud:** As it can be seen in fig. 2.11, hybrid cloud needs a capital expenditure as it is a combination of two or more CC types, which sustain unparalleled entities; however, they are restricted together by united or proprietary technology which enables data and possible applications.

On the basis of scale and applications operation, users can take either of the following:

1. **Commodity Cloud:** It means that if they get charged by hour, their servers will be accessed by the public.

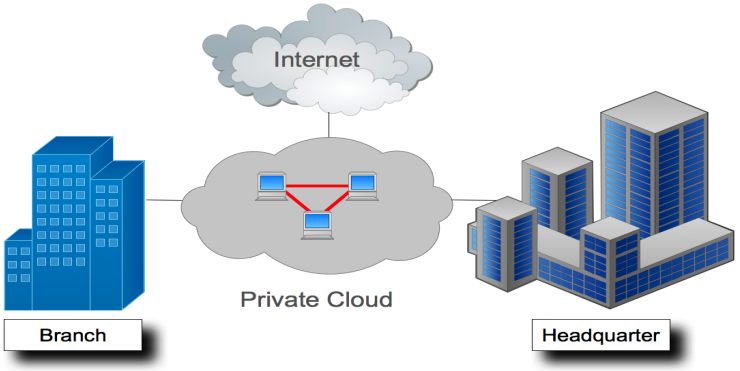


FIGURE 2.10: Private Cloud

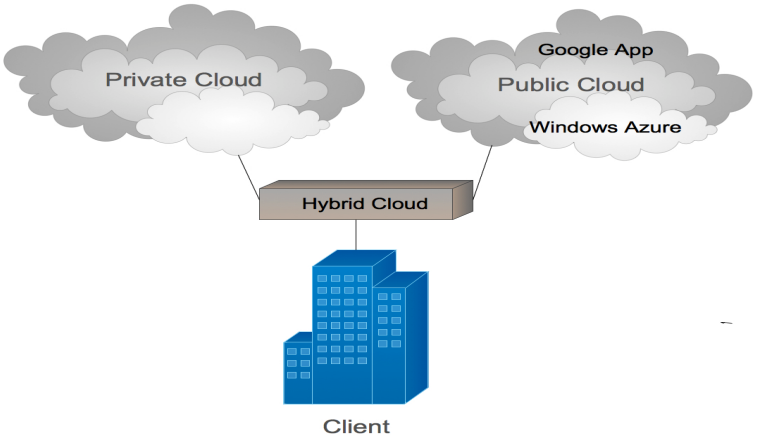


FIGURE 2.11: Hybrid Cloud

2. **Enterprise Cloud:** It means that if they are charged in a month having non-public access of servers that might not be accessible to public in general. This is chosen by the users who need high levels of service from CC.

On the basis of Infrastructure Location Operation (ILO), three choices are available to users out of which they can choose any one. These choices are:

1. **On-site** which means hardware is placed in the users information centre.
2. **Off-site** which means hardware is placed in the information centre of the provider.
3. **Co-location** which means hardware is placed in a location which is neutrally secured.

The Data Location option (DLO) and ILO are almost the same; but the DLOs decisions are highly interdependent. If a user choses private deployment of cloud, there will be no

scale/ or operation options because these are applicable to public cloud only. In addition to this, their data and infrastructure can be located only on-site or off-site. If the user chooses public deployment of cloud, then he can either choose an enterprise cloud or a commodity but data and infrastructure may get limited only for off-site locations.

TABLE 2.1: Cloud Computing Options

	Service	Development	Functionality	ILO	DLO
1	IaaS	Public Cloud	Commodity Cloud	On-Site	On-Site
2	PaaS	Private Cloud	Enterprise Cloud	Off-Site	Off-Site
3	SaaS	Public-Private Hybrid	None	Co-Location	Co-Location

2.3.2 Benefits and Challenges in Cloud Computing

New technologies affect the economy of the world in many positive ways, yet its negative factors should not be ignored. Moving the process of traditional computing into CC has a great impact on security, increase in efficiency, saving of energy, visibility, flexibility as well as community. However, the challenges that might be faced are privacy issues, cultural issues, reliability, availability and controllability. See table 2.2.

To start with CC advantages, security problems generally come when user does not have back-up and the system crashes resulting in data loss if it is not stored in some other location. CC solves these kind of issues by providing back-up. Sometimes the user may forget to take storage media to store data from home to his office. CC provides access to this data with security to store data in a data centre [Barnatt, 2010]. CC running is independent of location, and hence, another benefit of security is that CC makes the personal computing much safer. CC corporate data centres cannot securely be compromised whereas individual PC always can be [Barnatt, 2010, Catteddu, 2010].

The CC capital investment is minimal as users have multiple organisations available ensuring that competition services for customers are comparatively inexpensive. such an advantage motivates the users to only pay for what they use. Customers also do not need to worry about the updating of software, changing code or hardware capacity that are frequent issues for firms operating their own software. CC permits flexibility for users in other environment; for example, customers can access the files by using different types of devices such as smart-phones, tablets, etc [Catteddu, 2010].

Energy saving means capability of direct instrumental cooling affects the stability and speed of computers, and it is recognised well that computers take a lot of energy for this cooling if hardware has higher temperature. At this stage, vendors of CC keep their

TABLE 2.2: Benefits and Challenges in Cloud Computing

	Benefits	Challenges
1	Security In terms of Backup Information Data	Security, Privacy, and Trust Issues
2	Capital Investment Saving	Cultural Issues
3	Energy Saving	Availability
4	Increased Efficiency	Reliability
5	Flexibility	Controllability

servers in cold environment so that they can perform well. If they use liquid nitrogen method for this process, this can save the energy under some circumstances to run computers [Krutz and Vines, 2010].

Furthermore, efficiency increment in CC environment, people and organisations will likely become greater and therefore, CC provides a reliable and comprehensive set of resources for its users. CC may impose all analysis, development, deployment and monitoring to provide the opportunity to increase productivity and reduce risks in business [Catteddu, 2010, Krutz and Vines, 2010].

In the context of flexibility, CC provides the services where the customers can apply the cost-effective applications, hardware, and software. Hence, they do not have to interact with outdated hardware or software. If they still require, it can scale their software and hardware on a regular basis, thereby adding the provided flexibility [Catteddu, 2010].

As it can be see in table 2.2, there are five concerns in CC environment but they are not fixed or limited. The 1st concern is privacy, security and trust issues. Sometimes, CC does not offer certain safety mechanisms that help in tracking the servers of CC. Network and user access inside private cloud is specific and restricted; hence, the processes are handled in the organisations without any restriction of security, bandwidth and legal requirements that using the public cloud services could be needed [Catteddu, 2010, Zhang et al., 2010].

Unfortunately the public cloud is not able to constantly protect the confidentiality of data for all the users because the trust relations have temporary efficiency in cloud; and thus, the users cannot prevent the risk of receiving dangerous actions or malicious data. Additionally, the users may get the illegal data from the cloud. More dangerous is that this technology cannot completely prevent fraud, phishing, exploitation and service-hijacking as generally occurs in IT systems [Krutz and Vines, 2010].

Therefore, government have already taken action against hackers and made a legal jurisdiction where organisation takes an action to ensure the privacy rights of the users and give permission to appeal against the cloud merchants if required. Researchers have shown the method to handle these problems. They found that some firms in CC should comprehensively provide different levels of protection per degree of trust. Then these firms can manage change in trust with context to time and adjust, to monitor and to reflect properly the trust relation to become dynamic with the time [Zhang et al., 2010].

On the other end, possible protection issues with cloud return within the system. There is multi-instance in multi-tenancy virtual environments where all instances are assumed as completely isolated with one another. This can escape the limits of sandboxed environment, and provide complete access to the hosting system [Catteddu, 2010, Zhang et al., 2010].

As far as the disadvantages of cloud computing is concerned, traditional problems can also be taken as a drawback as the organisation and information need to be shared or manipulated their conventional methods of working, it might be a huge traditional challenge to adopt the CC technology [Catteddu, 2010].

2.3.3 Deploying IDS/IPS into Cloud Computing

CC is limited in terms of control the data and resources where the administrator can see IDS details. Roschke et al. [2009] proposed a solution for an IDS central management within CC. Such a solution focused on integration of sensor outputs on any single type of interface [Roschke et al., 2009].

Karen and Mell [2010] proposed a system consistency of CC application layer in different platform layers such as the system layer, the application layer and the platform layer [Karen and Mell, 2010]. IDS, in this system, generates alerts sent to the event of CC and then stores the data in the repository of the sender, receiver and helper types of plugs. Kozushko [2013] proposed a method of deployment through an analysis of complex components that are monitored by the help of a user [Kozushko, 2013].

Singh and Roy [2012] claimed that the issue of deploying IDS/IPS cloud computing lies on information confidentiality, which is received by the help of SaaS [Singh and Roy, 2012]. They said that all the various information types have to pass through the network with considering the lake of confidential information that need to be monitored. The encryption within the deployment may be considered as an effective method to tackle

the information confidentiality issue; however, packet analysis could be the best solution tackle the information confidentiality issue through the use of CC.

In recent years, there is a remarkable rise of the amount of vulnerabilities that impact the process of CC. One of the main reasons that influences the adoption process of cloud services is the distributed nature of CC service [Michael Armbrust, 2009] . Chen and Deng [2009] said that a large number of organisations deployed SaaS for IT management resources although there is a great deal of inherent issues in CC such as the security of data, the management of vulnerability, the system of disaster recovery, the process of business continuity and identity management [Chen and Deng, 2009]. Two years later, Murugan and Kuppusamy [2011] supported the say of Chen and Deng [2009] and said that a number of firms are not in a position to adopt IDS/IPS due to a plethora of ambiguities in the process of CC [Murugan and Kuppusamy, 2011]. The main challenges for deploying IDS in CC lies on security and data management, e.g. confidentiality, auditability, control over data lifecycle, privileged user Access, lack of standards and interoperability, and multi tenancy [Brodkin, 2009, Hall and Liedtka, 2011, Michael Armbrust, 2009, **?**]. In the event of virtual cloud, there are other common issues encountered in deploying CC which are as follows;

1. Difficulties in analysing huge logs in a virtual environment as IDS tools cannot differentiate between normal and threats effectively when scanning traffic [Kandukuri et al., 2009].

2. Generation of false alarms because of riggering unjustified alerts, whereby it diminishes the value and urgency of real alerts [Kretzschmar et al., 2011].

3. Low Detection Efficiency where the current IDS tools have very low efficiency in detection of all attacks in each attack class (e.g. DoS, U2L, R2L, and probe) and hence these attacks are successful in thwarting legitimate users from accessing the network resources [Lombardi and Di Pietro, 2011].

4. Finding correlation between attacks where it is hard to classify them in their classes: DoS, U2L, R2L or Probe [Ristenpart et al., 2009].

5. Difficulties in reassembling IP packets such as TCP, UDP and ICMP [Kaufman, 2010].

2.3.4 Virtual Cloud Computing (vCloud)

Cloud Computing basically means the utilisation of organised infrastructure software that is networked and the ability to professional provides assets to clients in an on-request condition. With Cloud Computing, data is for all time put away in servers on the Internet and reserved temporarily on desktop computers, note pads, handhelds, or other customer gadgets. In this design, vCloud regularly known as utility registering where clients can get to basic business applications virtually on-line from any end-client gadget , on a pay-per-use basis.

Businesses that are little and medium-sized or workgroups can utilise Cloud Computing to completely out-source their framework; however, numerous solutions today have significant issues in (1) proprietary application where stages require broad time to redevelop to function o -premise, (2) if Service-level agreement (SLAs) are not met, clients are regularly not able to move to another supplier (3) long lead times are needed to move or set up new. By outlining vCloud, they found that vCloud is adaptable and flexible, giving IT divisions an approach to expand limit or include abilities request without putting resources into new framework, and also preparing new faculty, or licensing new programs. In order to expand on-premise infrastructure and increase the capacity on demand, the present enterprises are starting to leverage the cloud computing model.

2.3.4.1 What is VMware vCloud?

VMware software promotes vCloud that has many services on the web; yet it is still similar to the meaning of Cloud Computing. To define this term precisely, vCloud is a brand and a part of a family of VMware products, with the most popular tools being on the vCloud Suite and vCloud IaaS offerings. vCloud providers are third-party companies (TPC) that offer IaaS solutions. These providers run the VMware vCloud Suite and allows thier consumers to move existing vSphere[3] virtual machines to their infrastructure clouds although TPC can use the vCloud Connector to connect the internal vSphere infrastructure with their vCloud datacenters[4]. vCloud Director also gives customers two advantages: (1) the ability to build secure private clouds that dramatically increase metacentre efficiency and business agility, and (2) it delivers CC for existing carpenters by pooling virtual infrastructure resources and delivering them to users as catalogue based services.

[3]it refers to Data Centre Server

[4]VMware houses a director of service providers at vCloud.VMware.com

2.3.4.2 The Components of vCloud

The major components are as follows:

1. **vSphere / ESXi 5.5:** The hypervisor was loaded on our physical servers.
2. **vCenter 5.5:** The centralised management console for all the vSphere hosts and virtual machines.
3. **vCloud Director:** The private cloud self-service portal.
4. **vCloud Networking and Security (vCNS):** Previously called vSphere, vCNS is where it keeps the cloud infrastructure secure.

2.3.4.3 Why vCloud?

vCloud assists enterprises get the opportunity to showcase quickly by virtualising and streamlining infrastructure, computerizing delivery of services, and giving high accessibility to both customary and new types of applications, for example, Big Data. This implies a business running vCloud reacts faster to client requests, invests more time on innovation, and is prepared for the up and coming era of applications. Three fundamental key advantages have been considered to apply vCloud which is called (M yCloud). These benefits are as per the following;

1. **Productivity:** vCloud Standardize and unite server firms with wise and policy based IT operations to considerably lessen CapEx by up to 49% and OpEx by up to 56% [Chaplot, 2015].
2. **Agility:** vCloud empowers IT to quickly arrangement framework, applications, and finish IT services, which abbreviates the marketing hours of new IT services from weeks to minutes. The outcome is increased efficiency that makes it possible for both IT and the business to concentrate on development and activities of high value [Singh, 2014].
3. **Control:** vCloud gives high accessibility to arranged relocations and business congruity/calamity recuperation through application, server, group and server firm disappointments that decreases downtime of all applications. Policy based governance and consistence observing guarantee that business standards are implemented and applications have the correct level of security. Usage metering and

costing of services of private cloud framework that is virtualised assist IT and line of entrepreneurs settle on critical cost benefit choices. The outcome is up to 30% more uptime for level 1 applications and up to 50% reserve funds of fiasco recuperation management costs [Singh, 2014].

Also, one of the reasons to create vCloud was to give an IaaS service on the highest point of vSphere. Therefore, the framework uses the protected separation of virtual machines and virtual systems given by vSphere. Moreover, vCloud Director exploits vShield to give extra networking controls that are not available in vSphere. Moreover, the vCloud Director design makes it possible for the multi-tenant detachment at the administration and conveyance layer required in a cloud domain.

2.4 Approaches for Evaluating Cloud Performance

Different methods have been developed recently in order to evaluate CC performance. These methods emphasise on different features of CC and attempt to design a paradigm to access the cloud performance. The research timeline goes back to 2009, when the CC performance was considered. For this new computing approach to be commercially successful, the capability to provide quality of service (QoS) assured services is crucial. Xiong and Perros [2009] proposed an approach to assure the QoS and analyse the CC performance in finding the relation between the following;

1. The maximum number of customers,
2. The minimum service resources, and
3. The maximum services level in an attempt to provide guaranteed services of QoS.

Stantchev [2009] also proposed an approach that emphasises on non-functional properties of individual services which may give more granulated and straightforward information [Stantchev, 2009]. QoS factors are part of non-functional run time related features of service and give one of the important research challenges in service-based computing.

Lee et al. [2009] conducted a practical evaluation on Amazon Elastic Cloud Compute (EC2) in order to to study the advantages of cloud service to its users. Such a study considered essential SaaS features such as availability, reusability, scalability, availability, customisability, pay-per-use and provider's data management. The results shows that

these SaaS features were affected by the latency sensitive processes as CC output [Lee et al., 2009].

Barker and Shenoy [2010] said that different virtual machines run different applications from independent customers which may share a physical CC server. In this study, a number of experiments were carried out in Amazon's EC2 system to measure the disk, system and network movement, and throughput fluctuations.

Iosup et al. [2011] proposed a cloud based component that was integrated with the traditional benchmark system to do an experimental evaluation of the services of CC. Three evaluation metrics in their study were set [Iosup et al., 2011];

1. Specific evaluation of cloud where multiple resource instances are acquired and released repeatedly to check whether huge clouds can avoid this problem.
2. Infrastructure agnostic evaluation where certain typical benchmarks are used with two types of workloads (Single Instance and Multiple Instance).
3. Performance metrics that use various benchmarks like Bonnie, LMBench, HPCC, Cache Bench etc.

Yang et al. [2013] proposed a fault recovery paradigm in order to calculate a CC service's performance. Such a paradigm contains two most important issues: service reliability and the service performance. To enhance the reliability of cloud service, a powerful fault tolerant technology has to be considered. Reliability of CC service was explained as how often the CC responded successfully to the request by the user. Then, service performance of CC was defined as how fast the response of CC from the request issued from the user. With the help of these methods and models, quantity of service performance can easily be evaluated and to make correct decisions on the CC such as determining the appropriate number of schedulers, and appropriate nodes choice for processing the sub-tasks, etc.

Li et al. [2012b] have made taxonomy of performance measures to evaluate the performance of commercial services of CC. The purpose of conducting such a study was that authors said that wrong and confused evaluation implementations could badly interfere and spoil the evaluation-related inclusion and interaction in context of commercial CC computing. The taxonomy was made across two dimensions: (1) performance feature that was further divided into physical property and capacity parts, and (2) experiment dimension that was divided into operational and environmental part. Fuzzy logic was

then used to evaluate the CC performance through various factors such as process, files and volumes. Such a study summarised that fuzzy logic helps the user in understanding the cloud quality easily on the desired.

Supriya et al. [2012] created a cloud trust management system through fuzzy logic. Such a model was built on three inputs: (1) performance, (2) agility and (3) financial which were given to Mamdani Fuzzy Inference system in order to generate set of values that are fed into Sugeno Fuzzy Inference system. Such a system gives the trust rating of the cloud service provider. Sugeno FIS output is the crisp value: very poor, poor, good, excellent or outstanding which helps the user to guess that how cloud is much trustworthy.

Alhamad et al. [2011] proposed e-learning application in order to evaluate the overall cloud provider trust values with the help of fuzzy logic. The proposed method of fuzzy logic in the study was using four inputs. These inputs are (1) availability, (2) scalability, (3) usability and (4) security factors for trust evaluation. This technology also includes neural network to minimise the number of generated rules of fuzzy logic.

Sethi et al. [2012] proposed a load balancing algorithm incorporated with Fuzzy logic. The purpose of this study was to address the load balancing issue of CC. This was based on the Round Robin technique of load balancing to get measurable enhancements in utilisation of resources and cloud environment availability. Factors such as speed of processor and load assigned to virtual machines were entered as an input to fuzzifier that performs the process of fuzzification and an output such as balanced load is generated.

In this thesis, author proposes Fuzzy Intrusion System in Cloud Computing (FIDSCC) model works on a virtual cloud. This model was built in an experimental lab in order to evaluate security performance concerning the uncertainties of intrusions. Such a model works on maximising the accuracy, specificity, detection rate and reducing the sensitivity and false alarms.

2.5 Related Work

Chiba et al. [2016] have listed and analysed several approaches existed for intrusion detection system in Cloud Computing such as Anomaly based IDS, Signature based IDS, Fuzzy based IDS,and Data Mining based IDS. [Chiba et al., 2016]. This section provides some studies that used the aforementioned approaches.

2.5.1 IDS Techniques in Cloud Computing

The vulnerability of Cloud Computing from a security and privacy perspective cannot work effectively and securely without using protection techniques such as IDS and IPS. [Vieira et al., 2010]. IDS/IPS in Cloud Computing can have a production of alerts which is based on the true alarms; however, false alarms are still existed in case of detection by IDS/IPS [Bakshi and Yogesh, 2010]. This is due to the fact that IDS/IPS can be judged by the degree of the identity and the lesser number of false alarms There can be a detection of intrusion patterns in Cloud Computing by the inspection of network packets through the use of signatures (pre-defined rules) and generation of alarms for system administrators [Lo et al., 2010].

There are two approaches for IDS and IPS: Anomaly Detection (AD) and Signature Detection (SD). AD is a system that detects misuse and those detectors look for any differences in activity on the network. It is based on the assumption that all of these attacks are different from any normal attack and if there is a need for identification of all the differences. These kinds of detectors are helpful in the detection of profiles that have a representation of the users, host and the other kinds of systems [Bosin et al., 2009]. These profiles are seen to be collected from the normal data over a certain period of time. This can be helpful in understanding the deviation from the criterion. There are various measures that are useful in the detection of anomalies such as threshold detection, statistical measures, rule-based measures and other kinds of measures [Karen and Mell, 2010].

Foster et al. [2009] proposed a system named Grid and Cloud Computing Intrusion Detection Systems (GCCIDS). This system was designed to cover the attacks for the host based IDS systems (HIDS) which cannot monitor intrusions. This method analyse knowledge and behaviour of intrusions that take place [Foster et al., 2009]. However, this system cannot detect any new kinds of attack nor have the creation of a database that needs to be taken into the consideration while creating the IDS.

Xin and Yun-jie [2010] argued that with the increasing popularity of a network, issues with security have become increasingly severe; and therefore,the traditional kind of intrusion and firewall systems has generally been sufficient to deal with the technology; yet there is a need to develop a new type of IPS. Such an opinion was supported by Jansen and Grance [2011] who claimed that the IPS is deemed to be an advanced combination of ID, personal firewalls and anti-viruses. The function of IPS is not only to detect the interruption of the services by an attacker, but also to take preventative action. This

should include the features such as logging off the user, the initiation of system shutdown, the process of halting the system and disabling of connections. Xin and Yun-jie [2010] also mentioned that the rational types of ID have a functioning style not unlike anti-virus software. This is an example of passive mode of data testing. Hence, if there is a detection of an attack, the prevention of attacks takes place in the traffic of data.

Waxman [2011] stated that computer network attack, also known as Cyber-Attack, refers to any unwanted or unethical activity that is intended to disturb, alter or hit someone's privacy or to steal others' important data either secretly or publically. These types of attacks are usually performed by anonymous hackers and it is very difficult to recognise the hackers or to catch them [Levy, 2010]. Cyber-attacks are performed using multiple ways such as, secretly installing spy software in the targeted systems [Runthala, 2010], secretly attempting to log in the targeted system successfully [Puzmanova and Mikhailovsky, 2014] or secretly monitoring the internet traffic of the targeted system [Garber, 2010]. Cyber-attacks include, but are not limited to Malware, Phishing, Password Attack, Denial-of-Service (DoS) Attack, Man in the Middle (MITM) Attack, Drive by Downloads, Malvertising, Rogue Software and many more [Pipyros et al., 2014].

Alqahtani et al. [2014a,b] proposed two models based on IDS/IPS called SIDSCC[5] and SIPSCC[6] in order to evaluate IDS/IPS detection and prevention once they detect/prevent the attacks within cloud computing (namely *SaaSCloud*). Theses two models were investigated separately depending on different methods of protection, levels, techniques, scenarios, and attacks. The main motivation to conduct these studies was to evaluate the efficiency of IDS/IPS within *SaaSCloud* based on three perspectives; the vulnerability detection, average time, and false negative. However, these two models need to be validated further against IDS/IPS dataset such as (DARPA, KDD, or ISCX) in order to re-evaluate and validate the functionality, working and the capability of IDS/IPS within cloud computing. These datasets generally have various known and unknown attacks with different protocols scenarios. The main deference between these mechanisms is that IDS is unlike IPS whereby IPS is considered as an extension of IDS and blocks connections or drop abnormal packets if they consist of unauthorised data.

v

[5]This Service is an Intelligent Service of Intrusion Detection System for Cloud Computing (SIDSCC)
[6]This Service is an Intelligent Service of Intrusion Prevention System for Cloud Computing (SIDSCC)

2.5.2 IDS Based On Fuzzy Logic

Klir and Yuan [1995] said that fuzzy logic has been widely used in the IDS systems that helps increase the intrusion detection rates and thus significantly strengthens the IDS systems. Fries [2009] presents a new Fuzzy-Genetics based hybrid approach that is considered to be superior then previously developed Genetic-Algorithm (GA) based approaches which do not have high capability of intrusion detection. The proposed approach adds in the GA based system, an ability to change according to the networking environment, to handle the noise and to detect intrusions in the system with significant accuracy. It is based on two major steps, including GA algorithm as an initial step to produce subset of the communication features by using traditional dimensional reduction technique and the next step as defining a set of fuzzy logic rules i.e., trapezoidal fuzzy sets that allow complete membership over all ranges. This approach has been tested by KDD Cup 1999 Dataset, and results show that the intrusion detection rate accuracy is above 90% whereas the false positive rate is below 1%.

Singh [2011] discusses a very major and most common security challenge i.e., blackhole attack, in Mobile Ad hoc Networks (MANETs) and also presents the corresponding solution by utilising the strengths of fuzzy logics. The proposed approach is comprised of four stages including, fuzzy parameter extraction where the initial parameters are extracted based on the incoming network traffic, fuzzy computation which calculates the fidelity level on the bases of extracted parameters where the fidelity level defines the intrusion level of the packet, fuzzy verification module where a decision is made that either the blackhole attack exists or not, finally an alarm module generates an alarm in case of blackhole identification. The approach has been applied on routing protocol and is simulated by varying the input parameters such as, mobility of nodes and traffic speed. It has been found that the blackhole attack detection is considerably accurate and the false detection ratio is also very low.

Tan et al. [2014] addressed the attack of Denial of Service (DoS) and proposed a fuzzy logic based intrusion detection approach to cater this attack [Tan et al., 2014]. The proposed approach leverages the fuzzy logic by applying it over an already developed IDS System with an aim to improve the detection rate of such attack whereas the IDS system is based on MCA-based DoS attack detection system. The MCA-based system works on triangle based MCA-based technique that involves the extraction of geometrical correlation of the mutually exclusive features. The proposed approach is

tested by exposing it to KDD CUP'99 data set and results indicate that the DoS attack detection rate has been considerably improved after applying fuzzy logic.

Kumar and Ramesh [2016] introduces different kinds of attacks on internet including, Probe Assaults, DoS Attacks, R2L Attacks, U2R Attacks, Checking Attacks, Dissent of Service Attack, Infiltration Attack and describes the kinds intrusion detection systems that are, grouping, example mining, information mining procedures, computerised reasoning systems and delicate registering methods. The paper also presents a fuzzy logic enabled, oddity based intrusion detection system that is developed using information mining procedures to increase the intrusion detection rate as well as accuracy. The proposed approach has been divided into four steps including, classification of preparing information where the interested information is gathered, strategy for era of fuzzy guidelines where all the fuzzy sets are generated, fuzzy choice module, where a decision is taken about the nature of incoming traffic and the last step is to find the suitable order for a test information where the final decision is taken that either the incoming packet is assaulted or not. The approach is applied in a network and tested by introducing different kinds of attacks and results shows that the assaults detection rate is very high as well as accurate.

This study discusses the SnortIDS system, its strengths and capabilities, a demonstration of Snort system using ISCX datasets [ISCX, 2012] and finally proposes a new technique to increase the SnortIDS malicious activities detection rate by utilising the strengths of fuzzy logics. ISCX datasets are basically sets of malicious activities that are offered to the IDS systems to analyse the capability of IDS systems to detect them [Shiravi et al., 2012]. If the detection rate is high and accurate, we can conclude that the IDS system is stronger enough to be used for live traffic. Several other datasets are also available for testing IDS systems such as, KDD CUP-1999, but they are not realistic [Chauhan et al., 2013].

Similarly, the statistical parameters that are used to describe the overall performance as well as the capability of the underlying IDS system are: (1) specificity and (2) Sensitivity. Specificity, commonly termed as true negative rate, is a parameter whose value represents the proportion of negatives that have been correctly identified as true. On the contrary, sensitivity, commonly termed as true positive rate, is a parameter whose value represents the proportion of positives that has been correctly identified as true. The proposed fuzzy logic based Snort system is better than previously developed Snort systems as it significantly increases the specificity and accuracy of the Snort system, and significantly decreases false alarm ratio.

2.5.3 IDS Based on Data Mining

The concept of IDS began with Anderson's seminal paper [Anderson, 1980] in which he explained a threat classification model. This model builds a protection monitoring surveillance system on the basis of AD under user behaviour. Lee et al. [2010] introduced a systematic framework that implements techniques of data mining for detecting intrusion Ėxperiments in their study, which were tested on tcpdump data and email system data, concluded that the detection models accuracy relies on ample training data and accurate feature set. Schultz et al. [2011] introduced an alternative framework in which data mining algorithms were used to train various classifiers on the group of abnormal and normal executables traffic to detect new examples.

Nadiammai and Hemalatha [2012] proposed a comparative analysis of some function-based and rule-based classifiers in predicting their efficiency on the basis of sensitivity, accuracy, time, specificity and error. Hwang et al. [2010] presented a three-tier architecture of IDS that contains black and white lists. The black list separates the known attacks from traffic and white list helps in identifying rest traffics including normal traffics, and detected anomalies. The anomalies were further analysed and classified using a SVM multi-class classifier.

Tavallaee et al. [2009] conducted a study that proposed the relevance of every attribute in KDD '99 dataset of IDS to detect every class. Subramanian et al. [2012] focused on classifying the NSL-KDD [NSL-KDD, 2012] dataset with the help of decision tree algorithms to build a model with respect to metric data and performance analysis of decision tree algorithms. Lippmann et al. [2000] also proposed a comparison between various data mining classification methods for IDS. Srinivasulu et al. [2009] suggested various data mining classification methods such Naive Bayes, CART and model of artificial neural network and assessing the performance of every classifier by means of confusion matrix.

Kalyani and Lakshmi [2012] also proposed the comparison of classification methods such as Descison Tree, Naive Bayes, PART, OneR and RBF network algorithm with the help of NSL-KDD dataset. Such a comparison mentioned the advantages that were taken by utilising NSL-KDD dataset instead of KDDCUP'99 dataset [KDDCUP99]. In this regard, Reddy et al. [2011] proposed a study of different data mining methods given for IDSs enhancement in order to support the previous comparison. Some of the machine learning algorithms such as principal component analysis for dimensionality reduction

helps reduce the memory requirements and increases the speed performance. Naive Bayes for attack classification were intensely investigated [Neethu, 2012].

2.5.4 Data Mining and Classification

Data mining which is known knowledge Discovery in Databases (KDD) is the process of extracting descriptive models from large quantity of data. Frawley et al. described data mining as "non-trivial extraction of implicit, previously unknown and potentially useful information from the large data" [Frawley et al., 1992]. Data mining generally includes five divisions of tasks known as regression, clustering, association rule learning, classification and visualisation to help perform better through prediction. Thus the prediction is the main difference and goal of data mining whereas IDS based on data mining needs less expert knowledge to label the traffic data to specify cyber-attacks instead of hand-coding rules.

Classification is a supervised data mining technique which is commonly applied on datasets. Its goal is to construct a classifier or model from classified objects to categorise unseen items as perfectly as possible. The classifier output can be shown in different forms depending on the type of classification and available information such as the form of Decision Trees or Rules. The accuracy of classification of most of the data mining algorithms is enhanced because it is tough to detect some new attacks as the changing of attack patterns from attackers.

In order to analyse the classification of cyber-attacks in our study, four different types of classification attacks were adopted. These classifiers are Decison Tree (J48), Naive Bayes (NB), OneR, and K-nearest neighbour (K-NN). Decision Tree, which is called as J48 in WEKA [environment for knowledge analysis , WEKA], is a statistical classifier that makes a decision tree from training datasets using information entropy concept. Such a classifier breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is gradually developed [Kalyani and Lakshmi, 2012]. The classifier of Naive Bayes depends on Bayes' theorem with independence assumptions between predictors [Langley et al., 2012]. Such a classifier is easy to implement with no complex iteration on parameter estimations, which makes it very beneficial for very large datasets. Despite its simplicity, such a classifier does well; and therefore, it is widely utilised and preferred due to it outperforms frequently more sophisticated classification methods.

OneR, short for One Rule, is a simple, yet accurate classification algorithm that generates one rule for each predictor in the data, then selects the rule with the smallest total error as its one rule. To create a rule for a predictor, WEKA constructs a frequency table for each predictor against the target [Subramanian et al., 2012]. Finally, K-NN is called Instance Based Knowledge (IBK) in WEKA where it represents what is learned rather than rule set inferring or decision tree by using instances. The process of instance-based learning in K-NN works once the training instance set is stored, and then the memory will be searched for the training instances [MeeraGandhi, 2010].

2.6 Summary

This chapter presents the security techniques including IDS/IPS that may improve the security performance of cloud computing. Some approaches includes anomly detection, signature detection, fuzzy based on IDS, data mining based on IDS and some classifier algorithms that are run against some public datasets such as DARPA, KDD-99 and ISCX. These methods were reviewed. Based on literature, it could be said that IDS is powerful to detect the abnormal traffic but it is limited in terms of differentiating between the wanted and unwanted alerts whereas fuzzy logic based IDS has the ability to differentiate the alerts and classify them. This is because fuzzy application based IDS is flexible and elastic to the uncertainties of intrusions in cloud computing. Although such a method is limited as it does need a human interaction to determine fuzzy sets, rules as well as the high resources consumption, this paves the way for studying a fuzzy approach based on IDS to deal with different levels, scenarios, and different attack classes in virtual cloud. Chapter 3 presents the environment and experimental design for this study as well as the investigative approach and performance metrics.

Chapter 3

Environment and Experimental Design

This chapter describes the problem and draws the experimental design of the study. It also covers the differentiation between the traditional cloud computing and virtual cloud computing that known as vCloud. In particular, a formal definition of vCloud is provided, along with a related literature review including vCloud types. Moreover, experimental settings of designing CC including configuring IDS/IPS and deploying IDS/IPS into CC are provided, together with the fundamental applications. Finally, an extensive explanations to set an experimental lab in Cloud Computing environment is given.

3.1 Experimental Lab Components

An experimental lab was set up to assure full control for resources required in all experiments. The experimental lab has the following main Components:

1. vCloud as an IaaS service called *MyCloud*
2. Four detection systems with/without FL (FL-SnortIDS, FL-SuricataIDS, SnortIDS, and SuricataIDS).
3. The Information Security Centre of Excellence Dataset (namely ISCX dataset) in order to evaluate the performance of four detection systems within *MyCloud*.

4. Fuzzy Intrusion Detection System in Cloud Computing model called FIDSCC system.

Therefore, the experimental lab had certain requirements with specified specifications. These requirements are ESXi5.5 Servers, vCentre Server, Active Directory, vShield and vCloud. These servers was run within VMwware Workstations because of the limitation in the lab resources. See table 3.1.

TABLE 3.1: Environment Lab Specifications

	Requirement	Specifications	
PC1	ESXi Server	Process	Intel(R) Core (TM) i7 CPU
		RAM	16 GB
		System Type	64-bit Operation System
		Hard Disk	1 TB
		IP Address	(192.168.1.48)
PC2	ESXi Server	Process	Intel(R) Core (TM) i7 CPU
		RAM	16 GB
		System Type	64-bit Operation System
		Hard Disk	1 TB
		IP Address	(192.168.1.49)
PC3	vCentre Server	Process	Intel(R) Core (TM) i5 CPU
		RAM	8 GB
		System Type	64-bit Operation System
		Hard Disk	1 TB
		IP Address	(192.168.1.50)
PC4	Active Directory, vShield, and vCloud	Process	Intel(R) Core (TM) i5 CPU
		RAM	16 GB
		System Type	64-bit Operation System
		Hard Disk	1 TB
		ESXi Server IP Address	(192.168.1.48)
		Active Directory	(192.168.1.51)
		vShield	(192.168.1.54)
		vCloud	(192.168.1.66)

In the experiments, we needed a piece of computer software called hypervisor. Such a software is a virtual machine monitor (VMM) that creates and runs virtual machines within *MyCloud*. A computer on which a hypervisor is running one or more virtual machines is defined as a host machine. Each virtual machine is called a guest machine as shown in fig. 3.1. There are two methods of hypervisors design: (1) Type-1: Native or Bare-Metal Hypervisors, and (2) Type-2: Hosted Hypervisors

1. **Type-1: Native or Bare-Metal Hypervisors:** These hypervisors run directly on the host's hardware to control the hardware and to manage guest operating

systems. For this reason, they are sometimes called bare metal hypervisors. A guest operating system runs as a process on the host.

2. **Type-2: Hosted Hypervisors:** These hypervisors run on a conventional operating system just as other computer programs do. Type-2 hypervisors abstract guest operating systems from the host operating system. VMware Workstation and VirtualBox are examples of type-2 hypervisors.

The experimental lab environment had the following configured servers as an infrastructure:

1. **2-ESXi Servers 5.5:** There were ESXi 5.5 servers, which were built upon VMware's ESXi 5.5 virtual server software. They are a bare metal hypervisors that include support for a variety of hardware combinations and include management via the VMware vSphere Client application, which can be installed on most any Windows-based system.

2. **1-vCenter Server 5.5:** There was a virtual data centre that assesses the status and overall health of VMware vCenter Server.

3. **1-vCloud Director 5.5:** There was a vCloud Director, which is VMware Inc.'s cloud computing management tool, that manages Infrastructure as a Service (IaaS) architectures by monitoring and controlling various cloud-computing components, such as security, virtual machine (VM) provisioning, billing and self-service access.

4. **1-vShield Manager 5.5:** There was a vShield Data Security server that protects sensitive data in the virtual and cloud infrastructure, tracking any violations.

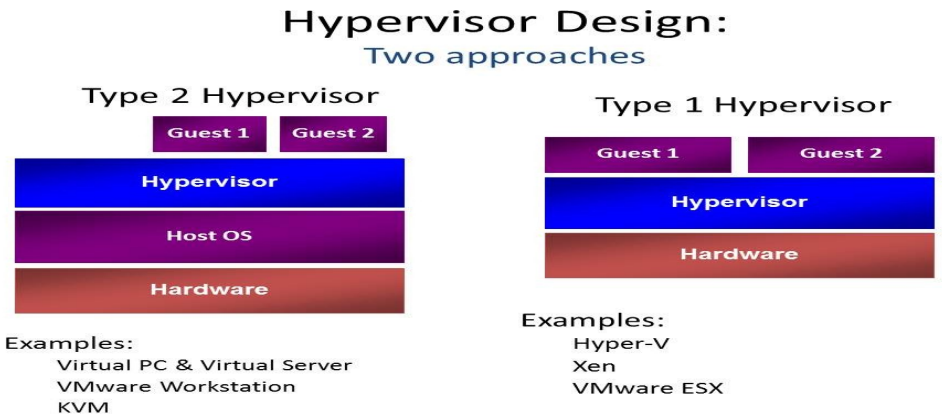


FIGURE 3.1: Approaches of Designing Hypervisors [Nunez et al., 2011]

5. **Active Directory on Win Server 2008 R2:** There was a directory service that Microsoft developed for Windows domain networks.

Then IDS/IPS Servers were deployed, Syslog Server and Attacker machine into *MyCloud*. Overall, the experimental lab had 10 virtual machines (VMs) as it is shown in table 3.2.

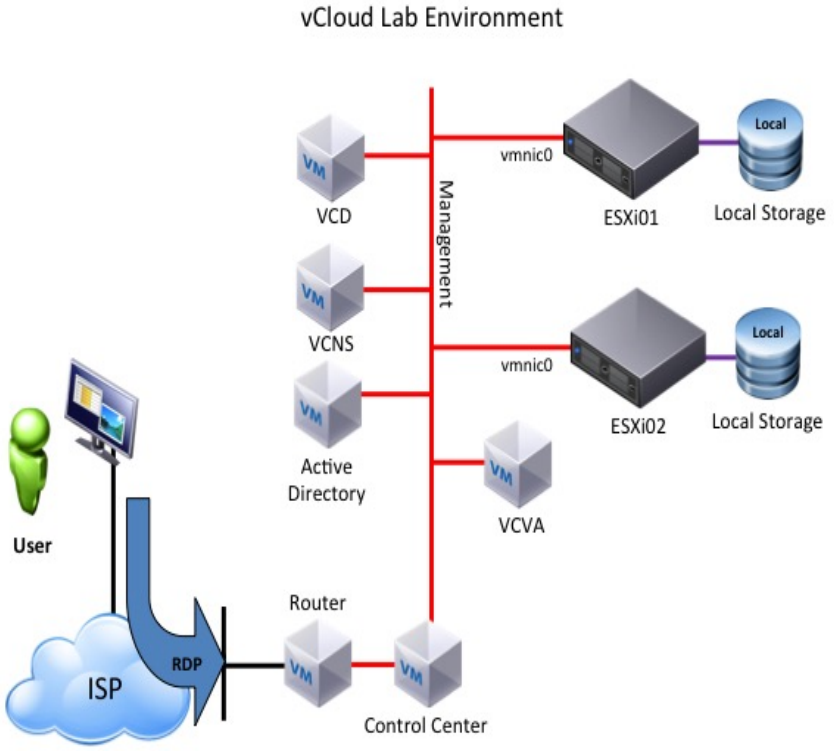


FIGURE 3.2: vCloud Lab Environment

The below information guides show how an administrator/user can connect to experimental lab environment. As shown in fig. 3.2, user can login into the lab environment remotely through RDP protocol using Remote Desktop Connection. Then Router will allow port 3389 i.e. Nating which is for RDP to do using TCP to Control Centre system (CCS). CCS is the physical system such as Laptop or Desktop with Windows 7 OS at the lab premises. From CCS, administrator was access to all the infrastructure virtual machines which is shown in the lab diagram. See fig. 3.2.

TABLE 3.2: Virtual Cloud Login Credentials

	Requirement	Specifications		Login Credentials	
PC1	ESXi Server	Process	Intel(R) Core (TM) i7 CPU	User Name	root
		RAM	16 GB		
		System Type	64-bit Operation System	Password	SaeeD1983
		Hard Disk	1 TB		
		IP Address	(192.168.1.48)		
PC2	ESXi Server	Process	Intel(R) Core (TM) i7 CPU	User Name	root
		RAM	16 GB		
		System Type	64-bit Operation System	Password	SaeeD1983
		Hard Disk	1 TB		
		IP Address	(192.168.1.49)		
PC3	vCentre Server	Process	Intel(R) Core (TM) i5 CPU	User Name	root
		RAM	8 GB		
		System Type	64-bit Operation System	Password	vmware
		Hard Disk	1 TB		
		IP Address	(192.168.1.50)		
	SSO	User Name	administrator@vsphere.local	Password	vmware
PC4	Active Directory (AD)	Process	Intel(R) Core (TM) i5 CPU	User Name	administrator
		RAM	16 GB		
		System Type	64-bit Operation System	Password	VMware1!
		Hard Disk	1 TB		
		IP Address	(192.168.1.51)		
	vShield Manager (VCINS)	Process	Intel(R) Core (TM) i5 CPU	User Name	admin
		RAM	16 GB		
		System Type	64-bit Operation System	Password	default
		Hard Disk	1 TB		
		IP Address	(192.168.1.54)		
	vCloud Director Appliance (VCD)	Process	Intel(R) Core (TM) i5 CPU	User Name	administrator
		RAM	16 GB		
		System Type	64-bit Operation System	Password	Vmware
		Hard Disk	1 TB		
		IP Address	(192.168.1.66)		
	IDS Server	Process	Intel(R) Core (TM) i5 CPU	User Name	sxa
		RAM	16 GB		
		OS Type	64-bit Operation System		
		System Type	Windows	Password	root
		Hard Disk	1 TB		
		IP Address	(192.168.1.75)		
	IPS Server	Process	Intel(R) Core (TM) i5 CPU	User Name	sxa
		System Type	16 GB		
		OS Type	64-bit Operation System		
		System Type	Windows	Password	root
		Hard Disk	1 TB		
		IP Address	(192.168.1.76)		
	Syslog Server	Process	Intel(R) Core (TM) i5 CPU	User Name	sxa
		RAM	16 GB		
		System Type	64-bit Operation System		
		OS Type	Windows	Password	Sa1983
		Hard Disk	1 TB		
		IP Address	(192.168.1.77)		
	Attacker	Process	Intel(R) Core (TM) i5 CPU	User Name	root
		RAM	16 GB		
		System Type	64-bit Operation System		
		OS Type	IinuxBackTrack 5	Password	toor
		Hard Disk	1 TB		
		IP Address	(192.168.1.78)		

3.1.1 The Information Security Centre of Excellence Dataset (ISCX)

Intrusion Detector Learning (IDL) is a software that detects network intrusions and protects a computer network from unauthorised users, or intruders. The role of IDL is to create a predictive model through public datasets such as DARPA1998, KDD1999 and ISCX2012 through classifier algorithms (CAs). CAs are capable to differentiate between normal connections and abnormal connections, e.g. intrusions or attacks [Shanmugavadivu and Nagarajan, 2012]. The 1998 DARPA Intrusion Detection Evaluation Program is a group of data which was not used because the aim of our study was to survey and evaluate the intrusion detection within Cloud Computing in a research. DARPA 1998 dataset was a benchmark that is consist of a set of data to be audited, which includes a wide types of intrusions simulated in a military network environment. The 1999 KDD[1] is a competition of intrusion detection that implements a version of this dataset [Shanmugavadivu and Nagarajan, 2012]. However, these aforementioned datasets have unrealistic data and lack of scenarios. ISCX is an Intrusion Detection Evaluation dataset that provides a set of complete traffic of real-time network, carefully acquired for the applications which include web browsing (HTTP, HTTPS) mails (SMTP, POP, IMAP), file sharing (FTP), and SSH protocols. This dataset was simulated to provide real time network traffic for IDS from which IDS can detect different anomalies in the patterns of traffic, and generate different alerts. ISCX dataset is traffic of 7 days of activity of an agent that contains these five types of traffic, that needs to be analysed,

1. Normal Traffic
2. Infiltrating Network from Inside
3. HTTP Denial of Service
4. Distributed Denial of Service
5. Brute Force SSH

This traffic is divided into 7 days of real-time traffic, each day file ranging from 4 GBs to 23.4 GBs. Now to analyse this traffic, SnortIDS was run in offline mode, which has its limits as SnortIDS cannot read a trace file greater than 200MBs which varies depending on the system. Therefore, the only option was to split the per day files into different

[1] it was published in the third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 The Fifth International Conference on Knowledge Discovery and Data Mining

small files, which then read by SnortIDS that can provide us alerts. An important feature of SnortIDS is, in a single run, can read multiple files provided in the folder while maintaining states of previous connections. This will be discussed later but up to now, ISCX Dataset was categorised with respect to dates in folder, and is split, ranging 50+files/day -450+files/day.

3.1.2 Attack Classes for Evaluation of IDSs

All IDS datasets were designed to evaluate IDS through four attack classes whereby each class has several types of attacks. These classes are as a follows;

1. **Denial of Service Attacks (DOS):** DOS is the first class of attack where an intruder makes memory or computing resources fully occupied and hard to deal with legitimate request; therefore, preventing legitimate user access to a machine [Raiyn, 2014].
2. **Remote to Local (User) Attacks (R2L):** R2L[2] attack is the second class of attack where an intruder dispatches malicious packets to a target machine through a network. Then, the vulnerability of the target machine will be exploited in order to illegally gain local access to that target machine [Raiyn, 2014].
3. **User to Root Attacks (U2R):** U2R attacks is the third class of attacks where an intruder initiates with access to a normal user account on the system and being able to take advantage of the vulnerability in order to gain root access to the system. In this case, the attacker will start up with access to a normal user account on the system through possible hacking methods such as gaining information by sniffing passwords, a dictionary attack, or social engineering, which will let the attackers to exploit several vulnerabilities to get root access to the system [Raiyn, 2014].
4. **Probing:** Probing is the fourth class of attacks where an intruder inspects the network in order to gather information or find known vulnerabilities. The attacker with a map of target machine and services can utilise the information which are available on a network in order to be noticed for exploitation [Raiyn, 2014].

[2]This attack happens when an intruder has the ability to dispatches malicious packets to a target machine over a network but does not have an account on that machine that exploits some vulnerability to gain local access as a user of that machine.

3.2 Intrusion Detection Systems (IDSs)

There is a great deal of open source IDS tools available. The use of these tools depends on the user or administrator. Some of them are for monitoring hosts and others are for the networks connecting them to identify the latest threats. The IDS systems: Snort and Suricata so-called (SnortIDS and SuricataIDS) were utilised twice (with/without FL) to create four detection systems (FL-SnortIDS, FL-SuricataIDS, SnortIDS, and SuricataIDS) suing ISCX dataset.

SnortIDS is an open source, rule based Intrusion Detection System provided by Cisco. It is now also being used as IIDS/IPS. SuricataIDS is also another open source IDS system that has been developed by a foundation i.e., Information Security Foundation (OISF). Both aforementioned IDSs are widely used around the globe making any network infrastructure safe and reliable by detecting and resisting the well-known cyber-attacks or malwares and evaluating the incoming network traffic. These IDSs makes decisions about the activities either to be regular or malicious, on the bases of some predefined rules. These rules were set by the respective community and are applied for the evaluation of incoming network traffic. With the ever growing on-line communication technologies, the network traffic has been become more and more complex day by day; hence the results obtained by applying such predefined rules and keeping track of the changes is a very tiresome effort and might become outdated up to some extent. Appendix A shows the basic working of both IDSs.

3.2.1 IDS Fuzzy Classifiers

Once SnortIDS and SuricataIDS demonstrated the experimental results against ISCX dataset, it concludes that the false detection rate is high enough that it cannot be ignored; and thus, it requires a serious attention. In order to cater this issue, IDS fuzzy classifiers were built for these IDS called FL-SnortIDS and FL-SuricataIDS. These fuzzy Logic are based IDS approaches have been presented in this section which refurnishes the alerts generated by the SnortIDS and SuricataIDS systems; and then it takes extra-cautious decisions that either the incoming traffic is actually a regular traffic or malicious. These approaches enhance the performance and accuracy of these two systems considerably. In terms of increased accuracy, specificity and sensitivity and reduced false alarms, some experiments have been presented to analyse the performance of these systems by

using ISCX dataset. The results indicate that FL-SnortIDS system outperforms FL-SuricataIDS system. Upcoming sections will focus on different processes of these fuzzy classifier systems.

3.2.1.1 Understanding Alerts of IDS Fuzzy Classifiers

The alerts generated by SnortIDS are not categorised in any manner, which may help us identify the real threats vs. alerts generated by bad network or sometimes a simple mistake in credentials that can cause an alert. Thus, these alerts need to be categorised by the types of attack they represent. The alerts generated by SuricataIDS is much like SnortIDS that is a list of long unsorted lines, which is very difficult for any network administrator to understand. Two samples are shown below, which essentially is very much like SnortIDS but formatting and attack names may vary.

```
06/11/2010-03:01:09.045867 [**] [1:2260002:1] SURICATA Applayer Detect protocol only one
direction [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP}
192.168.5.122:110 -> 192.168.2.107:4585
```

FIGURE 3.3: A Log of SnortIDS System

It is very important to learn to read the log provided by SnortIDS or SuricataIDS so we can classify and arrange them as desired. To give a typical example, the fig. 3.3 and fig. 3.4 are log samples of ISCX Dataset that guide to understand the alert.

```
[**] [129:12:1] Consecutive TCP small segments exceeding threshold [**],[Classification: Potentially
Bad Traffic] [Priority: 2] ,06/10-23:01:07.148725 192.168.5.122:22 -> 192.168.2.111:1566,TCP TTL:64
TOS:0x8 ID:23576 IpLen:20 DgmLen:72 DF,***AP*** Seq: 0x7EB71E3  Ack: 0x8E4571EB  Win:
0x1D50  TcpLen: 20,
```

FIGURE 3.4: A Log of SuricataIDS System

3.2.1.2 Alerts Classification of IDS Fuzzy Classifiers

After extensive analysis of the alert files which were generated by SnortIDS and SuricataIDS, these alerts were programmatically categorised on the basis of alert classification. This gives a very clear picture of the alerts generated by the IDSs. The alert

classified for SnortIDS and SuricataIDS are displayed in table B.1.

The table B.1 shows the meaning of IDS systems' alerts for SnortIDS and SuricataIDS. The ***Unknown Traffic*** alert in SnortIDS contains 46% of alerts where such an alert was being generated against ***HTTP_INSPECT*** rules, and its size of the transferred data was not the same as already communicated. For SuricataIDS system, the figure below shows that ***GENERIC Protocol Command Decode*** alert contains 97% of alerts. These alerts were being generated against ***HTTP_INSPECT*** and ***TCP_INSPECT*** rules, where size of transferred data was not the same as already communicated or the window size was different. There are many reasons for these alerts to be generated. It may be due to a bad network, or wrong configuration of HTTP server, but as the communication between server and client is established legitimately, so these are the alerts we can remove from the alert files of SnortIDS or SuricataIDS, as these are not the work of any intruder. It is just some server error. The table 3.3 and table 3.4 show the alerts before removing any unwanted and false alerts.

Table 3.3: Corrected Alerts Classified for SnortIDS System

Alert Types	**Count**	**Percentage**
Network Trojan	2075	0.82
Access to a Potentially Vulnerable Web Application	10	0.0039
Attempted Administrator Privilege	44887	17.87
Attempted Denial of Service	28	0.011
Attempted Information Leak	16208	6.45
Attempted User Privilege Gain	5	0.0019
Detection of a Network Activity	1	0.00039
Detection of a Network Scan	5	0.0019
Executable Code was Detected	115	0.045
Generic Protocol Command Decode	11661	4.64
Information Leak	3	0.001
Misc Activity	2011	0.80
Misc Attack	3	0.001
Not Suspicious Traffic	215	0.085
Potential Corporate Privacy Violation	9838	3.91
Potential Bad Traffic	45610	18.16
Unknown Traffic	116665	46.46
Unsuccessful User Privilege Gain	4	0.0015
Web Application Attack	1730	0.68

Potentially Bad Traffic alert generated by SnortIDS is 18% of the alerts. This alert was being generated by an FTP server that used to generate an extra reset flag to make sure the connection was terminated, a services hosted on servers such as AKAMI and

TABLE 3.4: Corrected Alerts Classified for SuricataIDS System

Alert Types	**Count**	**Percentage**
Network Trojan	2119	0.61
Access to a Potentially Vulnerable Web Application	2	0.00058
Attempted Administrator Privilege	336	0.098
Attempted Denial of Service	27	0.0078
Attempted Information Leak	220	0.064
Attempted User Privilege Gain	3	0.0008
Generic Protocol Command Decode	335570	97.93
Misc Activity	445	0.12
Misc Attack	4	0.0011
Not Suspicious Traffic	215	0.062
Potential Corporate Privacy Violation	2755	0.80
Potentially Bad Traffic	45610	18.16
Unknown Traffic	596	0.17
Unsuccessful User Privilege Gain	2	0.00058
Web Application Attack	34	0.0099
(Null)	321	0.093

such servers generate extra resets making sure that connection is terminated, where SnortIDS deals it as an unknown connection packet as SnortIDS has already removed that connection from its memory. Hence, SnortIDS classifies this alert as ***Potentially Bad Traffic***.

One more reason for the alert to be generated for SuricataIDS is an ill configure FTP server, which was generating an extra reset flag to make sure the connection was terminated, a services hosted on servers such as AKAMI will cause these issues, where SuricataIDS deals it as application error packet as SuricataIDS has already removed that connection from its memory. Hence, SuricataIDS will generate per packet threat. The table 3.3 and table 3.4 show the alerts before removing any unwanted and false alerts.

Similarly working on the alert files for both systems: SnortIDS and SuricataIDS, the following types of alerts were discarded by carefully analysis of the traffic of ISCX Dataset. This exercise is always done by network administrators when installing new IDS. The rules of IDSs was configured with respect to the traffic but SnortIDS and SuricataIDS were not a network aware IDS, hence the administrators cannot remove some rules randomly. For this reason, we used a fuzzy logic controller to carefully remove the rules. The unwanted alerts types for SnortIDS and SuricataIDS are shown in table 3.5.

These alerts for both systems: SnortIDS and SuricataIDS were generated mostly due to ill-configured services. Some alerts were being generated due to network congestion and

TABLE 3.5: Corrected Alerts Classified for IDS Systems Including Unwanted Alerts

No	Alert Types	Number of Alerts		Percentage	
		SnortIDS	SuricataIDS	SnortIDS	SuricataIDS
1	Network Trojan	2075	2119	0.82	0.61
2	Access to a Potentially Vulnerable Web Application	10	2	0.0039	0.00058
3	Attempted Administrator Privilege	44887	336	17.87	0.0098
4	Attempted Denial of Service	28	27	0.011	0.0078
5	Attempted Information Leak	16208	220	6.45	0.064
6	Attempted User Privilege Gain	5	3	0.0019	0.00087
7	Detection of a Network Activity	1	None	0.00039	None
8	Detection of a Network Scan	5	None	0.0019	None
9	Executable Code was Detected	115	None	0.045	None
10	Generic Protocol Command Decode	11661	335570	6.64	97.93
11	Information Leak	3	None	0.001195	None
12	Misc Activity	2011	445	0.80	0.12
13	Misc Attack	3	4	0.0011	0.0011
14	Not Suspicious Traffic	215	215	0.085	0.085
15	Potential Corporate Privacy Violation	9838	2755	3.91	0.80
16	Potentially Bad Traffic	45610	596	18.16	0.17
17	Unknown Traffic	116665	None	46.46	None
18	Unsuccessful User Privilege Gain	4	2	0.0015	0.00058
19	Web Application Attack	1730	34	0.68	0.0099
20	Detection of a Non-Standard Protocol or Event	1	None	0.00039	None
21	(Null)	None	321	None	0.093

drop packets. Besides these alerts, all other alerts posed a real threat to network and devices by injecting some kind of malware, or trying to access password protected files.

3.2.2 How IDS Fuzzy Classifier Works

First of all, we have a fuzzy classifier that makes the alerts generated by SnortIDS or SuricataIDS into understandable alerts. The fig. 3.5 illustrates that fuzzifier classifies the alerts into different categories. These categorised alerts are the inputs of the FL controller where on the basis of alert types; these alerts are further categorised as threat or false alerts. We have a basic minimum amount of 3 alerts per generated alert, to call it an illegal activity e.g. Network policy dictates, a user gets 3 passwords attempts per day over domain. Thus, if in case, a user mistakenly put the password wrong, an alarm is generated but it is not a threat because he is a legitimate user. If the retries count increased to 3, then the user gets blocked for that day. This means that if the total numbers of attempts to log in by a user are greater than allocated retries, it will be considered as a potential threat and will be presented on the threat screen; due to the fact that an authorised user can never miss hit the password thrice and still be unblocked. The whole process of accurate threat detection has been divided into three major stages, which are presented below:

1. Alert Classification
2. Threat Detection
3. Threat Severity

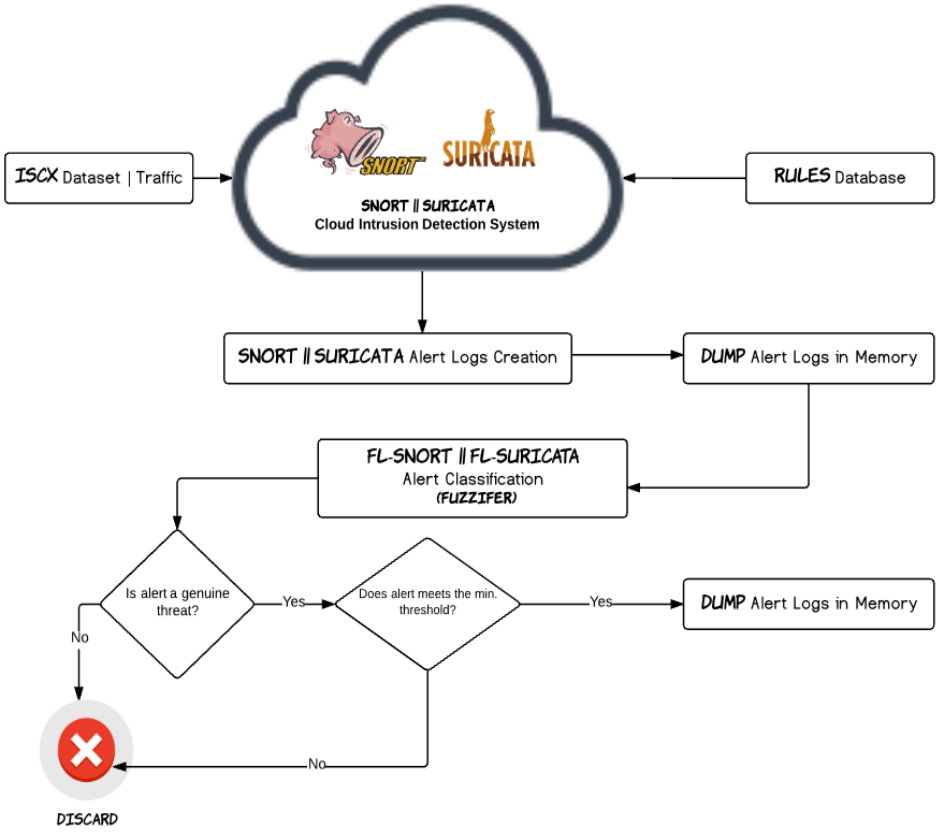


FIGURE 3.5: How FL-SnortIDS/FL-SuricataIDS Works within MyCloud

The initial stages regenerates the already generate alerts, as generated by the typical SnortIDS or SuricataIDS systems. This stage helps increase the accuracy of true threat detection and mitigates the inaccuracy of false threat detection. Afterwards, these classified alerts are passed through the threat detection engine which detects the potential threats. Finally, we checked the total number of potential threats generated against single activity such as, login. It helps us differentiate from alert and threat. For instance, if this number exceeds three, which is the predefined threshold, the potential threat is marked as a genuine threat; otherwise it is considered as a mistake and thus ignored.

3.3 Experimental Approaches

3.3.1 Methodology for The First Study

To evaluate the performance of the four systems, the author created the following metrics:

1. Numbers of threats detected (Accuracy)
2. False positives and false negatives ratio per system (False Alarms Ratio)
3. Sensitivity Ratio
4. Specificity Ratio
5. Threat Detection Rate (DR)

Accuracy of any system is determined by the ratio of true positives and true negatives detected vs. all connections; this provides us with a matrix that how accurate threats and non-threats are differentiated. It can be calculated by:

$$AccuracyRatio = \frac{(Numberofcorrectassessment)}{(Numberofallassessments)}$$

False Alarm ratio tells us how many connections are falsely categorised as threats or legitimate connections. False Alarm Ratio can be measured by the following equation:

$$FalseAlarmRatio = \frac{(Numberoffalsepositiveassessment)}{(Numberofallnegativeassessment)}$$

Sensitivity ratio tells us that our IDS detected how many threats vs. actual threats, while specificity ratio tells us our IDS treated legitimate connections as threats vs. all clean traffic. Sensitivity and specificity of a system can be measured using the following equations:

$$SensitivityRatio = \frac{(Numberoftruepositiveassessment)}{(Numberofallpositiveassessment)}$$

$$SpecificityRatio = \frac{(Numberoftruenegativeassessment)}{(Numberofallnegativeassessment)}$$

Threat detection rate is the rate of detection of threats per system, classified as low, medium, and high. In this study, our aim was to identify the performance of which of these systems: SnortIDS, SuricataIDS, FL-SnortIDS/FL-SuricataIDS is better than others. In order to do this, we set two hypotheses based on the comparison matrices above. The first hypothesis was designed for sensitivity, specificity, and accuracy while the other one was for the false alarm ratio. The first hypothesis was as a follows;

- Null Hypothesis : Performance of two methods are identical (i.e. $\mu_1 = \mu_2$).
- Alternative Hypothesis : Performance for one method significantly improves over other methods (i.e. $\mu_1 > \mu_2$).

For false alarm ratio, we set the following hypothesis;

- Null Hypothesis : False Alarm ratio of two methods are identical (i.e. $\mu_1 = \mu_2$).
- Alternative Hypothesis : False Alarm ratio for one method significantly lesser than the other methods (i.e. $\mu_1 < \mu_2$).

Our approach for testing the ISCX dataset against 4 systems was to compare the two independent results of each sensitivity, specificity, false alarm ratio and accuracy for SnortIDS vs FL-SnortIDS, SuricataIDS vs FL-SuricataIDS, SnortIDS vs SuricataIDS, SnortIDS vs FL-SuricataIDS respectively. As an essential criteria, we checked for the normality assumption with Shapiro Test for each of the category above and figure out that none of our sample data does satisfy the normality assumption, so we applied then the non-parametric test for two sample comparison for each category above against Mann-Whitney one tailed test. Mann-Whitney one tailed test has used the pairwise comparison in order to compare two population means that come from same population by using this equation.

$$U = n_1 n_2 + \frac{n_2(n_2+1)}{(2)} \sum_{i=n_1+1}^{n_2} R_i$$

where,

n_1 : sample size of sample 1

n_2 : sample size of sample 2

Ri : Rank of sample (whose rank is greater)

For detection rate, our approach was to calculate the detection rate number of threats detected vs total stream and then get them categorised in high, medium, low priority classes. In order to calculate the overall of each detection system, we then normalised the 3 steps of detection rate from 0-1. The cut-offs have been identified by Snorby[3] software. After getting these values for each detection system, we obtained a final result for each detection system. We defined the threshold for low, medium, and high as a follows;

$$\begin{aligned} 0.2 &\leqslant high \\ 0.01 < medium &< 0.2 \\ 0 \leqslant low &\leqslant 0.01 \end{aligned}$$

3.3.2 Methodology for The Second Study

We used three metrics of the first study (Accuracy, Sensitivity and Specificity) plus Precision and F-measure. Initially, experiments have been conducted based on several classifiers available in WEKA. Then we decided to select the top four classifiers among these classifiers, which outperformed better based on our performance metrics. Our ultimate goal was to evaluate the performance of classification algorithms for attack classification within Cloud Computing. In order for the detection algorithm to map the incoming events to attacks and normal activities, several performance metrics have been set for all results obtained from ISCX datasets. The Precision and F-measure are described as a follows;

1. **Precision:** It is defined as the ratio of elements correctly classified as positive out all the elements of algorithms classified as positive. In other word, it is the proportion of instances that are correctly classified as a positive class divided by

[3]Snorby is a modern web interface for Network Security Monitoring which also provides software based intrusion prevention system: Required packages Snort.

the total instances classified to positive class.

$$Precision = \frac{(TP)}{(TP+FN)}$$

2. **F-measure=F-Score:** The F-measure can be used as a single measure of performance of the test for the positive class. The F-measure is the harmonic mean of precision and recall:

$$F - measure = 2 * \frac{(Precision*Sensitivity)}{(Precision+Sensitivity)}$$

Once the results were compiled, they were compared on the basis of the following factors; accuracy, incorrect classification, mean absolute error, false positive rate, precision, sensitivity, specificity and ROC Area[4]. The classifier algorithms that were applied on the results for SnortIDS, FL-SnortIDS, SuricataIDS, and FL-SuricataIDS systems are Decision Tree, Naive Bayes, OneR, and K-NN. Then, the final results for all the attack classes of these four results were compared further on the basis of these final matrices:

1. Accuracy
2. Precision
3. Sensitivity
4. Specificity
5. F-measure

In this study, our aim was to identify the performance of which of these systems: SnortIDS, SuricataIDS, FL-SnortIDS and FL-SuricataIDS is better than others and which classifier algorithm outperforms others and presents better results. Our approach for testing the ISCX dataset against four results obtained from Cloud Computing systems was to compare the four independent results of each factor of our performance metrics in two stages. The first stage was to rank each class of attack in each result and compare based on our performance metrics. The second stage was a comparison for the following systems respectively:

[4]ROC Area is the weighted average classes under the ROC Area that have False Positive in the $x - axis$ and True Positive in the $y - axis$ (the weighted average classes under the Roc Area that have False Positive in the $x - axis$ and True Positive in the $y - axis$).

1. SnortIDS vs FL-SnortIDS
2. SuricataIDS vs FL-SuricataIDS
3. SnortIDS vs SuricataIDS
4. SnortIDS vs FL-SuricataIDS

3.3.3 Methodology for The Third Study

Final results for all these systems were compiled and the comparison of these systems was done on the basis of these matrices:

1. Threat Detection Rate (DR)
2. False Positive Ratio (FPR)

Threat detection rate is the rate of detection of threats per system, classified as ($low, medium$, *and* $high$). In this study, our aim was to identify which threat is best determined by which method and overall which method is best to identify the threat. In order to do this, we set two hypotheses based on the comparison matrices above. In order to achieve the goal of this study, we set the following hypothesis;

- ***Null Hypothesis:*** Threat detected by both the methods are same (i.e. $p_1 = p_2$ *or* $p_1 - p_2 = 0$).
- ***Alternative Hypothesis:*** Threat detected by method 1 is significantly better than the method 2 i.e. False positive ratio for method 1 is significantly lesser than the False positive ratio of method 2 (i.e. $p_1 < p_2$ *or* $p_1 - p_2 < 0$).

Our approach for testing the ISCX dataset against 4 systems within *MyCloud* is to compare the two independent results for SnortIDS vs FL-SnortIDS, SuricataIDS vs FL-SuricataIDS, SnortIDS vs SuricataIDS, SnortIDS vs FL-SuricataIDS respectively. As an essential criteria, we have used one tailed test of proportion for two samples to figure out the threat detected by which of the two methods are significantly better based on false positive ratio. The one tailed test for two proportions was used to compare the proportion of two independent samples.

$$Z = \frac{p_1 - p_2}{\sqrt{p(1-p)\left(\frac{1}{n_1}\frac{1}{n_2}\right)}}$$

where,

p_1 : proportion for sample 1

p_2 : proportion for sample 2

n_1 : size of sample 1

n_2 : size of sample 2

p : pooled proportion

$$p = \left(\frac{p_1 n_1 + p_2 n_2}{n_1 + n_2}\right) \tag{3.1}$$

For detection rate, our approach was to calculate the detection rate number of threats detected vs total stream and then get them categorised in $(high, medium, low)$ priority classes. This will be calculated overall of each system. We then normalised the 3 steps of detection rate from 0-1. After getting these values for each system, we obtained a final result for each system. We defined the threshold for $(law, medium, and\ high)$ as a follows;

$$\begin{aligned} 0.2 &\leqslant high \\ 0.01 < medium &< 0.2 \\ 0 &\leqslant low \leqslant 0.01 \end{aligned}$$

3.4 Summary

This chapter explains the lab settings, configuration, and deployment of IDS into Cloud Computing. This chapter also provides details of the configuration of ISCX daatset and how fuzzy logic based IDS works into virtual Cloud. ISCX benchmark were used with IDS but ut was not used with two IDSs (SnortIDS and SuricataIDS) and fuzzy classifiers (FL-SnortIDS and SuricataIDS) [Shiravi et al., 2012]. This leads the author to propose a model called FIDSCC system that relies on fuzzy logic based on IDS. This dataset was classified into four classes: DoS, U2L, R2L, and Probe. This way may boost the

security within Cloud Computing and could be a solution to secure Cloud Computing and reduce the false alarms and sensitivity before implementing it on an actual Cloud Computing. Chapter 4 illustrates a compartive copmarison between these four detection systems (SnortIDS, SuricataIDS, FL-SnortIDS, and FL-SuricataIDS).

Chapter 4

A Comparative Analysis for The Alerts of IDS Fuzzy Classifiers Approaches within Cloud Computing

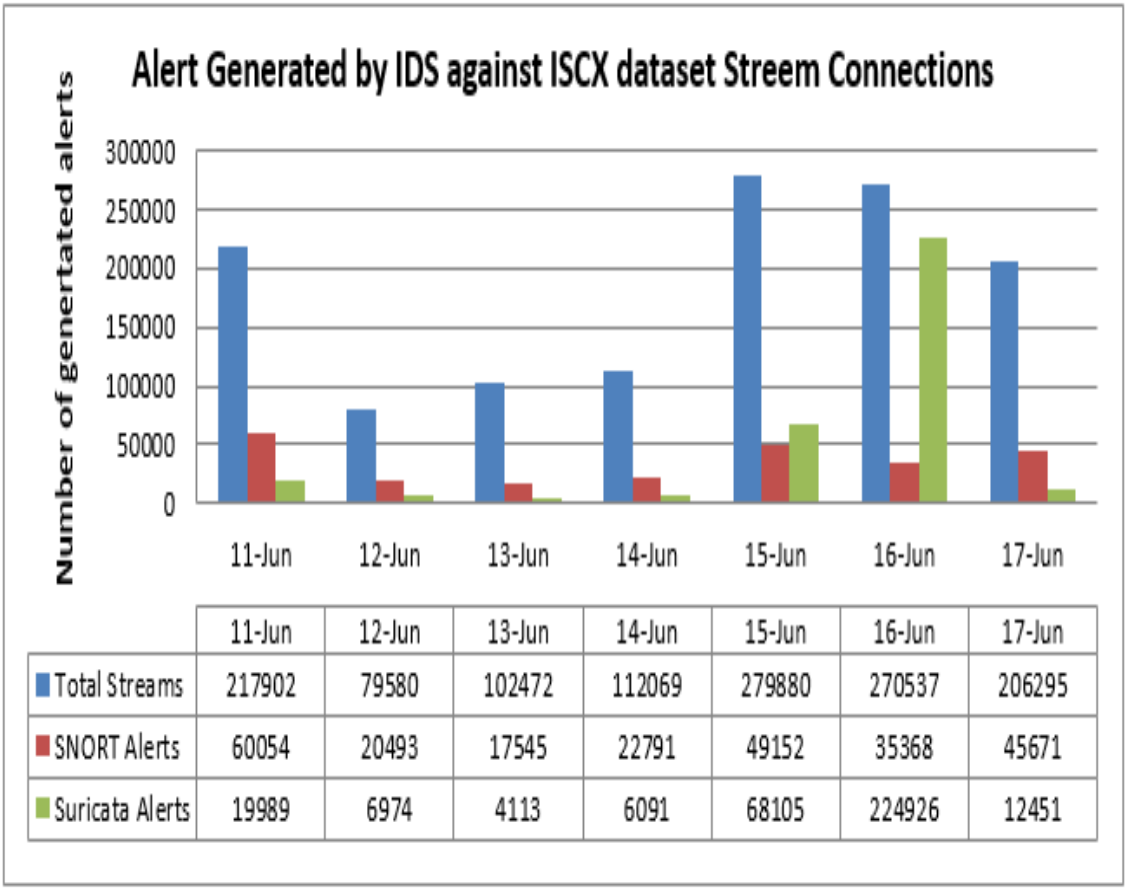
In this chapter, a comparative analysis for IDS fuzzy classifiers within *MyCloud* against the alerts of IDS is done. This chapter also covers the specific metrics for this study together with related work created and proposed by the author. The investigative approach in this chapter includes IDS fuzzy classifier, and how IDS fuzzy classifier studies. This chapter provides experimental results that includes the methodology, and statistics for all compared systems. Finally, the comparative analysis is done based on the results of four systems: SnortIDS, SuricataIDS, FL-SnortIDS, and FL-SuricataIDS.

4.1 Experimental Results and Analysis

4.1.1 Descriptive Statistics

4.1.1.1 Intrusion Detection Systems

As it can be seen in fig. 4.1 for the overall results of both IDS Systems: SnortIDS and SuricataIDS, it shows that IDS systems analysed the total of 1,268,735 connection streams of the ISCX Dataset, out of which IDS system generated 251,074 alerts connections for SnortIDS and 342,649 alerts connections for SuricataIDS. These alerts for both systems contain malicious or anomaly alerts. The numbers show that SnortIDS classifies 19.78% of traffic as malicious while SuricataIDS classifies 27.01% of traffic as malicious.



	11-Jun	12-Jun	13-Jun	14-Jun	15-Jun	16-Jun	17-Jun
Total Streams	217902	79580	102472	112069	279880	270537	206295
SNORT Alerts	60054	20493	17545	22791	49152	35368	45671
Suricata Alerts	19989	6974	4113	6091	68105	224926	12451

FIGURE 4.1: IDS Systems' Analysis on *MyCloud*

The total generated alert types for these IDS systems were 19 alert types for SnortIDS and 15 alert types of SuricataIDS. The alerts then were classified into 203 attack classifications for SnortIDS and 152 attack classifications for SuricataIDS. Based on these classifications, attacks were prioritised based on its priority. This priority shows how dangerous this attack can be for the network 1 being highest and 3 being the lowest. We went a step further to categories these alerts for each system into four attack classes that are DoS, Probe, U2R and R2L. The fig. 4.2 shows the number of these attack classes for Snort-IDS and SuricataIDS.

In the event of SuricataIDS system, the attack classes were not much alert classifications of DoS, R2L and U2R comparing to the class of probe. This is because of the fact that Probe class focuses on any network anomalies meaning at network layer level and transmission layer, while the other three are application level classifications. This means any alert generated by network may it be at IP (network) layer or TCP and UDP (Transmission) layer is classified as network probing.

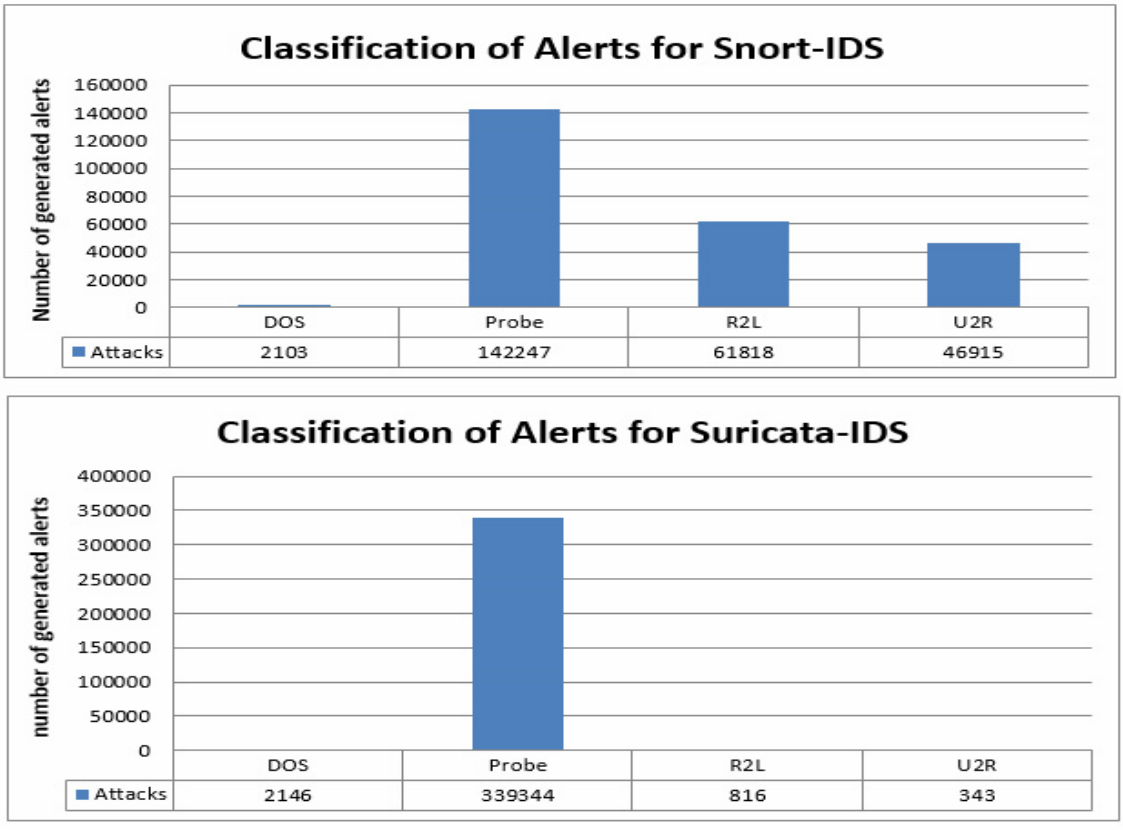


FIGURE 4.2: Classification of Alerts for IDS Systems on *MyCloud*

The table 4.3 shows the ratio analysis for these IDS systems in terms of sensitivity, specificity, accuracy and false alarms. The detection rate tells the network administrator that at what rate the alerts are generated greater. The detection rate means the higher numbers of alerts are generated. In both the cases on average more than 20% of traffic is marked malicious, and generating a high detection rate.

4.1.1.2 IDS Fuzzy Classifiers

As it can be shown in fig. 4.3, we removed, after classifying the alerts, any unwanted alerts caused by network health, unwanted privacy rules and server issues. We then set a threshold of 3 alerts per day for any alert to be classified as threat, to avoid any legitimate used in password entry.

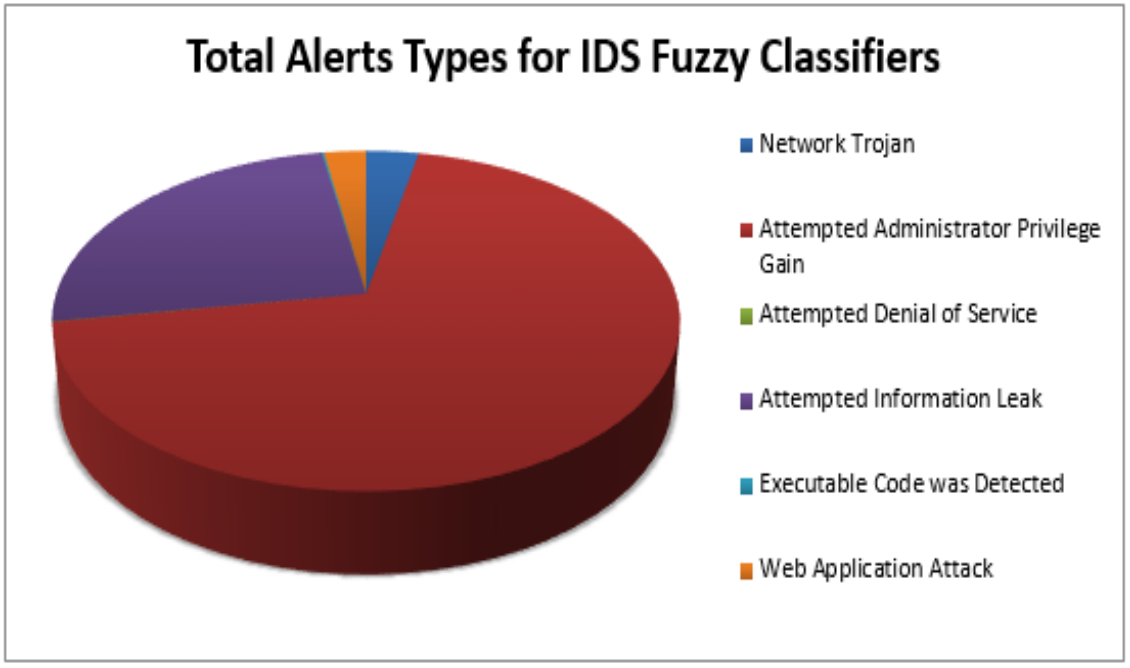


FIGURE 4.3: Total Alerts' Types for IDS Fuzzy Classifiers

As it can be seen in fig. 4.4, it illustrates the overall results for all systems: SnortIDS, SuricataIDS, FL-SnortIDS and FL-SuricataIDS. Noting that fig. 4.1 shows that IDS systems analysed the total of 1,268,735 connection streams of the ISCX Dataset, out of which Snort generated 65,066 alerts connections for FL-SnortIDS and 2,743 alerts connections for FL-SuricataIDS. These alerts for both systems contain malicious or anomaly alerts. The numbers show that SnortIDS classifies 19.78% of traffic as malicious

TABLE 4.1: Overall Descriptive Statistic for IDS Systems within *MyCloud*

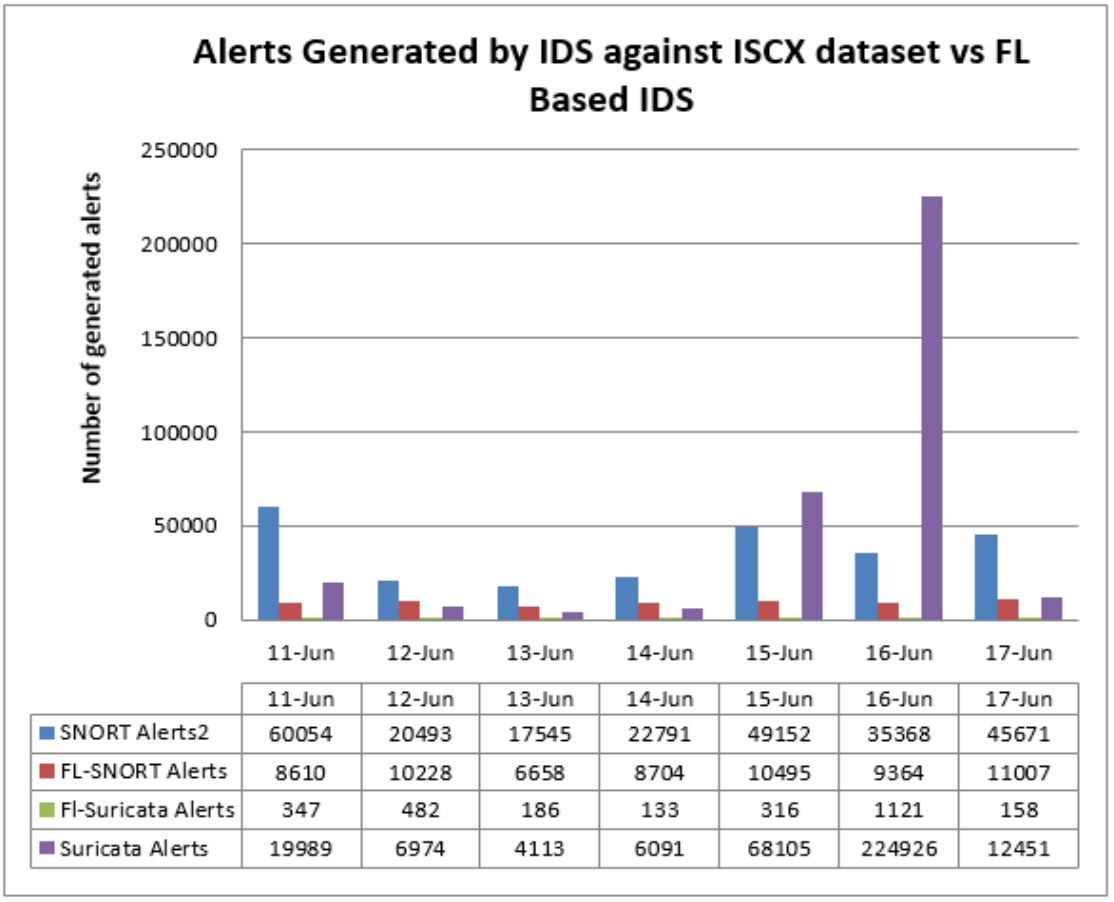
Snort-IDS Ratio Analysis

ISCX Dataset Date	Total Connections	Positive Alerts	Sensitivity		Specificity		Sensitivity Ratio	False Positive Ratio	Specificity Ratio	Accuracy Ratio	Threat Detection Rate
			TP	FP	FN	TN					
11	144,545	60,054	8,610	51,444	145	135,790	0.98	0.27	0.72	0.73	High
12	48,737	20,493	10,228	10,265	49	38,460	0.99	0.21	0.78	0.82	High
13	74,931	17,545	6,658	10,887	75	68,198	0.98	0.13	0.86	0.87	High
14	75,276	22,791	8,704	14,087	76	66,496	0.99	0.17	0.82	0.84	High
15	185,999	49,152	10,495	38,657	186	175,318	0.98	0.18	0.81	0.82	High
16	186,937	35,368	9,364	26,004	187	177,386	0.98	0.12	0.87	0.88	High
17	140,306	45,671	11,007	34,664	141	129,158	0.98	0.21	0.78	0.80	High
Total Stats	**856,731**	**251,074**	**65,066**	**1860,08**	**859**	**790,806**	**0.98**	**0.19**	**0.81**	**0.82**	**High**

Suricata-IDS Ratio Analysis

ISCX Dataset Date	Total Connections	Positive Alerts	Sensitivity		Specificity		Sensitivity Ratio	False Positive Ratio	Specificity Ratio	Accuracy Ratio	Threat Detection Rate
			TP	FP	FN	TN					
11	144,545	19,989	347	19,642	145	144,053	0.70	0.11	0.88	0.87	High
12	48,737	6,974	482	6,492	49	48,206	0.90	0.11	0.88	0.88	High
13	74,931	4,113	186	3,927	75	74,670	0.71	0.05	0.95	0.95	High
14	75,276	6,091	133	5,958	76	75,067	0.63	0.07	0.92	0.92	High
15	185,999	68,105	316	67,789	186	185,497	0.63	0.27	0.73	0.73	High
16	186,937	224,926	1,121	223,805	187	185,629	0.85	0.55	0.45	0.45	High
17	140,306	12,451	158	12,293	141	140,007	0.52	0.08	0.91	0.91	High
Total Stats	**856,731**	**342,649**	**2,743**	**339,906**	**859**	**853,129**	**0.76**	**0.28**	**0.71**	**0.71**	**High**

while SuricataIDS classifies 27.01% of traffic as malicious, while in case of FL-SnortIDS these numbers reduces to 5% and when FL is applied on SuricataIDS this number is less than 1%.



	11-Jun	12-Jun	13-Jun	14-Jun	15-Jun	16-Jun	17-Jun
SNORT Alerts2	60054	20493	17545	22791	49152	35368	45671
FL-SNORT Alerts	8610	10228	6658	8704	10495	9364	11007
Fl-Suricata Alerts	347	482	186	133	316	1121	158
Suricata Alerts	19989	6974	4113	6091	68105	224926	12451

FIGURE 4.4: *MyCloud* Corrected Alerts Generated by IDS vs IDS Fuzzy Classifiers

The fig. 4.5 illustrates the total generated alert types for these fuzzy classifiers. They were 5 alert types for FL-SnortIDS and 4 alert types of FL-SuricataIDS. The alerts then were classified into 77 attack classifications for FL-SnortIDS and 46 attack classifications for FL-SuricataIDS. Based on these classifications, attacks were prioritised based on its severity 1 as a high, 2 as a medium and 3 as a low. We then categorised these alerts classifications for each system into four attack classes that are DoS, Probe, U2R and R2L. The figure below shows the number of these attack classes for FL-SnortIDS. More alerts of FL-SuricataIDS attack classes originally were generated by Probe classifier where they were reduced to only 34 application level threats.

The table 4.2. shows the ratio analysis for these IDS systems in terms of sensitivity, specificity, accuracy and false alarms. The detection rate varies for both systems as in FL-SuricataIDS system was less than 1% of rate was recorded on average and in case of FL-SnortIDS on average less than 15% threat detection rate was detected.

4.1.2 Overall Approaches' Descriptive Statistics

The fig. 4.6 shows the ratio analysis for these four systems in terms of sensitivity, specificity, accuracy and false alarm. The detection rate tells the network administrator that at what rate the alerts are generated the greater the detection rate means the higher

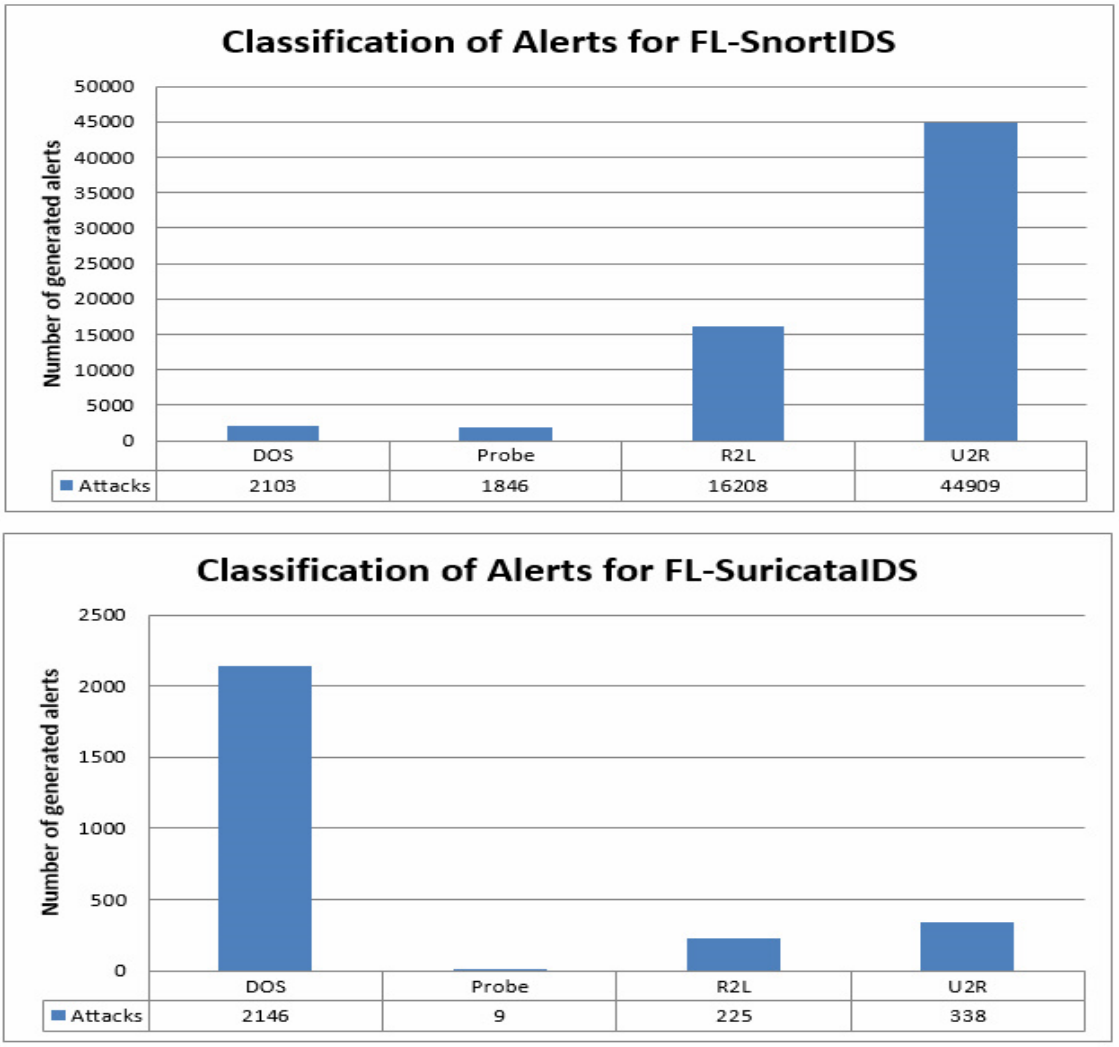


FIGURE 4.5: Classification of Alerts for IDS Fuzzy Classifiers within *MyCloud*

TABLE 4.2: Overall Descriptive Statistic for IDS Fuzzy Classifiers within *MyCloud*

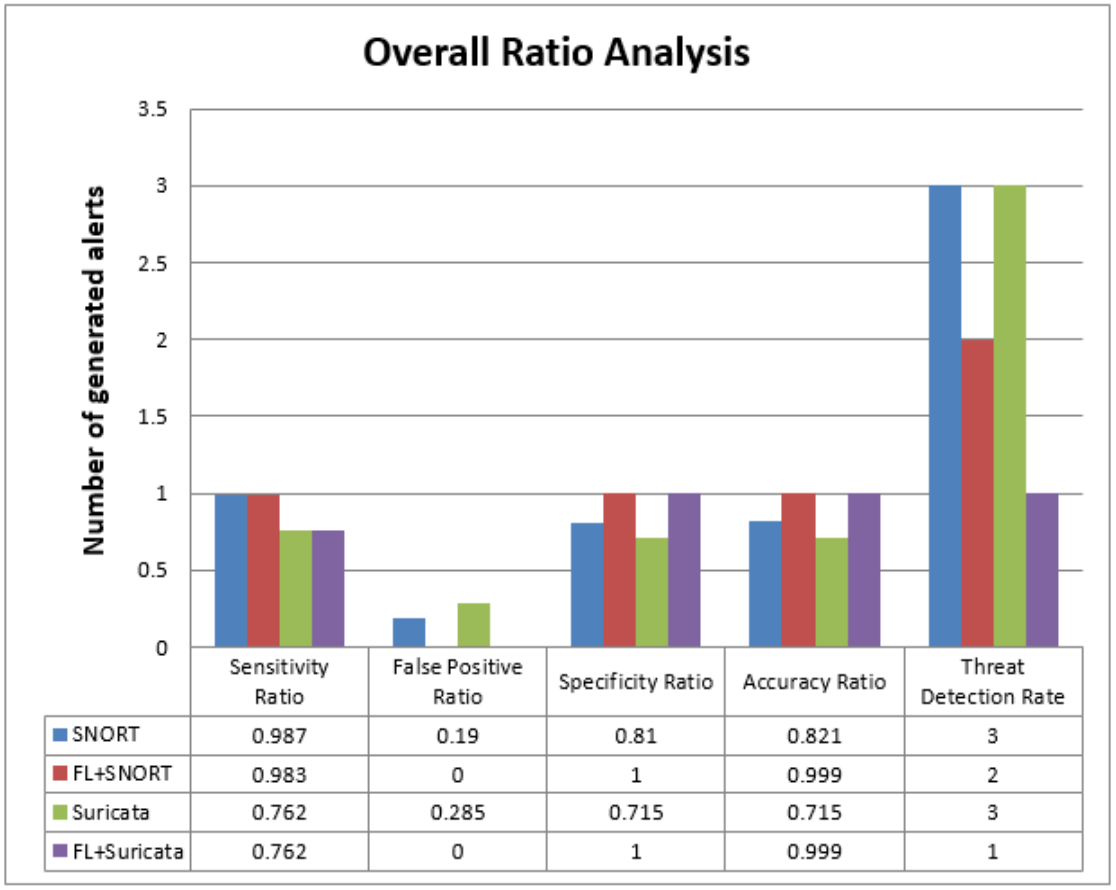
FL-SnortIDS Ratio Analysis

ISCX Dataset Date	Total Connections	Positive Alerts	Sensitivity		Specificity		Sensitivity Ratio	False Positive Ratio	Specificity Ratio	Accuracy Ratio	Threat Detection Rate
			TP	FP	FN	TN					
11	144,545	8,594	8,594	8,594	161	135,790	0.98	0.00006	0.99	0.99	Medium
12	48,737	10,217	10,217	10,217	60	38,460	0.99	0.00026	0.99	0.99	Medium
13	74,931	6,543	6,543	6,543	190	68,198	0.97	0.00009	0.99	0.99	Medium
14	75,276	8,672	8,672	8,672	108	66,496	0.98	0.00013	0.99	0.99	Medium
15	185,999	10,468	10,468	10,468	213	175,318	0.98	0.00005	0.99	0.99	Medium
16	186,937	9,336	9,336	9,336	215	177,386	0.97	0.00005	0.99	0.99	Medium
17	140,306	10,991	10,991	10,991	157	129,158	0.98	0.00008	0.99	0.99	Medium
Total Stats	**856,731**	**64,821**	**64,821**	**64,821**	**1,104**	**790,806**	**0.98**	**0.00008**	**0.99**	**0.99**	**Medium**

FL-SuricataIDS Ratio Analysis

ISCX Dataset Date	Total Connections	Positive Alerts	Sensitivity		Specificity		Sensitivity Ratio	False Positive Ratio	Specificity Ratio	Accuracy Ratio	Threat Detection Rate
			TP	FP	FN	TN					
11	144,545	347	347	1	145	144,053	0.70	0.00000	0.99	0.99	Low
12	48,737	482	482	1	49	48,206	0.90	0.00002	0.99	0.99	Low
13	74,931	186	186	1	75	74,670	0.71	0.00001	0.99	0.99	Low
14	75,276	133	133	1	76	75,067	0.63	0.00001	0.99	0.99	Low
15	185,999	316	316	1	186	185,497	0.62	0.00000	0.99	0.99	Low
16	186,937	1,121	1,121	2	187	185,629	0.86	0.00001	0.99	0.99	Low
17	140,306	158	158	1	141	140,007	0.53	0.00000	0.99	0.99	Low
Total Stats	**856,731**	**2,743**	**2,743**	**8**	**859**	**853,129**	**0.76**	**0.00001**	**0.99**	**0.99**	**Low**

numbers of alerts are generated. In both the cases on average more than 20% of traffic is marked malicious, generating a high detection rate.



	Sensitivity Ratio	False Positive Ratio	Specificity Ratio	Accuracy Ratio	Threat Detection Rate
SNORT	0.987	0.19	0.81	0.821	3
FL+SNORT	0.983	0	1	0.999	2
Suricata	0.762	0.285	0.715	0.715	3
FL+Suricata	0.762	0	1	0.999	1

FIGURE 4.6: Overall Ratio Analysis on *MyCloud*

The table 4.3 is the overall descriptive statistics for all generated alerts of each system implemented: SnortIDS, SuricataIDS, FL-SnortIDS and FL-SuricataIDS.

4.2 Comparative Analysis

Based on the experimental *MyCloud* datasets, we have conducted 5 comparisons: SnortIDS vs FL-SnortIDS, SuricataIDS vs FL-SuricataIDS, SnortIDS vs SuricataIDS, SnortIDS vs FL-SuricataIDS, and SnortIDS vs SuricataIDS vs FL-SnortIDs vs FL-SuricataIDS respectively.

TABLE 4.3: Overall Statistical Analysis for All Approaches within *MyCloud*

Systems	criteria	Minimum	Maximum	Median	Mean	Standard Deviation
SnortIDS	Sensitivity	0.9804	0.9952	0.9874	0.9870	0.0053
	Specificity	0.7252	0.8721	0.8193	0.8117	0.05
	False Alarm	0.1279	0.2748	0.1807	0.1883	0.0499
	Accuracy	0.7368	0.8770	0.8271	0.8258	0.0475
	Detection Rate			High		
FL-SnortIDS	Sensitivity	0.9718	0.9942	0.9816	0.9827	0.0073
	Specificity	0.9997	0.9999	0.9999	0.9999	0.0001
	False Alarm	0.0001	0.0003	0.0001	0.0001	0.0001
	Accuracy	0.9974	0.9988	0.9988	0.9985	0.0005
	Detection Rate			Medium		

Systems	criteria	Minimum	Maximum	Median	Mean	Standard Deviation
Suricata-IDS	Sensitivity	0.5284	0.9077	0.7053	0.711	0.1327
	Specificity	0.4534	0.95	0.8813	0.8204	0.1768
	False Alarm	0.0499	0.5466	0.1187	0.1796	0.1768
	Accuracy	0.4547	0.9493	0.8816	0.8202	0.1761
	Detection Rate			High		
FL-SuricataIDS	Sensitivity	0.5284	0.9077	0.7053	0.7110	0.1327
	Specificity	1	1	1	1	0.0000
	False Alarm	0.0000	0.0000	0.0000	0.0000	0.0000
	Accuracy	0.999	0.999	0.999	0.999	0.0000
	Detection Rate			Medium		

4.2.1 SnortIDS vs FL-SnortIDS

The fig. 4.7 states the alternative hypothesis as the true location shift was greater than 0. In sensitivity, the level of significance was greater than 0.5 ($p - value > 0.05$). Hence, we do not have sufficient evidence to reject our null hypothesis i.e. sensitivity performances on both methods are the same. This is can be clearly seen from the box-plot visualisation as well as Mann-Whitney test that there is no difference in performance of sensitivity between two methods. For specificity and accuracy, the level of confidence was $p - value < 0.05$, and therefore, we have sufficient evidence to reject our null hypothesis i.e. specificity and accuracy performances for SnortIDS is better than FL-SnortIDS. For false alarm performance, the true location shift is less than 0 and the level of confidence was $p - value < 0.05$. Therefore, we have sufficient evidence to reject our null hypothesis i.e. false alarm ratio for FL-SnortIDS is lesser than the SnortIDS.

4.2.2 SuricataIDS vs FL-SuircataIDS

The fig. 4.8 states the alternative hypothesis as the true location shift was greater than 0. In sensitivity, $p-value > 0.05$, hence we do not have sufficient evidence to reject our null hypothesis i.e. sensitivity performances on both methods are identical. The box-plot visualisation and Mann-Whitney test show that there is no difference in performance of sensitivity between two methods. For specificity and accuracy, the level of confidence was $p-value < 0.05$, and therefore, we have sufficient evidence to reject our null hypothesis i.e. specificity and accuracy performances for FL-SnortIDS is better than SnortIDS. For false alarm performance, the true location shift is less than 0 and the level of confidence was $p-value < 0.05$. Therefore, we have sufficient evidence to reject our null hypothesis i.e. false alarm ratio for FL-SuricataIDS is lesser than the SuricataIDS.

4.2.3 SnortIDS vs SuircataIDS

The fig. 4.9 states the alternative hypothesis as the true location shift was greater than 0. In sensitivity, $p-value < 0.05$, hence we have sufficient evidence to reject our null hypothesis i.e. sensitivity performance for SnortIDS is better than the SuricataIDS.

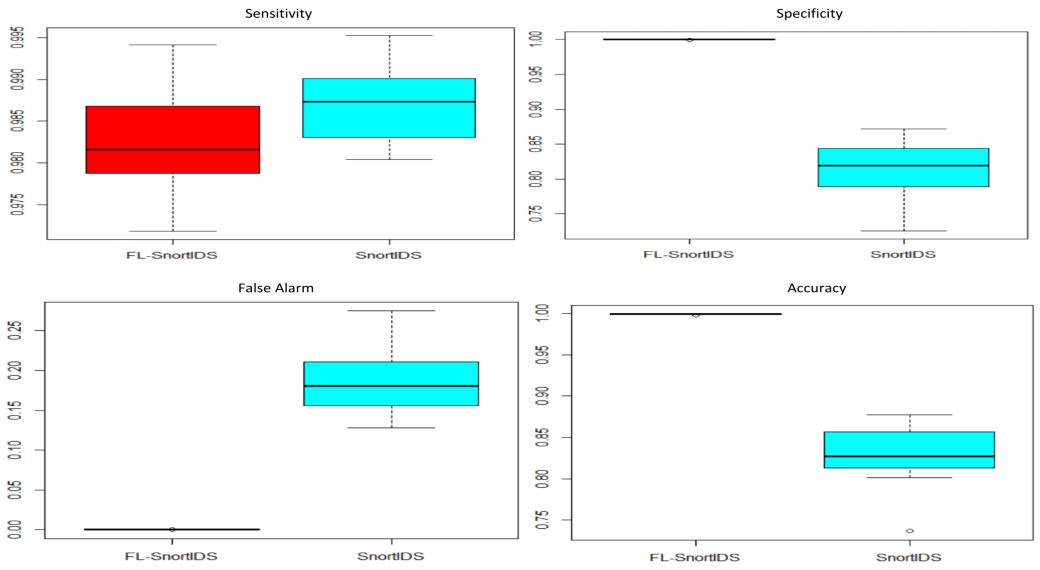


FIGURE 4.7: SnortIDS vs FL-SnortIDS

The box-plot visualisation and Mann-Whitney test show that sensitivity performance is better for SnortIDS is better than the SuricataIDS. For specificity and accuracy, the level of confidence was $p-value > 0.05$, and therefore, we do not have sufficient evidence to reject our null hypothesis i.e. specificity and accuracy performances for SnortIDS are similar to SuricataIDS. For false alarm performance, the true location shift is less than 0 and the level of confidence was $p - value > 0.05$. Therefore, we do not have sufficient evidence to reject our null hypothesis i.e. false alarm ratio for SnortIDS is same as of SuricataIDS.

4.2.4 FL-SnortIDS vs FL-SuircataIDS

The fig. 4.10 states the alternative hypothesis as the true location shift was greater than 0. In sensitivity, specificity and accuracy, $p - value < 0.05$, hence we have sufficient evidence to reject our null hypothesis i.e. sensitivity, specificity and accuracy performances for FL-SnortIDS are better than FL-SuricataIDS. The box-plot visualisation and Mann-Whitney test show that sensitivity, specificity and accuracy performances are better for FL-SnortIDS than FL-SuricataIDS. For false alarm performance, the true location shift is greater than 0 and the level of confidence was $p - value < 0.05$. Therefore, we have

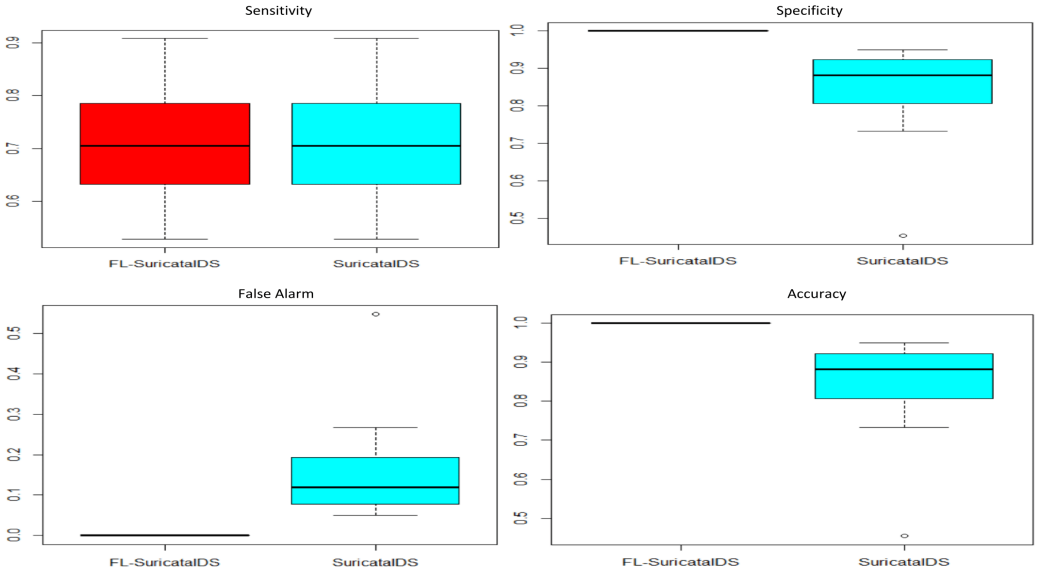


FIGURE 4.8: SuricataIDS vs FL-SuricataIDS

sufficient evidence to reject our null hypothesis i.e. false alarm ratio for FL-SuricataIDS is less than FL-SnortIDS.

4.3 Discussion

4.3.1 Intrusion Detection Systems

After a successful run, SnortIDS and SuricataIDS produce an alert file with a generic template, starting from time of alert generation, what type of alert was generated and what is the threat level for the system. It also provides the basic log of the packet generating alert.

This whole alert file is a simple text file; each alert is separated by a newline identifier but it is not easy to analyse. This is because of the fact that both detectors SnortIDS and SuricataIDS were not programmed as network-aware IDS, so that is why a huge rule based is added to prevent any threats going unnoticed; however, it can generate many false negatives, and careful analysis of the alert files is very difficult, in a network which is relatively large it is very difficult to read SnortIDS and SuricataIDS log files.

The results obtained by performing the above experiment detect the malicious traffic and generate alerts that do not readily provide any useful information. In particular,

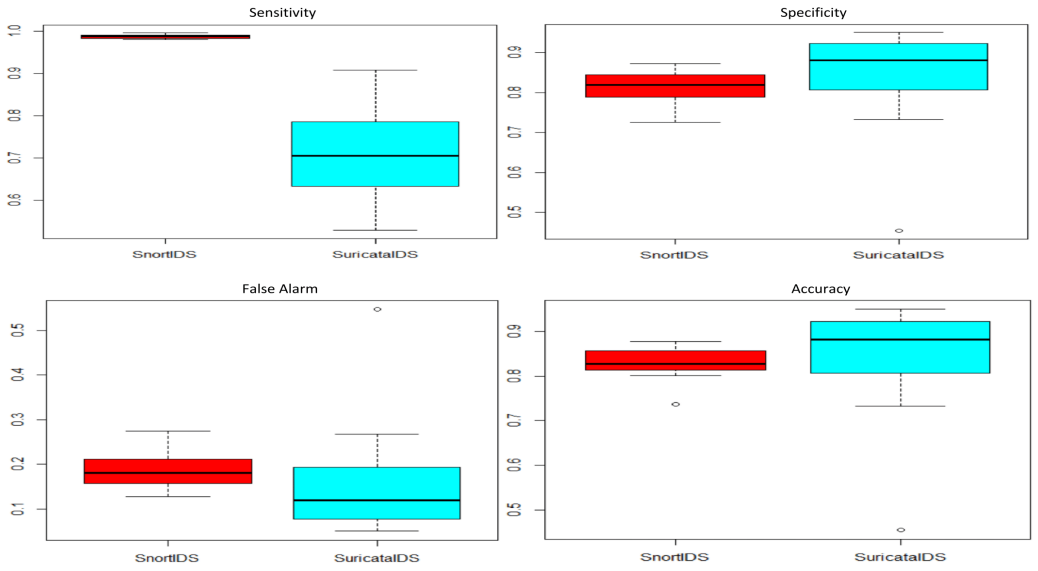


FIGURE 4.9: SnortIDS vs SuricataIDS

the alerts generated by the SnortIDS or SuricataIDS were not classified on the basis of types of the cyber-attacks, which means that we cannot find what kind of attack has been generated by the intruder such as, unauthorized login attempts, DoS attack, and privacy compromising attack. Similarly, the obtained results were not easily readable and required a high level of user interaction. In the next experiments, we proposed a new approach of detecting such malicious activities by leveraging the strengths of Fuzzy Logic with the aim to produce more detailed results as well as more accuracy.

4.3.2 IDS Fuzzy Classifiers

The alerts generated by typical FL-SnortIDS and FL-SuricataIDS systems also include a large number of false positives such as, the generation of an alert due to a single unsuccessful administration privilege gain attempt where it might be a mistake committed by an authorized user. Similarly, several other common mistakes might also result in alerts generation by FL-SnortIDS system. Besides that, the root cause of such false alerts generation is considered to be mostly because of packets drop or network congestion.

An IDS system that is not network aware will generate alerts that are of useful and thus mark that legitimate network traffic as potential alert. Therefore, the total generated

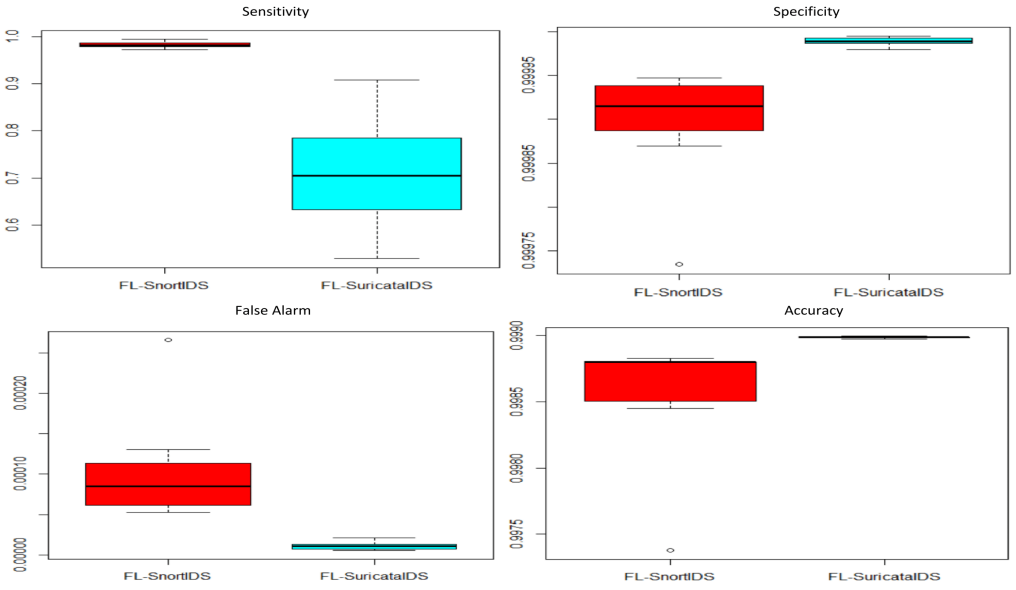


FIGURE 4.10: FL-SnortIDS vs FL-SuricataIDS

alerts may contain a large number of of false alerts that it is highly undesirable for the administration authorities. Fuzzy Logic based SnortIDS and SuricataIDS systems exhibit the tendency to detect the false alerts and ignore them. Moreover, it also classifies the alerts on the basis of cyber-attacks and allows the users to know the type of attacks generated and also produce the results that are easily understandable. It is very effective and greatly enhances the performance as well as accuracy of the SnortIDS and SuricataIDS systems.

As an initial step, towards the development of an enhanced IDS system, the deployed IDS systems on the underlying network infrastructure are made intelligent and aware of the incoming complex traffic. This step helps IDS system to easily distinguish between the real threats from network anomalies and network behavior. By carefully examining each generated alert of the ISCX data set, we were successfully able to separate the real threats against false alerts. This readily decreased the number of alerts generated, which made the alerts easier to be analyzed. As a network administrator, every alert generated by the IDS must be responded. With a lower number of alerts generated, it will be easier to analyse the presence of any malicious or irregular behavior of traffic and respond accordingly.

With respect to the number of alerts generate by SnortIDS, which were reported to be 250174 against ISCX data set, our Fuzzy Logic based SnortIDS reduced these alerts to the total of 64821 threats. It is approximately 25% of initially generated alerts. This means that the alerts generation has been reduced by approximately 75% compare to SnortIDS.

One of the most important factors to consider here is that none of the real threats reported by SnortIDS were removed from FL based IDS alerts. Only the alerts generated due to the high sensitivity of the rules of the Snort community and network congestion were removed to avoid any false positive generated by FL based SnortIDS.

Moving forward to analysis, we needed some basic matrices of the alerts generated by system and the level of accuracy these alerts provided, the given chart shows the number of alerts per system, and the number of true positives and false positives of the system. If an alert is analysed to be authorised traffic then that alert is called a false positive as a rightful stream is marked as warning. If this is not the case then the threat is rightly detected, classifying it as true positive. The graph also shows the number of true negatives and false negatives. A false negative is a stream which IDS have marked as safe for network but it is harmful for the system. With SnortIDS, which has a

large database of rules which is being updated continuously, it's very unlikely that any dangerous stream of packets passes. But for analysis purposes we marked 0.1% of total streams not classified as threats to be called false negatives for analysis. Any genuine traffic not being classified as risk is known as true negative. The graph shows complete detail of both of the IDS.

TABLE 4.4: Mann Whitney Test Result (Based on 95% Confidence Level)

Metrics	Method 1	Method 2			
		SnortIDS	FL-SnortIDS	SuricataIDS	FL-SuricataIDS
Sensitivity	SnortIDS	-	0.0825	0.0002914*	0.0002914*
	FL-SnortIDS	0.9175	-	0.0002914*	0.0002914*
	SuricataIDS	0.999709	0.999709	-	0.5
	FL-SuricataIDS	0.999709	0.999709	0.5	-
Specificity	SnortIDS	-	0.999709	0.9359	0.999709
	FL-SnortIDS	0.0002914*	-	0.0002914*	0.999709
	SuricataIDS	0.0641	0.999709	-	0.999709
	FL-SuricataIDS	0.0002914*	0.0002914*	0.0002914*	-
False Alarm	SnortIDS	-	0.999709	0.9359	0.999709
	FL-SnortIDS	0.0002914*	-	0.0002914*	0.999709
	SuricataIDS	0.0641	0.999709	-	0.999709
	FL-SuricataIDS	0.0002914*	0.0002914*	0.0002914*	-
Accuracy	SnortIDS	-	0.999709	0.9175	0.999709
	FL-SnortIDS	0.0002914*	-	0.0002914*	0.999709
	SuricataIDS	0.0825	0.999709	-	0.999709
	FL-SuricataIDS	0.0002914*	0.0002914*	0.0002914*	-

4.4 Summary

This chapter proposes a fuzzy logic engine that enhances the security performance by combining fuzzy logic to IDS when compared to IDS alone. We did pair-wise comparison as it can be seen in the graphical representation of all four methods in fig. 4.11. On analysing the below result, we can see for the first three comparisons we have clear results:

- FL-SnortIDS is better than SnortIDS
- FL-SuricataIDS is better than SuricataIDS
- SnortIDS is better than SuricataIDS

For the fourth comparison between FL-SnortIDS vs FL-SuricataIDS, we found FL-SnortIDS is better in terms of sensitivity while the other criteria are other way round. So to come up with the conclusion, the graph of specificity, false alarm ratio and accuracy and also the descriptive statistics show that there was a difference in these criteria; yet it is not too much comparing to the criterion of sensitivity. Therefore, based on the sensitivity performance, the FL-SnortIDS is better than FL-SuricataIDS to get false alarm rather than not getting the alarm when actually it should .

- FL-SnortIDS is better than FL-SuricataIDS
- SnortIDS is better than FL-SuricataIDS

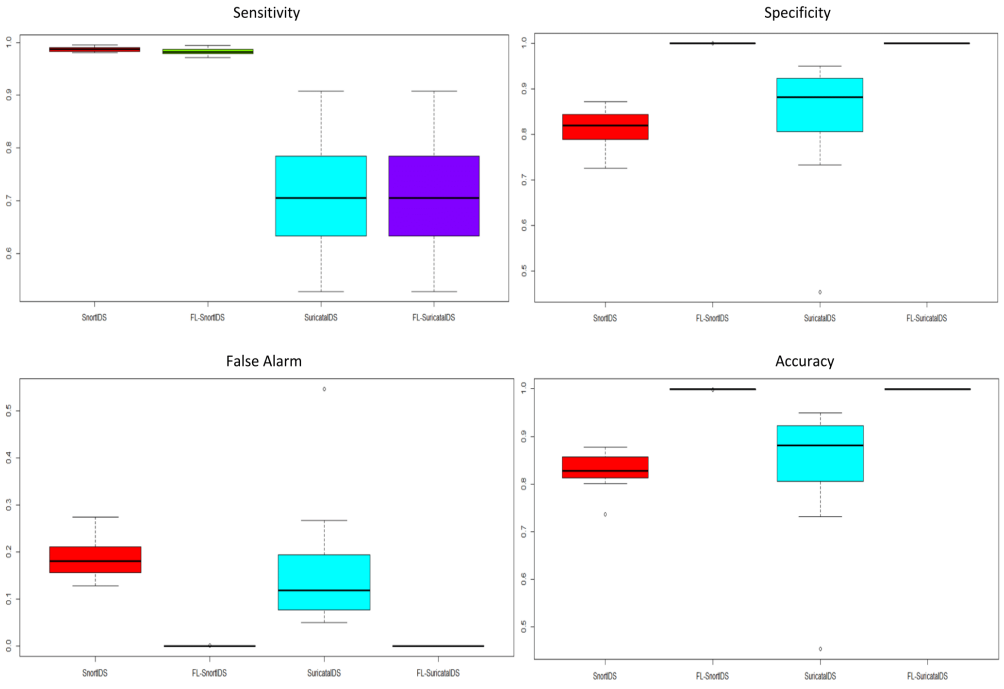


FIGURE 4.11: Final Results for All Systems

Combining results of all five category viz. sensitivity, specificity, False Alarm ratio, accuracy and detection rate, we have the following result ranked according to their performance:

1. FL-SnortIDS which detects the threat with Medium detection rate.

2. SnortIDS which detects the threat with High detection rate.

3. FL-SuricataIDS which detects the threat with Low detection rate.

4. SuricatIDS which detects the threat with High detection rate.

Chapter 5

A Comparative Analysis of Different Classification Techniques for Cloud Intrusion Detection Systems' Alerts and Fuzzy Classifiers

Intrusion detection is a crucial problem in network security. It is a method of parsing network traffic data to detect security abuses and data mining can play a very significant role in evolving an IDS. The dataset of IDSs or soft computing based IDS techniques can be classified into four main classes, which are DoS, U2L, R2L, and Probe, of abnormal traffic. In this chapter, we used each separate data of each detection system, which are SnortIDS, SuricataIDS, FL-SnortIDS, FL-SuricataIDS, against the the most common classification algorithms (CAs). These CAs are Decision Tree (J48), Naive Bayes, OneR, and K-Nearest Neighbour (K-NN). These algorithms were chosen after investigating the most effective classification algorithms that are widely used [Chauhan et al., 2013]. The aim of this study was to present a comparative study for the security performance of each detection system that was gained from our previous experiments: SnortIDS, SuricataIDS, FL-SnortIDS, and FL-SuricataIDS in order to test which classifier algorithm is the best for our systems' results, and investigate which system presents

significant results. The performance of these CAs was evaluated using 10-fold cross validation and experiments and assessments of these methods were performed in the WEKA environment using the ISCX dataset.

5.1 Experimental Results and Analysis

5.1.1 Snort Results (SnortIDS)

Different algorithm classifiers were applied to the SnortIDS system's result which has 203 observations. As shown in table 5.1, various measures of different classifiers have been observed. It is clear that the Decision Tree classifier is the highest performing among others, with an accuracy, weighted average precision, recall, specificity, and F-measure at 99.5%, and an ROC of 1. It also recorded the lowest number of incorrectly classified traffic, Mean Absolute Error, the number of errors to analysis algorithm classification accuracy, and FP Rate. Based on the presented tables, the Decision Tree classifier is the best algorithm for SnortIDS result among all systems in this case. As shown in table 5.1, Naive Bayes is the lowest performing classifier, being outperformed by all other algorithms in all metrics.

TABLE 5.1: Performance Comparison of Different Classifiers Based on Different Metrics with 10 Cross Validation for SnortIDS System

Measurements	**Naïve Bayes**	**OneR**	***K*-NN**	**Decision Tree**
Accuracy	95.56%	97.53%	98.52%	99.50%
Incorrect classified	4.43%	2.46%	1.47%	0.49%
Mean Absolute Error	0.567	0.0123	0.0076	0.0073
FP Rate	0.021	0.002	0.01	0.005
Precision	95.60%	98.10%	98.50%	99.50%
Recall (Sensitivity)	95.60%	97.50%	98.50%	99.50%
Specificity	95.10%	97.53%	98.50%	99.50%
F-Measure	95.60%	97.70%	98.50%	99.50%
ROC Area	0.991	0.986	1	1

As it can be seen in table 5.2, the accuracy value of Naive Bayes were the lowest value, so the default cross-validation of Naive Bayes was modified from 10 to 15 in order to optimise its accuracy that was the lowest value in table 5.1. For example, the accuracy in the 10 cross validation was 95.56%. In the 15 cross validation, accuracy slightly

increased to 96.55%, which is a 1.04% increase. There was a similar increase in precision, recall, specificity and F-measure. There was also an improvement in the lowest number of incorrectly classified traffic and FP Rate, which decreased by an average of 23%. Lastly, the mean absolute error had the largest improvement, from 0.567 in the 10 cross validation to 0.0564 in the 15 cross validation, which is a 90% decrease. This means that the predictive ability of this algorithm has been improved but still it is way behind Decision Tree in every performance metric.

TABLE 5.2: Optimising Naïve Bayes Classifier

Parameters	**Naïve Bayes with 15 Cross Validation**
Accuracy	96.55%
Incorrect classified	3.44%
Mean Absolute Error	0.0564
FP Rate	0.016
Precision	96.8%
Recall (Sensitivity)	96.6%
Specificity	96.6%
F-Measure	96.5%
ROC Area	99.1%

Furthermore, in order to precisely identify the best algorithm for SnortIDS result, all techniques for each performance measures (viz. accuracy, precision, sensitivity, specificity and F-measure) were ranked according to the attack classes (viz. Probe, DoS, R2L, and U2R) for each classifier. For accuracy, As shown in table 5.3, the Decision Tree had the highest overall ranking in all metrics compared to the other classifiers. Although it was outranked by the Naive Bayes in the Probe class in terms of precision and specificity, in fact it was outranked by the other classifiers in the latter. Additionally, it tied with the K-NN and the OneR classifier in the DoS and the R2L classes, respectively, in terms of precision. The four classifiers had the same consistent overall rankings in all metrics, with the Decision Tree as the highest overall ranking, followed by the K-NN classifier, and then it is followed by the OneR classifier. The Naive Bayes had the lowest overall rank in all metrics. The ranking results concluded that Decision Tree was more convenient for SnortIDS more than other classifiers' algorithms in terms of predicting the new income for this system.

Table 5.3: Results and Ranking for SnortIDS Dataset

Accuracy										
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	99.50%	1	100%	1	100%	1	100%	1	99.51%	1
NB	95.56%	4	97.47%	4	98.47%	3	95.54%	4	98.47%	4
OneR	97.53%	3	98.2%	3	97.54%	4	99.99%	2	99.49%	3
***K*-NN**	98.52%	2	98.52%	2	99.99%	2	99.01%	3	99.5%	2
Precision										
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	99.50%	1	99.01%	2	100%	1	100%	1	100%	1
NB	95.90%	4	100%	1	94.12%	2	88.46%	3	99.98%	3
OneR	98.10%	3	98.99%	3	78.26%	3	100%	1	99.96%	4
***K*-NN**	98.50%	2	98.52%	4	100%	1	97.18%	2	99.99%	2
Sensitivity										
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	99.50%	1	100%	1	100%	1	100%	1	93.75%	1
NB	95.60%	4	96.00%	4	88.90%	4	99.93%	4	81.25%	4
OneR	97.50%	3	96.01%	3	99.97%	3	99.97%	3	93.70%	3
***K*-NN**	98.50%	2	98.00%	2	99.99%	2	99.98%	2	93.74%	2
Specificity										
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	99.50%	1	99.02%	4	100%	1	100%	1	100%	1
NB	95.10%	4	100%	1	99.44%	3	93.23%	4	99.00%	4
OneR	97.50%	3	99.97%	2	97.30%	4	99.79%	2	99.91%	3
***K*-NN**	98.50%	2	99.03%	3	99.67%	2	98.50%	3	99.94%	2
F-Measure										
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	99.50%	1	99.50%	1	100%	1	100%	1	96.77%	1
NB	95.10%	4	97.44%	4	91.43%	3	93.88%	4	86.99%	4
OneR	97.50%	3	97.96%	3	87.80%	4	98.75%	3	96.73%	3
***K*-NN**	98.50%	2	98.49%	2	99.9%	2	99.92%	2	96.74%	2

5.1.2 Snort Fuzzy Logic Results (FL-SnortIDS)

The FL-SnortIDS result has 76 observations, which has were classified using the four algorithms in order to obtain the best performance metrics based on this system. As shown in table 5.4, the K-NN and the Decision Tree classifiers have the highest accuracy of 97.36% and the highest F-measure of 97.4%. They also recorded the lowest number of incorrectly classified traffic at 2.63%. Just as same as the result of SnortIDS, it outperformed all the other algorithms in terms of the weighted average precision, recall, specificity, and the ROC Area. It also recorded the lowest mean absolute error. However, it had the highest FP Rate at 0.017, whereas the K-NN had the lowest FP Rate at 0.011. Based on the numbers in table 5.4, the Decision Tree classifier can be considered to be

the best algorithm for the FL-SnortIDS result; however, one should take note that it has a higher chance of producing a false positive compared to the other algorithms. With that in mind, one can also consider the K-NN as a suitable alternative to the Decision Tree since both methods have the same accuracy. As shown in table 5.4, the Naive Bayes was the lowest performing classifier, being outperformed by all other algorithms in all metrics. Hence, we optimised its results in order to improve its overall accuracy.

As it can be shown in table 5.5, the result of Naive Bayes was not changed much although we modified the number of folds cross validation several times; and therefore, there was not any optimisation on its overall accuracy. As shown in table 5.4, it was concluded that Naive Bayes with 10 folds cross validation has the optimum values of accuracy. It can be suggested that, with this particular result, Naive Bayes could have been suitable when it is used with 10 folds cross validation.

TABLE 5.4: Performance Comparison of Different Classifiers Based on Different Metrics with 10 Cross Validation for FL-SnortIDS System

Measurements	**Naïve Bayes**	**OneR**	***K*-NN**	**Decision Tree**
Accuracy	93.42%	96.05%	97.36%	97.36%
Incorrect classified	6.57%	3.94%	2.63%	2.63%
Mean Absolute Error	0.0953	0.0197	0.0165	0.0147
FP Rate	0.012	0.012	0.011	0.017
Precision	95.40%	96.60%	97%	97.50%
Recall (Sensitivity)	93.40%	96.10%	97.40%	97.45%
Specificity	93.40%	96.10%	97.40%	97.45%
F-Measure	93.80%	96.10%	97.40%	97.40%
ROC Area	97%	97.4%	98.90%	99.90%

TABLE 5.5: Optimising Naïve Bayes Classifier with Different Number of Fold Cross Validation for FL-SnortIDS System

Parameters	**8, 9 Fold Cross Validation**	**10 Fold Cross Validation**	**11 Cross Validation**
Accuracy	92.10%	93.42%	92.10%
Incorrect Classified	7.89%	6.57%	7.89%
Mean Absolute Error	0.0961	0.0953	0.0943
FP Rate	0.021	0.012	0.016
Precision	94%	95.40%	94%
Recall (Sensitivity)	92.10%	93.40%	92.10%
Specificity	92.10%	93.40%	92.10%
F-Measure	92.40%	93.80%	92.40%
Roc Area	97.30%	97%	97.15%

By using depth analysis, all measures have been conducted for FL-SnortIDS result in order to find out the most excellent classifier algorithm to predict unseen item in this system evidently. As it can be seen in table 5.6, it shows that the Decision Tree had the highest overall ranking in all metrics compared to the other classifiers. However, it was outranked by the Naive Bayes in the Probe class in all metrics except Sensitivity. Furthermore, it was outranked by the Naive Bayes and the K-NN in the R2L class, in terms of sensitivity and the F-measure, respectively. The four classifiers had the same consistent overall rankings in all metrics, with the Decision Tree as the highest overall ranking, followed by the K-NN classifier, and then it was followed by the OneR classifier. The Naive Bayes had the lowest overall rank in all metrics. Hence, the results show that the Decision Tree outperforms all the other algorithms for the FL-SnortIDS result in terms of predicting the new income in this system.

5.1.3 Suricata Results (SuricataIDS)

The SuricataIDS dataset has 152 observations where such a dataset was classified using the four algorithms in order to obtain the best performance metrics. As shown in table 5.7, it shows that the K-NN and the Decision Tree classifiers have the highest accuracy of 98.68%. The K-NN, OneR, and Decision Tree reported the same F-measure of 98.7% and also recorded the lowest number of incorrectly classified traffic at 1.31%. While the Decision Tree outperforms all the other algorithms in terms of recall, specificity, and the ROC Area, it is outperformed by the OneR in terms of the weighted average precision. It also had less errors as it recorded the lowest mean absolute error and FP Rate among all the other algorithms.

The Decision Tree classifier can be considered to be the best algorithm for the SuricataIDS result because of its high accuracy, sensitivity, and specificity; however, one should take note that it has a higher chance of producing a false positive compared to the other algorithms. With that in mind, the OneR algorithm can be a suitable alternative to the decision tree since while it has slightly lower accuracy, it has a higher precision and is less likely to produce a false positive. Table IX also shows that the Naive Bayes is the lowest performing classifier, being outperformed by all other algorithms in all metrics.

TABLE 5.6: Results and Ranking for FL-SnortIDS Dataset

	Accuracy									
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	97.36%	1	97.36%	3	100%	1	98.86%	1	98.86%	1
NB	93.42%	4	98.61	2	97.26%	3	93.42%	4	98.83%	3
OneR	96.05%	3	98.64%	1	96.05%	4	98.64%	2	98.85%	2
***K*-NN**	97.36%	2	97.36%	4	99.99%	2	97.36%	3	98.64%	4
	Precision									
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	97.50%	1	93.80%	4	100%	1	100%	1	100%	1
NB	95.40%	4	100%	1	99.90%	3	70.60%	4	99%	4
OneR	96.60%	3	99.90%	2	85.70%	4	99.98%	2	99.60%	3
***K*-NN**	97%	2	96.70%	3	99.98%	2	92.30%	3	99.98%	2
	Sensitivity									
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	100%	1	100%	1	100%	1	98.70%	3	98.70%	1
NB	93.40%	4	99.90%	4	9.90%	4	100%	1	87.50%	4
OneR	96.60%	3	96.72%	3	99.99%	2	91%	4	93.80%	2
***K*-NN**	97.40%	2	96.75%	2	99.98%	3	99.99%	2	93%	3
	Specificity									
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	97.30%	1	99.02%	4	100%	1	100%	1	100%	1
NB	93.40%	4	100%	1	99.40%	3	92%	4	99%	4
OneR	96%	3	99.97%	2	97.30%	4	99%	2	99.91%	3
***K*-NN**	97.20%	2	99.03%	3	97.80%	2	98.40%	3	99.99%	2
	F-Measure									
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	97.40%	1	96.80%	3	100%	1	95.80%	2	96.90%	1
NB	93.80%	4	98.40%	1	94.10%	3	82.80%	4	93.30%	4
OneR	96.10%	3	98.30%	2	92.30%	4	95.70%	3	96%	3
***K*-NN**	97.30%	2	96.70%	4	99.90%	2	96%	1	96.80%	2

TABLE 5.7: Performance Comparison of Different Classifiers Based on Different Metrics with 10 Cross Validation for SuricataIDS System

Measurements	**Naïve Bayes**	**OneR**	***K*-NN**	**Decision Tree**
Accuracy	97.36%	98.6%	98.68%	98.68%
Incorrect classified	2.63%	1.31%	1.31%	1.31%
Mean Absolute Error	0.0678	0.0066	0.007	0.0072
FP Rate	0.005	0.001	0.012	0.022
Precision	97.60%	98.80%	98.70%	98.70%
Recall (Sensitivity)	97.40%	98.60%	98.70%	98.80%
Specificity	97.40%	98.60%	98.70%	98.80%
F-Measure	97.30%	98.70%	98.70%	98.70%
ROC Area	0.991	0.993	0.995	1

Different number of fold cross validations were amended to optimise the accuracy of Naive Bayes. The table 5.8 shows that the accuracy of Naive Bayes has been slightly optimised from default 10 folds cross validation to 3 folds cross validation. Thus, it can be said that 3 fold cross validation is the best fold cross validation for such a result of system.

TABLE 5.8: Optimising Naïve Bayes Classifier with Different Number of Fold Cross Validation for SuricataIDS System

Parameters	**3 Folds Cross Validation**	**9 Folds Cross Validation**	**10 Folds Cross Validation**	**11 Folds Cross Validation**
Accuracy	98.02%	97.36%	97.36%	97.36%
Incorrect classified	1.97%	2.63%	2.63%	2.63%
Mean Absolute Error	0.0746	0.0675	0.0678	0.0683
FP Rate	0.004	0.005	0.005	0.005
Precision	98.20%	97.60%	97.60%	97.60%
Recall (Sensitivity)	98%	97.40%	97.40%	97.40%
Specificity	98%	97.40%	97.40%	97.40%
F-Measure	98%	97.30%	97.30%	97.30%
ROC Area	0.992	0.991	0.991	0.991

In this system, all measures have been ranked according to the attack classes in order to certainly offer the most appropriate algorithm to classify the SuricataIDS result. As shown in table 5.9, the Decision Tree had the highest overall ranking in all metrics compared to the other classifiers. However, it had the highest rank in the Probe class in terms of sensitivity only. In addition to that, it was outranked in sensitivity by the OneR and the Naive Bayes classifiers in the DoS class and the R2L class, respectively. Lastly, it was outranked by the K-NN classifier in the U2R class in terms of specificity.

The four classifiers had the same consistent overall rankings in all metrics, with the Decision Tree as the highest overall ranking, followed by the K-NN classifier, and then it is followed by the OneR classifier. The Naive Bayes had the lowest overall rank in all metrics. Based on the outcomes of overall metrics, it could have been said that Decision Tree was the best classifier comparing to other algorithms for SuricataIDS result.

5.1.4 Suricata Fuzzy Logic Results (FL-SuricataIDS)

The FL-SuricataIDS result has 45 observations. The result of this system was classified implementing different algorithms in order to gain the best performance metrics. As

TABLE 5.9: Results and Ranking for SuricataIDS Dataset

Accuracy										
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	98.68%	1	98.68%	4	99.33%	1	100%	1	99.34%	1
NB	97.36%	4	100%	2	98%	3	98%	4	98%	4
OneR	98.60%	3	99.30%	1	94.80%	4	99.90	2	99.30%	3
***K*-NN**	98.64%	2	99%	3	99%	2	99%	3	99.33%	2
Precision										
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	98.70%	1	97.90%	4	100%	1	100%	1	100%	1
NB	97.60%	4	100%	1	93.30%	3	89.70%	4	99.70%	3
OneR	98.80%	3	99.90	2	88.20%	4	99.90%	2	99%	4
***K*-NN**	98.60%	2	99%	3	99.90%	2	96.30%	3	99.90	2
Sensitivity										
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	98.80%	1	100%	1	93.50%	2	100%	3	96.90%	1
NB	97.40%	4	99.70%	3	93%	4	99.70%	1	89.70	4
OneR	98.50%	3	98.90	4	100%	1	99.90%	4	89.90	3
***K*-NN**	98.70%	2	99.90%	2	93.30%	3	99.80%	2	96.80	2
Specificity										
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	98.68%	1	96.49%	4	100%	1	100%	1	99.34%	3
NB	97.36%	4	100%	1	99.30	3	97.60%	4	99.30%	4
OneR	98.60%	3	99.33%	2	94.82%	4	99.90%	2	99.90%	2
***K*-NN**	98.60%	2	98.20%	3	99.34%	2	99.20%	3	100%	1
F-Measure										
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	98.75%	1	99%	3	96.60%	1	100%	1	96.85%	1
NB	97.30%	4	100%	1	93.30%	4	94.50	4	89.70%	4
OneR	98.70%	3	99.40%	2	93.80%	3	99.90%	2	96.81%	3
***K*-NN**	98.73%	2	99.50%	4	96.40%	2	98.10	3	96.84%	2

shown in table 5.10, it shows that the OneR has the highest accuracy of 97. 77%. It outperformed all the other algorithms in all metrics except the ROC Area, where it was outperformed by the Decision Tree with its ROC value of 0.999.

Based on the numbers in table 5.10, the OneR classifier can be considered to be the best algorithm for the FL-SuricataIDS result because of its high accuracy, precision, sensitivity, and specificity compared to all the other algorithms. It is also less likely to produce a false positive. As shown in table 5.10, the Naïve Bayes shows the lowest performing classifier, being outperformed by all other algorithms in all metrics.

Due to the low accuracy of Decision Tree, K-NN and Naïve Bayes, their values were optimised to the best value from 10 to 4 fold cross validations as shown in table 5.11.

TABLE 5.10: Performance Comparison of Different Classifiers Based on Different Metrics with 10 Cross Validation for FL-SuricataIDS System

Measurements	**Naïve Bayes**	**OneR**	***K*-NN**	**Decision Tree**
Accuracy	91.11%	97.77%	95.55%	95.55%
Incorrect classified	8.88%	2.22%	4.44%	4.44%
Mean Absolute Error	0.1124	0.0111	0.0246	0.0148
FP Rate	0.019	0.011	0.014	0.023%
Precision	0.936	0.979	0.959	0.956%
Recall (Sensitivity)	91.10%	97.80%	95.60%	95.6%
Specificity	91.10%	97.80%	95.60%	95.60%
F-Measure	91.50%	97.80%	95.50%	95.60%
ROC Area	0.976	0.983	0.992	0.999

TABLE 5.11: Optimising Naïve Bayes Classifier with Different Number of Fold Cross Validation for FL-SuricataIDS System

Measurements	**Naïve Bayes**	***K*-NN**	**Decision Tree**
Accuracy	93.33%	97.77%	97.77%
Incorrect classified	6.66%	2.22%	2.22%
Mean Absolute Error	0.1211	0.0263	0.0147
FP Rate	0.008	0.003	0.012
Precision	95.80%	98.10%	97.90%
Recall (Sensitivity)	93.30%	97.80%	97.80%
Specificity	93.30%	97.80%	97.80%
F-Measure	93.90%	97.80%	97.80%
ROC Area	0.977	0.999	0.999

For example, the accuracy of the Decision Tree in the 10 folds cross validation was 95.55%. In the 4 folds cross validation, accuracy slightly increased to 97.77%, which is a 2.32% increase, it should be noted that its accuracy is now equal to the accuracy of the OneR in the 10 folds cross validation. The accuracy, precision, sensitivity, specificity, and F-measure of the 4 folds cross validation of the Naive Bayes, K-NN, and Decision Tree classifiers increased by an average of 2.37%. There was also a decrease in the lowest number of incorrectly classified traffic and FP Rate. However, the 4 folds cross validation of the mean absolute error of the Naive Bayes and K-NN, increased by an average of 7.3%. But this was not the case with the Decision Tree classifier, whose mean

absolute error decreased by 0.7%. This shows that the predictive ability of the three algorithms have been improved.

One thing to take note of is that the values of the measures of the 4 folds cross validation of the Decision Tree now resembles the values of the measures of the 10 folds cross validation of the OneR. The difference between the two is that the 10 folds cross validation of the OneR has a lower mean absolute error and FP Rate than the 4 folds cross validation of the Decision Tree. Therefore, while the 4 folds cross validation Decision Tree can be a good algorithm to apply to the FL-SuricataIDS result, the 10 folds cross validation of the OneR can be considered to be the better algorithm for the FL-Suricata result because it will have fewer errors and is less likely to come up with a false positive.

Based on our performance metrics, the table 5.12 clearly indicates that OneR could be the most suitable algorithm for FL-SuricataIDS result as it has the first rank in overall the attack classes. Additionally, OneR has also achieved the first rank in each class of attack in terms of accuracy, precision and sensitivity. Regarding specificity, OneR attained the first rank in all attack classes except DoS, where it was outranked by the Decision Tree. Regarding the F-measure, OneR was outranked by the Decision Tree and the Naive Bayes in the Probe and U2R classes, respectively. Exceptionally for R2L class, OneR has achieved the first rank in all the metrics. The outcomes of classifiers in this system regarding all the metrics indicates that OneR has considerably outperformed Decision Tree that achieved the first rank in the rest of systems' results.

5.2 Discussion

5.2.1 Classifier Algorithms vs *MyCloud* Systems' Results

In this section, the performance metrics of each classification algorithm was compared in order to find the best algorithm for each detection system systems with 10 folds cross validation. This comparison was done after pre-processing the systems' results and cleaning them properly based on statical test. This aim of doing this is to remove the noise in any result. Such a process was reflected in the prediction results through using the mentioned algorithms.

Based on the results of performance metrics against detection systems' dataset, the table 5.13 shows that the Decision Tree has the highest accuracy and specificity over other algorithms for SnortIDS, FL-SnortIDS and SuricataIDS systems. The K-NN has the

TABLE 5.12: Results and Ranking for FL-SuricataIDS Dataset

Accuracy										
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	95.60%	2	99.90%	2	95.50%	4	99.90%	2	95.55%	3
NB	91.11%	4	99.80%	3	99.90%	1	93.18%	4	93.18%	4
OneR	97.77%	1	100%	1	97.77%	2	100%	1	99.33%	1
***K*-NN**	95.56%	3	95.55%	4	100%	1	97.72%	3	97.72%	2
Precision										
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	95.60%	3	99.50%	3	93.35	2	99.90%	2	93.80%	4
NB	93.60%	4	99.40%	4	93.30%	4	62.50%	4	99.50%	3
OneR	97.90%	1	100%	1	93.80%	1	100%	1	100%	1
***K*-NN**	95.90%	2	99.90%	2	93.33%	3	83.3%	3	99.9%	2
Sensitivity										
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	95.70	2	99.70%	2	93.33%	3	99.90%	2	93.80%	3
NB	91.10	4	99.50%	4	93.35%	2	99.70%	4	81.30%	4
OneR	97.80	1	100%	1	100%	1	100%	1	97.80%	1
***K*-NN**	95.60	3	99.60%	3	93.30%	4	99.80%	3	95.60%	2
Specificity										
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	95.60%	2	99.90%	2	96.70%	1	99.90%	2	95.60%	4
NB	91.11%	4	99.40%	4	96.42	3	92.30%	4	99.60%	3
OneR	97.77%	1	100%	1	94.82%	4	100%	1	100%	1
***K*-NN**	95.55%	3	99.70%	3	96.66%	2	97.43%	3	99.70%	2
F-Measure										
	Overall	**Rank**	**Probe**	**Rank**	**DoS**	**Rank**	**R2L**	**Rank**	**U2R**	**Rank**
Tree	95.60%	2	100%	1	93.35%	2	99.90%	2	93.80%	4
NB	91.50%	4	99%	4	93%	4	76.90%	4	99.60%	1
OneR	97.80%	1	99.40%	3	96.80%	1	100%	1	96.80%	2
***K*-NN**	95.50%	3	99.90%	2	93.30%	3	90.90%	3	96.70%	3

highest precision and sensitivity for the SnortIDS and the FL-SnortIDS systems; it also has the highest sensitivity for the SuricataIDS system. The OneR has the highest precision for the SuricataIDS system; it also has the highest accuracy, precision, sensitivity, and specificity for the FL-SuricataIDS system. Hence it can be said that Decision Tree is the best algorithm for these three systems for prediction; however, for FL-SuricataIDS system as shown in table XV, OneR was the best classifier.

In terms of detection systems' results, SnortIDS system provides better accuracy value comparing to FL-SnortIDS system, whereas SuricataIDS system provides better accuracy comparing to FL-SuricataIDS system. Besides accuracy, other metrics such as precision, sensitivity, specificity, and F-measure of the classifiers have been conducted against detection systems' datasets. The result of these metrics concluded that Decision

Tree is matchless for SnortIDS, FL-SnortIDS and SuricataIDS systems and OneR is the best for FL-SuricataIDS system. In addition to these results, SnortIDS and SuricataIDS systems perform within the mentioned algorithms better outcomes than FL-SnortIDS and FL-SuricataIDS systems' results.

TABLE 5.13: Classifiers' Algorithms vs detection systems' Results

Performance Metrics	**Classifiers**			
Accuracy				
	Decision Tree	**OneR**	**Naïve Bayes**	***K*-NN**
SnortIDS	99.50%	97.53%	95.56%	98.52%
SuricataIDS	98.68%	98.68%	97.36%	98.68%
FL-SnortIDS	97.36%	96.05%	93.42%	97.36%
FL-SuricataIDS	95.55%	97.77%	91.11%	95.55%
Precision	**Classifiers**			
	Decision Tree	**OneR**	**Naïve Bayes**	***K*-NN**
SnortIDS	95.60%	98.10%	98.50%	99.50%
SuricataIDS	97.60%	98.80%	98.70%	98.70%
FL-SnortIDS	95.40%	96.60%	97%	97.50%
FL-SuricataIDS	93.60%	97.90%	95.90%	95.60%
Sensitivity	**Classifiers**			
	Decision Tree	**OneR**	**Naïve Bayes**	***K*-NN**
SnortIDS	95.60%	97.50%	98.50%	99.50%
SuricataIDS	97.40%	98.60%	98.70%	98.80%
FL-SnortIDS	93.40%	96.10%	97.40%	97.40%
FL-SuricataIDS	91.10%	97.80%	95.60%	95.60%
Specificity	**Classifiers**			
	Decision Tree	**OneR**	**Naïve Bayes**	***K*-NN**
SnortIDS	99.50%	97.50%	95.10%	98.50%
SuricataIDS	98.68%	98.60%	97.36%	98.6%
FL-SnortIDS	97.30%	96%	93.40%	97.20%
FL-SuricataIDS	95.60%	97.77%	91.11%	95.55%
F-Measure	**Classifiers**			
	Decision Tree	**OneR**	**Naïve Bayes**	***K*-NN**
SnortIDS	99.50%	97.50%	95.10%	98.50%
SuricataIDS	98.75%	98.70%	97.30%	98.73%
FL-SnortIDS	97.40%	96.10%	93.80%	97.30%
FL-SuricataIDS	95.60%	97.80%	91.50%	95.50%

5.2.2 Performance Metrics vs detection Systems' Results

Based on the experimental results shown in table 5.13 and table 5.14, there are clear observations:

- ***SnortIDS is better than FL-SnortIDS.***
- ***SuricataIDS is better than FL-SuricataIDS.***
- ***SuricataIDS is better than FL-SnortIDS.***

However, for the fourth comparison between SnortIDS vs SuricataIDS, It is found that SnortIDS is better SuricataIDS in terms of precision while the other criteria preform based on the chosen algorithm. Thus, to come up with the conclusion, the accuracy and F-measure of SuricataIDS are better than SnortIDS on behalf of OneR, Naive Bayes and K-NN while in sensitivity and specificity of SuricataIDS are better than SnortIDS in Decision Tree, OneR and K-NN. Therefore, based on the performance metrics upon these results, SuricataIDS shows more promising detections than SnortIDS.

- ***SuricataIDS is better than SnortIDS.***

For the fifth and sixth comparisons between FL-SnortIDS vs FL-SuricataIDS and SnortIDS and FL-SuricataIDS, it is found FL-SnortIDS outweigh FL-SuricataIDS and SnortIDS outweigh FL-SuricataIDS in three algorithms except OneR. Hence, it can be said that FL-SnortIDS and SnortIDS show more promising detections than FL-SuricataIDS respectively.

- ***FL-SnortIDS is better than FL-SuricataIDS.***
- ***SnortIDS is better than FL-SuricataIDS.***

5.3 Summary

According to the comparison of classifiers' algorithms and performance metrics against detection systems' dataset, we found that the Decision Tree is the best for SnortIDS, FL-SnortIDS and SuricataIDS systems whereby OneR is the best algorithm for

TABLE 5.14: Performance Metrics vs Detection Systems' Results

Classifiers' Algorithms	**Performance Metrics**				
Descison Tree Algorithm					
	Accuracy	**Precision**	**Sensitivity**	**Specificity**	**F-Measure**
SnortIDS vs FL-SnortIDS	SnortIDS	SnortIDS	SnortIDS	SnortIDS	SnortIDS
SuricataIDS vs FL-SuricataIDS	SuricataIDS	SuricataIDS	SuricataIDS	SuricataIDS	SuricataIDS
SnortIDS vs SuricataIDS	SnortIDS	SuricataIDS	SuricataIDS	SnortIDS	SnortIDS
FL-SnortIDS vs FL-SuricataIDS	FL-SnortIDS	FL-SnortIDS	FL-SnortIDS	FL-SnortIDS	FL-SnortIDS
Naïve Bayes Algorithm	**Performance Metrics**				
	Accuracy	**Precision**	**Sensitivity**	**Specificity**	**F-Measure**
SnortIDS vs FL-SnortIDS	FL-SnortIDS	FL-SnortIDS	FL-SnortIDS	FL-SnortIDS	FL-SnortIDS
SuricataIDS vs FL-SuricataIDS	SuricataIDS	SuricataIDS	SuricataIDS	SuricataIDS	SuricataIDS
SnortIDS vs SuricataIDS	SuricataIDS	SuricataIDS	SuricataIDS	SuricataIDS	SuricataIDS
FL-SnortIDS vs FL-SuricataIDS	FL-SnortIDS	FL-SnortIDS	FL-SnortIDS	FL-SnortIDS	FL-SnortIDS
OneR Algorithm	**Performance Metrics**				
	Accuracy	**Precision**	**Sensitivity**	**Specificity**	**F-Measure**
SnortIDS vs FL-SnortIDS	SnortIDS	SnortIDS	SnortIDS	SnortIDS	SnortIDS
SuricataIDS vs FL-SuricataIDS	SuricataIDS	SuricataIDS	SuricataIDS	SuricataIDS	SuricataIDS
SnortIDS vs SuricataIDS	SuricataIDS	SuricataIDS	SuricataIDS	SuricataIDS	SuricataIDS
FL-SnortIDS vs FL-SuricataIDS	FL-SuricataIDS	FL-SuricataIDS	FL-SuricataIDS	FL-SuricataIDS	FL-SuricataIDS
K-NN Algorithm	**Performance Metrics**				
	Accuracy	**Precision**	**Sensitivity**	**Specificity**	**F-Measure**
SnortIDS vs FL-SnortIDS	SnortIDS	SnortIDS	SnortIDS	SnortIDS	SnortIDS
SuricataIDS vs FL-SuricataIDS	SuricataIDS	SuricataIDS	SuricataIDS	SuricataIDS	SuricataIDS
SnortIDS vs SuricataIDS	SuricataIDS	SnortIDS	SnortIDS	SuricataIDS	SuricataIDS
FL-SnortIDS vs FL-SuricataIDS	FL-SnortIDS	FL-SnortIDS	FL-SnortIDS	FL-SnortIDS	FL-SnortIDS

FL-SuricataIDS system. In addition to these results, SnortIDS and SuricataIDS systems within the chosen algorithms show better outcomes than FL-SnortIDS and FL-SuricataIDS systems. The reason why that IDS systems' results are better than IDS fuzzy classifiers is because of the fact that IDS based fuzzy logic removed all unwanted alerts and have lower number of observations that IDS systems.

According to the comparison of performance metrics against detection systems' datasets as shown in table 5.14, it shows that detection systems' datasets were ranked. The conclusion was as a follows;

1. SuricataIDS outperforms other datasets in all given algorithms.
2. SnortIDS is the second best result to predict in given algorithms.
3. FL-SnortIDS is the third result to predict in given algorithms.
4. FL-SuricataIDS has the last ranking in predicting alarms in given algorithms.

Chapter 6

Security Performance Evaluation Using Type-1 Fuzzy Logic Approach (FIDSCC System)

On the 26th of November, 2012, few of the Google services got disconnected approximately for twenty minutes; sufficient time to affect everyone. Companies who were using Google Apps were failed to manage e-mail tasks and this phenomenon showed the dependency of different countries on Google services. Therefore, fragility became apparent; a question comes in mind as to what level can people depend on the cloud services. The purpose of this chapter is to analyse how to minimise vulnerability in the network and security issues in the cloud after employing fuzzy logic technique as a proposal to improve the cloud performance through threat detection rate and reducing the false positive alarms. Data integrity is a primary thing whereas information systems are related to physical as well as human failure and form a possible objective for malicious assault for acquisition of data.

6.1 How Does FIDSCC System Work within Cloud Computing

As explained in chapter 4, the techniques used in fuzzy classification of the alerts generated by both the IDSs can be categorised in two main categories.

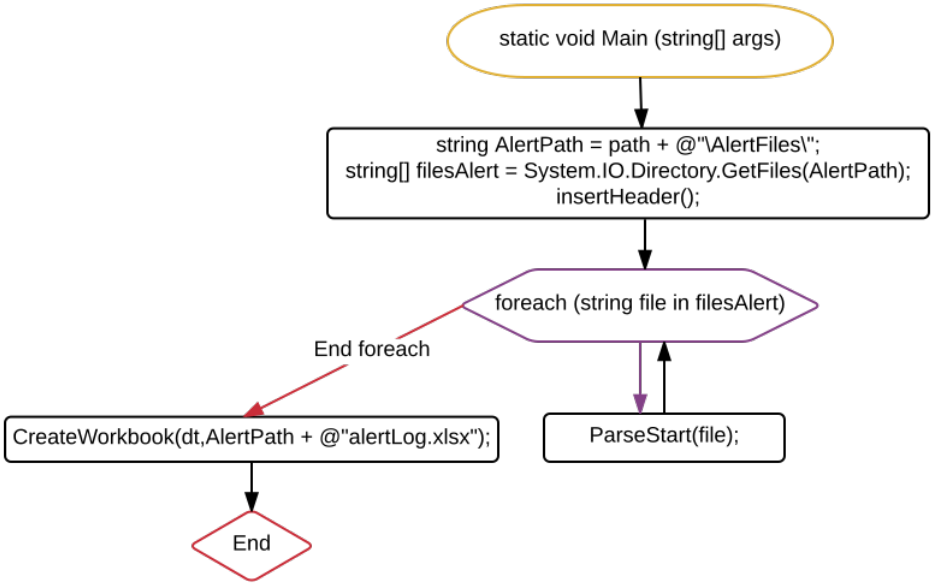


FIGURE 6.1: How Fuzzy IDS Reads IDSs' Alerts

1. An intelligent program to understand the alerts generated by the two IDS.
2. Removal of any unwanted or false alerts for better understanding of the problem.

For both steps we needed a unified mechanism and criteria for the alerts generated by any of the IDS, irrespective of the IDS output format. In order to solve this task, a system was developed, which when given the log file of the above mentioned IDSs that will generate a unified output which will be understandable by the system, and easy to manipulate by network administrator or forensic analysis. The first task is to read the alerts for both IDS systems and generate an output which can be fed to the next system for analysis purposes. See fig. 6.1.

The fig. 6.1 suggests that once given a path for a folder where the alerts file are placed it will parse all the files using function *parsestart()* with respect to the IDS and at the end will generate a unified alert log file. This file can be opened using excel and user can apply different filters on the data if required. Now, to see what happens in the *parsestart()* function, the next flow chart will be helpful to understand, this flow is for SnortIDS/SuricataIDS log files.

This piece of code will read the file each line, and will read till the end of file is received. After the first line of the alert is read, the program will iterate through the file to find the end of each alert. The alerts component can be seen in fig. 6.2 log provided by

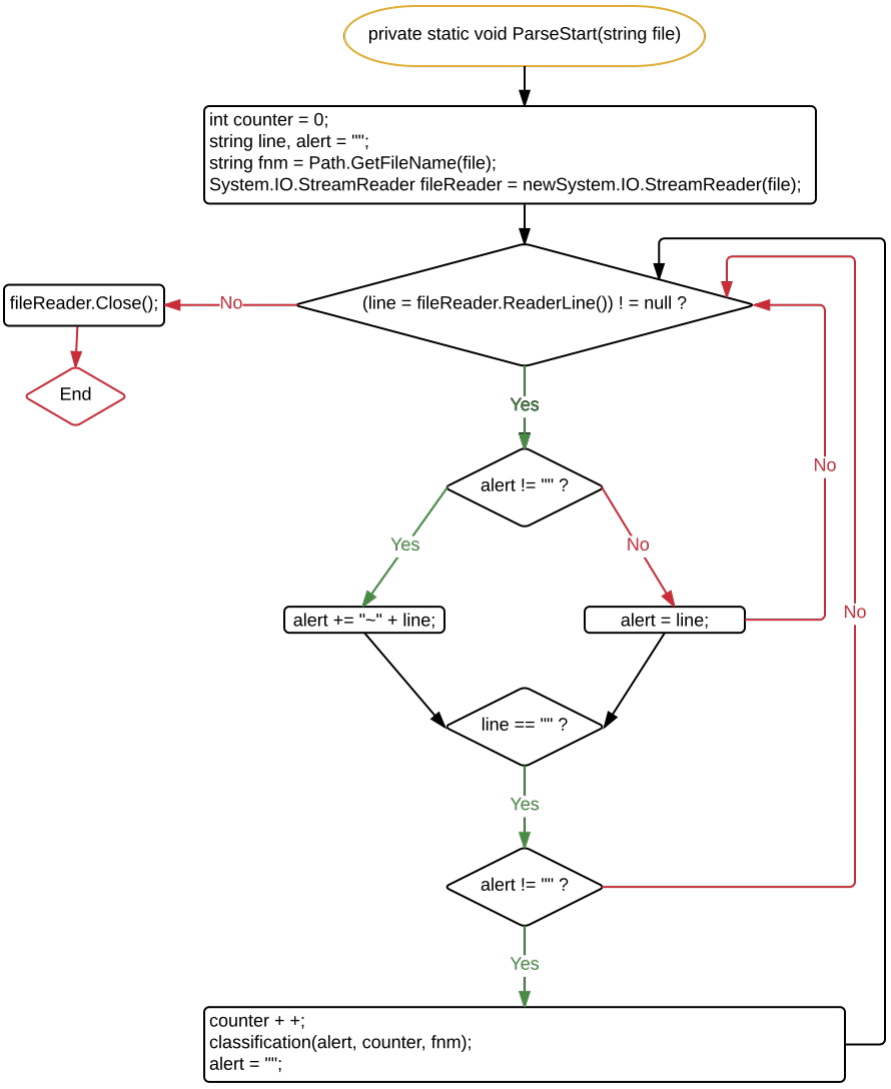


FIGURE 6.2: How Fuzzy Classifier Generates Log Files for IDSs' Systems

SnortIDS system. Such a log is explained in table 6.1 that explains the log of SnortIDS system. After the basic classification, the information extracted from this log along with the complete log itself is then dumped into a *datatable* which is used to create the alert log file later. The fig. 6.2 shows how fuzzy classifier generates the alert files for IDSs' systems.

This detailed and formatted log file can help any forensic team to understand what is wrong with the network or what went wrong in the network. After this basic classification of the data, the next task is to engineer some intelligence out of the data and remove any unrelated or unwanted alerts or threats. This however cannot be achieved without

TABLE 6.1: An Example of Analysing The IDS Systems' Logs

File Name	Alert Count	Alert Type	Alert Classification	Priority	Complete Log
alert_11	1	Consecutive TCP small segments exceeding threshold	Potentially Bad Traffic	2	[**] [129:12:1] Consecutive TCP small segments exceeding threshold [**],[Classification: Potentially Bad Traffic] [Priority: 2] ,06/10-23:01:07.148725 192.168.5.122:22 -> 192.168.2.111:1566,TCP TTL:64 TOS:0x8 ID:23576 IpLen:20 DgmLen:72 DF,***AP*** Seq: 0x7EB71E3 Ack: 0x8E4571EB Win: 0x1D50 TcpLen: 20,
alert_11	2	Consecutive TCP small segments exceeding threshold	Potentially Bad Traffic	2	[**] [129:12:1] Consecutive TCP small segments exceeding threshold [**],[Classification: Potentially Bad Traffic] [Priority: 2] ,06/10-23:01:07.152255 192.168.5.122:22 -> 192.168.3.114:4947,TCP TTL:64 TOS:0x8 ID:38480 IpLen:20 DgmLen:72 DF,***AP*** Seq: 0xB5AB593 Ack: 0x793B8F48 Win: 0x1D50 TcpLen: 20,
alert_11	3	Consecutive TCP small segments exceeding threshold	Potentially Bad Traffic	2	[**] [129:12:1] Consecutive TCP small segments exceeding threshold [**],[Classification: Potentially Bad Traffic] [Priority: 2] ,06/10-23:01:07.152737 192.168.5.122:22 -> 192.168.3.114:4947,TCP TTL:63 TOS:0x0 ID:38481 IpLen:20 DgmLen:128 DF,***AP*** Seq: 0xB5AB5B3 Ack: 0x793B8F48 Win: 0x1D50 TcpLen: 20,
alert_11	4	(http_inspect) INVALID CONTENT-LENGTH OR CHUNK SIZE	Unknown Traffic	3	[**] [120:8:2] (http_inspect) INVALID CONTENT-LENGTH OR CHUNK SIZE [**],[Classification: Unknown Traffic] [Priority: 3] ,06/10-23:01:14.390418 192.168.4.121:64401 -> 65.55.149.121:80,TCP TTL:128 TOS:0x0 ID:22036 IpLen:20 DgmLen:40 DF,***A**** Seq: 0x14F33638 Ack: 0xE24B3711 Win: 0x4273 TcpLen: 20,[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2013-2028],
alert_11	5	(http_inspect) INVALID CONTENT-LENGTH OR CHUNK SIZE	Unknown Traffic	3	[**] [120:8:2] (http_inspect) INVALID CONTENT-LENGTH OR CHUNK SIZE [**],[Classification: Unknown Traffic] [Priority: 3] ,06/10-23:01:14.617694 192.168.4.121:64402 -> 65.55.239.164:80,TCP TTL:128 TOS:0x0 ID:22068 IpLen:20 DgmLen:40 DF,***A**** Seq: 0x3EAC3D1B Ack: 0x2B526110 Win: 0x4289 TcpLen: 20,[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2013-2028],
alert_11	6	Consecutive TCP small segments exceeding threshold	Potentially Bad Traffic	2	[**] [129:12:1] Consecutive TCP small segments exceeding threshold [**],[Classification: Potentially Bad Traffic] [Priority: 2] ,06/10-23:01:21.601884 192.168.5.122:22 -> 192.168.2.111:1568,TCP TTL:64 TOS:0x8 ID:8884 IpLen:20 DgmLen:72 DF,***AP*** Seq: 0x173E270A Ack: 0x8E8221F0 Win: 0x1D50 TcpLen: 20,
alert_11	7	Consecutive TCP small segments exceeding threshold	Potentially Bad Traffic	2	[**] [129:12:1] Consecutive TCP small segments exceeding threshold [**],[Classification: Potentially Bad Traffic] [Priority: 2] ,06/10-23:01:47.167247 192.168.5.122:22 -> 192.168.2.113:3133,TCP TTL:63 TOS:0x0 ID:21195 IpLen:20 DgmLen:79 DF,***AP*** Seq: 0x32D41A33 Ack: 0x6BE9E88C Win: 0x16D0 TcpLen: 20,
alert_11	8	ET SCAN Potential SSH Scan OUTBOUND	Attempted Information Leak	2	[**] [1:2003068:6] ET SCAN Potential SSH Scan OUTBOUND [**],[Classification: Attempted Information Leak] [Priority: 2] ,06/10-23:01:56.971477 192.168.2.111:1571 -> 192.168.5.122:22,TCP TTL:128 TOS:0x0 ID:32007 IpLen:20 DgmLen:48 DF,******S* Seq: 0x8F17919C Ack: 0x0 Win: 0xFAF0 TcpLen: 28,TCP Options (4) => MSS: 1460 NOP NOP SackOK ,[Xref => http://doc.emergingthreats.net/2003068][Xref => http://en.wikipedia.org/wiki/Brute_force_attack],

Table 6.2: ISCX Dataset Attack Classes

Type	**Attacks**
DoS	A Network Trojan was detected Attempted Denial of Service
R2L	Attempted Information Leak Potentially Bad Traffic Information Leak
U2R	Access to a Potentially Vulnerable Web Application Attempted Administrator Privilege Gain Attempted User Privilege Gain Unsuccessful User Privilege Gain
Probe	(null) Detection of a Network Scan Detection of a Non-Standard Protocol or Event Executable Code was Detected Generic Protocol Command Decode Misc activity Misc Attack Not Suspicious Traffic Potential Corporate Privacy Violation Unknown Traffic Web Application Attack

understanding the network where IDS is deployed and its company policy. In this chapter, we removed any alerts generated by network congestion or ill configured services. Furthermore, we focused on application level attacks which helped us remove a lot of network related alerts. for this reason, we firstly categorised the alerts on the basis of the type of the attacks as DoS, Probe, R2L and U2R. This categorisation is shown in table 6.2.

This data is available in a file which can be changed any time by Network Administrator to help classify the attacks in future. For classification and better analysis of the alerts, we created a program called Fuzzy Classifier. The basic flow of this code is provided in fig. 6.3.

After basic system initialisation, which will populate the data tables and the unwanted alerts as in table 6.3. We then generated a classified output of each attack with the total number of alerts and the type it belongs to. This file is generated in *classifer()* and dumped into the memory later, this file is again an excel file and looks like the output shown in table 6.3.

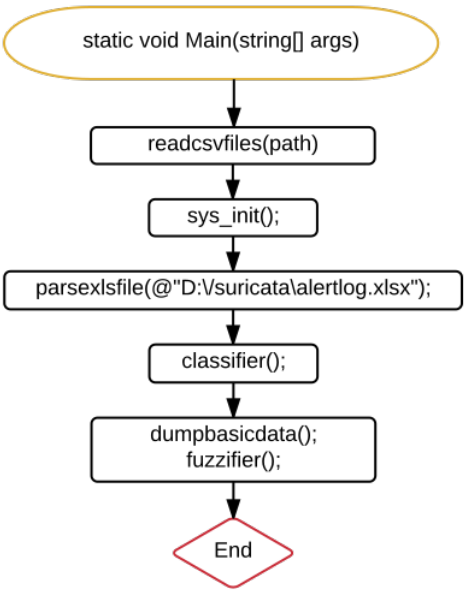


FIGURE 6.3: Fuzzy Classifier Flowchart

TABLE 6.3: IDS Systems' Outputs

Alert Type	Alert Classification	Count	Priority	Classifier
A Network Trojan was detected	ET TROJAN DustySky Checkin	1,655	1	DoS
Attempted Administrator Privilege Gain	ET SCAN LibSSH Based Frequent SSH Connections Likely BruteForce Attack	313	1	U2R
A Network Trojan was detected	ET TROJAN Blue Bot DDoS Blog Request	273	1	DoS
Attempted Information Leak	ET SCAN Potential SSH Scan	204	2	R2L
A Network Trojan was detected	ET TROJAN Win32.Sality.3 Checkin	113	1	DoS
A Network Trojan was detected	ET MALWARE Regnow.com Access	41	1	DoS
Attempted Denial of Service	ET DOS Microsoft Remote Desktop (RDP) Syn then Reset 30 Second DoS Attempt	27	2	DoS
Web Application Attack	ET WEB_SERVER HTTP 414 Request URI Too Large	21	1	Probe
A Network Trojan was detected	ET POLICY Microsoft user-agent automated process response to automated request	20	1	DoS

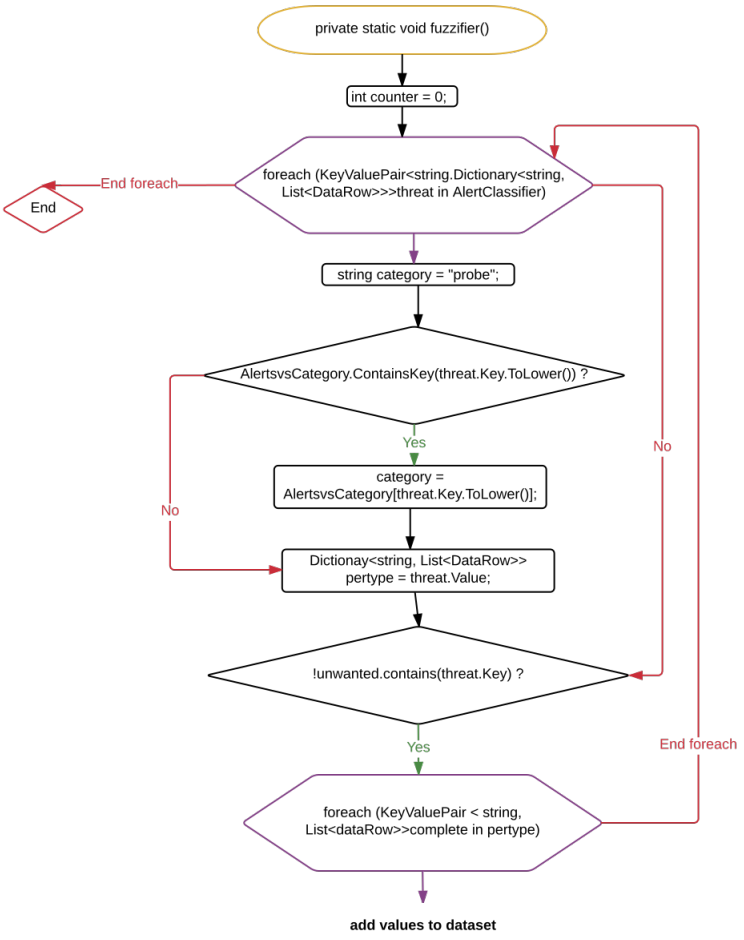


FIGURE 6.4: How Fuzzy IDS Removes Unwanted Alerts

After this classification, we used the fuzzy classifier using the information of unwanted alerts and the alert count, and then we generated a new alert log file which essentially looks the same as the first alert log file; yet the contents are very specific and the attacks specified are threat to the network and sensitive information. This task is achieved follows the steps as illustrated in fig. 6.4.

The program removes any unwanted alerts and keeps the real threats in the alert file. The results are discussed and analysed in the next section where it shows the efficiency and effectiveness of this system.

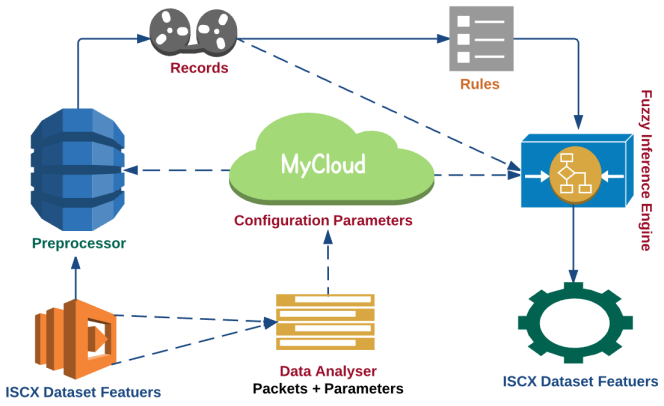


FIGURE 6.5: Simulation Modeling FIDSCC System within *MyCloud*

6.2 Modeling FIDSCC System

FIDSCC system basically works in two ways of operations: evaluating the process of detection ISCX attacks and generating rules. When FIDSCC is being operated in the rule-generation mode, it processes *MyCloud* data and uses a fuzzy application to generate rules. The detection mode uses this rule subset for intrusion detection. The following subsections of modelling the system describe the different components of FIDSCC architecture. The fig. 6.5 shows how FIDSCC system works.

In this experiment, FIDSCC system was modelled in order to assess the possibility of implementing a large system after creating and developing models for cloud computing issues. As it can be seen in table 6.4 and table 6.5, FIDSCC system has two systems in order to assess the overall *MyCloud* performance. The first system is to assess the effectiveness of the IDS performance through which the number of threat detected within *MyCloud* in each dataset.

TABLE 6.4: First System: *MyCloud* Detection Rate

System 1: IDS Evaluation (Number of Threat Detected)	
Input	**Output**
DoS Alerts	IDS Performance
Probe Alerts	
R2L Alerts	
U2R Alerts	

TABLE 6.5: Second System: Evaluating *MyCloud* Performance

System 2: Overall IDS Evaluation	
Input	**Output**
Number of Threat Detected	MyCloud Evaluation
False Positive Rate	

The second system is to assess *MyCloud* Performance through which the consequence of the number of threat detected within *MyCloud* in the first system and the false positive rate.

6.2.1 Features

The FIDSCC's values and its inputs as shown in table 6.4 and table 6.5 are represented as names and will be utilised as linguistic terms by the fuzzy inference engine. FIDSCC's features are stored as follows;

$V = \alpha_1, \alpha_2, and, \alpha_6$
where
$V = \alpha_{DoS}, \alpha_{Probe}, \alpha_{R2L}, and, \alpha_{U2R}$
is an example of FIDSCC's values

As shown in fig. 6.6, there are four inputs with their specified ranges. Its membership functions were simulated based on expertise and the consequences for each datasets. The output of these inputs illustrate the IDS performance through which the number of threat detected within *MyCloud*. Then the output of such a system would be used as an input for the second system along with the false positive ratio in order to assess *MyCloud* performance. (See fig. 6.6).

6.2.2 Data Analyser

Packets of ISCX dataset are placed in fixed size group ($f - group$) or in groups of packets captured in a certain period of time ($t - group$). Each ($f - group$) contains the same number packets covering a variable time range where each ($t - group$) contains a variable number of packets captured over a certain period of time. The module analyses data contained in each packet as a group in order to obtain domain information for

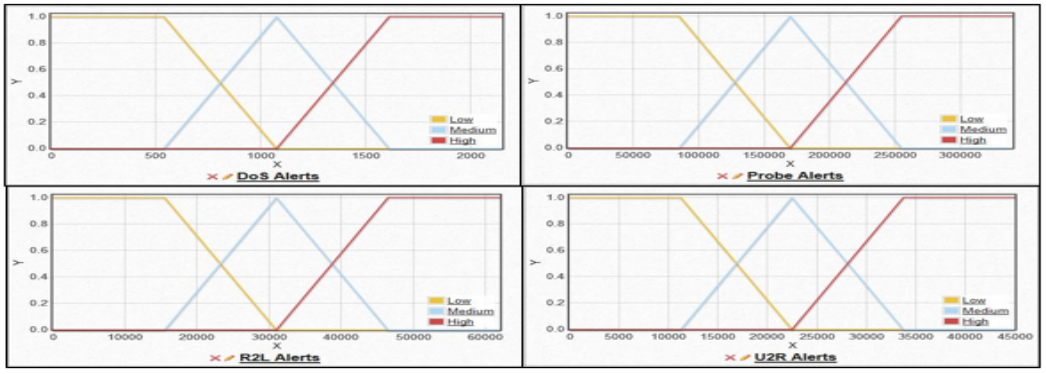


FIGURE 6.6: The First System's Membership Functions

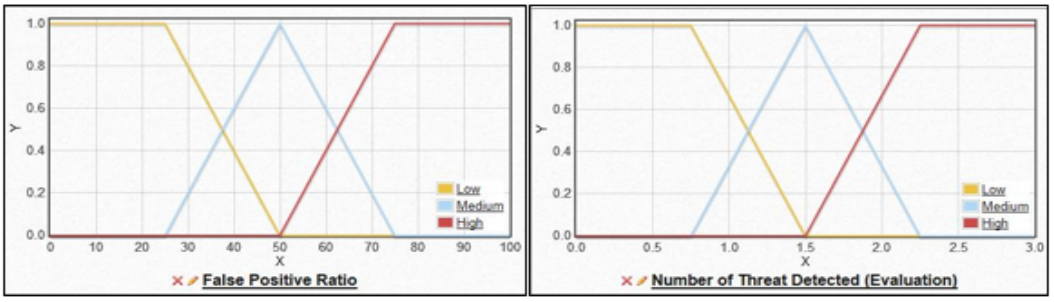


FIGURE 6.7: The Second System's Membership Functions

each features, e.g., $(low; medium; high)$. This information was used to help define a normalisation scheme for each features which will have a term set. Such feature is defined to help express rules using fuzzy variables. For instance, in order to fuzzify ISCX packets, it should be described as follows;

$$TERM(DoS) = Low, Medium, High$$
$$TERM(Probe) = Low, Medium, High$$
$$TERM(R2L) = Low, Medium, High$$
$$TERM(U2R) = Low, Medium, High$$

6.2.3 Configuration Parameters

Based on analysing data and features, the data analyser produces a file where all configuration parameters are stored in *MyCloud*. Parameter values then stored in this file regulate operation of the pre-processor and Fuzzy Inference Engine as shown in fig. 6.5. The configuration file specifies how features values are normalised, associates feature with a term set, and fuzzy membership functions associated with each term.

6.2.4 Pre-Processor

The Pre-processor is responsible for accepting raw packet data and producing records for each group as specified by the configuration file. This component is used in both rule generation mode and evaluating detection mode. The output produced by this component consists of records, each containing aggregate information for each packet group. Using records and concentrating only on features of interest greatly help in reducing the amount of information to be used by more computationally intensive components of the architecture. Records are only used to evaluate fuzzy logic rules with the exception of the case where one packet maps to one record.

6.2.5 Rules

With the fuzzy input sets defined, the next step is to write the rules for detecting ISCX attacks. A collection of fuzzy rules with the same input and output variables as shown in table 6.4 and table 6.5. The rules were created using JuzzyOnLine[1]. This tool contains a graphical user interface that allows the rule designer to create the membership functions for each input or output variable. Create the inference relationships between the various member functions, and to examine the control surface for the resulting fuzzy system. Based on the expertise, 31 rules have been generated for the first system in order to evaluate the performance of IDS and 9 rules for the second system to assess the performance of *MyCloud*. These rules were relevant to given data in FIDSCC system. See fig. 6.8 and fig. 6.9.

6.2.6 Fuzzy Inference Engine

With using JuzzyOnLine, an inference determines which rules are relevant to the given data in FIDSCC system where the engine uses six inputs in total as explained in the in table 6.4 and table 6.5. Additionally, the engine also presents the firing strength of each rule applied to each fact. FIDSCC system has been configured the inference engine to use Mamdani inference mechanism. See fig. 6.5.

[1]it is a website that uses the Juzzy toolkit, a Java-based library for type-1, interval and general type-2 fuzzy systems provided by Christian Wagner. It is open-source and free for non-commercial use http://ritweb.cloudapp.net:8080/JuzzyOnline/

Rules
1. If DoS Alerts is High and Probe Alerts is High and R2L Alerts is High and U2R Alerts is High then IDS Performance is High
2. If DoS Alerts is Medium and Probe Alerts is High and R2L Alerts is High and U2R Alerts is High then IDS Performance is High
3. If DoS Alerts is Low and Probe Alerts is High and R2L Alerts is High and U2R Alerts is High then IDS Performance is High
4. If DoS Alerts is High and Probe Alerts is Medium and R2L Alerts is High and U2R Alerts is High then IDS Performance is High
5. If DoS Alerts is High and Probe Alerts is Low and R2L Alerts is High and U2R Alerts is High then IDS Performance is High
6. If DoS Alerts is High and Probe Alerts is High and R2L Alerts is Medium and U2R Alerts is High then IDS Performance is High
7. If DoS Alerts is High and Probe Alerts is High and R2L Alerts is Low and U2R Alerts is High then IDS Performance is High
8. If DoS Alerts is High and Probe Alerts is High and R2L Alerts is High and U2R Alerts is Medium then IDS Performance is High
9. If DoS Alerts is High and Probe Alerts is High and R2L Alerts is High and U2R Alerts is Low then IDS Performance is High
10. If DoS Alerts is Medium and Probe Alerts is Medium and R2L Alerts is Medium and U2R Alerts is Medium then IDS Performance is Medium
11. If DoS Alerts is Low and Probe Alerts is Low and R2L Alerts is Low and U2R Alerts is Low then IDS Performance is Low
12. If DoS Alerts is Low and Probe Alerts is Low and R2L Alerts is Low and U2R Alerts is Medium then IDS Performance is Medium
13. If DoS Alerts is Low and Probe Alerts is Low and R2L Alerts is Low and U2R Alerts is High then IDS Performance is Medium
14. If DoS Alerts is Low and Probe Alerts is Low and R2L Alerts is Medium and U2R Alerts is Low then IDS Performance is Medium
15. If DoS Alerts is Low and Probe Alerts is Low and R2L Alerts is High and U2R Alerts is Low then IDS Performance is Medium
16. If DoS Alerts is Low and Probe Alerts is Medium and R2L Alerts is Low and U2R Alerts is Low then IDS Performance is Medium
17. If DoS Alerts is Low and Probe Alerts is High and R2L Alerts is Low and U2R Alerts is Low then IDS Performance is Medium
18. If DoS Alerts is Medium and Probe Alerts is Low and R2L Alerts is Low and U2R Alerts is Low then IDS Performance is Medium
19. If DoS Alerts is High and Probe Alerts is Low and R2L Alerts is Low and U2R Alerts is Low then IDS Performance is Medium
20. If DoS Alerts is Low and Probe Alerts is Low and R2L Alerts is Medium and U2R Alerts is Medium then IDS Performance is Medium
21. If DoS Alerts is Low and Probe Alerts is Low and R2L Alerts is High and U2R Alerts is Medium then IDS Performance is Medium
22. If DoS Alerts is Low and Probe Alerts is Low and R2L Alerts is Medium and U2R Alerts is High then IDS Performance is Medium
23. If DoS Alerts is Low and Probe Alerts is Low and R2L Alerts is High and U2R Alerts is High then IDS Performance is Low
24. If DoS Alerts is Low and Probe Alerts is Medium and R2L Alerts is High and U2R Alerts is High then IDS Performance is High
25. If DoS Alerts is Low and Probe Alerts is Medium and R2L Alerts is Medium and U2R Alerts is High then IDS Performance is Medium
26. If DoS Alerts is Low and Probe Alerts is Medium and R2L Alerts is Medium and U2R Alerts is Medium then IDS Performance is Medium
27. If DoS Alerts is Medium and Probe Alerts is Medium and R2L Alerts is High and U2R Alerts is High then IDS Performance is High
28. If DoS Alerts is Medium and Probe Alerts is Low and R2L Alerts is High and U2R Alerts is High then IDS Performance is High
29. If DoS Alerts is Low and Probe Alerts is Medium and R2L Alerts is High and U2R Alerts is High then IDS Performance is High
30. If DoS Alerts is High and Probe Alerts is Low and R2L Alerts is Low and U2R Alerts is High then IDS Performance is Medium
31. If DoS Alerts is High and Probe Alerts is High and R2L Alerts is Low and U2R Alerts is Low then IDS Performance is Medium

FIGURE 6.8: The First System's Rules

Rules
1. If Number of Threat Detected (Evaluation) is High and False Positive Ratio is Low then IDS Evaluation is High
2. If Number of Threat Detected (Evaluation) is High and False Positive Ratio is Medium then IDS Evaluation is Medium
3. If Number of Threat Detected (Evaluation) is High and False Positive Ratio is High then IDS Evaluation is Medium
4. If Number of Threat Detected (Evaluation) is Medium and False Positive Ratio is Low then IDS Evaluation is High
5. If Number of Threat Detected (Evaluation) is Medium and False Positive Ratio is Medium then IDS Evaluation is Medium
6. If Number of Threat Detected (Evaluation) is Medium and False Positive Ratio is High then IDS Evaluation is Low
7. If Number of Threat Detected (Evaluation) is Low and False Positive Ratio is High then IDS Evaluation is Low
8. If Number of Threat Detected (Evaluation) is Low and False Positive Ratio is Medium then IDS Evaluation is Low
9. If Number of Threat Detected (Evaluation) is Low and False Positive Ratio is Low then IDS Evaluation is Medium

FIGURE 6.9: The Second System's Rules

6.3 Comparative Analysis

6.3.1 DoS Attacks within Detection Systems

Since number of true positive and false positive are almost same for all the method, also with 99% confidence level the test result as shown in table 6.7 does not show any significant difference in performance by different method for detecting the Dos threat. Hence, DoS threat is detected by all the system similarly i.e. all are equivalently good to detect DoS.

TABLE 6.6: DoS Attacks within *MyCloud*

	True Positive	**False Positive**
SnortIDS	2,083	20
FL-SnortIDS	2,083	20
SuricataIDS	2,125	21
FL-SuricataIDS	2,125	21

TABLE 6.7: Sample Proportion Test Result (DoS)

	SnortIDS	FL-SnortIDS	SuricataIDS	FL-SuricataIDS
SnortIDS	-	0.50	0.46	0.46
FL-SnortIDS	0.50	-	0.46	0.46
SuricataIDS	0.54	0.54	-	0.50
FL-SuricataIDS	0.54	0.54	0.50	-

6.3.2 Probe Attacks within Detection Systems

TABLE 6.8: Probe Attacks within *MyCloud*

	True Positive	**False Positive**
SnortIDS	1,836	140,411
FL-SnortIDS	1,836	18
SuricataIDS	30	339,314
FL-SuricataIDS	30	4

TABLE 6.9: Sample Proportion Test Result (Probe)

	SnortIDS	FL-SnortIDS	SuricataIDS	FL-SuricataIDS
SnortIDS	-	1	0*	1
FL-SnortIDS	0*	-	0*	0.001*
SuricataIDS	1	1	-	1
FL-SuricataIDS	0*	0.999	0*	-

Based on test result in table 6.9, with 99% confidence level:

1. SnortIDS is better than SuricataIDS to detect Probe.
2. FL-SnortIDS is better than SnortIDS, SuricataIDS, FL-SuricataIDS to detect Probe.
3. FL-SuricataIDS is better than SnortIDS and SuricataIDS to detect probe.

6.3.3 R2L Attacks within Detection Systems

TABLE 6.10: R2L Attacks within *MyCloud*

	True Positive	**False Positive**
SnortIDS	44,458	448
FL-SnortIDS	44,458	448
SuricataIDS	340	3
FL-SuricataIDS	340	3

TABLE 6.11: Sample Proportion Test Result (R2L)

	SnortIDS	FL-SnortIDS	SuricataIDS	FL-SuricataIDS
SnortIDS	-	0.50	0.59	0.59
FL-SnortIDS	0.50	-	0.59	0.59
SuricataIDS	0.40	0.40	-	0.50
FL-SuricataIDS	0.40	0.40	0.50	-

With 99% confidence level, the test result in table 6.11 does not show any significant difference in performance by different method for detecting the R2L threat. Hence, R2L threat is detected by all the system similarly i.e. all are equivalently good to detect R2L.

6.3.4 U2R Attacks within Detection Systems

TABLE 6.12: U2R Attacks within *MyCloud*

	True Positive	**False Positive**
SnortIDS	16,046	45,772
FL-SnortIDS	16,046	162
SuricataIDS	218	598
FL-SuricataIDS	218	2

Based on test result in table 6.13, with 99% confidence level:

TABLE 6.13: Sample Proportion Test Result (U2R)

	SnortIDS	FL-SnortIDS	SuricataIDS	FL-SuricataIDS
SnortIDS	-	1	0.68	1
FL-SnortIDS	0*	-	0*	0.55
SuricataIDS	0.31	1	-	1
FL-SuricataIDS	0*	0.44	0*	-

1. FL-SnortIDS is better than SnortIDS and SuricataIDS to detect the U2R threat.
2. FL-SuricataIDS is better than SnortIDS and SuricataIDS to detect the U2R threat.
3. SnortIDS and SuricataIDS have same performance to detect the U2R threat.
4. FL-SnortIDS and FL-SuricataIDS have same performance to detect the U2R threat.

6.4 Experimental Results

6.4.1 Rules

As aforementioned, FIDSCC deals with outputs by specifying the inputs such as $V = \alpha_{DoS}, \alpha_{Probe}, \alpha_{R2L}$, *and* α_{U2R} as $low, medium$, and $high$ where all of these are fuzzy sets on the domain of inputs values. To give a typical example for the output of $MyCloud$, if the DoS and Probe specified as $(Low; Medium, High)$, then, FIDSCC system has to determine whether there is an attack goes to $MyCloud$ or not. After performing this, rules based on membership functions combinations will be setup in an IF THEN form. The fig. 6.10 shows the fuzzification process in particular the firing strengths of FIDSCC system where the Fuzzy Inference Engine compares each received record against all of the rules and then outputs a listing of the firing strengths of each rule for each record. These firing strengths are used to determine the likelihood that an attack is in process.

Therefore, it can be said that FIDSCC system has four phases for performing fuzzy IF THEN rules. The first step is to compare the input variables with the membership functions on the antecedent part to obtain the membership values of each linguistic label. Secondly, combining the membership values using min operator on the premise part to get the degree of fulfilment or the firing strength of each rule. Thirdly, generating the qualified consequents or each rule depending on the firing strength. Lastly, defuzzifying the crisp output by aggregating the qualified consequents.

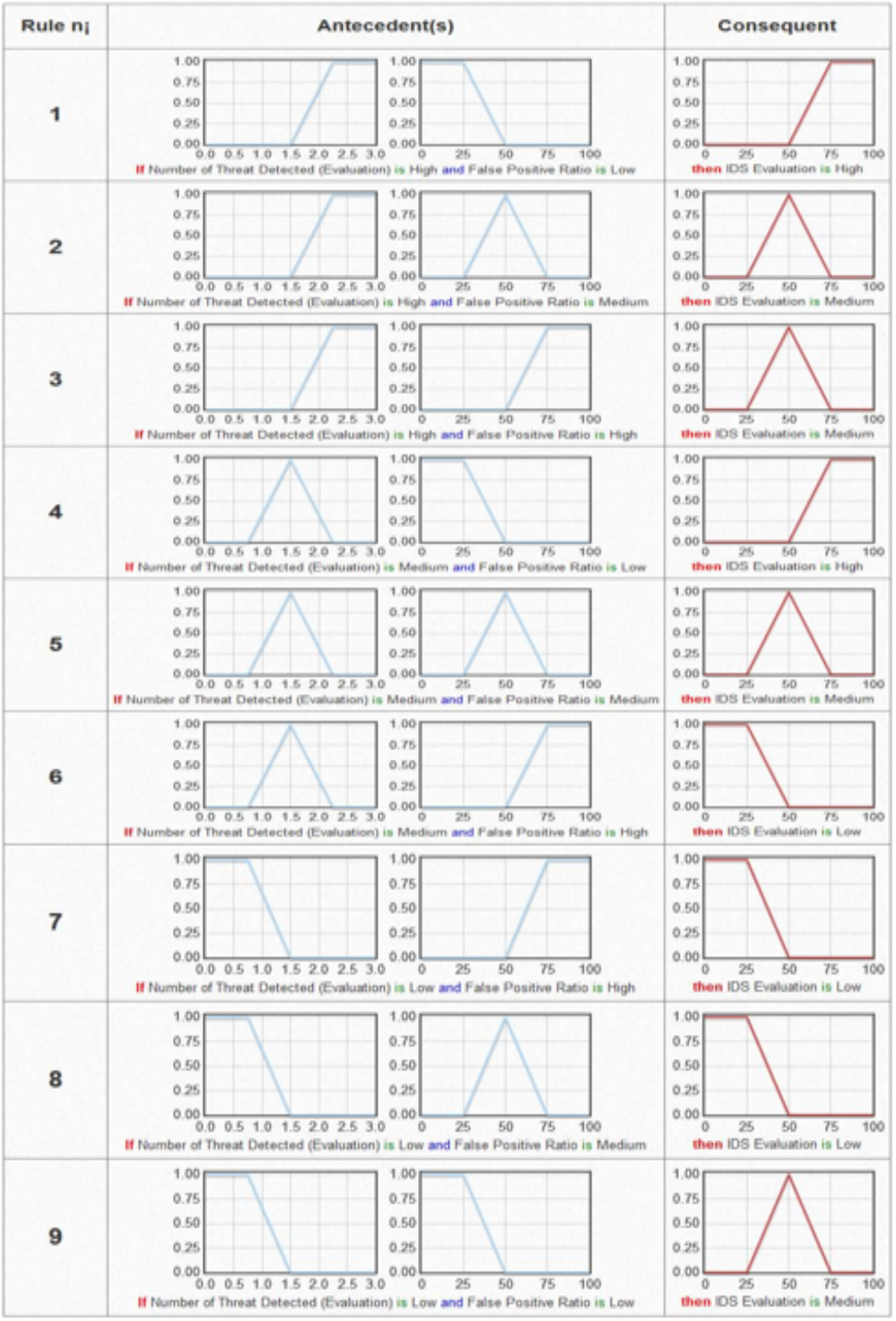


FIGURE 6.10: Firing Strength for FIDSCC System

6.4.2 FIDSCC Performance

Performance issues in *MyCloud* fundamentally relies on overall traffic patterns and peaks in the system [Brunette and Mogull, 2009]. The abnormal traffic plays an important role to affect the security performance of *MyCloud*.

6.4.2.1 Membership Functions

To simulate and measure the traffic of ISCX dataset within *MyCloud*, we built two systems;

1. **IDS-Evaluation Through Number of Threats Detected**.
2. **Overall IDS-Evaluation within** *MyCloud*

These first system was based on attacks types: DoS, Probe, U2L, and U2R. The output of this system was prioritised 3 as High, 1.5 as medium , and 0 as low. We then took the consonances of this system as an input of the second system along with another input of false positive ratio (FPR). See this first system[2] and the second system [3] in JuzzyOnline.

[2]http://goo.gl/tjsrtl
[3]http://goo.gl/BsXATF

6.4.2.2 First System: IDS-Evaluation Through Number of Threats Detected

Table 6.14: First System Inputs: IDS-Evaluation Through Number of Threats Detected

1st System of FIDSCC System's Inputs		**Type**	**Range of 1st System of FIDSCC System's Inputs**			
Alerts	**Scale**		**A**	**B**	**C**	**D**
DoS	High	Trapezoidal	1,075	1,612	2,150	2,150
	Medium	Trapezoidal	537	1,075	1,075	1,612
	Low	Trapezoidal	0	0	537	1,075
Probe	High	Trapezoidal	170,000	255,000	340,000	340,000
	Medium	Trapezoidal	85,000	170,000	170,000	255,000
	Low	Trapezoidal	0	0	85,000	170,000
R2L	High	Trapezoidal	31,000	46,500	62,000	62,000
	Medium	Trapezoidal	15,500	31,000	31,000	46,500
	Low	Trapezoidal	0	0	15,500	31,000
U2R	High	Trapezoidal	22,500	33,750	45,000	45,000
	Medium	Trapezoidal	11,250	22,500	22,500	33,750
	Low	Trapezoidal	0	0	11,250	22,500

6.4.2.3 Second System: Overall IDS-Evaluation within *MyCloud*

Table 6.15: Second System Outputs: IDS-Evaluation Through Number of Threats Detected

2nd System of FIDSCC System's Inputs		**Type**	**Range of 1st System of FIDSCC System's Inputs**			
Alerts	**Scale**		**A**	**B**	**C**	**D**
Number of Alerts Detected (NoT)	High	Trapezoidal	1.5	2.25	3	3
	Medium	Trapezoidal	0.75	1.5	1.5	2.25
	Low	Trapezoidal	0	0	0.75	1.5
False Positive Ratio (FPR)	High	Trapezoidal	50	75	100	100
	Medium	Trapezoidal	25	50	50	75
	Low	Trapezoidal	0	0	25	50

6.4.3 FIDSCC Output

TABLE 6.16: True and False Positives within *MyCloud*

System	Attacks Types	True and False Positives		False Positive Ratio (FPR)
		True Threats	False Alarms	
SnortIDS	DOS	2,083	20	0.95
	Probe	1,836	140,411	98.70
	R2L	44,458	448	0.99
	U2R	16,046	45,772	74.04
	Overall			**43.67**
FL-SnortIDS	DOS	2,083	20	0.95
	Probe	1,836	18	0.97
	R2L	44,458	448	0.99
	U2R	16,046	162	0.99
	Overall			**0.97**
SuricataIDS	DOS	2,125	21	0.97
	Probe	30	33,9314	99.99
	R2L	340	3	0.87
	U2R	218	598	73.28
	Overall			**43.78**
FL-SuricataIDS	DOS	2,125	21	0.97
	Probe	30	4	11.76
	R2L	340	3	0.87
	U2R	218	2	0.90
	Overall			**3.63**

TABLE 6.17: First System Outputs: IDS-Evaluation Through Number of Threats Detected

Detection System	Inputs for 1st System of FIDSCC System		Crisp	Centroid Type-Reduction	MyCloud Performance Number of Alerts Detected	
	Inputs	Inputs Value			Scale	Consequent
SnortIDS	DoS	2,103	[50,70]	79.88	High	3
	Probe	142,247				
	R2L	61,818				
	U2R	44,906				
FL-SnortIDS	DoS	2,103	[20,80]	49.99	Medium	1.5
	Probe	1,846				
	R2L	16,211				
	U2R	44,906				
SuricataIDS	DoS	2,146	[20,80]	49.99	Medium	1.5
	Probe	339,344				
	R2L	816				
	U2R	343				
FL-SuricataIDS	DoS	2,146	[50,70]	79.88	Medium	1.5
	Probe	34				
	R2L	220				
	U2R	343				

TABLE 6.18: Second System Outputs: Overall IDS-Evaluation within *MyCloud*

Detection System	Inputs for 2nd System of FIDSCC System		Crisp	Centroid Type-Reduction	MyCloud Performance Number of Alerts Detected	
	Inputs	Inputs Value			Scale	Consequent
SnortIDS	NoT	3	[25,78]	58.54	High	3
	FPR	43.67				
FL-SnortIDS	NoT	1.5	[50,75]	80.81	Medium	1.5
	FPR	0.97				
SuricataIDS	NoT	1.5	[20,80]	58.41	Medium	1.5
	FPR	43.78				
FL-SuricataIDS	NoT	1.5	[50,75]	80.81	Medium	1.5
	FPR	3.63				

6.5 Discussion

As it can be seen in fig. 6.11, we have found the following results of the threat detection on *MyCloud*, which are

1. SnortIDS detects the main part of the data as Probe threat.
2. FL-SnortIDS detects the main part of the data as U2R threat.
3. SuricataIDS almost detects all the data as Probe threat.
4. FL-SuricataIDS detects the main part of the data as DoS threat.

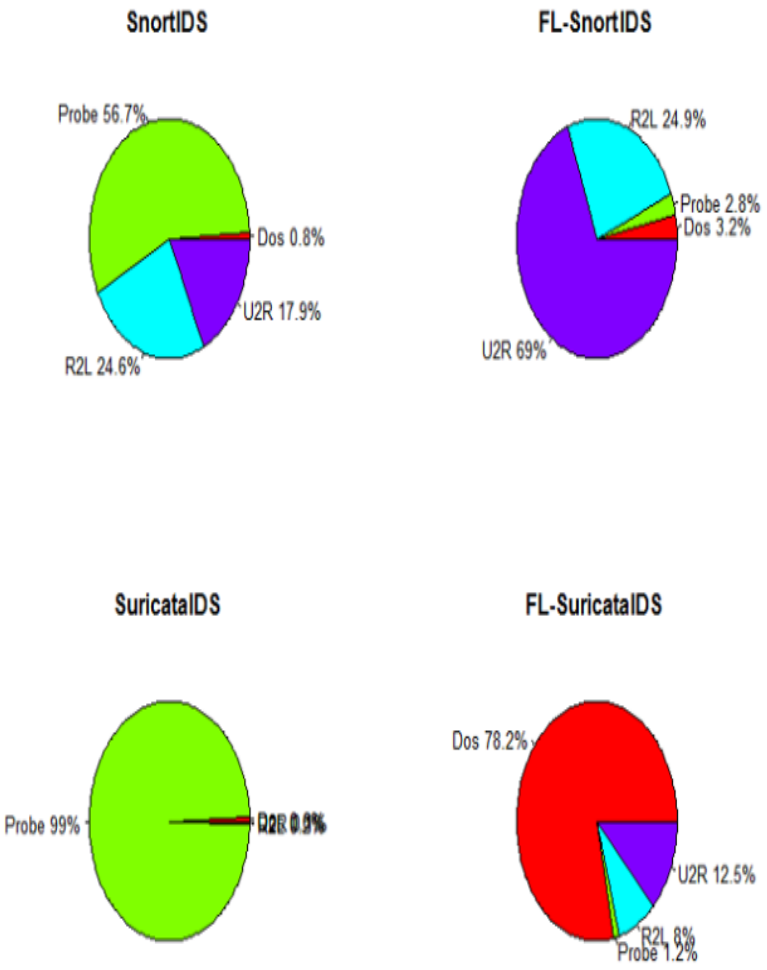


FIGURE 6.11: Detection Rate for *MyCloud* Through All Approaches

TABLE 6.19: Sample Proportion Test Result (Overall) which represents Method 1 is better than Method 2

Threat Classes		Method 2			
	Method 1	SnortIDS	FL-SnortIDS	SuricataIDS	FL-SuricataIDS
DoS	SnortIDS	-	0.50	0.46	0.46
	FL-SnortIDS	0.50	-	0.46	0.46
	SuricataIDS	0.54	0.54	-	0.50
	FL-SuricataIDS	0.54	0.54	0.50	-
Probe	SnortIDS	-	1	0*	1
	FL-SnortIDS	0*	-	0*	0.001*
	SuricataIDS	1	1	-	1
	FL-SuricataIDS	0*	0.99	0*	-
R2L	SnortIDS	-	0.50	0.59	0.59
	FL-SnortIDS	0.50	-	0.59	0.59
	SuricataIDS	0.40	0.40	-	
	FL-SuricataIDS	0.40	0.40	0.50	-
U2R	SnortIDS	-	0.50	0.68	1
	FL-SnortIDS	0*	-	0*	0.55
	SuricataIDS	0.31	1	-	1
	FL-SuricataIDS	0*	0.44	0*	-

Based on table 6.19, we have found the following results as it is shown in table 7.1;

TABLE 6.20: False Positive Ratio for *MyCloud* Through All Approaches

	DoS	**Probe**	**R2L**	**U2R**
SnortIDS vs FL-SnortIDS	Same	FL SnortIDS	Same	FL-SnortIDS
SuricataIDS vs FL-SuricataIDS	Same	FL-SuricataIDs	Same	FL-SuricataIDS
SnortIDS vs SuricataIDS	Same	SnortIDS	Same	Same
FL-SnortIDS vs FL-SuricataIDS	Same	FL-SnortIDS	Same	Same

Combining results of detection performances of all type of threat based on false positive ratio, we come up with following conclusion ranked according to their performance within *MyCloud*:

1. ***FL-SnortIDS***
2. ***FL-SuricataIDS***
3. ***SnortIDS***
4. ***SuricataIDS***

6.6 Summary

The objective of this study was to produce a model using fuzzy inference systems to trace intruders on Cloud Computing through four detection systems. To create such a model, there were two fuzzy inference systems were created: first is to evaluate the attack classes of ISCX datasets against four detection systems and another one is to assess the security performance. Combining the results of detection systems for all types of threats, based on detection rate and false positive ratio, showed that fuzzy classifiers perform better than IDS systems alone within Cloud Computing.

Chapter 7

Conclusion

7.1 Context

The focus of this research was to understand and find the best approach towards cloud security and its availability. We initially found the best rated open source intrusion detection systems on which we could run a simulated dataset and find where these systems lack or supersede others. The Information Security Centre of Excellence (ISCX) dataset was required. First step was to discuss the capability of IDS systems: SnortIDS and SuricataIDS systems to detect different kinds of attacks by exposing it to ISCX datasets and also highlight its strengths and weaknesses of both IDSs. After a successful configuration of both systems, ISCX dataset was ran on a virtual cloud called *MyCloud*. The alerts generated by both detection systems, then were standardised and formatted for better understanding of these systems. The comparison of these two IDS systems has also been presented and it has been concluded that SnortIDS system outperforms SuricataIDS system.

Table 7.1: Final Comparisons

Systems	Sensitivity	Specificity	False Alarm Ratio	Accuracy
SnortIDS vs FL-SnortIDS	Same	FL-SnortIDS	FL-SnortIDS	FL-SnortIDS
SuricataIDS vs FL-SuricataIDS	Same	FL-SuricataIDS	FL-SuricataIDS	FL-SuricataIDS
SnortIDS vs SuricataIDS	SnortIDS	Same	Same	Same
FL-SnortIDS vs FL-SuricataIDS	FL-SnortIDS	FL-SuricataIDS	FL-SuricataIDS	FL-SuricataIDS

After that, we proposed a fuzzy logic based IDSs in order to understand and enhance the security performance of these IDSs within *MyCloud* for both systems: SnortIDS and SuricataIDS, and minimise the alerts to threats. The proposed approach was simulated to demonstrate the higher level of accuracy, sensitivity and specificity achieved. The substantial decrease in false alarms was also achieved. By using fuzzy technique, unwanted alerts were removed while the others were categorised into 4 types of cyber-attacks; DoS, R2L, U2R and Probe. This improvement on SnortIDS and SuricataIDS were named to be FL-SnortIDS and FL-SuricataIDS respectively.

Results showed that the capabilities of IDSs were considerably increased after applying fuzzy logic over the alerts generated by any of the IDS systems. In particular, the main focus of this work was on the comparison between alerts generated by typical SnortIDS and SuricataIDS and similarly the alerts generate by Fuzzy Logic based SnortIDS and fuzzy logic based on SuricataIDS system. Experimental results showed the attainment of satisfactory detection rates based on the recent and most evaluated benchmark ISCX dataset on intrusions. The statistical values of accuracy, sensitivity, specificity and false alarm ratios justified that fuzzy logic based IDS works the best than any other IDS system. These results were further analysed using tools such as Mann-Whitney Test. These analyses showed these results:

- FL-SnortIDS is better than FL-SuricataIDS
- FL-SnortIDS is better than SnortIDS
- FL-SuricataIDS is better than SuricataIDS
- SnortIDS is better than SuricataIDS

Furthermore, we validated the results through four classifier algorithms such as OneR, Naive Base, Decsion Tree, and KNN. The purpose of using data mining and conducting a comparative study was to find out the best available classification technique applied to *MyCloud* systems' results. This study, which was performed by analysing the ISCX dataset, observes the performance of classification algorithms and shows that Decision Trees classifier was the best at classifying the intrusions in SnortIDS, SuricataIDS, and FL-SnortIDS systems. Out of which OneR has outperformed with respect to FL-SuricataIDS system, whereas Naive Bayes consumes least time in all results compared to others. The main objective was to get a better rate of intrusion detection for the classifier to reduce the rate of false negatives. In terms of the performance metrics upon

MyCloud systems' results, all the metrics for the first three comparisons were clear where SnortIDS does better than FL-SnortIDS, SuricataIDS was better than FL-SuricataIDS, and SuricataIDS was better than FL-SnortIDS. However, for the fourth comparison between SnortIDS vs SuricataIDS, the comparison was not clear where we found SnortIDS was better than SuricataIDS in terms of precision while other criteria perform better in any algorithm chosen. This means that the accuracy and F-measure of SuricataIDS were better than SnortIDS on behalf of OneR, Naive Bayes and K-NN while in sensitivity and specificity of SuricataIDS were better than SnortIDS in Decision Tree, OneR and K-NN. Therefore, based on the performance metrics upon these systems, we found that SuricataIDS system shows more promising detections than SnortIDS. For the fifth and sixth comparisons between FL-SnortIDS vs FL-SuricataIDS and SnortIDS and FL-SuricataIDS, we found FL-SnortIDS outweigh FL-SuricataIDS and SnortIDS outweigh FL-SuricataIDS in three algorithms except OneR. Hence, we sum up that FL-SnortIDS and SnortIDS show more promising detections than FL-SuricataIDS respectively.

Overall, we conclude that Cloud Computing has some concerns regarding the detection of security threats such as false positive and false negative, and hence, enhancing cloud security is very important for providers, users and organisations to improve the security performance and maximise the detection rate through novel approaches. In this thesis, conventional approaches such as IDS (SnortIDS or SuricataIDS) showed that they are not flexible to the uncertainty of intrusions. By proposing FIDSCC model that comprises of Type-1 fuzzy logic (T1FL) technique with IDS when compared to IDS alone, we found that this approach within Cloud Computing is more secure than the other IDS tools: Snort and Suricate. The effectiveness of this model is that FIDSCC can provide a better alternative to detecting threats and reducing the false positive rates more than the other conventional approaches.

However, a possible alternative option that can be adopted using the comparative work in this thesis is to implement Type-2 fuzzy logic based IDS using ISCX dataset. Indeed, this will reduce the generality of the whole approach by defining new fuzzy sets and rules. This technique would be fast and robust Cloud Computing by identifying new and unusual attacks but it requires a deeper understanding of the nature and the common features of FIDSCC model. Also, a deeper understanding of different emerging attacks and techniques used by network or forensic analyst is required in order to determine and restrict the intrusion in Cloud Computing.

7.2 Summary of Contributions

In chapter 4, we conducted a comparative study for IDS fuzzy classifiers in order to enhance the security that IDS provides. This is due to the level of threats that has been noticeably grown and become a serious danger t towards the services' providers, e.g., web services providers, email services providers, cloud service providers etc., whom have to deal with millions of users per second. Thus, in order to deal with this much number of users is a big challenge but, detection and prevention of such kinds of threats is even more challenging and also vital due to the fact that those threats might cause a severe loss to the service providers in terms of privacy leakage or unavailability of the services to the users. The chapter 4 was conducted in order to incorporate this issue, several IDSs: SnortIDS and SuricataIDS have been developed in which they differ in their detection capabilities, performance and accuracy. SnortIDS and SuricataIDS are well-known IDS systems that are used worldwide. The chapter 4 discusses the functionality, working and the capability of these two IDS systems to detect the intrusions within *MyCloud* network and also presents a comparison analysis of these two systems for the detection of different kinds of cyber-attacks. Furthermore, it also proposes a Fuzzy-Logic engine that enhances the performance and accuracy of these two systems considerably, in terms of increased accuracy, specificity, sensitivity and reduced false alarms. Some experiments in this chapter have also been conducted to analyse the performance of these systems by using ISCX dataset and results indicate that FL based SnortIDS system outperforms FL based SuricataIDS.

In chapter 5, we used WEKA as a data miner to analyse the results of IDSs' systems and fuzzy classifiers that were done in In chapter 4. WEKA played a very significant role in evolving the IDSs and fuzzy classifiers. The dataset of IDSs were classified into normal and abnormal traffic in order for generated alerts to detect threats. In this crefChapter5, we utilised the most common classification algorithms: Decision Tree (J48), Naive Bayes, OneR, and K-Nearest Neighbour (K-NN). These algorithms were chosen after investigating the most effective classification algorithms that are widely used. The aim of this study was to present a comparative study for the performance of each system that was gained from our previous experiments: SnortIDS, SuricataIDS, FL-SnortIDS, and FL-SuricataIDS in order to test which classifier algorithm is the best for our systems' results, and investigate which system presents significant results. The performance of these classification algorithms was evaluated using 10-fold cross validation and experiments and assessments of these methods were performed in the WEKA environment using the ISCX dataset.

In chapter 6, the rise of traffic in *MyCloud* was notable, and all the recorded information was not useful to analyse; thus, existing network forensics approaches were simple and disadvantageous to differentiate the detection rate within *MyCloud* and reduce the false positives ratio. We, as administrators, would have faced difficulty in managing the damaged state of system without knowledge expert. Therefore, this chapter 6 produced an approach using fuzzy inference systems to trace intruders on *MyCloud*. There were two fuzzy inference systems were created: first is to evaluate the attack classes of ISCX within *MyCloud* and another is to assess *MyCloud* performance. Certain tests have been made for the comparison purposes amongst IDSs' systems and Fuzzy classifiers. These were satisfactory tests where the combining results of detection performances of all type of threats based on detection rate and false positive ratio showed that fuzzy classifiers do better than IDSs' systems within *MyCloud*.

7.3 Extensions and Future Work

This work is by no means perfect and a number of possible improvements and extensions can be made to further enhance the emerging attacks. This is because of the fact that this work goes a long way in understanding different emerging attacks and techniques used by network or forensic analyst to try to determine and restrict the intrusion in their networks or cloud computing. It can be noticed that such a study shows that fuzzy logic along with the legacy intrusion detection systems yields better results and facilitates the network administrators to mitigate the issues. This technique in future can be used for other open source intrusion detection systems to yield better results for that system. Furthermore, the tool developed to format and removal of unwanted alerts generated by IDS which are of no use can be used by cyber response teams. Different logs from different sources are required to analyse and respond to any cyber emergency e.g. when a datacenter is being attacked by distributed denial of service attacks, the response team collects different types of logs including system logs, network logs, firewall logs, IDS logs, and router logs. These are all in different format and have many unwanted alerts and logs which are not required. Utilising the technique in the above mentioned studies, these logs could be unified and unwanted alerts or logs may be discarded. This approach makes it very easy for response team to collect relevant information for the incident and focus their effort on solving the problem rather than finding it.

Furthermore, a new genetic algorithm (GA) are being developed and extensive researches are being carried out to analyse the huge amount of data being transported over the

networks. This is because GA shall help in finding appropriate fuzzy rules; therefore, fuzzy logic along with genetic based approach might give more powerful performance to increase incidents of attacks of network to secure data [Cerli and Ramamoorthy, 2015]. In the future, fuzzy logic based IDS incorporated with GA will be designed and implemented to help understand the changing networks and complex attacks. This method might open a way for researchers to improve the IDS performance in any environment.

Appendix A

Experimental Settings

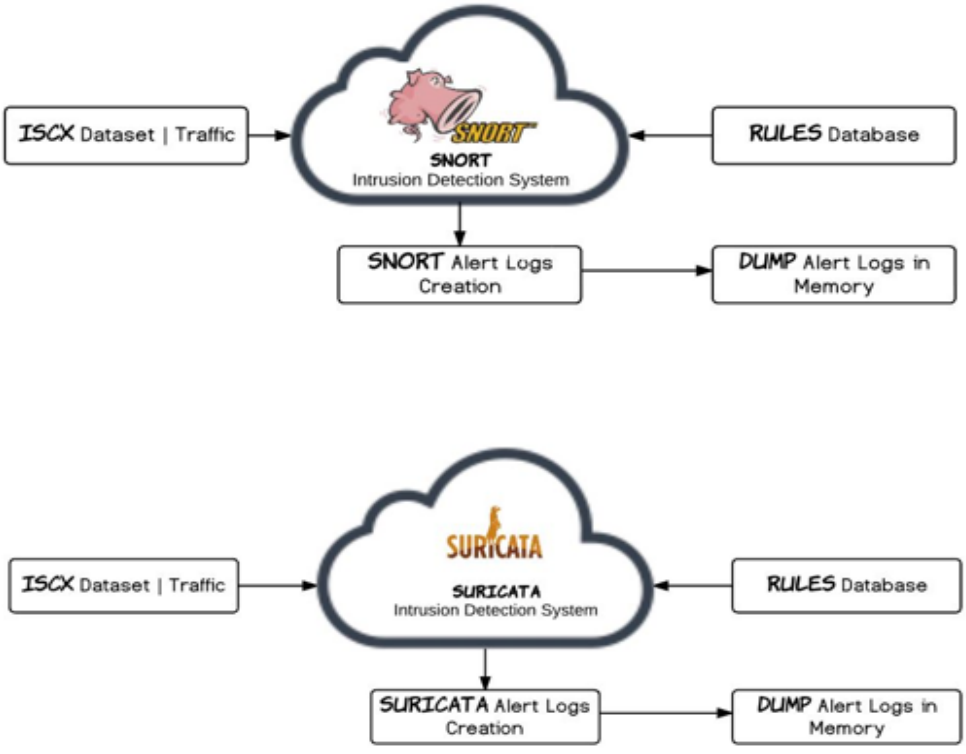


Figure A.1: Snort and Suricata Alert Logs

A.0.1 Snort (SnortIDS)

First of all, packages for SnortIDS was downloaded, using these two commands in the source that were downloaded:

1. *wget https://www.snort.org/downloads/snort/daq-2.0.6.tar.gz*
2. *wget https://www.snort.org/downloads/snort/snort-2.9.8.2.tar.gz*

Wget is a utility to download files from web; it supports HTTP, FTP and HTTPS. After this command ran successfully, we got ready to configure packages.

1. **Configuring Data Acquisition (DAQ)**

   DAQ is Data Acquisition library used by SnortIDS replacing the basic data acquisition library such as libpcap. Previous versions of data acquisition used interrupt based packet capturing, meaning that for every incoming packet, an interrupt was generated by hardware. This interrupt was captured by kernel and forwarded to application layer, where the interrupt was handled. With the increase in link speed, such interrupt based data acquisitions were not suitable any-more, so poll mode drivers were introduced. After a certain period of time, these drivers poll the NICs for any incoming packet, increasing the overall efficiency of system and application by reducing OS level interrupts. Before configuring DAQ, we made sure that all the necessary packages were installed. The DAQ library was configured using these steps.

   (a) *tar xvfz daq-2.0.6.tar.gz*

   (b) *cd daq-2.0.6*

   (c) *./configure && make && sudo make install*

2. **Configuring SnortIDS**

   Once DAQ was configured, we installed SnortIDS on the system; the steps were as it is shown in the commands below.

   (a) *tar xvfz snort-2.9.8.2.tar.gz*

   (b) *cd snort-2.9.8.2*

   (c) *./configure –enable-sourcefire && make && sudo make install*

   When SnortIDS was installed, basic directories was created, where rules and logs are to be maintained. This may change depends upon the user preference. The commands are provided below:

(a) *mkdir /usr/local/etc/snort*

(b) *mkdir /usr/local/etc/snort/rules*

(c) *mkdir /var/log/snort*

(d) *mkdir /usr/local/lib/snort_dynamicrules*

After these directories were made, from Snort community, all rules available for IDS were downloaded. These rules help SnortIDS determine either the connection is malicious or not. This completes the basic installation of SnortIDS. SnortIDS has 3 basic modes which are,

(a) sniffer mode to read and display the packets

(b) packet logger mode dumps these packets on disc

(c) Network Intrusion Detection System (NIDS) Mode performs analysis on the network and detects any anomalies.

3. **Running SnortIDS**

After a successful configuration of SnortIDS system, and incorporating the community rules, ISCX dataset was run against SnortIDS using this command:

snort –pcap-dir="/IscxDataset/11Jun" –c /usr/local/etc/snort.conf

This command runs SnortIDS on a directory full of pcaps, which we have splitted, with the rules present in snort.conf

A.0.2 Suricata (SuricataIDS)

First of all, packages for SuricataIDS was downloaded, using these commands as the latest version of SuricataIDS was downloaded using these commands:

1. *apt-get -y install libpcre3 libpcre3-dbg libpcre3-dev*
2. *build-essential autoconf automake libtool libpcap-dev libnet1-dev*
3. *libyaml-0-2 libyaml-dev zlib1g zlib1g-dev libcap-ng-dev libcap-ng0*
4. *make libmagic-dev libjansson-dev libjansson4 pkg-config*
5. *sudo apt-get -y install libnetfilter-queue-dev libnetfilter-queue1 libnfnetlink-dev libnfnetlink0*

6. *VER=3.0.1*
7. *wget"http://www.openinfosecfoundation.org/download/suricata-$VER.tar.gz"*
8. *tar -xvzf "suricata-$VER.tar.gz"*
9. *cd "suricata-$VER"*
10. *sudo apt-get update*
11. *sudo apt-get install suricata*
12. *./configure –enable-nfqueue –prefix=/usr –sysconfdir=/etc –localstatedir=/var*

After using these commands successfully, the latest stable version of SuricataIDS system was downloaded on Ubuntu.

1. **Basic Setup**

   After SuricataIDS was successfully installed, some basic steps were followed to get the working environment such as, to create a new directory for SuricataIDS and its configuration files. In order to do this, these commands were applied:

   (a) *sudo mkdir /var/log/suricata*
   (b) *sudo mkdir /etc/suricata*
   (c) *sudo cp classification.config /etc/suricata*
   (d) *sudo cp reference.config /etc/suricata*
   (e) *sudo cp suricata.yaml /etc/suricata*
   (f) *set the environmental variable*

   Finally, to start SuricataIDS, the below command was applied;

   (g) *sudo suricata -c /etc/suricata/suricata.yaml -i wlan0 –init-errors-fatal*

2. **Applying Rules on SuricataIDS**

   The detection rules were applied on the basis of which, the malicious activities were detected. These rules have been defined by the development team that was downloaded automatically using some automatic software such as, Pulled Pork and Oinkmaster. The following commands were used to setup the rules for SuricataIDS by using Oinkmaster mainly

   (a) *sudo apt-get install oinkmaster*

(b) *sudo mkdir /etc/suricata/rules*

(c) *cd /etc*

(d) *sudo oinkmaster -C /etc/oinkmaster.conf -o /etc/suricata/rules*

(e) *sudo nano /etc/suricata/suricata.yaml*

(f) *suricata -c /etc/suricata/suricata.yaml -i wlan0 (or eth0)*

3. **Running SuricataIDS**

   After the successful configuration of SuricataIDS on the system, and incorporating the community rules, ISCX dataset was run against SuricataIDS using this command:

   suricata –r /IscxDataset/11Jun.pcap –c /usr/local/etc/suricata.conf

   This command ran SuricataIDS on a single pcap file, with the rules present in suricata.conf

A.1 Creating *MyCloud*

MyCloud is vCloud infrastructure that includes the hardware and software components such as servers, storage, virtualisation software and operating systems. These components were used to build *MyCloud*. The below headings are some of snapshots of the *MyCloud* Infrastructure.

A.1.1 vCloud Director:

This is a management tool specified for all types of vCloud Infrastructure. fig. A.2 shows the main interface of the vCloud management tool.

A.1.2 vApps in Organisation:

A vApp is a virtual application comprised of virtual machines, their networks and their policies. While vApps were concepts that were present previously in VMware Desktop, vCloud Express and Lab Manager, they never truly represented isolated, independent virtual applications. A vApp can be connected to a vApp specific network or to an Organisation Network. See fig. A.3.

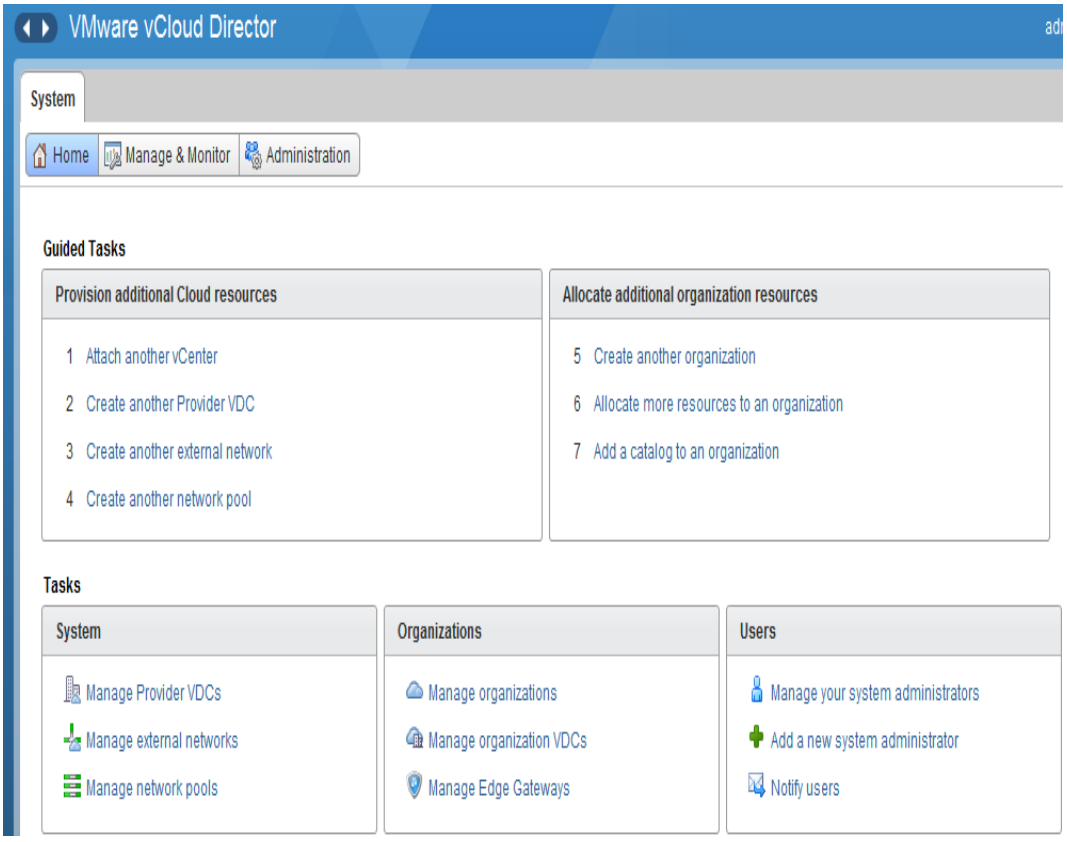


FIGURE A.2: vCloud Management Tool

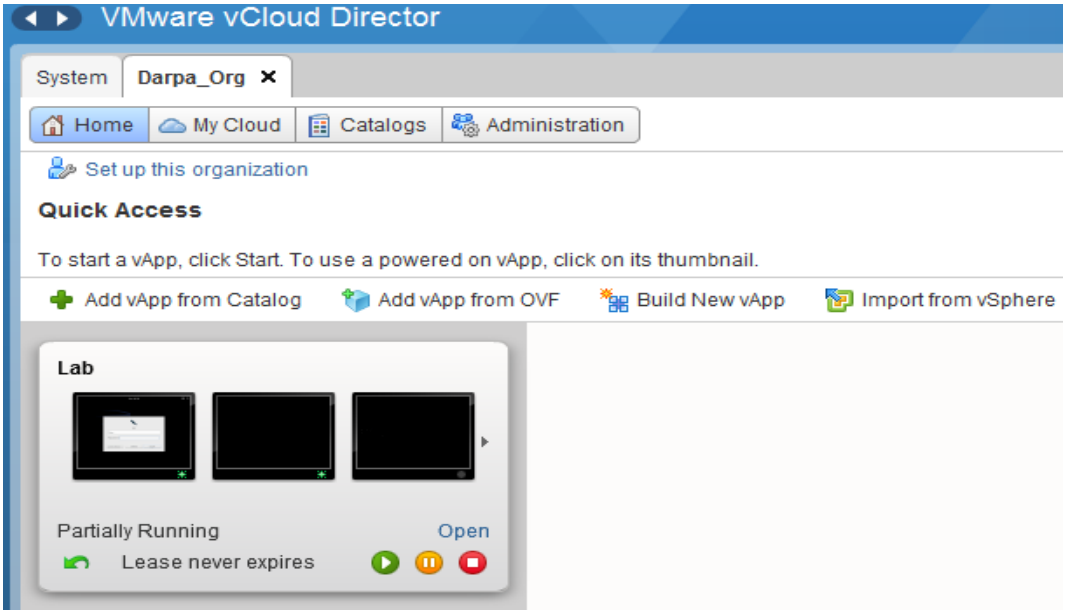


FIGURE A.3: vApp in Organisation within Cloud Environment

A.1.3 IDS Systems Deployment:

As shown in fig. A.4, vApp provides a graphical view of the virtual machines and networks within *MyCloud* environment. IDS/IPS systems were deployed and converted into *MyCloud* using VMware vCentre Converter.

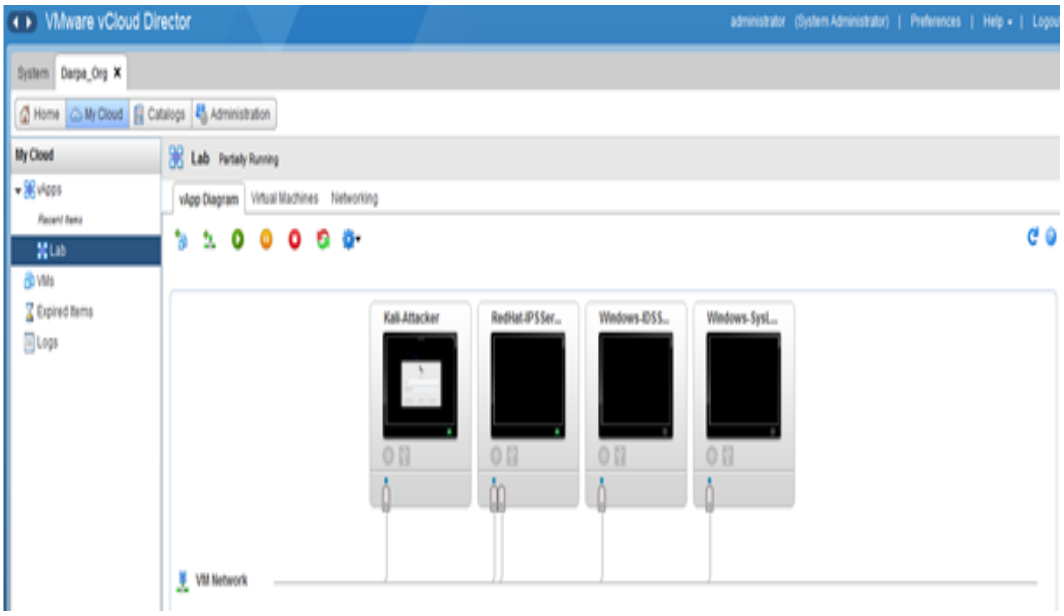


FIGURE A.4: The Lab Deployment in *MyCloud*

A.1.4 Network Pools Between Two ESXis:

In order to create organisation Networks or vApp Networks, a pool of network resources had to be available. These network pools were created in advance of the creation of Org and vApp networks in *MyCloud*. This is because of the fact that if they do not exist, the only network option available to an organisation is the direct connect to the provider network. See fig. A.5.

A.1.5 VMware vCenter Server:

vCenter Server provides a centralised platform for managing the vSphere environments in order to automate and deliver a virtual infrastructure with confidence. See fig. A.6.

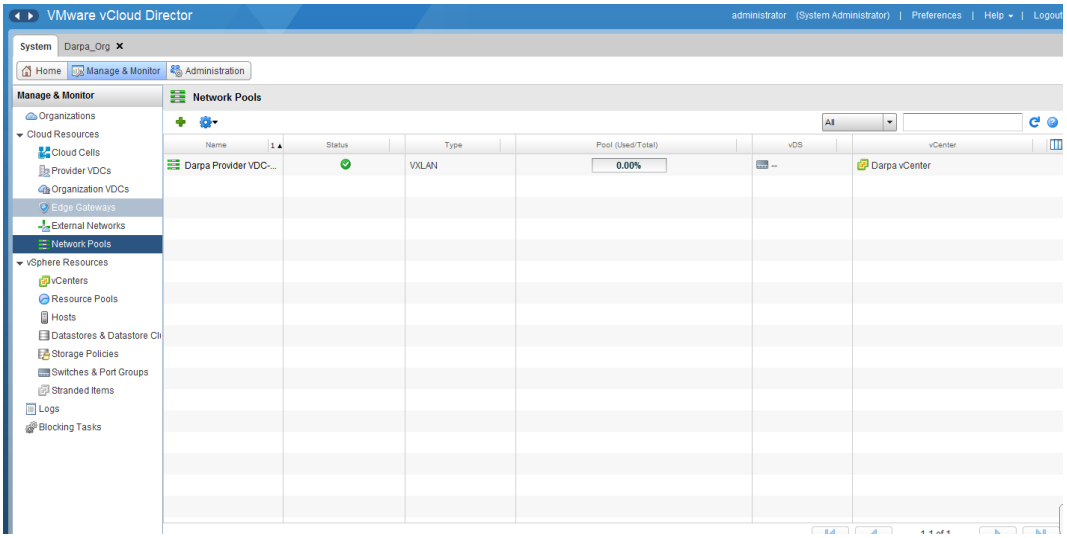


FIGURE A.5: vCloud Network Pools

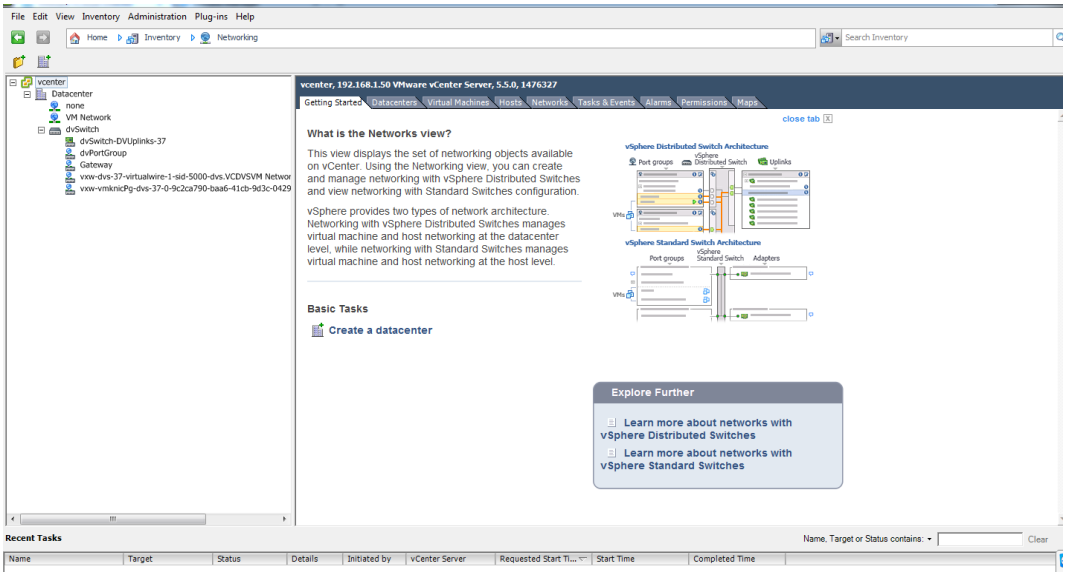


FIGURE A.6: vCenter Server Environment

A.1.6 The Storage Profiles:

The storage profiles feature was first introduced in vSphere. This is because it enables users to map the capabilities of a storage system to a profile. This was done although I was able to (1) create a user defined storage capability, (2) assign a storage capability, (3) create a virtual machine storage profile, and (4) enable the storage profile on the host/cluster at the vSphere layer only. The storage profiles were added to the provider virtual datacenter (VDC). Then subset of storage profiles was added to an organisation VDC. The storage profile was selected to match the users' requirements, and ensure

users that the created virtual machines were utilised the appropriate datastore. See fig. A.7.

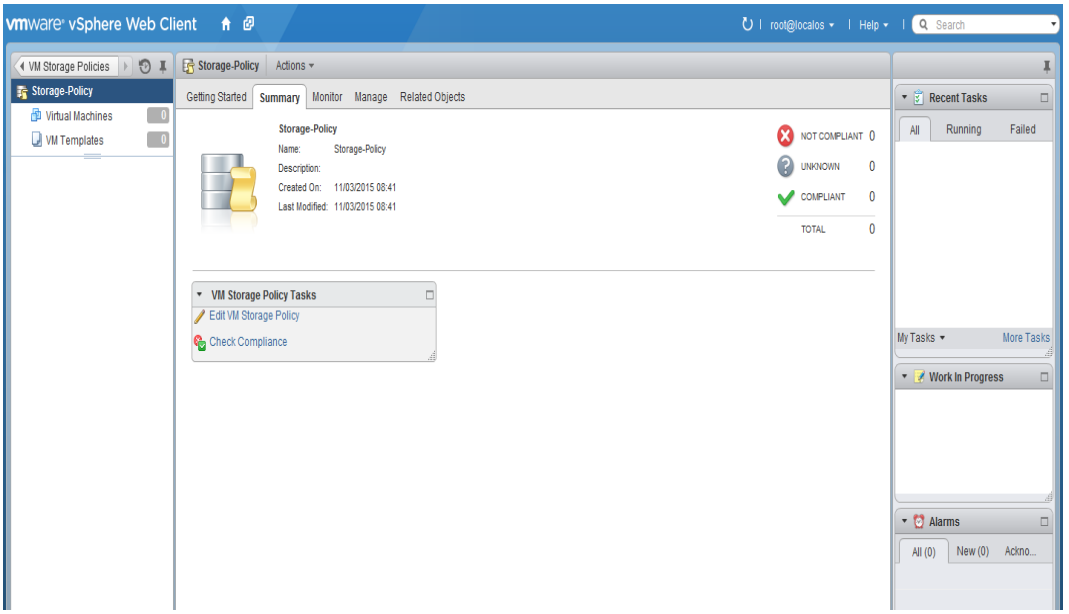


FIGURE A.7: vCenter Storage Profile

A.1.7 vCloud Networking and Security appliance (vShield):

The vShield is a group of networking and security products for virtualised IT infrastructures, comprising vShield Manager, vShield Edge, vShield Zones, vShield App, vShield Data Security and vShield Endpoint. Nowadays, vShield is known as vCloud Network and Security. See fig. A.8.

A.2 Installation and Configuration of *MyCloud*

A.2.1 ISP Router

Name of the router was **TalkTalk**. As this was configured with LAN IP: 192.168.1.254, so it is *HomeLab* gateway. Also, Native was configured to CCS IP: 192.168.1.251

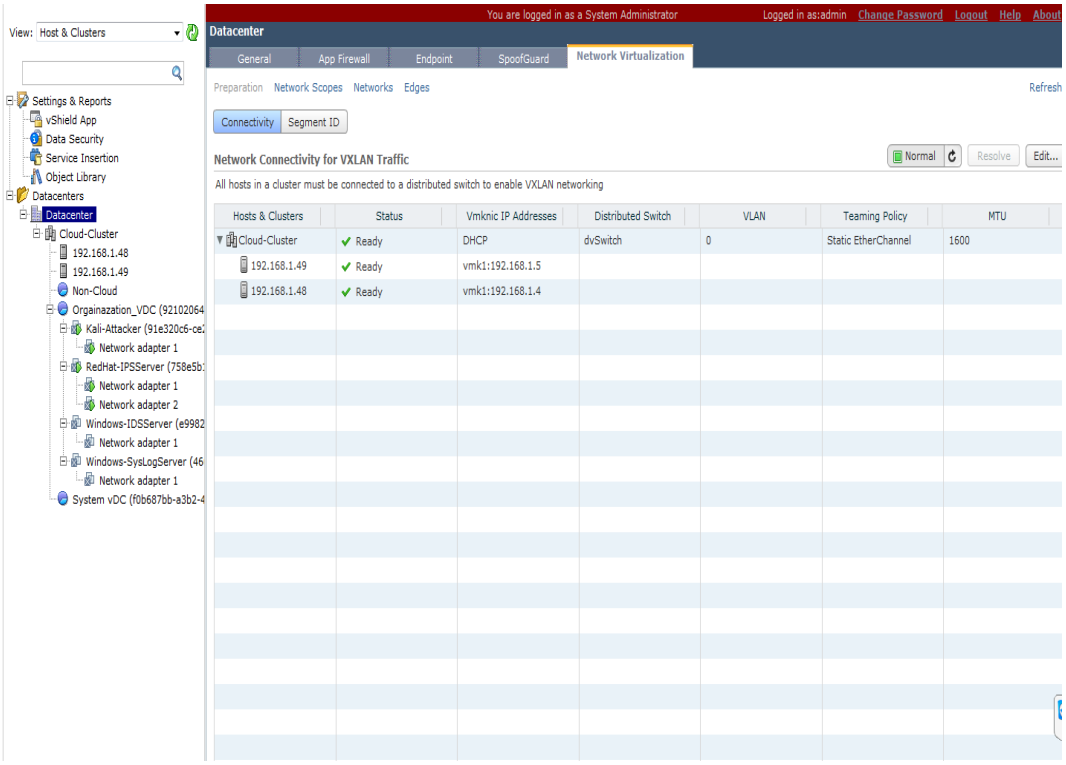


FIGURE A.8: VMware vShield

A.2.2 Active Directory

Initially, Windows Server 2008 R2 with Roles, e.g. Active Directory and DNS for Lab Authentication, was installed. Also, such a server we configured this as time server through. The following were used by Registry Editor command "**regedit**",

1. a command was written as shown in fig. A.9, then modified the configuration; **Enable dword=1**

```
C:\>Heky_Local_Machine\System\CurrentControlSet\Services\w32Time\TimeProviders\N
TPServer
```

FIGURE A.9: The First Command in **regedit**

2. a command was written as shown in fig. A.10, then modified the configuration; **Announceflags dword = 1**

```
C:\>Heky_Local_Machine\System\CurrentControlSet\Services\w32Time\config
```

FIGURE A.10: The Second Command in **regedit**

3. Then these commands were run **net stop w32time** & **net start w32time** in order to stop and start NTP.

A.2.3 Service DNS

DNS as part of the domain set-up procedure allows Windows to configure DNS automatically. Once the domain configured, the following reverse lookup zone were added: 192.168.1.x. including vcloud.local forward lookup zone, and then allowing the reverse entries to be created automatically.

- Name: ad with IP: 192.168.1.51 (**Active Directory**)
- Name: **esxi01** with IP: 192.168.1.48 (**ESXi Server**)
- Name: **esxi02** with IP: 192.168.1.49 (**ESXi Server**)
- Name: **vcva** with IP: 192.168.1.50 (**vCenter**)
- Name: **vcns** with IP: 192.168.1.54 (**vShield**)
- Name: **vcd** with IP: 192.168.1.66 (**vCloud**)

A.2.4 vSphere ESXi 5.5

As explained earlier ESXi01 and ESXi02 were installed on bare metal hardware with local storage.

- the DNS and Routing Information for an ESXi Host were configured .
- the ESXi Host was configured to use directory services.

A.2.5 vCenter Server Appliance 5.5

This is an appliance Open Virtual Format (OVF) that allows to deploy vCentre Server in the PC on top of VMware Workstations. The following vCentre Server software components were installed:

1. vCenter Single Sign On

2. VMware vCenter Inventory Service
3. VMware vCenter Server
4. VMware vSphere Web Client
5. VMware vSphere Syslog Collector

For the configuration details: a Data Centre Object was created, then, both ESXi Hosts were added to the vCentre Server Inventory. After that, both ESXi Hosts were configured as an NTP Client. A Cluster named "Cloud-Cluster" was then created and added both ESXi into this cluster. After that, Cluster was configured noticing that vCloud Director requires DRS to be enabled. DRS Automation was set in order to be fully automated. The Resource Pool was created and configured with default settings. Then vDistributed Switch was established with default name "dvswitch" and Number of uplink ports = 2. Both ESXi physical adaptors were added to dvswitch. Finally, Portgroup name "Gateway" was created by creating Storage Profile. This was done by creating Tags to both ESXi local datastore and adding it in to Category.

A.2.6 vCloud Networking and Security appliance (vShield)

This is also an appliance OVF in order to deploy vShield on the top of vData Centre Server. This was done by doing the following;

1. vCentre Server: I added require credentials to link it to vShield Manager.
2. Lookup Server: I added require credentials
3. DNS Server was enabled.
4. NTP Server was enabled.
5. Syslog Server was enabled.

A.2.7 vCloud Director Appliance 5.5

vCloud Director was applied on the top of vData Centre Server as an appliance OVF. See table 3.2.

Appendix B

The Meaning of IDS Systems' Alerts

As explained in section 3.2, The table B.1 illustrates the meaning of the content of each output for the alerts of IDS systems that are shown in fig. 3.3 and fig. 3.4.

Table B.1: The Meaning of IDS Systems' Alerts

Identifiers	Meanings
SnortIDS Log Explanations	
[**]	Generator ID, if multiple SNORT IDS are deployed it will show numbers
[129:12:1]	This is the signature ID, currently it means **TCP Small Segment Threshold Exceeded** Which is also described in next field
Classification	It tells what type of Alert is generated
Priority	Ranging from 1-4 the priority tells us the severity of alert. 1 being highest in severity and 4 being lowest.
Detail	After this basic classification, this log tells us the stream generating alerts
SuricataIDS Log Explanations	
06/11/2010-03:01:09.045867	The date and time of packet received at Network Interface Card
[**]	Generator ID, if multiple SURICATA IDS are deployed it will show numbers
[1:2260002:1]	This is the signature ID, currently it means **Generic Protocol Command Decode** Which is also described in next field
Classification	It tells what type of Alert is generated
Priority	Ranging from 1-4 the priority tells us the severity of alert. 1 being highest in severity and 4 being lowest.
Detail	After this basic classification, this log tells us the stream generating alerts

References

Alhamad, M., Dillon, T., Chang, E., 2011. A trust-evaluation metric for cloud applications, in: International Association of Computer Science and Information Technology (IACSIT Press), pp. 416–423.

Alqahtani, S.M., Balushi, M.A., John, R., 2014a. An intelligent intrusion detection system for cloud computing (sidscc), in: Computational Science and Computational Intelligence (CSCI), 2014 International Conference on, IEEE. pp. 135–141.

Alqahtani, S.M., Balushi, M.A., John, R., 2014b. An intelligent intrusion prevention system for cloud computing (sipscc), in: Computational Science and Computational Intelligence (CSCI), 2014 International Conference on, IEEE. pp. 152–158.

Anderson, J.P., 1980. Computer security threat monitoring and surveillance. Technical Report 51-73. James P. Anderson Company, Fort Washington, Pennsylvania.

Anonymous, 2012. Cloud computing is not a new idea!,¨[online]. february 13, 2012. URL: `http://astrocompute.wordpress.com/2012/02/13/cloud-computing-is-not-a-new-idea/`.

Augustin, M., Baláž, A., 2011. Intrusion detection with early recognition of encrypted application, in: 2011 15th IEEE International Conference on Intelligent Engineering Systems, IEEE. pp. 245–247.

Bakshi, A., Yogesh, B., 2010. Securing cloud from ddos attacks using intrusion detection system in virtual machine, in: Communication Software and Networks, 2010. ICCSN'10. Second International Conference on, IEEE. pp. 260–264.

Barker, S.K., Shenoy, P., 2010. Empirical evaluation of latency-sensitive application performance in the cloud, in: Proceedings of the first annual ACM SIGMM conference on Multimedia systems, ACM. pp. 35–46.

Barnatt, C., 2010. A brief guide to cloud computing: An essential guide to the next computing revolution., Robinson. pp. 52–55.

Bezborodov, S., et al., 2016. Intrusion detection systems and intrusion prevention system with snort provided by security onion., Mikkelin ammattikorkeakoulu. pp. 1–12.

Bosin, A., Dessì, N., Pes, B., 2009. A service based approach to a new generation of intrusion detection systems, in: on Web Services, 2008. ECOWS'08. IEEE Sixth European Conference, IEEE. pp. 215–224.

Bray, R., Cid, D., Hay, A., 2009. Ossec host-based intrusion detection guide, Syngress. pp. 190–216.

Brodkin, J., 2009. Gartner: Seven cloud-computing security risks. URL: `http://www.networkworld.com/article/2281535/data-center/gartner--seven-cloud-computing-security-risks.html`.

Brunette, G., Mogull, R., 2009. Security guidance for critical areas of focus in cloud computing v2. 1. Technical Report 1-76. Cloud Security Alliance.

Brutch, P., Ko, C., 2013. Challenges in intrusion detection for wireless ad-hoc networks, in: Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on, IEEE. pp. 368–373.

Burkadze, K., 2016. Cyber security and international law, Journal of Technical Science and Technologies. pp. 5–10.

Burks, D., 2012. Doug Burks: Security Onion Network Security monitoring in 229 minutes. URL: `https://www.youtube.com/watch?v=mazSRVFYmLQ`.

Bustince, H., Barrenechea, E., Pagola, M., Fernandez, J., Xu, Z., Bedregal, B., Montero, J., Hagras, H., Herrera, F., De Baets, B., 2016. A historical account of types of fuzzy sets and their relationships. IEEE Transactions on Fuzzy Systems 24, 179–194.

Cai-dong, G., Yu, Z., Jian-ping, W., 2009. Image knowledge management and rapid precise image-mining technology investigation in internal cloud computing, in: Information Science and Engineering (ICISE), 2009 1st International Conference on, IEEE. pp. 2940–2942.

Casey G., Cegielski, L.J.F.A.W.Y..H.B., 2012. Adoption of cloud computing technologies in supply chains, IEEE. pp. 184–211.

Catteddu, D., 2010. Cloud computing: benefits, risks and recommendations for information security, in: Web Application Security. Springer, pp. 17–17.

Cerli, A.A., Ramamoorthy, S., 2015. Intrusion detection system by combining fuzzy logic with genetic algorithm. Global Journal of Pure and Applied Mathematics (GJPAM) 11.

Chaplot, S., 2015. Basics about cloud computing and microsoft azure URL: `https://www.linkedin.com/pulse/basics-cloud-computing-microsoft-azure-siddhartha-chaplot`.

Chauhan, H., Kumar, V., Pundir, S., Pilli, E.S., 2013. A comparative study of classification techniques for intrusion detection, in: Computational and Business Intelligence (ISCBI), 2013 International Symposium on, IEEE. pp. 40–43.

Chen, Q., Deng, Q.n., 2009. Cloud computing and its key techniques, Journal of Computer Applications. pp. 25–65.

Chiba, Z., Abghour, N., Moussaid, K., El Omri, A., Rida, M., 2016. A survey of intrusion detection systems for cloud computing environment, in: Engineering & MIS (ICEMIS), International Conference on, IEEE. pp. 1–13.

Cloud, H., 2011. The nist definition of cloud computing, National Institute of Science and Technology. Special Publication, 800. pp. 33–37.

Cox, E., 1992. Fuzzy fundamentals. IEEE spectrum 29, 58–61.

Dawoud, W., Takouna, I., Meinel, C., 2010. Infrastructure as a service security: Challenges and solutions, in: Informatics and Systems (INFOS), 2010 The 7th International Conference on, IEEE. pp. 1–8.

Demirkan, H., Cheng, H.K., Bandyopadhyay, S., 2010. Coordination strategies in an saas supply chain, ME Sharpe. Journal of Management Information Systems. pp. 119–143.

Durowoju, O.A., Chan, H.K., Wang, X., 2011. The impact of security and scalability of cloud service on supply chain performance., pp. 27–31.

Feng, G., 2006. A survey on analysis and design of model-based fuzzy control systems. IEEE Transactions on Fuzzy systems 14, 676–697.

Ferguson, D.F., Hadar, E., 2011. Optimizing the it business supply chain utilizing cloud computing, in: Emerging Technologies for a Smarter World (CEWIT), 2011 8th International Conference & Expo on, IEEE. pp. 1–6.

Foster, I., Zhao, Y., Raicu, I., Lu, S., 2009. Cloud computing and grid computing 360-degree compared, in: Grid Computing Environments Workshop, 2008. GCE'08, Ieee. pp. 1–10.

Frawley, W.J., Piatetsky-Shapiro, G., Matheus, C.J., 1992. Knowledge discovery in databases: An overview, pp. 57–63.

Fries, T.P., 2009. A fuzzy-genetic approach to network intrusion detection, in: Proceedings of the 10th annual conference companion on Genetic and evolutionary computation, ACM. pp. 2141–2146.

Gagnon, S., Nabelsi, V., Passerini, K., Cakici, K., 2011. The next web apps architecture: Challenges for saas vendors, Institute of Electrical and Electronics Engineers, Inc.,— y USA United States. pp. 44–50.

Garber, L., 2010. Denial-of-service attacks rip the internet, IEEE Computer. pp. 12–17.

Google Patents, U..B., 2009. Intrusion Detection System. URL: `http://www.google.co.uk/patents/US6405318`.

Graham, D.E., 2010. Cyber threats and the law of war, HeinOnline, J. Nat'l Sec. L. & Pol'y. pp. 87–99.

Hagras, H., 2007. Type-2 flcs: A new generation of fuzzy controllers. IEEE Computational Intelligence Magazine 2, 30–43.

Hagras, H., Wagner, C., 2012. Towards the wide spread use of type-2 fuzzy logic systems in real world applications. IEEE Computational Intelligence Magazine 7, 14–24.

Hagras, H.A., 2004. A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots. IEEE Transactions on fuzzy systems 12, 524–539.

Hall, J.A., Liedtka, S.L., 2011. The sarbanes-oxley act: implications for large-scale it outsourcing, ACM, Communications of the ACM. pp. 95–100.

Hellendoorn, H., Thomas, C., 1993. Defuzzification in fuzzy controllers. Journal of Intelligent & Fuzzy Systems 1, 109–123.

Holmblad, L.P., 1982. Control of a cement kiln by fuzzy logic. Fuzzy information and decision processes , 389–399.

Hong, T.P., Lee, C.Y., 1996. Induction of fuzzy rules and membership functions from training examples. Fuzzy sets and Systems 84, 33–47.

Hosmer, H.H., 1993. Security is fuzzy!: applying the fuzzy logic paradigm to the multipolicy paradigm, in: Proceedings on the 1992-1993 workshop on New security paradigms, ACM. pp. 175–184.

Hwang, K., Kulkareni, S., Hu, Y., 2009. Cloud security with virtualized defense and reputation-based trust mangement, in: Dependable, Autonomic and Secure Computing, 2009. DASC'09. Eighth IEEE International Conference on, IEEE. pp. 717–722.

Hwang, T.S., Lee, T.J., Lee, Y.J., 2010. A three-tier ids via data mining approach, in: Proceedings of the 3rd annual ACM workshop on Mining network data, ACM. pp. 1–6.

Iosup, A., Ostermann, S., Yigitbasi, M.N., Prodan, R., Fahringer, T., Epema, D., 2011. Performance analysis of cloud computing services for many-tasks scientific computing, IEEE, IEEE Transactions on Parallel and Distributed systems. pp. 931–945.

ISCX, I.S.C.o.E.I., 2012. UNB ISCX Intrusion Detection Evaluation DataSet. URL: `http://www.unb.ca/research/iscx/dataset/iscx-IDS-dataset.html`.

Iyoob, I., Zarifoglu, E., Dieker, A., 2013. Cloud computing operations research, INFORMS, Service Science. pp. 88–101.

Jacobson, D.H., 2010. A view on cloud computing, Advisory Services. Pricewaterhouse-Coopers, Toronto. pp. 1–12.

Jang, J.S., 1992. Self-learning fuzzy controllers based on temporal backpropagation. IEEE Transactions on neural networks 3, 714–723.

Jang, J.S.R., Sun, C.T., Mizutani, E., 1997. Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence .

Jansen, W., Grance, T., 2011. Guidelines on security and privacy in public cloud computing, pp. 108–144.

John, R., 1998. Type 2 fuzzy sets: an appraisal of theory and applications. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6, 563–576.

John, R., Coupland, S., 2007. Type-2 fuzzy logic: A historical view. Computational Intelligence Magazine, IEEE 2, 57–62.

Kalyani, G., Lakshmi, A.J., 2012. Performance assessment of different classification techniques for intrusion detection, IEEE Computer. pp. 148–155.

Kandukuri, B.R., Paturi, V.R., Rakshit, A., 2009. Cloud security issues, in: Services Computing, 2009. SCC'09. IEEE International Conference on, IEEE. pp. 517–520.

Karen, S., Mell, P., 2010. Guide to intrusion detection and prevention systems (idps), pp. 2–17.

Karray, F.O., De Silva, C.W., 2004. Soft computing and intelligent systems design: theory, tools, and applications. Pearson Education.

Kaufman, L.M., 2010. Can a trusted environment provide security?, IEEE, Security & Privacy. pp. 50–52.

KDDCUP99, . URL: `https://goo.gl/jgYFz5`.

King, P.J., Mamdani, E.H., 1977. The application of fuzzy control systems to industrial processes. Automatica 13, 235–242.

Klir, G., Wierman, M., 1999. Uncertainty-based information: elements of generalized information theory. volume 15. Springer Science & Business Media.

Klir, G., Yuan, B., 1995. Fuzzy sets and fuzzy logic, Prentice hall New Jersey. pp. 55–72.

Klir, G.J., 2005. Uncertainty and information: foundations of generalized information theory. John Wiley & Sons.

Kozushko, H., 2013. Intrusion detection: host-based and network-based intrusion detection systems, IEEE Computer Society. pp. 12–29.

Kretzschmar, M., Golling, M., Hanigk, S., 2011. Security management areas in the inter-cloud, in: Cloud Computing (CLOUD), 2011 IEEE International Conference on, IEEE. pp. 762–763.

Krutz, R.L., Vines, R.D., 2010. Cloud security: A comprehensive guide to secure cloud computing, Wiley Publishing. pp. 87–103.

Kumar, D.N.P.S., Ramesh, D.G.P., 2016. Intrusion detection analysis by implementing fuzzy logic, Springer. pp. 38–51.

Langley, P., Iba, W., Thompson, K., 2012. An analysis of bayesian classifiers, in: Aaai, IEEE Computer Society. pp. 223–228.

Leavitt, N., 2009. Is cloud computing really ready for prime time, IEEE Computer Society. pp. 113–127.

Lee, C.C., 1990. Fuzzy logic in control systems: fuzzy logic controller. i. IEEE Transactions on systems, man, and cybernetics 20, 404–418.

Lee, J.Y., Lee, J.W., Kim, S.D., et al., 2009. A quality model for evaluating software-as-a-service in cloud computing, in: Software Engineering Research, Management and Applications, 2009. SERA'09. 7th ACIS International Conference on, IEEE Computer Press. pp. 261–266.

Lee, W., Stolfo, S.J., et al., 2010. Data mining approaches for intrusion detection., in: Usenix security, IEEE, IEEE Transactions on Parallel and Distributed systems. pp. 22–29.

Levy, S., 2010. Hackers: Heroes of the computer revolution, Penguin Books New York. pp. 114–128.

Li, Q., Zhang, X., Chen, M., 2012a. Design on enterprise knowledge supply chain based on cloud computing, in: Business Intelligence and Financial Engineering (BIFE), 2012 Fifth International Conference on, IEEE. pp. 93–97.

Li, Z., OBrien, L., Cai, R., Zhang, H., 2012b. Towards a taxonomy of performance evaluation of commercial cloud services, in: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, IEEE. pp. 344–351.

Lindner, M., Galán, F., Chapman, C., Clayman, S., Henriksson, D., Elmroth, E., 2010. The cloud supply chain: A framework for information, monitoring, accounting and billing, in: 2nd International ICST Conference on Cloud Computing (CloudComp 2010), IEEE Computer Society. pp. 97–110.

Lippmann, R., Haines, J.W., Fried, D.J., Korba, J., Das, K., 2000. The 1999 darpa off-line intrusion detection evaluation, Elsevier, Computer networks. pp. 579–595.

Lo, C.C., Huang, C.C., Ku, J., 2010. A cooperative intrusion detection system framework for cloud computing networks, in: 2010 39th International Conference on Parallel Processing Workshops, IEEE. pp. 280–284.

Lombardi, F., Di Pietro, R., 2011. Secure virtualization for cloud computing, Elsevier, Journal of Network and Computer Applications. pp. 1113–1122.

Mamdani, E.H., 1974. Application of fuzzy algorithms for control of simple dynamic plant, in: Proceedings of the institution of electrical engineers, IET. pp. 1585–1588.

Mamdani, E.H., Assilian, S., 1975. An experiment in linguistic synthesis with a fuzzy logic controller. International journal of man-machine studies 7, 1–13.

Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., Ghalsasi, A., 2011. Cloud computing—the business perspective, Elsevier, Decision Support Systems. pp. 176–189.

MeeraGandhi, G., 2010. Machine learning approach for attack prediction and classification using supervised learning algorithms, Int. J. Computer Society and Communication. pp. 202–211.

Mell, P., Grance, T., 2009. The nist definition of cloud computing, National Institute of Standards and Technology. McKinsey Global Institute. pp. 50–55.

Menascé, D.A., Ngo, P., 2009. Understanding cloud computing: Experimentation and capacity planning, Computer Measurement Group Conference. pp. 176–182.

Mendel, J., Hagras, H., Tan, W.W., Melek, W.W., Ying, H., 2014. Introduction to type-2 fuzzy logic control: theory and applications. John Wiley & Sons.

Mendel, J.M., 2001. Uncertain rule-based fuzzy logic systems: introduction and new directions. Prentice Hall PTR Upper Saddle River.

Mendel, J.M., 2007a. Advances in type-2 fuzzy sets and systems. Information sciences 177, 84–110.

Mendel, J.M., 2007b. Computing with words: Zadeh, turing, popper and occam. IEEE computational intelligence magazine 2, 10–17.

Mendel, J.M., 2007c. Type-2 fuzzy sets and systems: An overview [corrected reprint]. IEEE computational intelligence magazine 2, 20–29.

Michael Armbrust, Armando Fox, R.G.A.D.J.R.H.K.A.K.G.L.D.A.P.A.R.I.S.M.Z., 2009. Above the clouds: A Berkeley view of Cloud Computing,¨[Online]. UC Berkeley EECS, 2012. Technical Report 1-23. UC Berkeley Reliable Adaptive Distributed Systems Laboratory. URL: `http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf`.

Mohamed, A., 2009. A history of cloud computing¨[online]., 2009. URL: `http://www.computerweekly.com/feature/A-history-of-cloud-computing`.

Mouzouris, G.C., Mendel, J.M., 1994. Non-singleton fuzzy logic systems, in: Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the Third IEEE Conference on, IEEE. pp. 456–461.

Mouzouris, G.C., Mendel, J.M., 1997. Nonsingleton fuzzy logic systems: theory and application. IEEE Transactions on Fuzzy Systems 5, 56–71.

Murugan, S., Kuppusamy, K., 2011. System and methodology for unknown malware attack, in: Sustainable Energy and Intelligent Systems (SEISCON 2011), International Conference on, IET. pp. 803–804.

Nadiammai, G., Hemalatha, M., 2012. Perspective analysis of machine learning algorithms for detecting network intrusions, in: Computing Communication & Networking Technologies (ICCCNT), 2012 Third International Conference on, IEEE. pp. 1–7.

Nair, P.R., 2014. Tackling supply chain through cloud computing: Management: Opportunities, challenges and successful deployments, in: ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India-Vol II, Springer. pp. 761–767.

Neethu, B., 2012. Classification of intrusion detection dataset using machine learning approaches, International Journal of Electronics and Computer Science Engineering. pp. 1044–51.

Negnevitsky, M., 2005. Artificial intelligence: a guide to intelligent systems. Pearson Education.

Nozaki, K., Ishibuchi, H., Tanaka, H., 1997. A simple but powerful heuristic method for generating fuzzy rules from numerical data. Fuzzy sets and systems 86, 251–270.

NSL-KDD, 2012. Nsl-kdd dataset,¨[online]. published. retrieved august 15, 2012. URL: `http://www.unb.ca/research/iscx/dataset/iscx-NSL-KDD-dataset.html`.

Nunez, A., Castane, G., Vazquez-Poletti, J., Caminero, A., Carretero, J., Llorente, I., 2011. Design of a flexible and scalable hypervisor module for simulating cloud computing environments, in: Performance Evaluation of Computer & Telecommunication Systems (SPECTS), 2011 International Symposium on, IEEE. pp. 265–270.

O. A. Hathaway, R. Crootof, P.L.H.N.A.N.W.P., Spiegel, J., 2012. The law of cyber-attack. California Law Review URL: `www.californialawreview.org`.

Paulitsch, M., 2009. Dynamic coordination of supply chains in cloud computing. Ph.D. thesis. WU Vienna University of Economics and Business.

Paxson, V., Campbell, S., Lee, J., et al., 2013. Bro intrusion detection system. Technical Report. Lawrence Berkeley National Laboratory.

Pipyros, K., Mitrou, L., Gritzalis, D., Apostolopoulos, T., 2014. A cyber attack evaluation methodology, in: Proc. of the 13th European Conference on Cyber Warfare and Security, pp. 264–270.

Popovic, K., Hocenski, Z., 2010. Cloud computing security issues and challenges, in: MIPRO, 2010 proceedings of the 33rd international convention, IEEE. pp. 344–349.

Puzmanova, Rita, A.V.K.V.G., Mikhailovsky, A.A., 2014. Review of wi-foo: The secrets of wireless hacking, ACM, Queue. pp. 70–70.

Raiyn, J., 2014. A survey of cyber attack detection strategies., International Journal of Security & Its Applications. pp. 18–32.

Rake-Revelant, J., Holschke, O., Offermann, P., Bub, U., 2010. Platform-as-a-service for business customers, in: Intelligence in Next Generation Networks (ICIN), 2010 14th International Conference on, IEEE. pp. 1–6.

Reddy, E.K., Iaeng, M., Reddy, V., Rajulu, P., 2011. A study of intrusion detection in data mining, in: World Congress on Engineering (WCE), pp. 6–8.

Ristenpart, T., Tromer, E., Shacham, H., Savage, S., 2009. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds, in: Proceedings of the 16th ACM conference on Computer and communications security, ACM. pp. 199–212.

Roesch, M., et al., 2009. Snort: Lightweight intrusion detection for networks., in: LISA, pp. 229–238.

Roschke, S., Cheng, F., Meinel, C., 2009. Intrusion detection in the cloud, in: Dependable, Autonomic and Secure Computing, 2009. DASC'09. Eighth IEEE International Conference on, IEEE. pp. 729–734.

Ross, T.J., 2009. Fuzzy logic with engineering applications. John Wiley & Sons.

Runthala, A., 2010. Hacking: A threat to industrial work forces., CURIE Journal. pp. 3–13.

Sahab, N., Hagras, H., 2011. Adaptive non-singleton type-2 fuzzy logic systems: A way forward for handling numerical uncertainties in real world applications. International Journal of Computers Communications & Control 6, 503–529.

Salah, A., Shouman, M., Faheem, H., 2010. Surviving cyber warfare with a hybrid multiagent-base intrusion prevention system, IEEE, Potentials, IEEE. pp. 32–40.

Schultz, M.G., Eskin, E., Zadok, F., Stolfo, S.J., 2011. Data mining methods for detection of new malicious executables, in: Security and Privacy, 2001. S&P 2011. Proceedings. 2001 IEEE Symposium on, IEEE. pp. 38–49.

Sebastian, S., John, R., 2016. Multi-fuzzy sets and their correspondence to other sets. Annals of Fuzzy Mathematics and Informatics 11, 341–348.

Sethi, S., Sahu, A., Jena, S.K., 2012. Efficient load balancing in cloud computing using fuzzy logic, IOSR Journal of Engineering. pp. 65–71.

Shackelford, S., 2009. From nuclear war to net war: analogizing cyber attacks in international law, Berkley Journal of International Law (BJIL). pp. 125–135.

Shanmugavadivu, R., Nagarajan, D.N., 2012. Learning of intrusion detector in conceptual approach of fuzzy towards intrusion methodology, International Journal of Advanced Research in Computer Science and Software Engineering. pp. 114–129.

Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A.A., 2012. Toward developing a systematic approach to generate benchmark datasets for intrusion detection, Elsevier, Computers & Security. pp. 357–374.

Singh, A., 2010. Cloud computing for supply chain solutions, in: Supply & Demand Chain Executive, pp. 776–784.

Singh, A., 2014. vcloud suite architecture URL: `http://technologyandarchitecture.blogspot.co.uk/2014/07/vcloud-suite-architecture-1.html`.

Singh, A.K., Roy, S., 2012. A network based vulnerability scanner for detecting sqli attacks in web applications, in: Recent Advances in Information Technology (RAIT), 2012 1st International Conference on, IEEE. pp. 585–590.

Singh, J., 2011. Fuzzy logic based intrusion detection system against blackhole attack on aodv in manet, IEEE Cloud Computing. pp. 28–35.

Sinha, D., Dougherty, E.R., 1993. Fuzzification of set inclusion: theory and applications. Fuzzy sets and systems 55, 15–42.

Srinivasulu, P., Nagaraju, D., Kumar, P.R., Rao, K.N., 2009. Classifying the network intrusion attacks using data mining classification methods and their performance comparison, International Journal of Computer Science and Network Security. pp. 11–18.

Stantchev, V., 2009. Performance evaluation of cloud computing offerings, in: Advanced Engineering Computing and Applications in Sciences, 2009. ADVCOMP'09. Third International Conference on, IEEE. pp. 187–192.

Stats, I.L., 2016. Internet Users in the World. URL: `http://www.internetlivestats.com/internet-users/`.

Subramanian, S., Srinivasan, V.B., Ramasa, C., 2012. Study on classification algorithms for network intrusion systems, Journal of Communication and Computer. pp. 1242–1246.

Sugeno, M., 1985. An introductory survey of fuzzy control. Information sciences 36, 59–83.

Sumter, L., 2010. Cloud computing: security risk, in: Proceedings of the 48th Annual Southeast Regional Conference, ACM, Network Security. pp. 112–125.

Supriya, M., et al., 2012. Estimating trust value for cloud service providers using fuzzy logic, Citeseer. pp. 1027–1040.

Takagi, T., Sugeno, M., 1985. Fuzzy identification of systems and its applications to modeling and control. IEEE transactions on systems, man, and cybernetics , 116–132.

Talwar, M., 2015. Security issues in internet of things, International Journal on Emerging Technologies. pp. 413–422.

Tan, Z., Jamdagni, A., He, X., Nanda, P., Liu, R.P., 2014. A system for denial-of-service attack detection based on multivariate correlation analysis, IEEE, IEEE transactions on parallel and distributed systems. pp. 447–456.

Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A., 2009. A detailed analysis of the kdd cup 99 data set, in: Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009, IEEE, Network Security, IEEE Transactions on. pp. 71–79.

Truong, D., 2014. Cloud-based solutions for supply chain management: A post-adoption study , 697–708URL: `http://www.http://asbbs.org/files/ASBBS2014/PDF/T/TruongD(P697-708).pdf`.

Tsao, H., Parikh, S., Ghosh, A.S., Pal, R., Ranalkar, M., Tarapore, H., Venkatsubramanyan, S., 2010. Streamlining grain supply chains of india: Cloud computing and distributed hubbing for wholesale-retail logistics, in: Service Operations and Logistics and Informatics (SOLI), 2010 IEEE International Conference on, IEEE. pp. 252–257.

Van Leekwijck, W., Kerre, E.E., 1999. Defuzzification: criteria and classification. Fuzzy sets and systems 108, 159–178.

Vieira, K., Schulter, A., Westphall, C., Westphall, C., 2010. Intrusion detection for grid and cloud computing, IEEE Computer Society, IT Professional Magazine. pp. 38–47.

Wagner, C., Hagras, H., 2010. Uncertainty and type-2 fuzzy sets and systems, in: Computational Intelligence (UKCI), 2010 UK Workshop on, IEEE. pp. 1–5.

Wang, L.X., 1994. Adaptive fuzzy systems and control: design and stability analysis. Englewood Cliffs .

Wang, L.X., 1999. A course in fuzzy systems. Prentice-Hall press, USA.

Wang, L.X., Mendel, J.M., 1992a. Back-propagation fuzzy system as nonlinear dynamic system identifiers, in: Fuzzy Systems, 1992., IEEE International Conference on, IEEE. pp. 1409–1418.

Wang, L.X., Mendel, J.M., 1992b. Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. IEEE transactions on Neural Networks 3, 807–814.

Wang, L.X., Mendel, J.M., 1992c. Generating fuzzy rules by learning from examples. IEEE Transactions on systems, man, and cybernetics 22, 1414–1427.

Waxman, M.C., 2011. Cyber-attacks and the use of force: Back to the future of article 2 (4), Yale Journal of International Law. pp. 143–151.

environment for knowledge analysis (WEKA), W., . URL: `http://www.cs.waikato.ac.nz/ml/weka/`.

Wu, D., Mendel, J.M., 2009. A comparative study of ranking methods, similarity measures and uncertainty measures for interval type-2 fuzzy sets. Information Sciences 179, 1169–1192.

Wu, Y., Cegielski, C.G., Hazen, B.T., Hall, D.J., 2013. Cloud computing in support of supply chain information system infrastructure: Understanding when to go to the cloud, Wiley Online Library, Journal of Supply Chain Management. pp. 25–41.

Xin, G., Yun-jie, L., 2010. An new intrusion prevention attack system model based on immune principle, in: e-Business and Information System Security (EBISS), 2010 2nd International Conference on, IEEE. pp. 1–4.

Xiong, K., Perros, H., 2009. Service performance and analysis in cloud computing, in: 2009 Congress on Services-I, IEEE. pp. 693–700.

Yang, B., Tan, F., Dai, Y.S., 2013. Performance evaluation of cloud service considering fault recovery, Springer, The Journal of Supercomputing. pp. 426–444.

Yang, Y., Liu, S., John, R., 2014. Uncertainty representation of grey numbers and grey sets. IEEE transactions on cybernetics 44, 1508–1517.

Yinglei, B., Lei, W., 2011. Leveraging cloud computing to enhance supply chain management in automobile industry, in: Business Computing and Global Informatization (BCGIN), 2011 International Conference on, IEEE. pp. 150–153.

YiPeng, L., 2011. The impact of" cloud computing"-based information sharing on supply chain, in: Management of e-Commerce and e-Government (ICMeCG), 2011 Fifth International Conference on, IEEE. pp. 173–175.

Zadeh, L., 2008. Is there a need for fuzzy logic?, Information Science. pp. 2751 – 2779.

Zadeh, L.A., 1965. Fuzzy sets. Information and control 8, 338–353.

Zadeh, L.A., 1973. Outline of a new approach to the analysis of complex systems and decision processes. IEEE Transactions on Systems, Man and Cybernetics 1100, 38–45.

Zadeh, L.A., 1975. The concept of a linguistic variable and its application to approximate reasoning, Springer, Information Science. pp. 199–249.

Zadeh, L.A., 1984. Coping with the imprecision of the real world, ACM, Communications of the ACM. pp. 304–311.

Zadeh, L.A., 1994. Fuzzy logic, neural networks, and soft computing. Communications of the ACM 37, 77–85.

Zhang, Q., Cheng, L., Boutaba, R., 2010. Cloud computing: state-of-the-art and research challenges, Springer, Journal of internet services and applications. pp. 7–18.

Zimmermann, H.J., 1987. Fuzzy sets, decision making, and expert systems, Springer. pp. 267–283.

Zimmermann, H.J., 2011. Fuzzy set theory—and its applications. Springer Science & Business Media.