Thesis Title	Fuzzy Genetic Algorithm for Real-Time Intrusion Detection System
Thesis Credits	12
Candidate	Miss Pawita Jongsuebsuk
Thesis Advisor	Assoc. Prof. Dr. Naruemon Wattanapongsakorn
Program	Master of Engineering
Field of Study	Computer Engineering
Department	Computer Engineering
Faculty	Engineering
B.E.	2555

Abstract

Internet has become one of the main communication modes in our society. Various types of internet applications and usage are available. Increasing usage of the internet also increases threats in the internet. To prevent unwanted or dangerous threats, we have to be able to detect them first. Therefore, designing an effective intrusion detection system is a challenge because the threats have different characteristics and they evolve every day. The intrusion detection at present must be robust for new or unknown attacks. In this thesis, a real-time network-based intrusion detection approach using fuzzy genetic algorithm is proposed to detect DoS attacks and Probe attacks. The detection accuracy of the fuzzy genetic algorithm with KDD99 dataset and current online dataset is demonstrated. The experimental results show that the fuzzy genetic technique gives high detection rates and is robust for both known and unknown attacks. Then, the fuzzy genetic algorithm technique for real-time and online intrusion detection, i.e., the data is detected right after it arrived to the detection system, is developed. In an actual network environment, the network traffic is preprocessed into 12 features by counting connections in each source-destination IP-pair within 2 second time interval. The IDS is evaluated in terms of the detection speed, CPU consumption, memory consumption, the false alarm rate and the detection rate.

Keywords: Fuzzy Genetic Algorithm/ Intrusion Detection System/ Dos Detection/ Probe Detection

หัวข้อวิทยานิพนธ์	ฟัซซี่จีนีติกอัลกอริทึมสำหรับระบบการตรวจจับการบุกรุกในเครือข่ายแบบทันที
หน่วยกิต	12
ผู้เขียน	นางสาวภาวิตา จงสืบสุข
อาจารย์ที่ปรึกษา	รศ. ดร. นฤมล วัฒนพงศกร
หลักสูตร	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมคอมพิวเตอร์
คณะ	วิศวกรรมศาสตร์
พ.ศ.	2555

บทคัดย่อ

อินเตอร์เน็ตเป็นศูนย์กลางการสื่อสารในสังคมปัจจุบัน โดยมีการใช้งานในหลายรูปแบบ ปริมาณการใช้งานในเครือข่ายที่สูงขึ้นทำให้ปริมาณการโจมตีสูงขึ้นด้วย หากต้องการป้องกันการโจมตีที่ไม่พึงประสงค์ระบบต้องสามารถตรวจจับการโจมตีเหล่านั้นได้ก่อน ดังนั้นจึงจำเป็นต้องออกแบบระบบการตรวจจับการบุกรุกในเครือข่ายแบบทันที ระบบการตรวจจับการบุกรุกในเครือข่ายเป็นสิ่งที่ท้าทายเนื่องจากการบุกรุกและการโจมตีในเครือข่ายมีหลากหลายรูปแบบ หลากหลายพฤติกรรมและพัฒนาตัวเองอยู่ทุกวัน ดังนั้นระบบการตรวจจับการบุกรุกในเครือข่ายในปัจจุบันจึงต้องสามารถตรวจจับการบุกรุกรูปแบบใหม่ที่ไม่รู้จักได้ ในวิทยานิพนธ์เล่มนี้สามารถตรวจจับการบุกรุกแบบ DOS และ PROBE ได้ โดยใช้ฟัซซี่จีนีติกอัลกอริทึมและทดสอบความแม่นยำของระบบด้วยข้อมูลจาก KDD99 และข้อมูลการระบบจริง ผลการตรวจสอบพบว่าฟัซซี่จีนีติกอัลกอริทึมมีความแม่นยำในการตรวจจับและสามารถตรวจจับได้ทั้งการบุกรุกที่รู้จักและการบุกรุกที่ไม่รู้จัก นอกจากนี้ยังได้พัฒนาฟัซซี่จีนีติกอัลกอริทึมสำหรับการตรวจจับแบบทันทีในระบบเครือข่ายจริง โดยประมวลผลข้อมูลในเครือข่ายเป็น 12 รูปแบบ โดยการนับจำนวนการติดต่อระหว่างสองคู่ไอพีภายในเวลา 2 วินาที ทั้งนี้ทำการตรวจสอบระบบการตรวจจับการบุกรุกนี้ด้วยความเร็วในการตรวจจับ ปริมาณการใช้งานของ CPU ปริมาณการใช้งานหน่วยความจำ อัตราความผิดพลาดในการตรวจจับและอัตราความถูกต้องในการตรวจจับ

คำสำคัญ: ฟัซซี่จีนีติกอัลกอริทึม/ การตรวจจับการบุกรุก/ การตรวจจับแบบดอส/
การตรวจจับแบบโพรพ

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to a few people for their contribution to this thesis. The first one is Assoc. Prof. Dr. Naruemon Wattanapongsakorn, my research supervisor, for her patience, guidance, enthusiastic encouragement, useful critiques of this research work to keep my progress on schedule. I would also like to thank Dr. Chalermpol Charnsripinyo for his advice and his valuable and constructive suggestions. I have to thank all thesis committees for their useful suggestions and comments. My grateful thanks are also extended to Mr. Phurivit Sangkatsanee for his help in offering me the resources in preprocessing the network data, to Mr. Ekgapark Wonghirunsombat and his project team who helped me in data collecting process. I also would like to extend my thanks to all officers who have always offered me their help. Finally, I wish to thank my parents for their support and encouragement throughout my study.

CONTENTS

CONTENTS (Cont.)

LIST OF TABLES

LIST OF TABLES (Cont.)

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligent
ANN	Artificial Neural Network
ART	Adaptive Resonance Theory
DoS	Denial of Service
DR	Detection Rate
FA	False Alarm
HIDS	Host-Based Intrusion Detection System
IDPS	Intrusion Detection and Prevention System
IDS	Intrusion Detection System
KDD99	International Conference on Knowledge Discovery and Data Mining 1999
LD2	Lightweight Detection System
MAWI	Measurement and Analysis on WIDE Internet
METROSEC	Metrology for Security and Quality of Service
MOGFIDS	Multi-Objective Genetic Fuzzy Intrusion Detection System
N/A	Not Available
NIDS	Network-Based Intrusion Detection System
Probe	Port Scan
PSO	Particle Swarm Optimization
R2L	Remote to Local Attack
RT-UNNID	Real-Time Unsupervised Neural-Net-Based Intrusion Detector
SOM	Self Organizing Maps
SVM	Support Vector Machine
TN	True Negative Rate
U2R	User to Root

LIST OF ABBREVIATIONS (Cont.)

UI	User Interface
UNN-Engine	Unsupervised Neural-Net-Based Engine
WIDE	Widely Integrated Distributed Environment
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
ICMP	Internet Control Message Protocol
ARP	Address Resolution Protocol
IP	Internet Protocol

CHAPTER 1 INTRODUCTION

1.1 Motivation

Internet has rapidly become one of the main communication methods in our society. More and more types of internet applications and usage are available. The more usage of network applications, the more security risks the internet users may face.

To prevent unwanted or dangerous threats, we have to detect them first. Therefore, developing an intrusion detection method is a challenging research issue. There are four challenging issues about designing IDS. The first is the high accuracy and low false alarm rates, especially, the false positive rate (which should be less than 1%) and the false negative rate. Second, the IDS should be able to detect new/unknown attacks because new threats evolve every day. In addition, the performance of classification algorithm in the IDS should be good enough for real-time detection, such as computation speed, memory consumption, etc., because there are a lot of data packets over the real network. A bad performance can cause the system clash. Finally, the IDS should provide more information about the attacks in order to prevent the malicious activities such as attack type, target computer, etc.

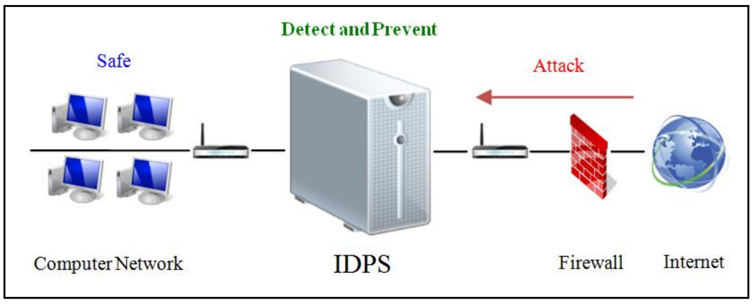


Figure 1.1 Network Environments and Intrusion Detection System

1.2 Objective and Research Scope

In this thesis, we focus on real-time and unknown detection. The algorithm is able to handle the attack and send an alarm message with useful information within three seconds after the packet arrives to the system. There are two output classes from the system including the normal class and the attack class. We are interested in designing an IDS algorithm using fuzzy genetic algorithm. The fuzzy rule is a supervised learning technique and a genetic algorithm which help the system find the best rule from a training dataset. This technique has a high detection rate, a low false alarm rate, fast processing and is robust for unknown data. Therefore, we develop the fuzzy genetic algorithm approach to implement our real-time intrusion detection system where the input network data is captured in the online network, and it will respond to the attack within 2-3 seconds. We evaluate our IDS in terms of the detection speed, CPU consumption, memory consumption, false alarm rate and overall detection rate.

1.3 General Approach

KDD99 dataset is a benchmark dataset that is used in various research studies. However, there were some research groups which generated their own datasets because the KDD99 dataset was now too old. Moreover, it did not contain the present network activities and present attack.

In general, IDS can be classified into two types which are host-based (HIDS) and network-based (NIDS). The HIDS analyzes information that is available on individual computers, such as system calls and log file while the NIDS monitors information in network traffic. The IDS can be further classified into misuse-based and anomaly-based. The misuse-based is a pattern matching. When the packets are matched with the patterns, they will be classified as the attacks. This technique has high accuracy and a low false alarm rate; however, it is not robust for new attacks. The anomaly-based IDS is designed to detect new/unknown attacks. However, it has a low detection rate when comparing with the misuse-based IDS. The general techniques for the unknown detection are as follows:

- Clustering is the algorithm that clusters input data into groups without training data (unsupervised-learning) such as k-means, k-nearest neighbors. However, these techniques have low accuracy.
- Neural network is a group of nodes which are associated with each other. The algorithm will create a neural network structure to recognize the given information. The neural network can work well with noisy data and incomplete data. However, it uses high computation time.
- Fuzzy set is used to create a rule (s). The behavior which agrees with the rule will be considered as an attack. There are many researchers using the fuzzy method because of its robustness and efficiency in detecting unknown data.
- Artificial immune system is a concept of simulating immunology which is inspired by a biological immune system. The intrusion detection system can be considered as an immune system and the attack packet is pathogens. The algorithm only creates a model of normal behavior. When the matching behavior is not found, it will be labeled as an attack.

1.4 Research Contribution

In summary, we make the following contribution:

1. We develop a real-time ID that detects both known and unknown attacks.

2. We improve performance of unknown detection with our proposed approach and compare the results with those from the existing methods.

3. We demonstrate the real time in a real-time network environment.

1.5 Report Overview

This research proposal is organized as follows: in chapter two, there are background study and literature review. Then, chapter three describes the detection approach. The experimental designs and results will be presented in chapter four, and chapter five is the conclusion.

CHAPTER 2 LITERATURE SURVEY AND BACKGROUND STUDY

2.1 Literature Survey

In this chapter, we present literature review consisting of two parts which are offline IDS and online IDS (real-time). The offline data uses KDD99 dataset which is a benchmark dataset. In this section, we focus on comparing various techniques for intrusion detection. Besides, we compare the performances of each technique, such as a detection rate, a false alarm rate and limitation. For the online IDS, we focus on a new technique to preprocess data in an actual network environment and testing environment.

2.1.1 IDS for Offline Data

Gómez and León [1] proposed a Fuzzy Genetic Algorithm to classify behavior of intrusion into two classes (normal class and attack class). This algorithm could be trained by one class (normal class). The behavior different from the training class would be classified as an attack. They used KDDC99 dataset which had four attack types including DoS, Probe, R2L and U2R. In the KDD99 dataset, they found that there were some features that had the same value for each record, so they reduced the number of the features into 33 features. The dataset was divided into two sets including the training set and the testing set. The training set had only the normal data containing 2,000 records. The highest obtained detection rate was 98.28% with 5% of the false alarm rate.

Banković et al. [2] proposed an interesting Fuzzy Genetic Algorithm Approach to reduce the number of the features in the dataset and maintain the high detection rate. From the experiment, they found that there were three features that were relevant. There were two experiments: the first experiment had two output classes (normal class and attack class). The accuracy of the detecting attack (TN) was 94.87% with 1.62% of the false positive. The second experiment had four classes (the fuzzy rule could identify each type of the attacks including the normal class, the *portsweep* class, the *smurf* class and the *neptune* class). From this experiment, the maximum detection rate was 87.6% because there was only 30% of the detection rate of the *portsweep*. These two experiments used the KDD99 dataset. However, the training dataset had 976 records (137 of attack records and 839 of normal records) and the testing dataset had 977 records (234 of attack records and 743 of normal records). Moreover, they considered only three types of the attacks which were the *portsweep*, *smurf* and *neptune*.

Tsang et al. [3] proposed Multi-Objective Genetic Fuzzy Intrusion Detection System (MOGFIDS) for detecting anomaly attack. There were three objectives for MOGFIDS: having the high classification rate, reducing the number of fuzzy rules and reducing complexity of fuzzy rules. This experiment used 10% version of KDD99 dataset for training including four attack types (DoS, Probe, R2L and U2R). However, they found that the dataset was biased against DoS (Neptune attack and Smurf attack). In order to make the training set more realistic, they sampled 1,000 records for each type of the DoS, 10,000 records of the normal and the remaining intact number of the records of other attacks (the number of the training set was 20,752 records). The testing set used 311,029 records with additional 14 unseen attack types. The result showed that this algorithm with 27 features gave 92.77% of the detection rate and 1.6 of the false positive rate.

Ensafi et al. [4] proposed optimizing fuzzy K-means for network anomaly detection using particle swarm optimization (PSO). Two versions of the KDD99 dataset were used (full version and 10% version). The training dataset had only the normal class from the 10% version. The testing dataset consisted of 60,592 records of the normal class and 250,436 of the attack class. Figure 2.1 presents the diagram of the proposed work. Particles swarm and K-means clustering was used together to cluster the dataset in each generation. A genetic algorithm was used to find the best solution. The output classes were Normal, DoS, R2L, U2R and Probe, and the detection rate was 95 % with 2.12% of the false alarm rate.

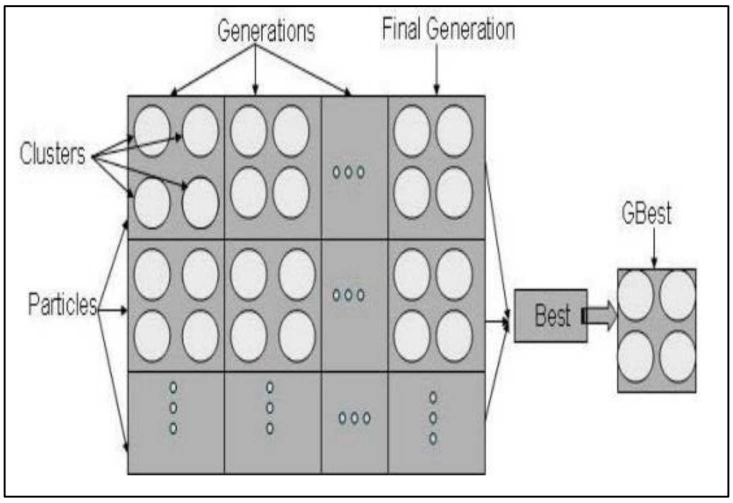


Figure 2.1 Optimizing fuzzy K-means for network anomaly framework [4]

Fries [5] proposed a Fuzzy Genetic Algorithm Approach. This work had two phases: preprocessing phase and detection phase. In the preprocessing phase, they used clustering and genetic algorithm to find the significant features. The result showed that there were 8 relevant features. In the detection phase, they evaluated the algorithm by using the 10% version of the KDD99 dataset as the training set (about 500,000 records) and the full version of the KDD99 dataset as the testing set (about 5 million records). In the testing set, there were 14 types of new attacks that were not presented in the training set. The detection rate was 99.6% with 0.2 of the false positive rate. They found that this algorithm had the high detection rate and was robust for an untrained attack.

Abadeh et al. [6] proposed a genetic fuzzy algorithm. They used three different kinds of genetic fuzzy systems based on Michigan, Pittsburgh and iterative rule learning. The algorithm could be classified into five classes (Normal, U2R, R2L, DoS, and Probe). The distribution of the training dataset and the testing dataset is shown in Table 2.1. The result showed that the Pittsburgh method had the highest detection rate of 99.53% with 1.94% of the false alarm rate.

Table 2.1 Distribution of different classes in training and testing datasets [6]

Attack Type	**Train**	**Test**
Normal	100	1,000
U2R	50	59
R2L	100	1,000
DoS	300	6,500
PORBE	100	1,000
Total	**650**	**9,559**

Ngamwitthayanon and Wattanapongsakorn [7] proposed a Fuzzy-Adaptive Resonance Theory (ART) in network anomaly detection with feature-reduction dataset. The Adaptive Resonance was a type of the neural network algorithm. The main algorithm was the ART algorithm while the Fuzzy was used to simplify a network structure of the ART. Moreover, they applied a feature reduction method with the KDD99 dataset. This approach increased the detection rate to 98.96% and used only14 features. However, this algorithm indicating the similar problem as the previous algorithm was impractical in the real network. Also, it did not provide enough information for a protection system.

Table 2.2 Detection rate with different numbers of KDD99 features [7]

Dataset	**Number of Features**	**Detection Rate (%)**
1	7	98.87
2	9	99.44
3	12	98.98
4	14	98.93
5	22	99.12
6	24	99.20
7	41	97.96

Muda et al. [8] proposed a detection solution by combining of the K-means algorithm and the Naïve Bayes algorithm. The first step of the algorithm was using the K-means algorithm to categorize data into two classes; normal class and attack class. Then, the Naïve Bayes algorithm was used to classify the previous results into attack types. They sampled 49,402 records of the training set from the 10% version of the KDD99 dataset and another 49,402 records from the full version of the KDD99 which had more 14 types of new attacks. The detection rate was 99.6%. However, this solution was impractical for a real network environment because the K-Means algorithm required time to process. It could cause the bottleneck problem in network traffic or system clash.

Seungmin et al. [9] proposed a self-organizing map (SOM) combined with the K-means algorithm to classify untrained attacks. The system was able to learn from the new data. There were three phases consisting of an adjusting SOM network, updating centroid (K-means algorithm) and splitting normal cluster. The cluster system could divide the output into two classes (normal class and attack class). They sampled the dataset from the KDD99 dataset. The size of the sampling dataset was 20,000 records which consisted of 1% of the attack and 99% of the normal class. They reduced the number of features into eight features (2, 3, 4, 10, 12, 23, 33 and 35). The average detection rate in this work was 89.7% with 2.43 of the false positive rate.

Chandrasekhar and Raghuveer [10] proposed an intrusion detection technique using the K-means, fuzzy neural network and the SVM algorithm. They found that a rule based system was worse when encountering with a large scale of the data, so they introduced the artificial neural network (ANN) for this system [Figure 2.2]. First, they used the K-mean algorithm to cluster the dataset into n clusters (each cluster was the type of intrusion). In each cluster, there was a neuro-fuzzy to learn the pattern. The neuro-fuzzy in each cluster was used to generate the SVM vector to classify attacks (the neuro-fuzzy algorithm helped to decrease a number of attributes in SVM). They sampled the training dataset and testing dataset from a 10% version file of the KDD99 dataset which consisted of 26,114 records for the training dataset and 27,112 records for the testing dataset (Table 2.3). The accuracy of each attack was 98% for DoS attack, 97.31% for Probe, 97.51 for R2L and 97.52 for U2R. Total detection rate was 98.48% with 2.41 % of the false positive rate.

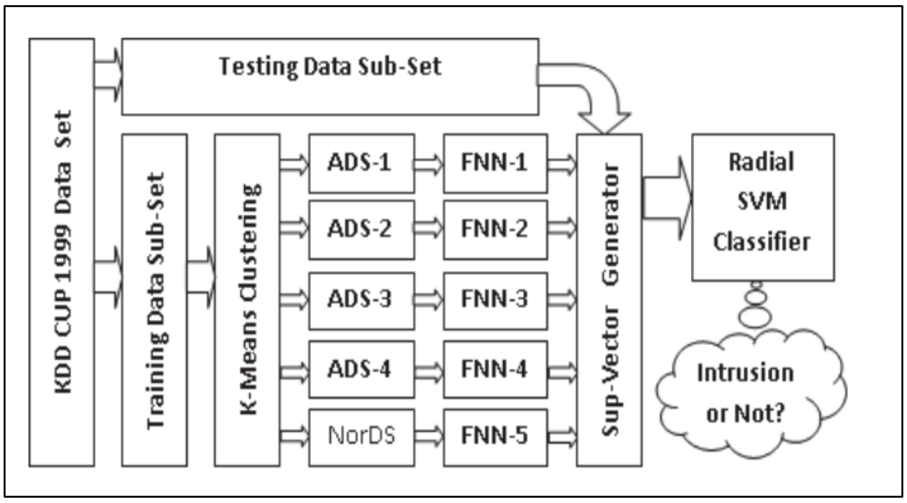


Figure2.2 Block diagram of proposed IDS from using K-means, fuzzy neural network and SVM algorithm [10]

Table 2.3 Data record taken for training and testing in [10]

	Normal	**DoS**	**PROBE**	**R2L**	**U2R**	**TOTAL**
Training	12,500	12,500	1,054	39	21	26,114
Testing	12,500	12,500	2,053	38	21	27,112

Table 2.4 Summary of Offline IDS

Year	Author	Algorithm	DR(%)	FP(%)	Feature	Output
2006 [1]	Gómez and León	Genetic Fuzzy	98.28	N/A	33	Normal, Attack
2007 [2]	Banković et al	Genetic Fuzzy	94.87	0-1.62	3	Normal, Attack
			87.6	0		Normal, Portsweep, Smurf and Neptune
2007 [3]	Tsang et al.	Genetic Fuzzy	92.77	1.6	27	Normal, Probe, DoS, U2R, R2L
2008 [4]	Ensafi et al	Fuzzy K-means and PSO	95.9	2.12	33	Normal, Probe, DoS, U2R, R2L
20010 [5]	Fries	Genetic Fuzzy	99.6	0.2	8	Normal, attack
2010 [6]	Abadeh et al.	Genetic Fuzzy	99.53	1.94	21	Normal, Probe, DoS, U2R, R2L
2011 [7]	Ngamwitthayanon and Wattanapongsakorn	Fuzzy and ART	98.96	N/A	14	Normal, Attack
2011 [8]	Muda et al.	K-means+ naïve bayes technique	99.8	0.09	41	Normal, Probe, DoS, U2R, R2L
2011 [9]	Seungmin et al.	SOM and K-means	89.7	2.43	8	Normal, attack
2013 [10]	Chandrasekhar et al.	K-means, fuzzy neural network and SVM	98.48	2.41	N/A	Normal, Probe, DoS, U2R, R2L

** DR = Detection Rate
** FA = False Alarm
** N/A not available

2.1.2 IDS for Online Data

Labib and Vemuri [11] proposed a real-time intrusion detection system by considering 10 features of header packets. Each record was the statistic data which was collected in every 50 packets. Then, they used SOM as an algorithm to classify attacks. The outputs were normal and DoS attacks. On the other hand, it needed a human expert to visualize the output data.

Amini et al. [12] proposed a real-time intrusion detection system using neural network algorithms (Adaptive Resonance Theory (ART) and Self-Organizing Map (SOM)) to classify normal packets and attack packets (two classes) as shown in Figure 2.3. They generated the attacks and collected the attack data by using attack tools as shown in Table 2.5 (left). They collected normal traffic in a real traffic network within 4 days. So, they created their own dataset which consisted of training data (5,000 packets) and testing dataset (3,000 packets). They preprocessed the packets into 27 features as shown in Table 2.5 (right). The result showed that the ART had the higher detection of 97.42%. The result is shown in Table 2.5.

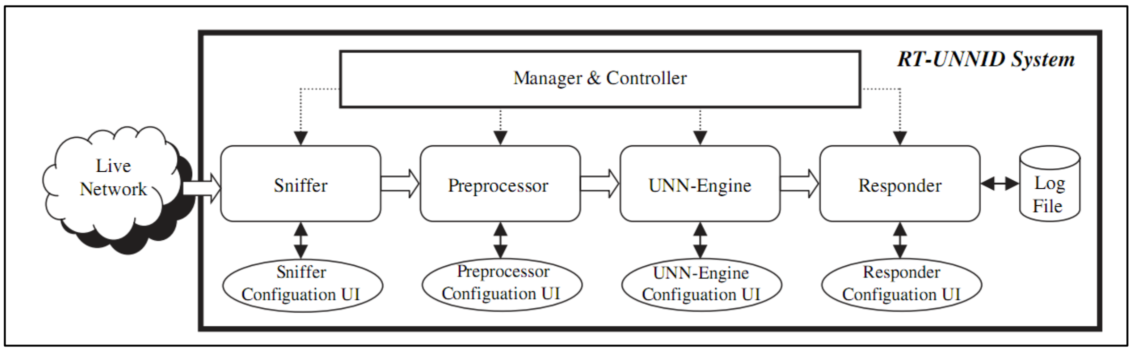


Figure 2.3 RT-UNNID systems [12]

Table 2.5 Real-time detection rate of RT-UNNID using SOM ART-1 and ART-2 [12]

	ETTR	**TR**	**FPR**	**FNR**
ART-1	71.71	97.42	1.99	0.59
ART-2	73.18	97.19	2.3	0.51
SOM	83.44	95.74	3.5	0.77

**ETTR is exact true types detection rate
TR is true detection rate
FPR is false positive detection rate
NFR is false negative detection rate

Table 2.6 Attack name (left) and feature name in proposed approach (right) [12]

#	Attack name	Attack generation tools	Train dataset	Test dataset
1	Bonk	targa2.c	√	√
2	Jolt	targa2.c	√	√
3	Land	targa2.c	√	√
4	Saihyousen	targa2.c	√	√
5	TearDrop	targa2.c	√	√
6	Newtear	targa2.c	√	√
7	1234	targa2.c	√	√
8	Winnuke	targa2.c	√	√
9	Oshare	targa2.c	√	√
10	Nestea	targa2.c	√	√
11	SynDrop	targa2.c	√	√
12	Octopus	Octopus.c	√	√
13	KillWin	KillWin.c	√	√
14	Twinge	Twinge.c	√	√
15	TcpWindowScan	Nmap	√	√
16	SynScan	Nmap	√	√
17	Neptune	FireHack	√	√
18	Dosnuke	FireHack	√	√
19	Smbdie	Smbdie.exe	√	√
20	XmassTree-Scan	Namp	√	√
21	LinuxICMP	linux-icmp.c	-	√
22	Moyari13	Moyari13.c	-	√
23	Sesquipedalian.c	Sesquipedalian	-	√
24	Smurf	smurf4.c	-	√
25	OverDrop	overdrop.c	-	√
26	OpenTear	opentear.c	-	√
27	ExhoChargen	FireHack	-	√

Category	Feature
-	protocol
IP	diff-time stamp
	ip id
	IP tos
	ipttl
	ipheaderlen
	iplen
	is home srcip
	is home dstip
	is land
	ip frag flag
TCP	tcpsrc port
	tcpdst port
	tcp fin
	tcpsyn
	tcprst
	tcp push
	tcpack
	tcpurg
	tcp offset
	tcp win size
UDP	udpsrc port
	udpdst port
ICMP	icmp type
	icmp code
	icmp id
	icmp sequence

Pukkawanna et al. [13] proposed the Lightweight Detection system (LD^2) to detect Denial of Service Attack (DoS). The target attacks included SYN Flood, ICMP flood, Port scan Host scan, UDP flood and smurf. The system preprocessed the network into five features (srcIP, protocol, dstIP, srcPort, and dstPort). The background traffic environment had two types: controlled environment and real traffic environment. In the controlled network environment, they used Iperf to generate the UDP traffic in various rates. In the real network environment, they replied traces by using tcpreplay. The trace was sampled from WIDE Backbone (100-150 Mbps). In each experiment, they generated a DoS attack on the top of a single background trace. Figure 2.4 showed the graph pattern that the system used for detecting each type of the attacks. For example, SYN flood had the same (srcIP, prot, dstIP, dstPort) but various srcPort. Thus, the detection system needed the training process in order to find a threshold for each attack type (Table 2.7). They generated multiple attacks at once (12 instances). The experiment result showed that the LD^2

performed well with the 100% detection rate (except a host scan that could not detect some activities) with no false positive. They also evaluated a system performance in term of CPU consumption and memory consumption by using a systat tool. It showed that the increasing packet rate of a background also increased the CPU usage [Figure 2.5]. The maximum CPU utilization of the LD^2 was 16% at 7,000 pps and the memory consumption was 20 MB. The behavior of the memory consumption is shown in [Figure 2.6].

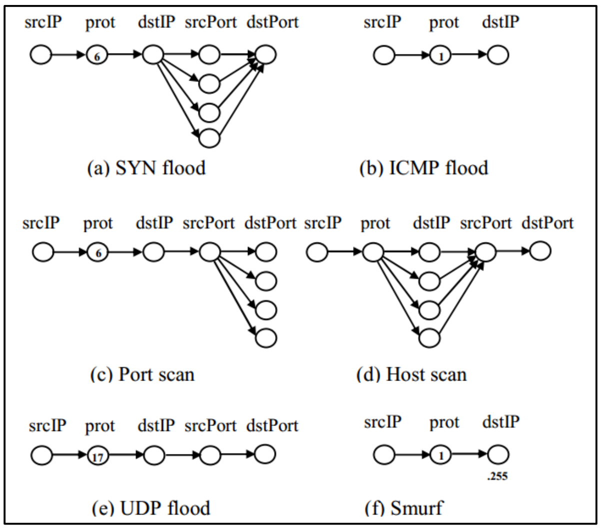


Figure 2.4 DoS attack graphlets [13]

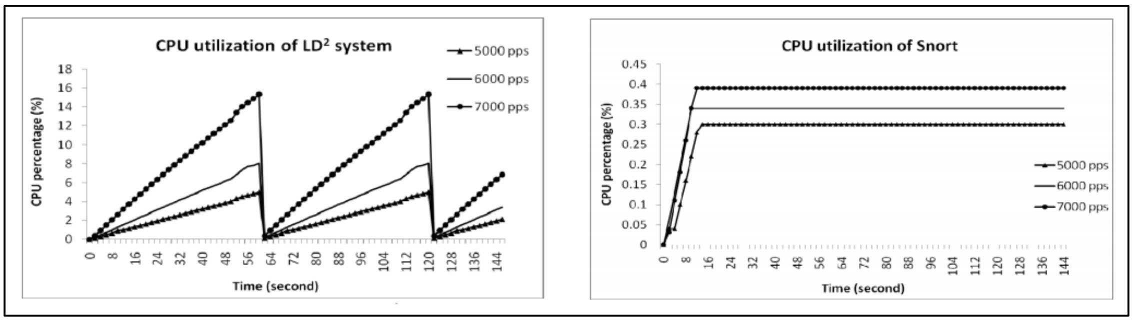


Figure 2.5 CPU initialized for LD^2 (left) and Snort (right) [13]

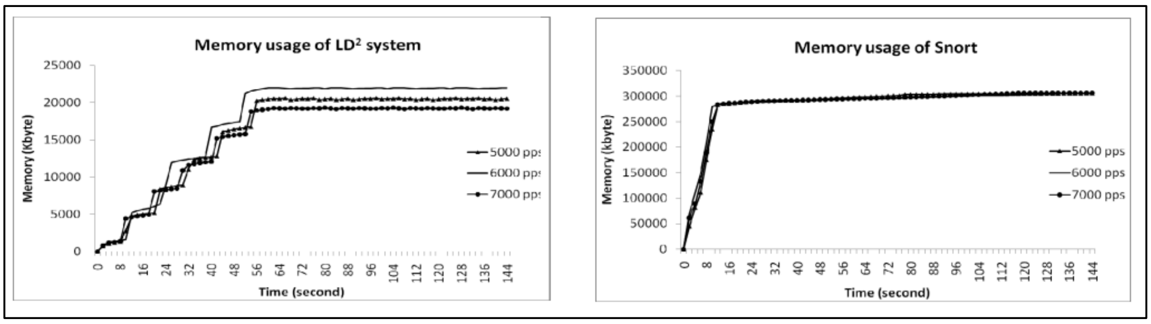


Figure 2.6 Memory usage for LD^2 (left) and Snort (right) [13]

Table 2.7 Threshold for attack graphlets [13]

Dos Type	**Threshold Parameters (per minute)**	**Upper Bound**	**Suggest Value**
SYN flood	Source ports	1,998	1,598
UDP flood	Number of UDP packets	1,918	1,534
ICMP flood	Number of ICMP packets to broadcast address	2,151	1,721
Smurf	Number of ICMP packets to broadcast address	2,151	1,721
Port scan	Destination ports	394	313
Host scan	Destination IP adresses	5	4

Su [14] proposed the real-time IDS for large-scale attacks by using fuzzy association rules. The technique derived features from a packet header from the open network within every 2 seconds (one record per two seconds). There were 16 features used in this technique as shown in Table 2.8. The system architecture is shown in Figure 2.7. The computer A preprocessed data from a real network and sent a record to the computer B to create a fuzzy rule. The computer D compared the rules between the computer B and C to find the attacks. This experiment was tested on 30 DoS attacks. A network topology is shown in Figure 2.8. IP traffic (a sender) was a computer used to generate the background traffic, such as TCP packets, UDP packets, ICMP packets and ARP packets. It connected to the internet. There was the IP traffic (a receiver) located in the local network. An attack generator was used to generate attacks (DoS) where the victim was found in the local network. The system was also located in the local network. It monitored the traffic in the local network. The traffic rate during the experiment was 0-80 Mbps. The result is shown in Figure 2.9. We can see that the system responded to the attack five time units (10 seconds) after the system was attacked. This system could only give an alarm signal when the network was under attack. However, it could not provide any useful information to prevent the network from malicious network activities.

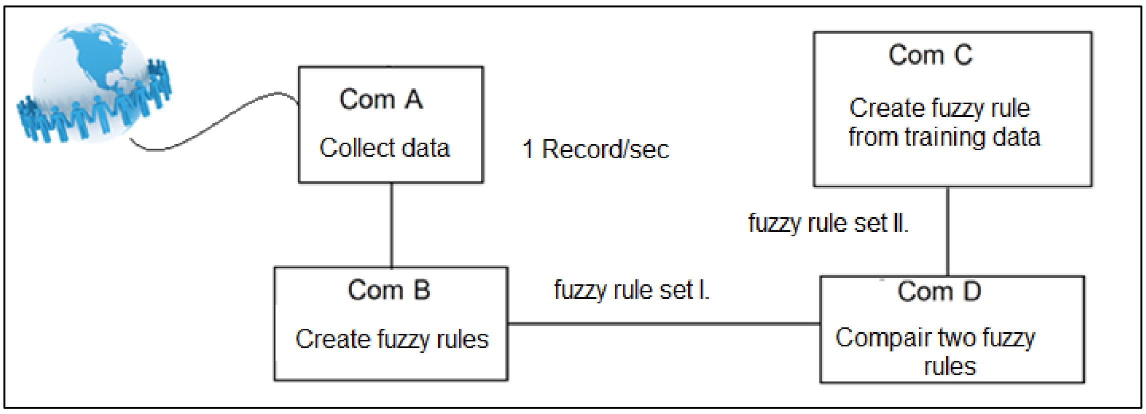


Figure 2.7 Architecture of NIDS [14]

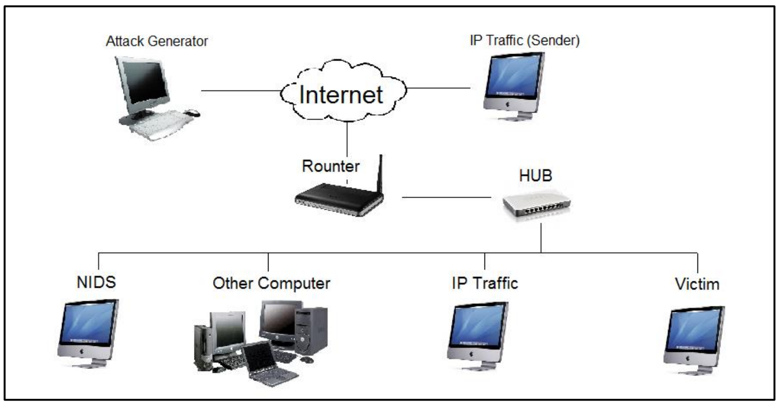


Figure 2.8 Network topology for simulation [14]

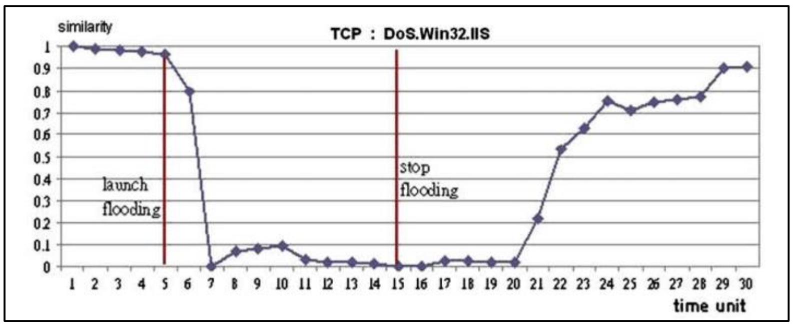


Figure 2.9 Similarity degradation during flooding for DoS.Win32.IIS [14]

Table 2.8 Feature list of real-time network IDS for large-scale attacks based on an incremental mining approach [14]

#	**Protocol**	**Feature**
1	TCP	source IP+SYN count
2	TCP	source IP+URG_Flag+URG_data count
3	TCP	Source IP+ACK-Flag+ACK count
4	ARP	Source IP+ARP count
5	IP	Destination IP slots hit
6	IP	Header length 1=20 count
7	IP	MF_Flag count
8	IP	(total length > 1400||<40)&&TTL=64 count
9	IP	Checksum_error count
10	TCP	ACK_Flag+ACK count
11	TCP	Checksum_error count
12	UDP	Same_length_interval count
13	ICMP	Type error count
14	ICMP	Checksum_error count
15	ICMP	Checksum_error count
16	ICMP	Length>1000count

Komviriyavut et al. [15] proposed a real-time detection. They used a packet sniffer to sniff the packets in the network every 2 seconds and preprocessed it into 13 features by counting the number of connections between two IP addresses every 2 seconds [Table 2.9]. They also used the decision tree algorithm to classify the data. In order to evaluate the performance, they collected the normal data from the network traffic in the Department of CPE from KMUTT. They simulated the attacks in a closed environment by using attack tools which consisted of 18 types of attacks [Table 2.10]. The dataset could be categorized into 3 types; DoS, Probe and normal data. The result showed that this algorithm had 97.5 percent of the detection rate. This technique was efficient to be used in an actual network environment in terms of speed, memory consumption and CPU consumption.

Examples of the record of the normal network data from the preprocessing phase are shown below.

```
2138,33,33,4,4,644,2136,0,0,0,0,0,0,Normal
12,2,2,0,0,1,12,0,0,0,0,0,0, Normal
```

Table 2.9 Features in online dataset [15]

No.	Feature Description	Data Type
1	Number of TCP packets	Integer
2	Number of TCP source ports	Integer
3	Number of TCP destination ports	Integer
4	Number of TCP fin flags	Integer
5	Number of TCP syn flags	Integer
6	Number of TCP reset flags	Integer
7	Number of TCP push flags	Integer
8	Number of TCP ack flags	Integer
9	Number of TCP urgent flags	Integer
10	Number of UDP packets	Integer
11	Number of UDP source ports	Integer
12	Number of UDP destination ports	Integer
13	Number of ICMP packets	Integer

Kachurka and Golovko [16] proposed a neural network approach for real-time network intrusion detection. This algorithm could detect the attacks without the training dataset. This experiment considered three different types of the attacks: tcp scan, sysn flood and udp flood (500 records of each attack). The feature names of each record were timestamp, duration of connection in seconds, source’s and destination’s IP-addresses, name of the service used, port number, the number of bytes transferred and the result flag of the connection. They used both KDD99 dataset and real-time dataset to evaluate the algorithm. This technique was able to detect unknown attacks at least 97% of the detection rate for each type of the attacks (use the KDD99 dataset to evaluate).

Casas et al. [17] proposed Unsupervised Network Intrusion Detection (NIDSs) using Sub-Space Clustering Algorithm and Multiple Evidence Accumulation Algorithm. The NIDSs was able to detect attacks without the training dataset. The system was tested in an offline environment (with the KDD99 dataset) and an online environment. In the online environment, they used the traffic trace from the MAWI repository of the WIDE project and the METROSEC project. These two network traces were generated over the past ten years. They preprocessed the data network into 9 features [Table 2.11]. The algorithm could be classified into two classes which were a positive class (attack) and a negative class. The result showed that 90% of the attacks were correctly detected.

Table 2.10 Attack names in the dataset [15]

No.	Data	Tools (to Generate)	Category
1	Smurf	Smurf.c	DoS
2	UDP Flood	Net Tools 5	DoS
3	HTTP Flood	Net Tools 5	DoS
4	Jping	Jping.c	DoS
5	Port Scan	Net Tools 5	Probe
6	Advance Port Scan	Net Tools 5	Probe
7	Host Scan	Host Scan 1.6	Probe
8	Connect	NMapWin 1.3.1	Probe
9	SYN Stealth	NmapWin 1.3.1	Probe
10	FIN Stealth	NmapWin 1.3.1	Probe
11	UDP Scan	NmapWin 1.3.1	Probe
12	Null Scan	NmapWin 1.3.1	Probe
13	Xmas Tree	NmapWin 1.3.1	Probe
14	IP Scan	NmapWin 1.3.1	Probe
15	ACK Scan	NmapWin 1.3.1	Probe
16	Window Scan	NmapWin 1.3.1	Probe
17	RCP Scan	NmapWin 1.3.1	Probe
18	Normal	Actual Environment	Normal

Table 2.11 Features used in NIDSs [17]

No.	Feature Description	Abbreviation
1	Number of source IP	nSrcs
2	Number of destination IP	NDsts
3	Number of TCP source ports	nSrcPorts
4	Number of TCP destination ports	nDstPorts
5	Ratio of number of sources to number of destination	nSrcPorts/nDstPorts
6	packet rate	nPkts/sec
7	fraction of ICMP packets	nICMP/nPkts
8	number of SYN packets	nSYN/nPkts
9	average packets size	avgPktsSize

Table 2.12 Summary of Online IDS

Year	**Author**	**Algorithm**	**DR(%)**	**FP(%)**	**Number of Features**	**Output**
2002 [11]	Labib and Vemuri	NSOM	-		10	Normal, DoS
2005 [12]	Amini et al.	Neural Network (ART and SOM)	97.427	1.99	27	Normal, Attack
2007 [13]	Pukkawanna et al.	BLINd classification	100 (accept host scan)	0 (accept host scan)	5	SYN Flood, ICMP flood, Port scan Host scan, UDP flood and smurf
2009 [14]	Su et al.	Fuzzy association rules	N/A	N/A	16	Normal, DoS
2009 [15]	Komviriyavut et al.	Decision Tree and Rule Based	97.5	0.6	13	Normal, DoS, Probe
2011 [16]	Kachurka and Golovko[14]	Neural Network	N/A	N/A	16	Normal, Attack
2012 [17]	Casas et al.	Clustering	N/A	N/A	9	Normal, Attack

2.2 Background Study

2.2.1 Artificial Intelligence (AI) [18]. Major AI researchers and textbooks define the field as "*The study and design of intelligent agents*" where an intelligent agent is a system that learns from giving knowledge and takes action that maximizes its chances to achieve its goal.
John McCarthy : "the science and engineering of making intelligent machines"

2.2.2 Machine Learning [19], a branch of artificial intelligence, is about the construction and study of systems that can automatically learn from experiences and get more accurate results. The definition of the machine learning is described as follows:

Arthur Samuel : "*Field of study that gives computers the ability to learn without being explicitly programmed*"

Tom M. Mitchell : "*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E*"

The learning process of the machine learning can be categorized into four types of the machine learning described as follows: [20]

1. Supervised learning: during the learning process, the system will be told by the training dataset what is correct and what is not correct.

2. Unsupervised learning: during the learning process, the correct answers are not provided; the algorithm will identify similarity of the input data and categorize the similar input together instead.

3. Reinforcement learning: during the learning process, the algorithm will be told what is wrong but not be told what is correct. It has to explore and try out different possibilities until it works out how to get the right answer.

4. Evolutionary learning: biological evaluation can be considered as a learning process such as the process that living things adapt their generation to survive in an environment.

There are many ideas proposed to make the algorithm learn. In this work, we are interested in combining fuzzy logic and genetic algorithms together which is a supervised learning approach.

2.2.3 Fuzzy Logic can help in decision making or reasoning in an uncertain situation. From Figure 2.10, the fuzzy value is in a range of completely true and completely false but Boolean logic has only true or false.

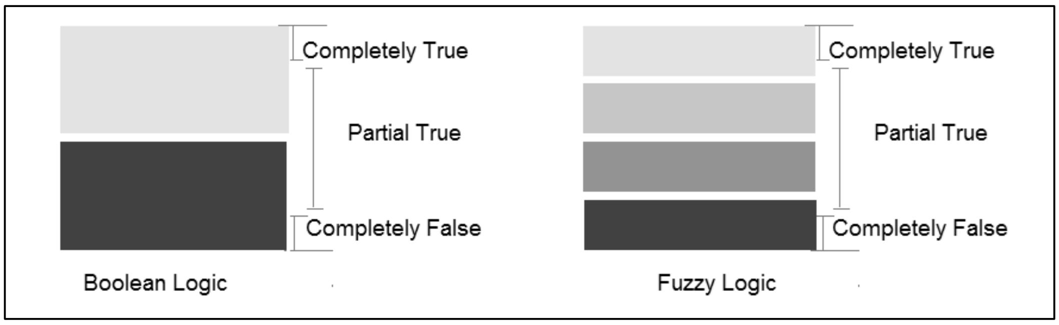


Figure 2.10 Boolean logic and fuzzy logic

Fuzzy logic uses a membership function to find a solution in an uncertain situation. There are many types of fuzzy functions such as a triangular membership function and a trapezoidal membership function.

For example:

> The trapezoidal membership function has three parameters {a, b, c, d} and x is an input value. The fuzzy value (from the input x) will be calculated using the conditions from Figure 2.11.

$$\text{trapezoidal}\,(x;a,b,c,d)=\begin{Bmatrix} 0, & x\le a \\ \frac{x-a}{b-a}, & a\le x\le b \\ 1, & b\le x\le c \\ \frac{d-x}{d-c}, & c\le x\le d \\ 0, & d\le x \end{Bmatrix}$$

Figure 2.11 Trapezoidal membership function [22]

2.2.4 Fuzzy Rule contains many fuzzy logics by using an if-then condition. Figure 2.12 presents a fuzzy rule by using many fuzzy logics where x_i is a fuzzy value that is calculated from the fuzzy logic i, A_i is a threshold value from the fuzzy logic i. All input values will be calculated using the fuzzy logic. When all fuzzy values match to rule 1 then the rule will classify it in to Class A.

Rule 1: if x_1 is A_{i1} and x_2 is A_{i2} and ... then Class A

Figure 2.12 Fuzzy rule

2.2.5 Genetic Algorithm (GA) Genetic algorithms are the evolutionary technique that uses the crossover and mutation operators to solve the optimization problems including NP-hard (non-polynomial) problems. It uses a natural evolution concept of only a "strongest or best solution" will survive among evolution of various populations. The technique does not guarantee an optimal solution. However, it can give a well-enough solution in the given time period. The genetic main algorithm process consists of the following approaches:

- Encoding: each gene is a parameter that a genetic algorithm uses for solving problems. The sequence of the genes is called a chromosome. A chromosome is one solution of that problem.

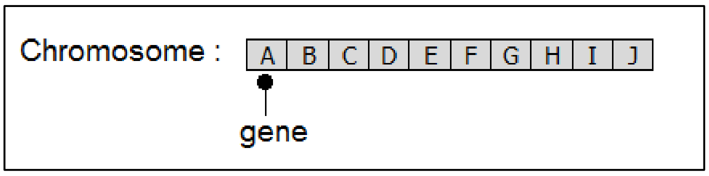
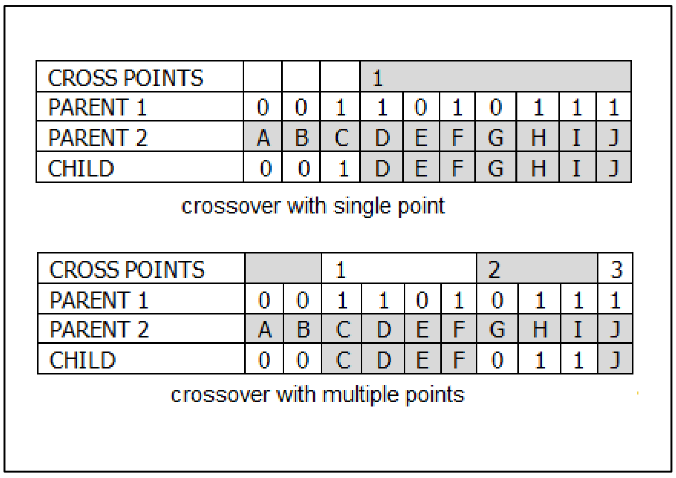


Figure 2.13 Example of chromosome

- Crossover: the approach to create a new chromosome from an existing chromosome by exchanging parts of the chromosomes (genes) between two chromosomes. In Figure 2.10, parent 1 and parent 2 exchange the chromosomes in a single point and multiple points.



CROSS POINTS				1						
PARENT 1	0	0	1	1	0	1	0	1	1	1
PARENT 2	A	B	C	D	E	F	G	H	I	J
CHILD	0	0	1	D	E	F	G	H	I	J

crossover with single point

CROSS POINTS			1				2			3
PARENT 1	0	0	1	1	0	1	0	1	1	1
PARENT 2	A	B	C	D	E	F	G	H	I	J
CHILD	0	0	C	D	E	F	0	1	1	J

crossover with multiple points

Figure 2.14 Genetic algorithm crossover multi values

- Mutation: the approach to create a new chromosome from an existing chromosome by randomly choosing the chromosome and randomly changing the gene.
- Evaluation: the function plays an important role in genetic algorithms. It is used to define the value of the chromosome.

2.2.6 KDD99 Dataset

KDD99 dataset is a benchmark dataset for an intrusion detection system. It was established in 1999 from MIT Lincoln labs in order to evaluate research results in intrusion detection. The Lincoln labs used the TCP dump to capture the local-area network in the Air Force environment. It was also used with multiple attacks. There were two file versions of the KDD99 dataset: 10% version file (about 500,000 records) and full version file (about 5 million records). Table 2.13 shows a number of the records and a number of the distinct records of each attack type in the 10% version file. Table 2.14 shows 41 features of the dataset.

Table 2.13 Number of each attack in 10% version file of KDD99 dataset [21]

Attack	#Original Records	#Distinct Records	Class
normal	97,277	87,831	Normal
back	2,203	994	DoS
land	21	19	DoS
neptune	107,201	51,820	DoS
pod	264	206	DoS
smurf	280,790	641	DoS
teardrop	979	918	DoS
satan	1,589	908	Probe
ipsweep	1,247	651	Probe
nmap	231	158	Probe
portsweep	1,040	416	Probe
guess_passwd	53	53	R2L
ftp_write	8	8	R2L
imap	12	12	R2L
phf	4	4	R2L
multihop	7	7	R2L
warezmaster	20	20	R2L
warezclient	1,020	1,020	R2L
spy	2	2	R2L
buffer_overflow	30	30	U2R
loadmodule	9	9	U2R
perl	3	3	U2R
rootkit	10	10	U2R
Total	**494,020**	**145,740**	

Examples of the data records in the KDD99 dataset:

0,tcp,http,SF,241,261,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,34,169,1.00,0.00,0.03,0.04,0.00,0.00,0.00,0.00,normal.
0,tcp,other,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,511,1,0.14,0.00,0.86,1.00,0.00,1.00,0.00,255,1,0.00,1.00,0.00,0.00,0.13,0.00,0.87,1.00,satan.
0,icmp,ecr_i,SF,1032,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,510,510,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,smurf.
0,tcp,private,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,132,8,0.00,0.00,1.00,1.00,0.06,0.07,0.00,255,8,0.03,0.06,0.00,0.00,0.00,0.00,1.00,1.00,neptune.
0,udp,private,SF,28,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,34,34,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,1,0.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,teardrop.

Network attacks fall into four main categories.

- **Denial of Service (DoS)** is a network attack that causes computer resources to be unavailable. DoS can happen from a person or multiple people. The target of the DoS attack is to serve a host on a high-profile web server such as banks, credit card payment gate way. Attackers attempt to force victims to either reset or consume network resources in order to destroy services. There are many methods used for this attack such as SYN flood, Tear drop attack and Peer to per attack.
- **Port Scan (Probe).** Port scanner is a tool designed to probe a server for an open port. Attackers can use this application to monitor behavior of the target and exploit vulnerability of that target.
- **Remote to Local Attack (R2L).** Attackers send packets to a machine and exploit machine’s vulnerability to gain the local access as an authenticated user, such as a password guessing attack.
- **User to Root (U2R).** Attackers will start normal access to a user account and exploit vulnerability in order to gain unauthorized access to the root. In common, this kind of the attack can cause the buffer overflow.

Table 2.14 Forty one features of KDD99 dataset [21]

#	Feature	Description	Type
1	Duration	duration of the connection.	Cont.
2	protocol type	connection protocol (e.g. tcp, udp)	Disc
3	Service	destination service (e.g. telnet, ftp)	Disc.
4	Dlag	status flag of the connection	Disc.
5	source bytes	bytes sent from source to destination	Cont.
6	destination bytes	bytes sent from destination to source	Cont.
7	Land	1 if connection is from/to the same host/port; 0 otherwise	Disc.
8	wrong fragment	number of wrong fragments	Cont.
9	Urgent	number of urgent packets	Cont.
10	Hot	number of “hot” indicators	Cont.
11	failed logins	number of failed logins	Cont.
12	logged in	1 if successfully logged in; 0 otherwise	Disc.
13	# compromised	number of “compromised’’ conditions	Cont.
14	root shell	1 if root shell is obtained; 0 otherwise	Cont.
15	su attempted	1 if “su root’’ command attempted; 0 otherwise	Cont.
16	# root	number of “root’’ accesses	Cont.
17	# file creations	number of file creation operations	Cont.
18	# shells	number of shell prompts	Cont
19	# access files	number of operations on access control files	Cont.
20	# outbound cmds	number of outbound commands in an ftp session	Cont.
21	is hot login	1 if the login belongs to the “hot’’ list; 0 otherwise	Disc.
22	is guest login	1 if the login is a “guest’’ login; 0 otherwise	Disc.

Table 2.14 Forty one features of KDD99 dataset [21] (Continued)

#	Feature	Description	Type
23	Count	number of connections to the same host as the current connection in the past two seconds	Cont.
24	srv count	number of connections to the same service as the current connection in the past two seconds	Cont.
25	serror rate	% of connections that have “SYN’’ errors	Cont.
26	srvserror rate	% of connections that have “SYN’’ errors	Cont.
27	rerror rate	% of connections that have “REJ’’ error	Cont.
28	srvrerror rate	% of connections that have “REJ’’ error	Cont.
29	same srv rate	% of connections to the same service	Cont.
30	diff srv rate	% of connections to different services	Cont.
31	srv diff host rate	% of connections to different hosts	Cont.
32	dst host count	count of connections having the same destination host	Cont.
33	dst host srv count	count of connections having the same destination host and using the same service	Cont.
34	dst host same srv rate	% of connections having the same destination host and using the same service	Cont.
35	dst host diff srv rate	% of different services on the current host	Cont.
36	dst host same src port rate	% of connections to the current host having the same src port	Cont.
37	dst host srv diff host rate	% of connections to the same service coming from different hosts	Cont.
38	dst host serror rate	% of connections to the current host that have an S0 error	Cont.
39	dst host srvserror rate	% of connections to the current host and specified service that have an S0 error	Cont.
40	dst host rerror rate	% of connections to the current host that have an RST error	Cont.
41	dst host srvrerror rate	% of connections to the current host and specified service that have an RST error	Cont.

CHAPTER 3 RESEARCH MOTHODOLOGY

From the literature review presented in chapter 2, the Fuzzy Logic was often chosen as an approach for network intrusion detection with low research consumption. It was also robust for unknown attack detection. Thus, we are interested in the Fuzzy Logic and the Genetic Algorithm. The Genetic Algorithm can help the fuzzy logic to learn a new data/solution in changing an environment. Therefore, the fuzzy genetic algorithm is chosen for our network-based intrusion detection. Moreover, we test the performances of our intrusion detection approach with both known and unknown network data.

Chapter three is organized as follows: the overview of our IDS system is described at the beginning of the chapter. Then, section 3.1 explains how our IDS preprocesses the online data and shows examples of the online dataset. Section 3.2 shows an IDS algorithm (fuzzy genetic algorithm), section 3.3 explains the testing method and evaluation criteria of our IDS system and section 3.4 displays simulation tools.

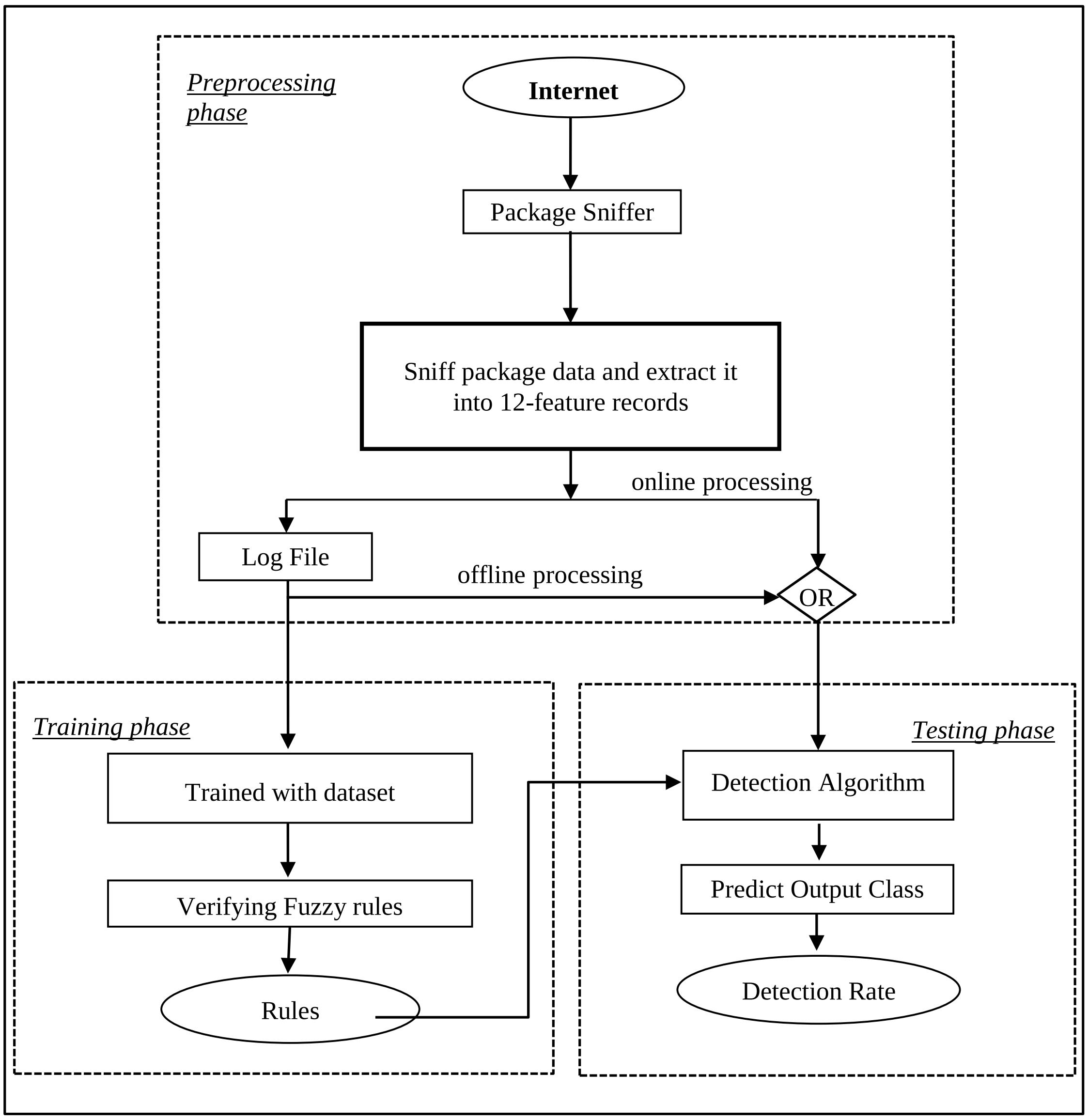


Figure 3.1 Real-time detection model

Our Real-time IDS, shown in Figure 5, consists of three phases: the pre-processing phase, the training phase and the testing phase. First, we create a network dataset by capturing the network data in CPE department of King Mongkut's University of Technology Thonburi in different time in a day for 1 month. The data packets are pre-processed using a packet header. The essential features which represent the network activity are extracted from this data. The extracted features are considered as the key-signature features, representing the main characteristics of the data. Then, the pre-processed data with the key signature extraction is sent to the training phase so that we can obtain fuzzy rules. In the training phase, the fuzzy rules are evolved by a genetic evolution concept. We can evaluate the performances of the fuzzy rules in the testing phase. Moreover, we can use the fuzzy rules to detect network attacks in an actual network environment.

3.1 Preprocessing Phase

In the pre-processing phase, we use a packet sniffer to extract network packet information as described in Komviriyavut et al [15]. This is shown in Figure 5. Each record consists of 12 data features. The features along with the data types are shown in Table 1.

Table 3.1 Twelve essential features in pre-processed data [15]

No.	Feature Description	Data Type
1	number of tcp packets	integer
2	number of tcp source ports	integer
3	number of tcp destination ports	integer
4	number of tcp fin flags	integer
5	number of tcpsyn flags	integer
6	number of tcp push flags	integer
7	number of tcpack flags	integer
8	number of tcp urgent flags	integer
9	number of udp packets	integer
10	number of udp source ports	integer
11	number of udp destination ports	integer
12	number of icmp packets	integer

The packet will be captured using Jpcap library [ref] for information extraction, the program will consider a connection between any two IP addresses (source IP and destination IP) and form a record for every 2 seconds. Then, the record will be sent to the detection phase in order to classify the attacks.

Examples of the data records where each record has 12 feature values and is labeled with its type (i.e. a normal data or an attack) can be shown as follows:

21,21,15,0,21,0,0,0,0,0,0,0, attack
169,2,90,169,0,0,0,0,0,0,0,0, attack
0,0,0,0,0,0,0,0,0,0,0,12683, attack
6,2,2,2,0,2,6,0,0,0,0,0, normal
111,2,2,0,0,2,111,0,0,0,0,0, normal
102,2,2,0,0,1,102,0,0,0,0,0, normal

3.2 Training Phase

In the detection phase, we use the fuzzy genetic algorithm as described in Fries [2]. The algorithm uses the data from a log file with the fuzzy genetic algorithm to train the rule. In this section, we will describe an idea of the fuzzy algorithm in section 3.2.1, an idea of the genetic algorithm in section 3.2.2, a methodology to encode the fuzzy rules in section 3.2.3 and the Fuzzy Genetic Algorithm in section 3.2.4.

3.2.1 Fuzzy Logic Algorithm

The fuzzy parameter is in between a range of 0-7 and the fuzzy value is between 0-1. The fuzzy rule is applied to each feature by using this set of the parameters {a, b, c, d}.

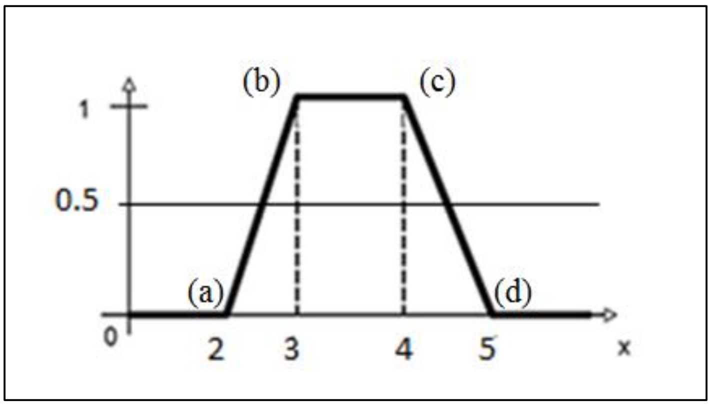


Figure 3.2 Trapezoidal fuzzy set {a=2, b=3, c=4, d=5}

From Figure 6, we can calculate the fuzzy value using these four equations:

1. if the data records between b and c, *probe =1;*
2. if the data records between a and b, $prob = \frac{attribute\_value - a}{b - a}$
3. if the data records between c and d, $prob = \frac{d - attribute\_value}{d - c}$
4. otherwise, *probe = 0;*

3.2.2 Genetic Algorithm

Procedure GA:

```
    Initialize:
        Initialize population P(t)

while (not (termination condition))
        {
        Create offspring F(t)
        Evaluate offspring F(t)
        Insert offspring in the population F(t)→ P(t)
        }
```

A Genetic Algorithm uses an evolutionary method to find the best solution. Each solution is encoded into a string called *"chromosome"*. At the beginning, the chromosomes are randomly initiated. A group of the chromosomes is called *"population-P(t)"* and *"offspring-F(t)"*. In creating an offspring step, the algorithm creates a new set of the chromosome using a reproduction method as described in next paragraph. Then, it evaluates values of the offspring, and inserts the offspring F(t) to the population P(t). The new generation of the chromosomes will be created and replaced the old generation until it reaches stopping criteria. The stopping criterion in this experiment is set to a certain number, such as 5,000 generations.

There are five ways of reproduction as follows:

I. Crossover: the approach to create a new chromosome from an existing chromosome by exchanging parts of the chromosomes (genes) between two chromosomes. We use one-point crossover, i.e.

Parent	p1:	A-B-C-D-|E-F-G-H-I-J
	p2:	D-E-H-A-|B-J-G-F-I-C
Child	c1:	A-B-C-D- B-J-G-F-I-C

II. Mutation: the approach to create a new chromosome from an existing chromosome by randomly choosing the chromosome to mutate and randomly changing its gene(s). This approach is applied to avoid the GA trapping in a local optimum. In our approach, we randomly choose two genes for mutation.

Parent	p1:	D-E-**H**-A-B-**J**-G-F-I-C
Child	c1:	D-E-**F**-A-B-**A**-G-F-I-C

- At point 3rd gene H mutate to F
- At point 6th gene J mutate to A

III. Alien: the program creates a new chromosome by randomly choosing every gene in the chromosome.

IV. Elitism best chromosome: the program keeps the best chromosome from the current population. The strongest chromosome will exist in the next generation.

3.2.2 String Encoding

a. Each feature will be encoded into the string as follows:

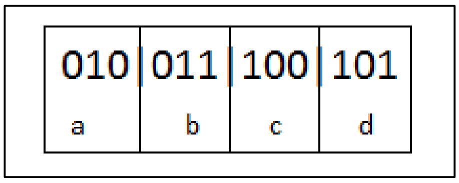


Figure 3.3 Fuzzy encoding for each feature {a=2, b=3, c=4, d=5}

b. Each chromosome refers to each feature. The records will be encoded as the chromosomes below which are series of fuzzy parameters for 12 features and the class at the end of the string.

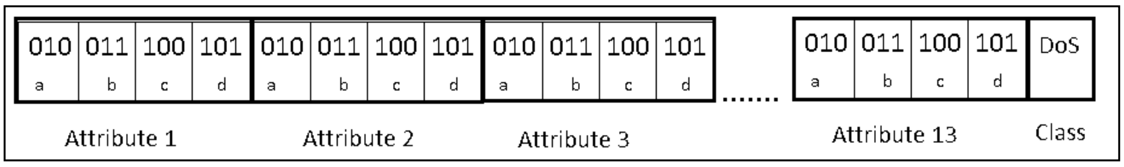


Figure 3.4 Encoding string

3.2.4 Fuzzy Genetic Algorithm

A Fuzzy Rule was developed by the genetic algorithm in order to find best fuzzy rules. The step of the algorithm is described below.

1. Initial population: Each generation has 20-50 chromosomes.
2. Finding probability: we calculate probabilities of beginning an attack for each feature, and then summarize all features in that record.
3. Classify Attack: we set the threshold = 0.5. When the total probabilities are less than 0.5, the record will be classified as normal.

```
Initial rules ();
while{
    for each record {
        for each rule{
        for each attribute{
                prob = fuzzy();
                totalprob = totalprob + prob;
            }
            If (totalprob> threshold)
            class is attack;
      else
            class is normal;
           }
compare the predicted result with actual result
find  A, B, $\alpha$, and $\beta$,
     }
                Calculate fitness
    // create next generation
    Evolutionary process();
```

Figure 3.5 Fuzzy genetic algorithm pseudo code

4. Finding A, B α, and β, where
   - A is a number of attack records in the dataset.
   - B is a number of normal records in the dataset.
   - α_i is a number of correctly identified attacks (α_i) for each chromosome
   - β_i is a number of normal connections incorrectly characterized as attacks (false positive, β_i) for each chromosome
   - Summarize α for this generation and summarize β for this generation

$$\alpha = \sum_{i=1}^{n} a_i$$

where n is a number of records in the dataset

$$\beta = \sum_{i=1}^{n} \beta_i$$

5. Calculate a fitness value:

   The program will calculate a fitness value for each rule by using equation below.

$$fitness \quad function = \frac{\alpha}{A} - \frac{\beta}{B} [3]$$

6. Evolutionary process:

   The program will generate next generation. The next generation includes

   20% of population from the current rule that has the highest value of fitness

   30% of population from the crossover method

   20% of population from the mutation method

   30% of population from the alien method

3.3 Detecting Phase

The testing phase is a process to evaluate performances of our algorithm. In this phase, a user can select the dataset from a log-file in order to evaluate accuracy of the fuzzy rule or connect to a real network environment in order to evaluate other performances of the IDS such as resource consumption and computation time. There are three steps in the testing phase as described below.

3.3.1 Data Normalization: The system will normalize each testing record to range 0-7, the maximum and minimum bounds are imported from the training phase. If the value of a testing record is greater than the maximum, the normalized value will be 7;

3.3.2 Data Classification: We use the rules from the training phase to classify the attack class and the normal class. There are 2 types of classification processes.

a. One rule classification: the program uses one rule to classify.

b. Two-rule classification: the program uses two rules together to classify network attacks. Then, the program will compare probability of being attacked from each rule with the threshold below.

```
If probability$_{rule1}$>threshold or probabilityes$_{rule2}$>threshold)
        then classify as attack.
    else
        classify as normal
```

3.3.3 Evaluation Criteria: There are four parameters that are used to evaluate accuracy of this algorithm and are described below.

- Detection rate (DR) is the percentage of the normal and attack classes correctly classified from the total number of the data records.
- True-positive rate (TP) is the percentage of the normal class correctly classified from the total number of the data records.
- True-negative (TN) is the percentage of the attack class correctly classified from the total number of the data records.
- False-positive (FP) is the percentage that the normal data records are classified as attacks from the total number of the normal data records.
- False-negative rate (FN) is the percentage that the attacks are misclassified from the total number of the attack records.

3.4 Simulation Tools

In this work, we use simulation tools to generate attacks in close environments in order to create training datasets and also use simulation tools to test our intrusion detection system. There are 17 types of the attacks that are interesting for real-time datasets including 4 types of DoS attack type and 13 types of Probe attack type. Table 3.2 shows each name of the attacks and the simulation tools.

Table 3.2 Attack type and simulation tools [15]

No.	Data	Tools	Category
1	Smurf	Smurf.c	DoS
2	UDP Flood	Net Tools 5	DoS
3	HTTP Flood	Net Tools 5	DoS
4	Jping	Jping.c	DoS
5	Port Scan	Net Tools 5	Probe
6	Advance Port Scan	Net Tools 5	Probe
7	Host Scan	Host Scan 1.6	Probe
8	Connect	Nmap Win 1.3.1	Probe
9	SYN Stealth	Nmap Win 1.3.1	Probe
10	FIN Stealth	Nmap Win 1.3.1	Probe
11	UDP Scan	Nmap Win 1.3.1	Probe
12	Null Scan	Nmap Win 1.3.1	Probe
13	Xmas Tree	Nmap Win 1.3.1	Probe
14	IP Scan	Nmap Win 1.3.1	Probe
15	ACK Scan	Nmap Win 1.3.1	Probe
16	Window Scan	Nmap Win 1.3.1	Probe
17	RCP Scan	Nmap Win 1.3.1	Probe
18	Normal	Actual Environment	Normal

CHAPTER 4 EXPERIMENTAL RESULTS

From chapter 3, our proposed algorithm is the fuzzy genetic algorithm. In this chapter, we will demonstrate our proposed approach (fuzzy genetic algorithm) with various views. We perform two types of experiments to evaluate our IDS including Offline IDS (section 4.1) and Online IDS (section 4.2). In the offline IDS, we implement and test our system using an operating system: Ubuntu ver.12.04. We use two datasets including the KDD99 dataset (discussed in section 4.1.1) and the real-time dataset (discussed in section 4.1.2). Moreover, we test our IDS with unknown attacks, shown in section 4.1.3 and compare it with various algorithms in section 4.1.4. In the online IDS, the operating system is Windows 7. We demonstrate the efficient IDS in terms of the detection rate and resource consumption by using an actual network environment from the Computer Engineering Department of KMUTT.

4.1 Offline Detection

4.1.1 Fuzzy GA with KDD99 dataset

In this experiment, we use the Fuzzy Genetic Algorithm with the KDD99 dataset. The table shows a number of each attack in the KDD99 dataset from two different versions. We can see that the number of records in the KDD99 dataset have different ratios for each attack type. For example, there are many *Smurf* and *Neptune* data records and few numbers of *Land* and *Pod*. The different numbers of the records are shown in Table 4.1.

In this experiment, we reduce the original 41 features in the KDD99 dataset into 8 features including *duration, src_bytes, num_failed_logins, root_shell, num_access_files, serror_rate, same_srv_rate and srv_count* [5].

Table 4.1 Number of records of each attack in KDD99 dataset (A-full version and B-10% version contain approximately 5,000,000 records and 200,000 records respectively)

Normal	Attack Type	#of records A	#of record B
Normal	Normal		97,278
Smurf	Dos	2,807,886	280,790
Neptune	Dos	1,072,017	107201
Teardrop	Dos	979	979
Back	Dos	2,203	2,203
Land	Dos	21	21
Pod	Dos	264	264
Ipsweep	Probe	12,481	1,247
Nmap	Probe	2,316	231
Portsweep	Probe	10,413	1,040
Satan	Probe	15,892	1,589
Total		**3,924,472**	**494,021**

4.1.1.1 One-rule

First, we train our fuzzy genetic algorithm with a training dataset. Table 2 shows a result of the fuzzy genetic algorithm in a training process. The dataset from this experiment was randomly collected from 10% file version for training including 160,147 records of attack data and 39,387 records of normal network data. The dataset was reduced to 8 features as obtained [ref] and its value was normalized to be in the range of 0-7. The rule was established and evaluated by the training dataset. There were two output classes, normal class and attack class. From Table 2, the detection rate is 98.72% with the low false positive of 0.13%. Table 3 shows the fuzzy rule of this experiment. There are four parameters (a, b, c and d) for each feature. Information of the fuzzy rules is described in section 3.2.3.

Table 4.2 Experimental result from Fuzzy Genetic Algorithm with KDD99 dataset

Name	#Attack	#Normal	TP(%)	TN(%)	FN(%)	FP(%)	DR(%)
KDD99 dataset	160147	39,387	99.87	98.45	1.55	0.13	98.72

Table 4.3 Detection rule of KDD99 dataset obtained from training process

Fuzzy parameter	Features							
	0	1	2	3	4	5	6	7
A	5	2	1	1	4	5	4	0
B	5	5	1	4	4	5	4	0
C	7	7	1	6	4	7	7	1
D	7	6	3	6	6	6	7	1

Next, we investigate in detail for each attack name used in the training dataset. In Table 4.2-1, we extract each attack from the first experiment. This experiment uses the same dataset as shown in Table 2 (160,147 of attack records and 39,387 of normal records). From the table, we can see that most attack types have the high detection rate except Back and Land with the detection rate of only 16.56% and 15.58% respectively. There is the low false negative rate of most types of the attacks except Nmap having 16.13% of the false negative rate. However, there are a lot of attacks that have the high false positive rate including Back (FP: 85.33%), Pod (FP: 84.66%), Ipsweep (FP: 6.64%), Nmap (FP: 6.3%) and Portsweep (FP: 6.4%). In summary, in the KDD99 dataset, there are some attacks that the fuzzy rule cannot distinguish them from the normal network behavior including Back, Pod, Ipsweep, Nmap and Portsweep.

These two experiments show that the KDD99 dataset have the high detection rate. However, when investigating in detail, there is misclassification in the Back attacks and the Pod attacks. The proportion of the attacks in the KDD99 testing dataset is affected by the detection rate. In this case, the detection rate is biased by Smurf and Neptune which are the main part of the whole dataset and have the high detection rate. Therefore, we cannot use only the detection rate to evaluate the IDS.

Table 4.4-1 Experimental results of Fuzzy Genetic Algorithm with KDD99 dataset

Name	**Class**	**# Attack**	**Evaluation Criteria**				**DR(%)**	**Data input**
			TP(%)	**TN(%)**	**FN(%)**	**FP(%)**		
Back	DoS	893	14.67	100.00	0.00	85.33	16.56	back + normal
Land	DoS	4	99.61	100.00	0.00	0.39	99.61	land + normal
Pod	DoS	112	15.34	100.00	0.00	84.66	15.58	: :
Smurf	DoS	113,842	99.24	99.90	0.10	0.76	99.73	: :
Teardrop	DoS	371	98.81	100.00	0.00	1.19	98.83	: :
Neptune	DoS	43,375	99.85	99.66	0.34	0.15	99.75	: :
Ipsweep	Probe	479	93.36	100.00	0.00	6.64	93.44	: :
Nmap	Probe	93	93.70	83.87	16.13	6.30	93.67	nmap+normal
Portsweep	Probe	392	93.60	100.00	0.00	6.40	93.66	: :
Satan	Probe	586	99.26	96.25	3.75	0.74	99.22	: :
Total		**160,147**						

From Table 4.4-2, we investigated the reason that the Back attack and the Pod attack have low detection rates. We found that the 8 features that we selected from the original 41 features were affected by the detection rate as shown in Table 4.4-2. The result showed that when using the original 41 features with the fuzzy genetic algorithm, the detection increased in both back attack (96.07%) and pod attack (85.05%). While using the 8 features, the detection rate decreased for the back attack (15.47%) and the pod attack (12.19%).

Table 4.4-2 Experimental results comparing different numbers of features used for Back attack and Pod attack

#Feature	**TP (%)**	**TN (%)**	**FN (%)**	**FP (%)**	**DR (%)**
41	95.99	99.68	0.32	4.01	96.07
8	13.56	100.00	0.00	86.44	15.47

a. Back attack

#Feature	**TP (%)**	**TN (%)**	**FN (%)**	**FP (%)**	**DR (%)**
41	85.01	100.00	0.00	14.99	85.05
8	11.94	100.00	0.00	88.06	12.19

b. Pod attack

Note:

Pod attack is Ping of Death attack. The attacker sends a large size of a ping packet to a victim. The victim cannot handle the ping packet that is larger than the maximum IPv4 packet size causing a system clash.

Back attack [24] is Denial of Service against Apache web server where a client requests a URL containing many backslashes. The server will try to respond to these requests until it clashes. The features that are relevant to the back attack are the following [23]:

- Feature 5 (bytes sent from source to destination)
- Feature 6 (bytes sent from destination to source)
- Feature 10 (number of “hot” indicators)
- Feature 13 (number of “compromised’’ conditions)
- Feature 32 (count connections having the same destination host)

4.1.1.2 Two-rule

In this experiment, we use the fuzzy genetic algorithm to classify the KDD99 dataset in a different way in order to find a new approach to increase the detection rate. We use two different datasets that are sampled from 10-percent file version of the KDD99 dataset including 199,534 records of the training dataset and 199,514 records of the testing set.

- The training dataset contains 158,597 of DoS records, 1,550 of Probe records and 39,387 of normal records.
- The testing dataset contains 158,503 of DoS records, 1,674 of Probe records and 39,337 of normal records.

There are three steps in this experiment.

- **DoS Training Process:** we use the training dataset to train the DoS rule by focusing on only the DoS attack in the dataset (Class A). So, we group Probe and Normal into the same class (Class B) in order to find the DoS rule. The rule is shown in Table 4.6.
- **Probe Training Process:** we use the training dataset to train the Probe rule by focusing on only the Probe attack (Class A). So, we group DoS and Normal into the same class (Class B) in order to find the Probe rule. The rule is shown in Table 4.7.
- **Testing Process:** In the testing dataset, we use both DoS and Probe rules obtained previously to classify the testing dataset. The testing using these two rules is described in section 3.3.2.

Table 4.5 Experimental results of Fuzzy Genetic Algorithm with KDD99 dataset

Dataset Name	#Class A (records)	#Class B (records)	DR(%)	FN(%)	FP(%)
DoS Training process	158,597	40,937	91.93	0.21	3.91
Probe Training process	1,550	197,984	95.31	98.18	0.00
Testing process	160,177	39,337	95.88	20.45	22.85

Table 4.6 DoS rule with KDD99 dataset obtained from DoS training process

Fuzzy parameter	Features							
	0	1	2	3	4	5	6	7
A	4	0	2	1	4	4	4	0
B	4	1	2	5	4	6	5	1
C	4	2	2	2	4	7	7	2
D	4	3	2	6	6	6	6	2

Table 4.7 Probe rule with KDD99 dataset obtained from Probe training process

Fuzzy parameter	Features							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
A	1	2	1	2	1	0	5	3
B	2	3	2	3	1	0	5	6
C	5	5	4	3	2	0	6	1
D	6	5	6	5	7	1	6	7

4.1.2 Fuzzy GA with Real-time Dataset

In this experiment, we use the Fuzzy Genetic Algorithm with the real-time dataset. The output has two classes which are attack and normal. We collect the real-time dataset in the actual network environment in our research laboratory. It is the online network data from the Computer Engineering Department at King Mongkut’s University of Technology Thonburi (KMUTT). There are 17 types of attacks (4 types are DoS and 13 types are Probe). There are two sets of the data including:

Training dataset with 14,300 records including

- 6,300 records of the attacks, consisting of 300 records of each Probe name and 600 of each DoS record)
- 8,000 records of the normal data

Testing set with 26,500 records including

- 10,500 records of the attacks (500 records of each Probe name, 1000 of each DoS record)
- 16,000 records of the normal data

4.1.2.1 One-rule

We use the Fuzzy Genetic Algorithm with the real-time dataset (Training dataset) to find a rule for classifying the normal class and the attack class, as shown in Table 4.8. Then, we use the Testing dataset to evaluate performances of a rule. It shows that the Fuzzy Genetic Algorithm can classify the real-time dataset with the high detection rate (97.97%) and the low false alarm rate (the false negative rate is 3.39% and the false positive rate is 1.14%). Table 4.9 presents parameters of the rule obtained from the training process. There are 4 parameters (a, b, c and d) for each feature, and we have twelve features of the network data.

Table 4.8 Experimental results of Fuzzy Genetic Algorithm with real-time dataset

Name	#Attack	#Normal	Evaluation Criteria				DR(%)
			TP(%)	TN(%)	FN(%)	FP(%)	
Real-time dataset	10500	16000	98.86	96.61	3.39	1.14	97.97

Next, we investigate in detail each attack name used in the training dataset. In Table 4.8, we extract each attack from the first experiment. This experiment uses the same dataset as shown in Table 4.8 (10,500 of the attack records and 16,000 of the normal records). From the table, we can see that most of the attack types have the high detection rate but UDP-flood and IPscan have the low detection rates of 89.59% and 86.89% respectively. There are three types of the attacks that have the high false negative rate including Advances Port Scan (FN: 10.20%), Connectscan (FN: 16:20%) and IPscan (FN: 16.40%).

Moreover, there are two types of the attacks that have the high false positive rates which are UDP-flooded (FP: 11.06%) and IPscan (FP: 13.01%).

Table 4.9 Detection rule of real-time dataset obtained from training process

Fuzzy parameter	Features											
	0	1	2	3	4	5	6	7	8	9	10	11
A	4	0	0	0	0	4	3	0	0	6	0	1
B	4	1	1	1	1	4	3	1	6	6	2	1
C	6	4	2	4	3	5	3	1	7	7	7	0
D	6	7	3	6	7	7	4	2	7	7	7	7

Table 4.10 Experimental results of Fuzzy Genetic Algorithm with real-time dataset

Dataset name	Type	# attack	Evaluation Criteria				DR (%)	Data input
			TP(%)	TN(%)	FN(%)	FP(%)		
HTTPflooded	DoS	1,000	99.64	96.50	3.50	0.36	99.46	httpflooded+normal
Jping	DoS	1,000	99.98	100.00	0.00	0.02	99.98	jping+normal
Smurf	DoS	1,000	99.98	100.00	0.00	0.02	99.98	: :
UDPflood	DoS	1,000	88.94	100.00	0.00	**11.06**	**89.59**	: :
Ackscan	Probe	500	99.97	100.00	0.00	0.03	99.97	ackscan+normal
AdvancePortscan	Probe	500	100.00	89.80	**10.20**	0.00	99.69	: :
Connectscan	Probe	500	99.86	83.80	**16.20**	0.14	99.38	: :
Finscan	Probe	500	97.42	100.00	0.00	2.58	97.5	: :
Hostscan	Probe	500	100.00	97.00	3.00	0.00	99.91	: :
IPscan	Probe	500	86.99	83.60	**16.40**	**13.01**	**86.89**	: :
Nullscan	Probe	500	99.01	96.00	4.00	0.99	98.92	: :
Portscan	Probe	500	99.98	100.00	0.00	0.02	99.98	: :
RCPscan	Probe	500	98.63	99.00	1.00	1.38	98.64	: :
Synscan	Probe	500	99.35	95.80	4.20	0.65	99.24	: :
UDPscan	Probe	500	97.52	100.00	0.00	2.48	97.59	: :
Winscan	Probe	500	99.98	100.00	0.00	0.02	99.98	: :
XmasTree	Probe	500	99.11	99.40	0.60	0.89	99.12	: :

4.1.2.2 Two-rule

In Table 4.11, we use two rules to classify the network dataset as we perform in section 4.1.1 but change the dataset to the real-time dataset. In this experiment, we use the training dataset to create rules and use the testing dataset to test in a testing process. The result is shown in Table 4.11. We can increase detection rate to 95.88% with low false positive rate. In Tables4.12 and 4.13 present Probe rule and DoS rule obtained from the training process, respectively.

Table 4.11 Detection rate of real-time dataset from using two rules of Fuzzy Genetic Algorithm

Dataset name	#C-Attack	#C-Normal	DR(%)	FN(%)	FP(%)
Training DoS process	2,400	11,900	91.93	30.69	1.48
Training Probe process	3,900	10,400	95.31	10.53	2.34
Testing process	10,500	16,000	95.88	6.28	2.70

#C-Attack (considered as attack) is number of records that is trained as attack.
#C-Normal (considered as normal) is number of record that is trained as normal.

Table 4.12 Probe rule of real-time dataset from training process

Fuzzy parameter	Features											
	0	1	2	3	4	5	6	7	8	9	10	11
A	5	5	0	0	3	3	2	0	3	1	0	4
B	6	5	1	2	3	5	4	1	3	2	1	5
C	6	6	6	3	4	6	7	2	3	2	4	5
D	7	6	6	7	6	7	7	6	3	4	7	5

Table 4.13 DoS rule of real-time dataset from training process

Fuzzy parameter	Features											
	0	1	2	3	4	5	6	7	8	9	10	11
A	6	0	4	2	0	4	5	2	1	4	2	0
B	6	1	4	2	1	4	5	2	6	4	2	1
C	6	1	4	2	1	4	5	2	7	5	2	7
D	6	2	4	2	1	3	5	2	7	6	2	7

4.1.3 Fuzzy GA with Unknown Detection

In this experiment, we consider detecting unknown attacks with three different algorithms (Decision Tree Algorithm, Naïve Bayes Algorithm and Fuzzy Genetic Algorithm$_{2 \text{ rules}}$). We use 26,500 data records from the real-time dataset including 16,000 records of the normal dataset and 10,500 records of the attack dataset. The number of the records in each attack type is shown in Table 14. In Table 14, seven test cases are used in this experiment which are T1, T2, ..., T7. For each test case, 13 attack types as well as the normal network data are provided in the training dataset, while the other 3 attack types are used as an unknown testing dataset for our Fuzzy Genetic Algorithm. For example, in the first test case, we use the training set which does not have Advances Port Scan, Ack Scan and Xmas Tree. Then we use these three types of the attacks for the testing dataset. Moreover, we test each type. The output classes are attack and normal.

Table 4.14 Seven test cases with unknown data types

No.	Data Type	Category	#Record	T1	T2	T3	T4	T5	T6	T7
1	Normal Activity	Normal	10,500							
2	Smurf	DoS	1,000			√				
3	UDP Flood	DoS	1,000				√		√	
4	HTTP Flood	DoS	1,000		√					√
5	Jping	DoS	1,000					√		
6	Port Scan	Probe	500			√				
7	Host Scan	Probe	500				√			
8	Connect	Probe	500			√				
9	SYN Stealt	Probe	500					√		
10	FIN Stealt	Probe	500					√	√	√
11	UDP Scan	Probe	500				√			
12	Null Scan	Probe	500		√					
13	IP Scan	Probe	500		√					
14	Window Scan	Probe	500							
15	RCP Scan	Probe	500						√	√
16	Adv Port Scan	Probe	500	√						
17	Xmas Tree	Probe	500	√						
18	ACK Scan	Probe	500	√						

Table 4.16 shows that **Decision Tree Algorithm** has the low detection (9.59%-26.17%). It has less than 1% of the false negative rate in the test cases 4, 5, 6 and 7, but has the high false positive rate in every test case (about 90%). **With Naïve Bayes Algorithm**, the high detection rates are obtained in the test cases 1, 2, 3, 5 and 7, with the detection rates of 91.11%, 90.35%, 93.11%, 90.90% and 93.17% respectively. However, this algorithm still has the high false alarm rate in every test case. **Fuzzy Genetic Algorithm** has the high detection rate. Its lowest detection rate is the test case 4 with 92.17%. It also has the low false positive rate of 0.25%-3.24% and the low false negative rate of 2.40%-61.15%. From this experiment, we can see that the Fuzzy Genetic Algorithm is the most robust algorithm for the unknown detection comparing with the Decision Tree algorithm and the Naïve Bayes algorithm.

4.1.4 Intrusion Detection with various Approaches

In this experiment, we compare various algorithms for intrusion detection with the KDD99 dataset and the real-time dataset. The algorithms include Decision Tree Algorithm, Naïve Bayes Algorithm and Fuzzy Genetic Algorithm$_2$ rules. There are two output classes which are normal and attack.

There are four datasets used in this experiment, where the two datasets from the KDD99* and the two datasets are collected on-line recently from an actual network environment.

- KDD99* Training dataset with 20,000 data records sampling from 10% file version.
- KDD99* Testing dataset with 50,000 data records sampling from 10% file version.
- Real-time Training dataset with 14,300 data records
- Real-time Testing dataset with 26,500 data records.

(*The KDD99 dataset was reduced into 8 features as shown in section 4.1.1)

Table 4.15 Experimental results with unknown attack type with real-time dataset

Test Case	Unknown Attacks	Decision Tree[7]						Naïve Bayes						Fuzzy Genetic					
		DR (%)		FP(%)		FN(%)		DR (%)		FP(%)		FN(%)		DR (%)		FP(%)		FN(%)	
	Advance Port Scan	6.67		96.25		0.00		90.95		9.09		8.00		99.44		0.26		10.20	
1	Ack Scan	6.67	9.59	96.25	96.25	0.00	28.07	91.19	91.11	9.09	9.08	0.00	6.80	99.75	99.46	0.26	0.26	0.00	3.53
	Xmas Tree	4.12		96.25		84.20		90.81		9.09		12.40		99.74		0.26		0.40	
	HTTP Flood (DoS)	22.30		82.56		0.00		93.12		7.22		1.50		98.15		1.81		2.50	
2	IP Scan	19.47	26.17	82.56	82.56	15.60	4.00	93.00	90.35	7.22	7.21	0.00	29.15	97.15	97.10	1.81	1.81	36.20	11.65
	Null Scan	19.93		82.56		0.40		92.59		7.22		13.60		98.08		1.81		5.40	
	Smurf (DoS)	9.44		96.22		0.00		93.21		7.21		0.00		98.48		0.54		17.30	
3	Port Scan	6.70	14.16	96.22	96.22	0.00	2.85	93.01	93.11	7.21	7.21	0.00	4.40	99.48	98.09	0.54	0.54	0.00	12.90
	Connect Scan	6.35		96.22		11.40		92.47		7.21		17.60		98.97		0.54		16.80	
	UDP Flood	9.44		96.23		0.00		35.99		61.96		96.80		93.10		1.16		98.70	
4	Host Scan	6.62	14.41	96.23	96.23	2.20	0.55	39.82	36.94	61.96	61.96	3.00	71.90	98.79	92.17	1.16	1.16	2.80	61.15
	UDP Scan	6.69		96.23		0.00		37.16		61.96		91.00		97.53		1.16		44.40	
	Jping (DoS),	9.43		96.23		0.00		90.83		7.21		40.50		96.91		3.24		0.70	
5	Syn Scan,	6.67	14.44	96.23	96.23	0.60	0.15	92.90	90.90	7.21	7.21	3.40	24.25	96.54	96.79	3.24	3.24	10.60	3.00
	Fin Scan	6.68		96.23		0.00		92.62		7.21		12.60		96.86		3.24		0.00	
	UDP Flood,	9.44		96.23		0.00		55.07		43.06		74.90		93.59		0.64		98.70	
6	RCP Scan,	6.69	14.47	96.23	96.23	0.00	0.00	58.24	57.21	43.06	43.05	0.40	40.70	99.08	93.51	0.64	0.64	10.00	53.30
	Fin Scan	6.69		96.23		0.00		57.87		43.06		12.60		99.21		0.64		5.80	
	Http Flood,	9.44		96.23		0.00		93.15		7.21		1.10		97.16		2.96		1.00	
7	RCP Scan,	6.69	14.47	96.23	96.23	0.00	0.00	92.99	93.17	7.21	7.21	0.40	3.80	96.90	97.11	2.96	2.96	7.60	2.40
	Fin Scan	6.69		96.23		0.00		92.62		7.21		12.60		97.13		2.96		0.00	

Table 4.16 Number of KDD99 data records in training dataset and testing dataset

KDD99 Dataset				Real-time Dataset		
Attack Name	**Training**	**Testing**		**Attack Name**	**Training**	**Testing**
Normal	3,919	9,851		Normal	8,000	16,000
Back	84	241		Smurf	600	1,000
IPsweep	45	134		UDP Flood	600	1,000
Land	5	3		HTTP Flood	600	1,000
Neptune	4,419	11,062		Jping	600	1,000
Nmap	12	22		Port Scan	300	500
Pod	13	29		Host Scan	300	500
Portsweep	32	88		Connect	300	500
Satan	67	137		SYN Stealt	300	500
Smurf	11,359	28,348		FIN Stealt	300	500
Teardrop	45	85		UDP Scan	300	500
Total	**20,000**	**50,000**		Null Scan	300	500
				Adv Port Scan	300	500
				Xmas Tree	300	500
				ACK Scan	300	500
				Total	**14,300**	**26,500**

Table 4.17 Results from various detection algorithms

Dataset	Decision Tree	Naïve Bay	Fuzzy GA	
			2 rules	**1 rule**
KDD99 dataset	83.19	95.94	79.77	**99.77**
Real-time dataset	**99.71**	99.17	97.3	98.86

(a) True Positive rate

Dataset	Decision Tree	Naïve Bay	Fuzzy GA	
			2 rules	**1 rule**
KDD99 dataset	**98.84**	98.64	98.77	98.28
Real-time dataset	98.75	88.31	93.72	96.61

(b) True Negative rate

Dataset	Decision Tree	Naïve Bay	Fuzzy GA	
			2 rules	**1 rule**
KDD99 dataset	1.16	**1.36**	1.23	1.72
Real-time dataset	**1.25**	11.69	6.28	3.39

(c) False Negative rate

Dataset	Decision Tree	Naïve Bay	Fuzzy GA	
			2 rules	**1 rule**
KDD99 dataset	16.81	4.06	98.52	**0.23**
Real-time dataset	**0.29**	0.83	2.71	1.14

(d) False Positive

In Table 17, the highest true positive rate in the KDD99 dataset is the Fuzzy Genetic Algorithm with 1 rule (99.77%) and the Naïve Bayes Algorithm (95.94%). Moreover, every algorithm has high values of the true negative rate with the low false negative rate. In addition, the false positive rates from the different algorithms are different. The false positive rate in the Fuzzy Genetic Algorithm with 1 rule is as low as 0.23% while the Decision Tree Algorithm gives 16.81%.

From the table, we can see that the Decision Tree can classify the real-time dataset better than other algorithms (highest values of true positive (99.71%) and true negative (98.75%); lowest values of false negative (1.25%) and false positive (0.29%)). However, the Fuzzy Genetic Algorithm with 2 rules has the same rate of true positive as the Decision Tree Algorithm but has the lower false negative rate which is 93.72%.

4.2 Online Detection

In online IDS, we would like to test performances of IDS in term of CPU Consumption, Memory Consumption and Network Consumption.

4.2.1 Experimental Setting:

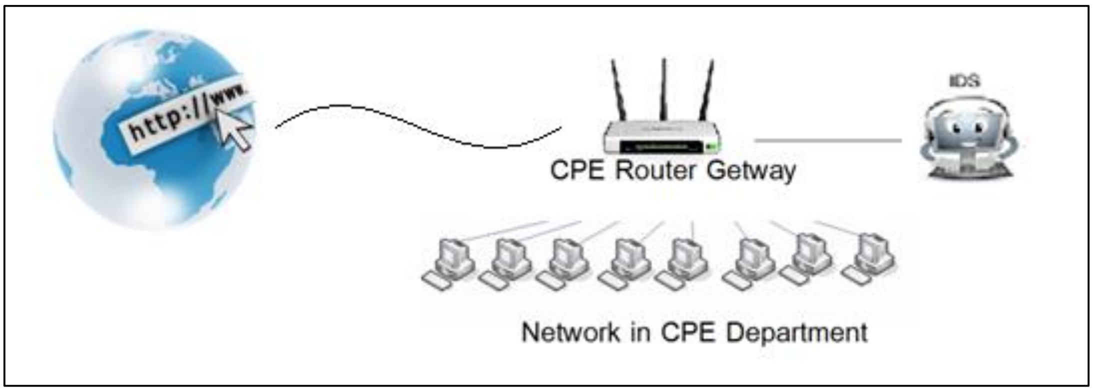


Figure 4.1 Real-time network environments

In this experiment, we monitor every packet in the CPE Department of KMUTT in both in and out of the network gate way. (The speed of the traffic is between 5-100 Mbit/sec). We connect our IDS to a gateway router using a mirror port during 12.30 pm. - 17.30 pm. on April 24, 2013. The IDS computer used Intel® Core™ i7-3770k CPU@ 3.5GHz 3.90 GHz RAM 8 GB Windows 7 Ultimate 67-bit with Network Interface card: Atheros AR8151 PCI-E Gigabit Ethernet Controller (NIDS 6.20).

In Table 4.18, there are 52,564,018 packets during the experimental time. It consists of 47,822,054 packets of TCP, 4,634,052 packets of UDP and 107912 of ICMP. In our real-time IDS, the system preprocessed these network data into 1,201,208 records. Moreover, our IDS classifies 1,519 records into the attack class. The CPU Consumption is between 7-14% while using only 2-2.5 GB of memory.

4.2.2 Experimental Result

Table 4.18 Experimental result from CPE network environment

TCP	UDP	ICMP	Total	Attack	Normal	CPU	Memory
47,822,054	4,634,052	107,912	1,201,208	1,519	1,199,689	7-14%	2-2.5 GB
Total: 52,564,018 packets							

Note:

TCP: Number of TCP packets
UDP: Number of UPD packets
ICMP: Number of ICMP packets
Total: Total number of records after preprocessing
Attack: Number of records that was detected as attack
Normal: Number of records that was detected as Normal
CPU: CPU Consumption
Memory: Memory Consumpt

CHAPTER 5 CONCLUSION

In this thesis, we proposed the fuzzy genetic algorithm to detect DoS and Probe attacks in both offline and online network environments. Our IDS can detect the attack in the real-time network environment. We began by evaluating accuracy of the fuzzy genetic algorithm in an offline dataset. The offline dataset includes a benchmark dataset (KDD99 dataset which was reduced to 8 features), and our real-time dataset. The result showed that the fuzzy genetic algorithm offered the high detection rate with the low false alarm rate on both datasets.

In addition, we explored in detail for each attack name in the dataset. We found that there were two attacks in the KDD99 dataset, namely Back and Pod, that were misclassified with the fuzzy genetic algorithm, while the algorithm could detect all attack types in our real-time dataset. From previous study, we have learned that the detection rate could be biased by the dataset. Therefore, we had to consider the false alarm rate and the proportion of the dataset. Next, we evaluated our fuzzy genetic algorithm by comparing with other algorithms considering both datasets. The accuracy of the fuzzy genetic algorithm is close to the results obtained from the decision tree algorithm.

We also compared our fuzzy genetic algorithm with other algorithms for detecting unknown attacks. We used only the real-time dataset to evaluate with seven test cases. Each of the training sets contained 13 attack types while the other 3 attack types were used as an unknown testing dataset. The results showed that the fuzzy genetic algorithm was the most robust algorithm for the unknown attack detection.

In the online network environment, we used our IDS to monitor the real-time traffic in the CPE Department of KMUTT (IDS PC spec: Intel® Core™ i7-3770k CPU@ 3.5GHz 3.90 GHz RAM 8 GB Windows 7 Ultimate 67-bit and Network Interface Card is Atheros AR8151 PCI-E Gigabit Ethernet Controller) in order to demonstrate performances of our IDS. The speed of the traffic was between 5-100 Mbit/sec. Our IDS consumed less than 14% of CPU resources while using only 2.5 GB of memory. In the real-time detection, our IDS could raise an alarm message within 2-3 seconds. This did not affect the PC performance when other applications were running.

REFERENCES

1. Gómez, J. and León, E., 2006, “A Fuzzy Set/Rule Distance for Evolving Fuzzy Anomaly Detectors”, **IEEE International Conference on Fuzzy Systems**, Art. No. 1682017, pp. 2286-2292.

2. Banković, Z., Stepanović, D., Bojanić, S. and Nieto-Taladriz, O., 2007, “Improving Network Security Using Genetic Algorithm Approach”, **Computers and Electrical Engineering**, pp. 438-451.

3. Tsang, C.-H., Kwong, S. and Wang, H., 2007, “Genetic-Fuzzy Rule Mining Approach and Evaluation of Feature Selection Techniques for Anomaly Intrusion Detection” **Pattern Recognition**, pp. 2373-2391.

4. Ensafi, R., Dehghanzadeh, S., Akbarzadeh-T, M.-R., 2008, “Optimizing Fuzzy K-means for Network Anomaly Detection using PSO”, **AICCSA 08 – 6th IEEE/ACS International Conference on Computer Systems and Applications**, Art. No. 4493603, pp. 686-693.

5. Fries, T.P., 2008, “A Fuzzy-Genetic Approach to Network Intrusion Detection”, **Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation**, pp. 2141-2146.

6. Abadeh, M.S., Mohamadi, H. and Habibi, J., 2011, “Design and Analysis of Genetic Fuzzy Systems for Intrusion Detection in Computer Networks”, **Expert Systems with Applications**, pp. 7067-7075.

7. Ngamwitthayanon, N. and Wattanapongsakorn, N., 2011, “Fuzzy-ART in Network Anomaly Detection with Feature-Reduction Dataset”, **Proceedings – 7th International Conference on Networked Computing, INC2011**, Art. No. 6058956, pp. 116-121.

8. Muda, Z., Yassin, W., Sulaiman, M.N. and Udzir, N.I., 2011, “Intrusion Detection Based on K-Means Clustering and Naïve Bayes Classification” **7th International Conference on Information Technology in Asia: Emerging Convergences and Singularity of Forms - Proceedings of CITA'11**, Art. No. 5999520.

9. Lee, S., Kim, G. and Kim, S., 2011, “Self-adaptive and Dynamic Clustering for Online Anomaly Detection”, **Expert Systems with Applications**, pp. 14891-14898.

10. Chandrasekhar, A. M. and Raghuveer, K., 2013, "Intrusion Detection Technique by Using K-means, Fuzzy Neural Network and SVM Classifiers", **International Conference on Computer Communication and Informatics, ICCCI 2013**.

11. Labib, K. and Vemuri, R., 2002, "NSOM: A Real-Time Network-Based Intrusion Detection System Using Self-Organizing Maps", **Networks and Security**.

12. Amini, M., Jalili, A. and Shahriari, H.R., 2005, "RT-UNNID: A Practical Solution to Real-Time Network-Based Intrusion Detection Using Unsupervised Neural Networks", **Computer & Security**, pp. 459-468.

13. Pukkawanna, S., Pongpaibool, P. and Visoottiviseth, V., 2008, "LD2: A System for Lightweight Detection of Denial-Of-Service Attacks", **In the Proceedings of Milcom**.

14. Su, M.-Y, 2009, "A Real-Time Network Intrusion Detection System for Large-Scale Attacks based on an Incremental Mining Approach", **Computers and Security**, pp. 301-309.

15. Komviriyavut, T., Sangkatsanee, P., Wattanapongsakorn, N. and Charnsripinyo, C., 2009, "Network Intrusion Detection and Classification with Decision Tree and Rule-based Approaches", **9th International Symposium on Communications and Information Technology**, Art. No. 5341005, pp. 1046-1050.

16. Kachurka, P. and Golovko, V., 2011, "Neural Network Approach to Real-Time Network Intrusion Detection and Recognition", **Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications**, IDAACS'2011 1, Art. No. 6072781, pp. 393-397.

17. Casas, P., Mazel, J. and Owezarski, P., 2012, "Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge", **Computer Communications**, pp. 772-783.

18. **Artificial Intelligence** [Online], Available: http://en.wikipedia.org/wiki/ Artificial_ intelligence [2013, April 18].

19. **Machine Learning** [Online], Available: http://en.wikipedia.org/wiki/Machine_ learning / [2013, April 18].

20. Marsland, S.P., 2009, **Machine Learning, 2nd ed.**, Massey Universities Palmerston North, pp. 95-122.

21. Adetunmbi A.O., Adeola S.O. and Daramola O.A., 2010, “Analysis of KDD ’99 Intrusion Detection Dataset for Selection of Relevance Features”, **Proceedings of the World Congress on Engineering and Computer Science 2010 Vol. I**.

22. **Fuzzy Logic** [Online], Available: http://alaska.reru.ac.th/text/fuzzylogic.pdf [2013, April 10]

23. Nguyen, H.T., Petrović, S. and Franke, K., 2010, “A comparison of feature-selection methods for intrusion detection”, **5th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security**, pp. 242-255

24. Christian Mèuller-Scholer, 2011, **Organic Computing -- a Paradigm Shift for Complex Systems**, p. 627.

CURRICULUM VITAE

NAME	Miss Pawita Jongsuebsuk
DATE OF BIRTH	23 July 1988
EDUCATIONAL RECORD	
HIGH SCHOOL	High School Graduation Sa-Nguan Ying School, 2007
BACHELOR’S DEGREE	Bachelor of Engineering (Computer Engineering) King Mongkut’s University of Technology Thonburi, 2011
MASTER’S DEGREE	Master of Engineering (Computer Engineering) King Mongkut’s University of Technology Thonburi, 2012

PUBLICATIONS

Wattanapongsakorn, N., Srakaew, S., Wonghirunsombat, E., Sribavonmongkol, C., Junhom, T., Jongsubsook, P. and Charnsripinyo, C., “A Practical Network-Based Intrusion Detection and Prevention System”, Trust, Security and Privacy in Computing and Communications (Trust Com), 2012 IEEE 11th International Conference, Liverpool, United Kingdom, 25-27 June 2012.

Jongsuebsuk, P., Wattanapongsakorn, N. and Charnsripinyo, C., “Network intrusion detection with Fuzzy Genetic Algorithm for unknown attacks”, Information Networking (ICOIN), 2013 International Conference, Bangkok, Thailand, 28-30 January 2013.

Jongsuebsuk, P., Wattanapongsakorn, N. and Charnsripinyo, C., “Real-Time Intrusion Detection with Fuzzy Genetic Algorithm”, International Conference on Electrical Engineering/Electronics, Computer,

Telecommunications and Information Technology, ECTI-CON 2013, Krabi, Thailand, 15-17 May 2013.