universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„A Fuzzy Logic-based Framework for Intrusion Classification in Corporate Network"

verfasst von / submitted by

Milos Avakumovic BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Diplom-Ingeneur (Dipl.-Ing.)

Wien, 2018 / Vienna 2018

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 926
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Wirtschaftsinformatik
Betreut von / Supervisor:	Univ.-Prof. Dipl.-Ing. Dr. Dr. Gerald Quirchmayr

ABSTRACT

As the number of hacking and intrusion attacks is increasing each year, Intrusion Detection Systems are becoming an extremely important component of the network security system. It is necessary to design system security mechanisms in a manner that identify unauthorized access to computer resources and data. Since complete prevention of unauthorized access is impossible, today's security systems aim to detect unauthorized intrusions and undertake a certain action before an unauthorized action causes damage.

No matter how effectively may Intrusion Detection Systems be identifying malicious activities, false alarms are a significant limitation nowadays, though. With an intention of making a step forward in overcoming this obstacle, the thesis proposes an Intrusion Detection System based on Fuzzy Logic that is able to provide a better classification rate in intrusion detection focusing on anomaly detection issues, i.e. the situations when a regular traffic is wrongly classified as an intrusion. Arriving packets are correctly treated as the system is firstly trained using a specific dataset and then verified by the predefined Fuzzy Logic Controller and Fuzzy Rules.

The Fuzzy framework establishment is based on the selection of most relevant input data which will contribute to higher precision of the classification rate. For this purpose, it is demonstrated how Fuzzy models can be used as an approach for intrusion classification and to improve the understanding and analysis of network input data.

Durch die stetig wachsende Zahl an Hacker Angriffen und unautorisierten Zugriffen entgegenzuwirken, kommen Segmente wie das Intrusion Detection Systems immer mehr zum Einsatz. Man ist gezwungen, eine sichere Methode zu wählen die für die Autorisierung sowie für den Schutz der Daten verantwortlich ist. Eine Volle Sicherheit vor unautorisierten Zugriffen gibt es leider nicht. Dadurch fokussieren sich Netzwerkspezialisten solche Vorfälle immer schneller aufzudecken und zu unterbinden.

Unabhängig davon wie effektiv die Sicherheitssysteme heute funktionieren, werden diese durch falsche Alarmsignale sabotiert (eingeschränkt). Um das Ganze zu vereinfachen und um dieses Hindernis zu überwinden, gibt das Intrusion Detection System auf der Basis von Fuzzy Logic Vorschläge. Dadurch ist eine bessere Klassifikationsrate bei der Erkennung von unautorisierten Zugriffen gegeben. Ankommende Datenpakete werden erstmal durchgeschleift, da das System zunächst anhand eines bestimmten Datensatzes „lernt“. Des weiteren wird ein vordefinierter Fuzzy Logic Controller mit den dazugehörigen Fuzzy-Regeln für die Verifizierung der Datenpakete genutzt.

Die Fuzzy-Framework basiert auf der Auswahl der relevantesten Eingabedaten, die zu einer höheren Genauigkeit der Klassifikationsrate beiträgt. Aus diesem Grund wird veranschaulicht, wie Fuzzy-Modelle funktionieren und uns helfen können, solche Angriffe zu klassifizieren und eingehende Datenpakete besser zu durchleuchten.

ACKNOWLEDGEMENTS

First I wish to express gratitude to my thesis advisor Univ.-Prof. Dipl.-Ing. DDr. Gerald Quirchmayr of the Faculty of Computer Science at University of Vienna. Prof. Quirchmayr was always open for any kind of question or consultation about my research or writing. He supported me during the whole process and steered me in the right direction whenever it was needed.

TABLE OF CONTENTS

Abstract 1

Acknowledgements 3

Table of Contents 4

List of Abbreviations 6

List of Figures 7

List of tables 8

Chapter 1 Introduction 9

1.1. Background and Motivation 9

1.2. Goals and Expected Outcomes 11

Chapter 2 Literature Review 13

Chapter 3 Intrusion Detection System: The Theoretical Background 14

3.1. Intrusion Detection 14

3.1.1. Network-based Intrusion Detection 15

3.1.2. Anomaly Detection 15

3.2. Fuzzy Logic 16

3.3. Existing Approaches for Intrusion Detection 17

3.3.1. Data Mining Techniques for Network Intrusion Detection 18

3.3.2. Artificial Neural Networks for Network Intrusion Detection 20

Chapter 4 Problem Description 22

Chapter 5 Framework Concept 24

5.1. Framework Design 24

5.2. Framework Architecture 25

5.3. Model Structure 26

5.3.1. Tools for Cleaning and Classification of Data 27
5.3.2. Fuzzy Logic Controller 27
Chapter 6 Prototype Implementation 30
6.1. Training Data 30
6.1.1. Data Attributes 30
6.1.2. Types of Attacks 32
6.2. Construction of the Model 33
6.2.1. Data Processing 34
6.2.2. Fuzzy Inference System 36
6.2.3. Evaluation of FIS 38
Chapter 7 Test and Evaluation 39
7.1. Setup and Validation of Fuzzy Inference System 39
7.2. NSL-KDD Dataset 43
7.3. Analysis of Two Specific Groups of Attacks 45
Conclusion 48
Bibliography 50
Appendix A Testing Documentation 53
Appendix B Source Code 60

LIST OF ABBREVIATIONS

ALDAPA	Algorithms, Data mining and Parallelism research group
ANN	Artificial Neural Network
ANSI	American National Standard Institute
DARPA	Defense Advanced Research Projects Agency
DM	Data Mining
DOS	Denial of Service
DR	Detection Rate
FAR	False Alarm Rate
FCM	Fuzzy C-Mean
FIS	Fuzzy Inference System
FN	False Negative
FP	False Positive
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
KDD	Knowledge Discovery and Data Mining
MIT	Massachusetts Institute of Technology
NSL	Network Socket Layer
R2L	Remote to Local (User)
SIGKDD	Special Interest Group on Knowledge Discovery and Data Mining
TN	True Negative
TP	True Positive
U2R	User to Root

LIST OF FIGURES

Figure 1 Annual number of recorded data breaches in the United States [2] 10
Figure 2 Survey conducted by Forrester's Business Technographics (sample of 818 participants in November 2003 and 639 participants in June 2004) 11
Figure 3 Conventional membership function 16
Figure 4 Continuous membership function 17
Figure 5 Real-time anomaly detection system 24
Figure 6 IDS multilayered architecture 25
Figure 7 Intrusion detection system 26
Figure 8 Fuzzy logic controller 28
Figure 9 Number of normal connections and attacks in the original KDD CUP '99 10% dataset, very small values are enlarged in figure right 34
Figure 10 Some results extracted from data printed by kddcup_analysis.m 34
Figure 11 Number of normal connections and attacks in the cleaned KDD CUP '99 10% dataset, very small values are enlarged in figure right 35
Figure 12 Classification of the KDD CUP '99 10% dataset (normal and four groups of attacks) 36
Figure 13 Number of detected normal connections with respect to number of clusters for the KDD CUP '99 10% dataset 41

LIST OF TABLES

Table 1 An overview of Data mining techniques 18
Table 2 Standard Metric and Confusion Matrix 22
Table 3 Basic features of individual TCP connections 31
Table 4 Content features within a connection suggested by domain knowledge 31
Table 5 Traffic features computed using a two-second time window 31
Table 6 Traffic features (Table 5) for destination host 32
Table 7 Attacks classified by four groups with definition from MIT Lincoln Laboratory 32
Table 8 Confusion matrix for cleaned and classified the KDD CUP '99 10% dataset when subclustering option is used 40
Table 9 Confusion matrix for cleaned and classified the KDD CUP '99 10% dataset when 16 clusters are selected 42
Table 10 Confusion matrix for cleaned and classified the KDD CUP '99 10% dataset when 4 clusters are selected and only two groups normal and attack 43
Table 11 Confusion matrix for the cleaned and classified NSL-KDD datasets (16 clusters) 44
Table 12 Confusion matrix for the cleaned and classified NSL-KDD datasets (16 clusters) – symbolic attributes included in calculations 44
Table 13 Comparison of specific types of attack between the original KDD Cup '99 and the 10% KDD Cup '99 datasets 45
Table 14 Confusion matrix for the cleaned and classified KDD CUP '99 10% dataset with 24 attributes are included in the calculation (attributes from Table 4 excluded) 46
Table 15 Frequency of occurrence for attributes belonging to the group in Table 5...47

CHAPTER 1
INTRODUCTION

Information technology development in the field of data collection, processing and distribution is additionally accelerated by the needs of modern business. Modern business is increasingly based on Internet, i.e. there exist various forms of electronic commerce. This way of doing business produces new risks to the security of information systems. Rapid development of information technology and the unstoppable growth of its application in all areas of human activity are increasing its vulnerability and exposure to potential hazards, especially because of the inevitable interdependence of the human factor and information system.

1.1. BACKGROUND AND MOTIVATION

Successful business operations of any organization are based on availability and proper functioning of all information system elements. However, new threats are posed every day by individuals and organizations that attack and abuse information systems. Since the information system supplies the necessary information to all other parts of an organization for its decision-making process, its security is of a great significance.

A massive flow of information between information systems is exposed to attacks by unauthorized users. Attackers access information systems, causing great damage to the overall operations of an organization. According to the reports by the U.S. Federal Agencies [1], the number of security incidents has been increasing over time. As presented in Figure 1, the trend is not consistent, but it is obvious that the number of network intrusions has increased sharply in the last decade. The increase in the number of breaches implies the greater number of data records exposed to attackers. [2]

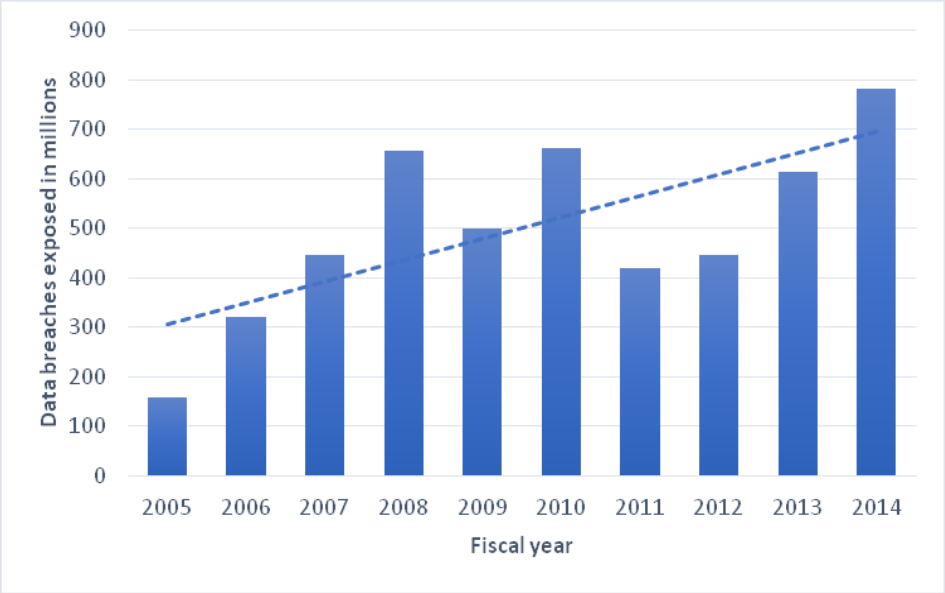

Figure 1 Annual number of recorded data breaches in the United States [2]

It is also important to emphasize the fact that the tools for carrying out attacks on information systems are becoming easier to use. According to reports from the Nato [3], extensive IT skills are not required anymore in order to be able to attack a system because of the constant growth of new and more sophisticated tools which can be used to carry out these attacks.

Because of this, organizations recognize a need for implementing information security management systems. This type of system reduces the possibility of an attack, either external or internal. Apart from that, by managing data security it also allows management to monitor and supervise all processes and reduce business risks to a minimum level. By using information security management system, a corporation is able to achieve information security in three main aspects: confidentiality, integrity and availability.

However, the main consideration of corporations used to be the costs of system implementation and maintenance. The cost of introducing information protection systems was mistakenly considered to be high in comparison to the cost incurred by security breaches. But an increase in the amount of attacks recovery costs has led to a growth of interest in the introduction of preventive and protective mechanisms. Organizations today realize that internal threats can be equally dangerous as external ones, or even more. According to a survey conducted by Forrester in November 2003 and June 2004, investments in strengthening the security of information systems increased. [4] An overview is presented in Figure 2.

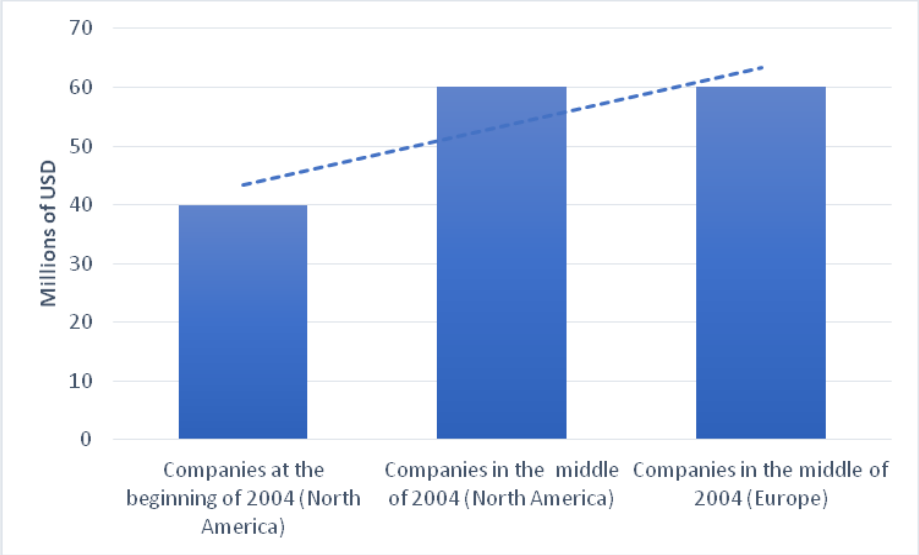

Figure 2 Survey conducted by Forrester's Business Technographics (sample of 818 participants in November 2003 and 639 participants in June 2004)

These pivotal background factors (rise in the number of attacks and growth of investment in security systems) represent an incentive for further research of the subject**.** Having in mind that no system can be perfectly secured, there is a need for constant improvement of protection mechanisms. Hence, the motive behind this thesis is to contribute to current knowledge in this field by researching the possibilities of usage of Fuzzy Logic technique for the purpose of intrusion detection.

1.2. GOALS AND EXPECTED OUTCOMES

The main goal of this thesis is to produce a theoretical framework of the Intrusion Detection System (IDS) which is based on Fuzzy Logic.

Additionally, the practical part of the thesis is focused on the improvement of the successfully classified network intrusions rate. For this purpose, four types of remotely launched attacks will be used: Denial of Service, User to Toot, Remote to User and Probe.

The primary focus of the practical part is to present the techniques for more effective analysis of network input data and to identify which input attributes are the most relevant for the Fuzzy rules generation process. As a result of this analysis, the rules are going to be more specified and improved.

The expected outcome of the research is to improve the classification rate for all types of attacks by generating a set of more reliable Fuzzy rules. These Fuzzy rules are

obtained by processing input data and selecting the most relevant attributes from the given base of inputs.

CHAPTER 2
LITERATURE REVIEW

The theoretical part of the thesis was written by consulting all the relevant sources such as books, scientific papers and web sources. In addition to that, the conclusions and main points made by experts at various workshops, conferences and seminars are also part of the used literature. The results of various pieces of research were also taken into account when analyzing the theoretical background of this subject. The most relevant sources were also used to present the existing approaches for improvement of successful classification rate.

The research and construction of the framework concept are based on the usage of the KDD Cup ’99 dataset as the input for further experimental analysis. The Fuzzy C-Mean algorithm is used as a main data mining technique to improve the successful classification rate of all types of attacks.

The main research based on the implementation of Fuzzy Logic used in the theoretical background as well as in the construction of the proposed framework was conducted and presented by the authors Shanmugavadivu and Nagarajan. [5]

CHAPTER 3
INTRUSION DETECTION SYSTEM: THE THEORETICAL BACKGROUND

Considering the complexity of any information security management system, it can be easily concluded that a single line of defense is not sufficient. That is why organizations use "defense in depth" – a layered protection mechanism for the critical components of the information system. It does not rely on a single security control but combines complementary security mechanisms, strengthening the security of information systems.

To withstand attacks, information security management system applies "defense in depth" by ensuring the following:

- Defense in more places - set up protective mechanisms at multiple locations to protect the information system against internal and external attacks
- Layered defense - set up multiple protective mechanisms so that an attacker must go through several layers to get to critical information
- Intrusion Prevention System (IPS) - set up a system for prevention of intrusions into an information system
- Intrusion Detections System - set up a system for detection of intrusions into information system

Therefore, IDS is considered to be the essential part of the successful maintenance of information system security.

3.1. INTRUSION DETECTION

An Intrusion detection system represents a part of technologies used to raise the overall level of security of information system. It gathers information from defined input and analyzes it to detect illegal activities and abuses of the system in which it is located. Main operations are based on monitoring a specific part of a system, and analysis of headers and content of packets and data at various layers of the system stack to identify unusual activities and attacks.

RFC 2828 defines intrusion detection as “a security service that monitors and analyzes system events for finding and providing real-time or near real-time warning of attempts to access system resources in an unauthorized manner”. [6]

The American National Standard Institute (ANSI) defines intrusion detection as “a process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network”. [7]

3.1.1. Network-based intrusion detection

There are different approaches that can be used as the basis for division of IDSs. The most common one is based on the information source. In other words, the part of the system that is monitored by the IDS is used as the information source. Apart from Host-based IDSs that supervise the work of a host, the most important are Network-based IDSs that supervise the processes within a network.

A Network-based IDS, as its name suggests, monitors the entire network or its segment, depending on its position in the network topology. The main operation is based on capturing and analyzing packets that traverse the network.

A Network-based IDS is often composed of a series of simple sensors, located in different points inside the network. Sensors monitor and analyze network traffic locally, and then report detected attacks to the central management console. Many sensors are designed for "deceptive" mode, so that an attacker would not be able to detect their presence and location.

The main advantage of a network-based IDS is that several well-distributed network-based IDSs can monitor a large network. The implementation of a network-based IDS has small impact on the existing network. In fact, it is usually a passive device that scans the network traffic without affecting normal operations in the network.

3.1.2. Anomaly detection

An Anomaly detection system discovers computer or network intrusions. This is a process where tracked activities are compared to the ones considered as expected behavior patterns. IDS technologies based on anomaly detection have profiles that

represent normal behavior. These profiles are usually produced by using audit records that are already generated by the system. The main benefit of this approach is that unknown threats can be discovered very effectively. [8] For example, if a computer becomes infected by a new malicious program which consumes a lot of resources, sends large number of e-mails, initiates many network connections or another manner of behavior that is different from the already established profile for the computer, it is then clear that such behavior is not compatible with the usual one. The malicious program would be detected due to a significant deviation from the previously established profile and behavior.

As said, a generated default profile can be static or dynamic. Once generated, the static profile cannot be changed, unless an IDS is triggered to generate a new profile. A dynamic profile adjusts itself as new events are observed, so it learns and adapts constantly. As the systems and networks are changing during time, the proper behavior is changing too, so a static profile will eventually become inaccurate, which implies that it should be periodically generated. Dynamic profiles do not have this problem.

3.2. FUZZY LOGIC

There are situations where it is not possible to represent a knowledge of system in a precise manner. In other words, sometimes it is not enough to rely on Binary logic where something is either black or white. To overcome the limit of classical Binary logic, Fuzzy Logic can be used to widen the range of options (all shades of gray).

To compare Binary and Fuzzy logic, a typical example is reviewed. It is a process of determination of belonging to a set of tall people. Conventional boundaries are strictly determined (Figure 3), so two people are classified differently even though their height varies with just a few centimeters. [9]

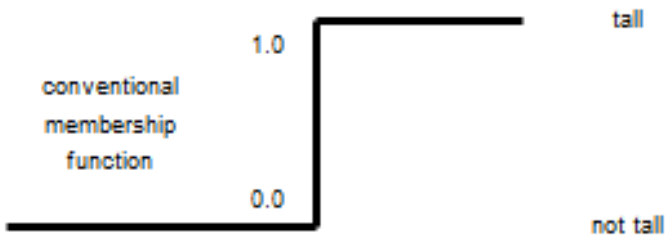

Figure 3 Conventional membership function

The approach above would make sense in a case of an abstract representation, such as numbers. It could be said that all numbers greater than a specific number are in general "larger" (than it number) and that smaller numbers in comparison with the specific number are in general "smaller" (than it number). However, when something is conditioned by age and social characteristics, such as estimation whether a person is high or not, setting such a sharp boundary does not make sense. That is why a continuous membership function is introduced to determine whether and how tall the person is (Figure 4).

Figure 4 Continuous membership function

Continuous membership function gives an opportunity to consider to whom it applies (children, female persons or to all adults, etc.). The only requirement is that membership function needs to be scaled between 0 and 1, which defines membership level of a variable to the function. [10]

3.3. EXISTING APPROACHES FOR INTRUSION DETECTION

The development of the Intrusion System has gone through different phases. In 1972, James P. Anderson pointed out the gravity of computer security issues. The main problem, still present nowadays, was segmenting the network into domains, providing unobstructed information flow between them, but with keeping the integrity and security of every domain. [11]

During the 1980s the same author was working on improving security auditing and surveillance. He takes credit for the original idea behind automated intrusion detection. This postulate represents the core of misuse detection. With the analysis of audit data, the first attack patterns were made, and they were used in the process of intrusion detection. [12]

The first model of real-time intrusion detection was developed between 1984 and 1986 by Dorothy Denning and Peter Neumann. The aim was to detect various types of

security violations. The idea behind this model was to track regular activities in the system and identify malicious activities. It focused on the basic system activities without having information about system security shortcomings. [13]

In the mid-1990s, the US Army was developing a prototype which was commercialized during the year 1995. This model was working real-time, using misused detection as engine. In 1997 RealSecure tool was released for commercial use and was running on Windows platform. [14]

Until the 2000s firewalls were used mostly because of their capability of processing traffic more quickly since they did not do deep packet inspection. But, at the beginning of the 2000s, new types of attacks able to pass the firewall started to appear, which made IDS main security mechanism. [15] Some organizations still used IPS which is positioned between home network and the internet. It functions in a way that drops each packet which it recognizes as an attack. Every packet needs to be checked and compared with signature entries in a database, which is constantly growing due to novel attacks. A problem that occurs is a large number of dropped packets that are in fact not malicious. Also, a large signature database is used to hinder IPS performance. Due to these problems, organizations turned to IDS, which is located aside. When a threat is detected, IDS does not drop the packet, but alerts the organization so that the management could decide how to proceed. Later, popularity of IPS began to grow again, after signature database was optimized and only most relevant signatures were used. [16]

3.3.1. Data Mining Techniques for Network Intrusion Detection

One of recent algorithms which is applied for patterns discovery from big data for intrusion detection is Data mining (DM). DM extracts knowledge from data. [17] It establishes a relationship within data samples which enables it to detect anomalous patterns. There are numerous DM techniques. An overview is presented in Table 1.

Table 1 An overview of Data mining techniques

Data Mining technique	Characteristic
Feature selection data analysis	"Discard all data attributes that have insufficient level of predictive information or

	do not have it all to create a group of suitable attributes" [18]
Classification analysis	"Assign attacks to classes according to values of attack's data attributes; could be used for anomaly and misuse; in misuse, training data is used to learn classifiers of different types used for detection of known intrusions; in anomaly, training data is used to establish normal behavior pattern; classification could be used for learning and detection of intrusions" [19]
Clustering analysis	"Assign attacks to clusters based on distance measurements made on attacks; unsupervised learning process; similarity measure represents an important factor in grouping observations" [19]
Association and correlation analysis	"Discover association relationships between specific attributes in dataset" [19]
Stream data analysis	"Attacks are dynamic by nature, so it perceives data streams as a whole since a record might be normal on its own but malicious if viewed as part of sequence" [19]
Distributed data mining	"Attacks could be performed from different locations and target different destinations, so it analyses data from several network locations" [19]
Visualization and querying tools	"Graphical user interface enables users to view classes, associations, clusters, etc." [19]

Currently, clustering is the most used data mining technique for intrusion detection. Various researchers have proposed many different clustering techniques so far. In the group of many algorithms, Fuzzy C-Mean is considered very efficient. [20], [21], [5],

[22] Some of notable frameworks are constructed by using other algorithms such as Classification and Regression Tree [23] and Genetic algorithm combined with Fuzzy Logic. [24], [25]

In spite of improvements which can be made by implementation of these data mining techniques, there are still some downsides that should be taken into consideration. The main weakness of data mining approaches refers to data correlation. When the system has not collected sufficient audit trail data, it cannot reach full potential. Another drawback is that correlation between entities does not imply causation. So, it can be possible to have hundreds of data correlated, with only a few of them that are worthwhile.

3.3.2. Artificial Neural Networks for Network Intrusion Detection

Artificial Neural Network (ANN) is also a very intensively researched approach. The concept of Intrusion detection using neural networks is based on the fact that a user leaves a print when using a system. So, neural network is used to identify the print and the users based on their specific behavior patterns.

ANN represents a collection of artificial neurons which are connected and interactive throughout operations of processing the signals. It is modeled like a human brain. Neurons and connections have a weight that adjusts as learning process proceeds. The weight represents the strength of the signal at a connection. The connections between neurons are activated if the condition set by the so-called activation function has been fulfilled. [26]

Some of efficient applications of the ANN are also done by using different algorithms such as Multilayer Perception, Radial Base Function, Logistic Regression, Voted Perception [27], Radial Basis Functions [28] and Multy Layer Back Propagation [29]. There is also a tendency to compare outputs generated by different algorithms such as Feed Forward Neural Network, Probabilistic Neural Network and Radial Basis Neural Network classifiers. [30]

The neural network approach can accomplish an excellent job in structuring a profile of user behavior that is adaptable over time. However, the potential drawback of ANN

might be the scalability of neural network systems. That problem might appear in a situation when the number of users exceeds the size of small or medium enterprises.

Intrusion Detection System is an essential part of “defense in depth” architecture. When malicious behavior is noticed, an alarm is raised allowing administrators to react according to the security policy. Because of that, the main objective of such a system is to treat the input data properly.

CHAPTER 4
PROBLEM DESCRIPTION

To achieve good performance predictions, IDS must meet two criteria:

- It must be able to accurately identify an intrusion
- It must not identify a regular action in the network environment as an intrusion

Assessing IDS performance prediction includes Detection Rate (DR) and False Alarm Rate (FAR). DR is defined as the ratio of the number of correctly detected attacks and the total number of attacks, while FAR is defined as the ratio of the number of normal connections that are incorrectly classified as attacks and the total number of normal connections. [31] The data on which the DR and FAR are determined can be presented via the confusion matrix. It consists of the following elements: True Negative (TN), False Positive (FP), False Negative (FN) and True Positive (TP), where:

- TN - correctly indicates connections that are normal (regular) traffic
- FP - indicates normal connections that are wrongly classified as non-regular (intrusions)
- FN - indicates non-regular connections (intrusions) that are wrongly classified as regular
- TP - correctly indicates connection that are non-regular (intrusions)

Table 2 Standard Metric and Confusion Matrix

STANDARD METRIC		**IDS OUTPUT**	
		NORMAL	*INTRUSION*
ACTUAL STATUS OF TRAFFIC RECORD	*NORMAL*	TN	FP
	INSTRUSION	FN	TP

Among all issues that might appear, FP alarms are the most common problem someone must deal with when implementing an IDS. Almost every rule can cause an

FP alarm. The main issue is that FP alarms can undermine valid IDS alerts. It is possible to have one IDS sensor that generates thousands of alerts caused by a single rule. Additionally, reviewing large volume of alerts and logic can be overwhelming and time consuming to an analyst. As an assumption, if there are around 100 alarms on daily basis, an analyst has a few minutes to review each of them. Due to this approach it is a common situation that alerts causing repetitive FP alarms are overlooked or ignored, so that a company is not able to realize and examine the actual problem.

CHAPTER 5
FRAMEWORK CONCEPT

As it is mentioned in the Chapter 1, one of the goals of this research is to provide theoretical framework of corporate network-based Intrusion Detection System which uses Fuzzy Logic as its engine.

5.1. FRAMEWORK DESIGN

The anomaly or outlier detection technique, in case of intrusion detection, identifies anomalous user behavior which does not match the expected behavior patterns. This technique ensures that the process will take place in real time. All input data should pass directly through anomaly detection system, undergo an assessment, and be isolated and checked again if there is any doubt of intrusion (Figure 5).

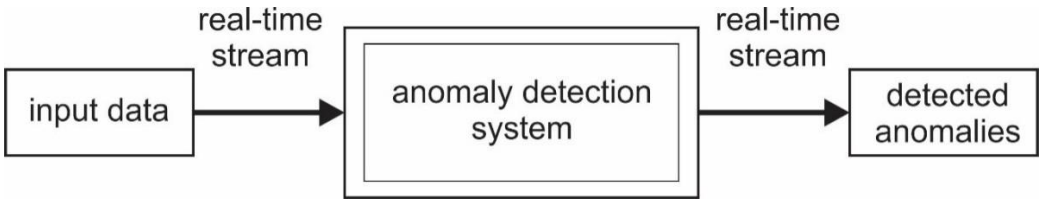

Figure 5 Real-time anomaly detection system

The amount of data passing through the system is usually large. Data is very different in nature. Construction of anomaly-based intrusion detection system requires a model of normal and anomalous behavior. Fortunately for a corporate network, but delicate from the mathematical point of view, the system is dealing with a large number of normal data (up to 99%). Anomalous data represents low percentage of data. Because of such uneven distribution, it is difficult to get a system that correctly detects the difference between normal and anomalous. It is even more difficult to apply this system in real-time.

Before moving on to the construction of the system, it is necessary to provide a representative data sample that consists of different types of attacks, as well as normal packets. This data will be used for the training of the system. It is important to treat

the data properly and prepare it for further processing. All irregularities of data and unnecessarily data should be filtered out.

The next step is the selection of relevant attributes that will be applied for intrusion detection. Fuzzy logic is applied as anomaly-based intrusion detection data mining technique.

As a final step, it is necessary to choose the environment in which the intrusion detection code will be developed. The code should be as universal as possible, based on reliability and scalability. Scalability of the solution is of great importance since both the corporate network that should be secured and the attackers' techniques have been constantly changing.

5.2. FRAMEWORK ARCHITECTURE

The proposed intrusion detection system represents a multilayered security mechanism based on very simple but powerful Pipe-And-Filter architecture. [32] This way of implementing the system enables more efficient and sophisticated analysis, since the data could be tracked after every iteration during the processing. The system architecture consists of three layers: Sensor, Detection and Reaction (Figure 6). [33]

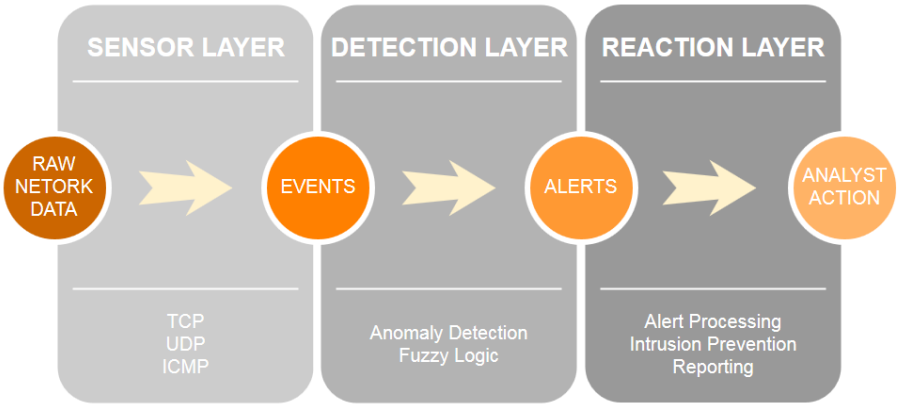

Figure 6 IDS multilayered architecture

The Sensor layer represents the interface to network elements. Raw data is collected by various agents that acquire different types of packets based on protocol types used on Network (ICMP) and Transport layer (TCP, UDP). Agents filter out unnecessary input data and trigger initial data processing that provides valuable information needed to construct an event. The event consists of predefined attributes.

The Detection layer is the core architecture element. Data processing is done here in order to prepare the data for Fuzzy Logic part of the layer. Additionally, classifiers assess at this layer the events passed from the Sensor layer and check if malicious behavior exists (anomaly detection). Fuzzy logic is the main data mining technique used for the process of determination whether a packet is corrupted or not. In case of an attack, an alert is generated and forwarded to the Reaction layer.

It is the Reaction layer where final processing is done. The alerts are aggregated according to the type of an attack that they belong to. The final output is reflected in security analyst action made according to the provided information. Learned signatures could be added to the IPS engine database. An important characteristic of this layer is its possibility of reporting, which could be useful for forecasting and generation of custom reports for the management.

5.3. MODEL STRUCTURE

Intrusion detection system needs appropriate data for the system training and testing. Agents in the Sensor layer are in charge of this step. Due to an inability to simulate real-traffic scenarios, certain datasets will be used. They consist of already generated events. The Detection layer consists of the tools for cleaning and classification of training/testing data and the tools for intrusion detection. These tools are the engine of the system. The final output, an alert, is forwarded to the Reaction layer. Based on the implemented logic or simply an analyst's estimation, it is decided how an alert will be treated.

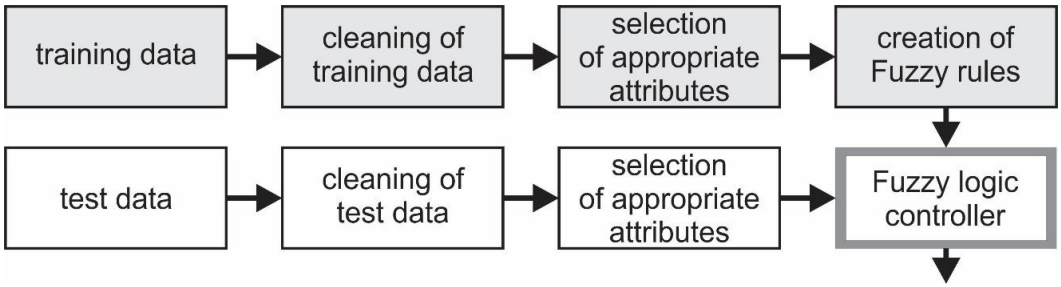

Figure 7 Intrusion detection system

The steps which take place in the system are illustrated in Figure 7. The system consists of two flows. One flow is related to the training data and the other to the testing data. However, it is obvious that the training and testing data pass through the same processing mechanisms and that the tools for data training may also be used for processing of the testing data.

5.3.1. Tools for cleaning and classification of data

In the absence of streaming data from a corporate network, a database is selected as a training and testing dataset. The database is used for the design of data processing tools: a tool for data analyzing, for duplicates cleaning and for incomplete data cleaning. In case of training data, this type of processing is carried out using some existing software. Since the same tools are applied for testing and for real-time intrusion detection, they are developed and included in the framework.

The second step is attribute analysis. The attributes have symbolic as well as numerical values. Symbolic attributes can be used in anomaly detection system, built on Fuzzy logic, only if the symbolic values are replaced by numerical ones. The tool for analysis of symbolic attributes is constructed in order to collect as much information as possible and to get familiarized with common trends in data.

The scalability of the system also represents an important part of implementation. This characteristic applies especially to the tools intended for numerical attributes. The tool for classification and selection of this type of attributes is built in such a way that it can be easily adapted and modified according to additional needs or requirements.

In order to use Fuzzy C-Mean clustering, the method that is selected to be the main data mining technique as a part of intrusion detection system, data must be converted into an appropriate format. The first series of formatting is performed with the previously mentioned tools. Final formatting, or the normalization of data, is performed on the data to which clustering is applied.

5.3.2. Fuzzy logic controller

Clustering method is used as data mining technique and embedded into the intrusion detection system. The clustering methods are extensively studied since they can perform successful natural grouping of data from large databases into meaningful subgroups called clusters. Fuzzy C-mean clustering enables each data point to belong to several clusters and the degree of membership can be defined and controlled by a certain parameter. The number of clusters can also be defined, or other techniques can be applied for determination of cluster number.

The heart of intrusion detection system, Fuzzy logic controller, is illustrated in Figure 8. The all Fuzzy processing elements are also presented through the controller illustration.

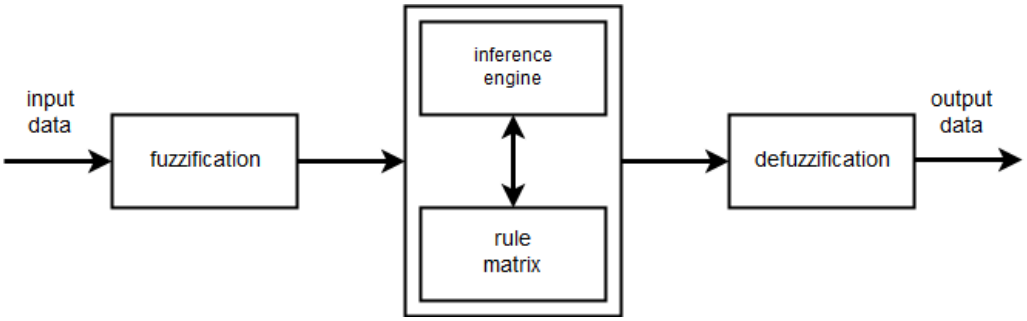

Figure 8 Fuzzy logic controller

The input data are non-fuzzy numbers. At the beginning, data need to be fuzzified. This means that a degree to which they belong to certain Fuzzy set needs to be determined. This part of procedure is performed using Fuzzy C-Mean clustering. Membership functions and a rule matrix are also derived from the results obtained using Fuzzy C-Mean clustering. In the end, a Fuzzy signal is transformed back to a non-fuzzy data.

5.3.2.1. Fuzzy C-Mean clustering algorithm

The Fuzzy C-Mean clustering [34], [7] is based on the minimization of the following objective function. The Matlab help and documentation website were used as a resource of algorithm explanation. [35]

$$J_m = \sum_{i=1}^{D}\sum_{j=1}^{N} \mu_{ij}^{m} \left\| x_i - c_j \right\|^2$$

"D is the number of data points, N is the number of clusters, x_i is i-th data point, c_j is the center of j-th cluster. The Fuzzy partition matrix exponent m controls the degree of fuzzy overlap, the number of data points that have significant membership in more than one cluster. The value of the exponent, marked m, is grater than 1. The degree of membership of x_i in j-th cluster is given by coefficient μ_{ij}. The sum of μ_{ij} values for a data point is one." [35]

The Fuzzy C-mean clustering is based on the following algorithm defined in [35]:

1. “Initialization of the cluster membership coefficient μ_{ij}

2. Calculation of cluster center

$$c_j = \frac{\sum_{i=1}^{D} \mu_{ij}^{m} x_i}{\sum_{i=1}^{D} \mu_{ij}^{m}}$$

3. Applying following formula to update coefficient μ_{ij}

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{N} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}}$$

4. Calculating objective function J_m

5. Repeating steps 1-4 until solution converge or maximum number of iterations is reached” [35]

CHAPTER 6
PROTOTYPE IMPLEMENTATION

The prototype implementation process consists of the following components: selection of appropriate training data, selection of an appropriate environment for the code development and development of the testing system.

6.1. TRAINING DATA

The dataset from the KDD CUP 1999 contest is selected as the training data. This dataset was uploaded from the community for data mining, data science and analytics (SIGKDD) website. [36] The contest data represents a version of data prepared by Massachusetts Institute of Technology (MIT) Lincoln Labs for the 1998 DARPA Intrusion Detection Evaluation Program. The complete explanation of DARPA can be found at the website of MIT Lincoln Lab. [37]

6.1.1. Data attributes

The dataset consists of entries recorded during seven weeks of real network traffic. It has almost half a billion records. There are 41 attributes plus an attribute determining whether connection is normal or an attack. One row of data from the KDD CUP ‘99 dataset is presented below

Table 3:	0, tcp, http, SF, 181, 5450, 0, 0, 0,
Table 4:	0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
Table 5:	8, 8, 0, 0, 0, 0, 1, 0, 0,
Table 6:	9, 9, 1, 0, 0.11, 0, 0, 0, 0, 0,
Connection:	normal

The lists of the attributes of the KDD Cup ‘99 contest data, as well as the description and type, are presented in Tables 3, 4, 5 and 6. As data source [36] was used.

Table 3 Basic features of individual TCP connections

	Attribute name	Description	Type
1	duration	length (number of seconds) of the connection	continuous
2	protocol_type	type of the protocol, e.g. tcp, udp, etc.	symbolic
3	service	network service on the destination, e.g., http, telnet, etc.	symbolic
4	flag	normal or error status of the connection	symbolic
5	src_bytes	number of data bytes from source to destination	continuous
6	dst_bytes	number of data bytes from destination to source	continuous
7	land	1 if connection is from/to the same host/port 0 otherwise	symbolic
8	wrong_fragment	number of "wrong" fragments	continuous
9	urgent	number of urgent packets	continuous

Table 4 Content features within a connection suggested by domain knowledge

	Attribute name	Description	Type
10	hot	number of "hot" indicators	continuous
11	num_failed_logins	number of failed login attempts	continuous
12	logged_in	1 if successfully logged in 0 otherwise	symbolic
13	num_compromised	number of "compromised" conditions	continuous
14	root_shell	1 if root shell is obtained 0 otherwise	symbolic
15	su_attempted	1 if "su root" command attempted 0 otherwise	symbolic
16	num_root	number of "root" accesses	continuous
17	num_file_creations	number of file creation operations	continuous
18	num_shells	number of shell prompts	continuous
19	num_access_files	number of operations on access control files	continuous
20	num_outbound_cmds	number of outbound commands in an ftp session	continuous
21	is_host_login	1 if the login belongs to the "host" list 0 otherwise	symbolic
22	is_guest_login	1 if the login is “guest login” 0 otherwise	symbolic

Table 5 Traffic features computed using a two-second time window

	Attribute name	Description	Type
23	count	number of connections to the same host as the current connection in the past two seconds	continuous

24	serror_rate	% of connections that have "SYN" errors	continuous
25	rerror_rate	% of connections that have "REJ" errors	continuous
26	same_srv_rate	% of connections to the same service	continuous
27	diff_srv_rate	% of connections to different services	continuous
28	srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
29	srv_serror_rate	% of connections that have "SYN" errors	continuous
30	srv_rerror_rate	% of connections that have "REJ" errors	continuous
31	srv_diff_host_rate	% of connections to different hosts	continuous

Table 6 Traffic features (Table 5) for destination host

	Attribute name	**Desription**	**Type**
32	dst_host_count	count for destination host	continuous
33	dst_host_srv_count	srv_count for destination host	continuous
34	dst_host_same_srv_rate	same_srv_rate for destination host	continuous
35	dst_host_diff_srv_rate	diff_srv_rate for destination host	continuous
36	dst_host_same_src_port_rate	same_src_port_rate for destination host	continuous
37	dst_host_srv_diff_host_rate	diff_host_rate for destination host	continuous
38	dst_host_serror_rate	serror_rate for destination host	continuous
39	dst_host_srv_serror_rate	srv_serror_rate for destination host	continuous
40	dst_host_rerror_rate	rerror_rate for destination host	continuous
41	dst_host_srv_rerror_rate	srv_serror_rate for destination host	continuous

6.1.2. Types of attacks

The data set consists of 22 types of attacks. All 22 types of attacks are classified into 4 major groups: Denial of service (DoS), Probing, Remote to local (R2L) and User to root (U2R). [37] Short definitions of the groups are given in Table 7.

Table 7 Attacks classified by four groups with definition from MIT Lincoln Laboratory

Group	**Attacks**	**Definitions**
DoS	back, land, neptune, pod, smurf, teardrop	"Attacker makes some computing or memory resource too busy or too full to handle legitimate requests, or denies legitimate users access to a machine"

		[38]
Probe	ipsweep, nmap, portsweep, satan	"Programs that can automatically scan a network of computers to gather information or find known vulnerabilities" [38]
R2L	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster	"Attacker who has the ability to send packets to a machine over a network but who does not have an account on that machine and exploits some vulnerability to gain local access as a user of that machine" [38]
U2R	buffer_overflow, loadmodule, perl, rootkit	"Attacker starts out with access to a normal user account on the system (perhaps gained by sniffing passwords, a dictionary attack, or social engineering) and is able to exploit some vulnerability to gain root access to the system" [38]

Since the original KDD CUP '99 dataset contains around 5 million samples, the operation with this amount of data is far beyond the capabilities of an average desktop computer. Instead of that, the KDD CUP '99 10% dataset with 494021 samples is used in further calculations.

There are also datasets derived from the KDD CUP '99 in which it is attempted to eliminate problems identified in the original dataset like a great number of duplicates or incomplete data. The Dataset used in parallel to the KDD CUP '99 data is the NSL-KDD dataset (25192 samples) [39] from the Canadian Institute for Cybersecurity, University of New Brunswick.

6.2. CONSTRUCTION OF THE MODEL

Intrusion detection system is developed in the Matlab using Fuzzy logic toolbox. The entire code is given in Appendix B. It is separated into two main parts. The first part represents the code for analysis and cleaning of data and selection of relevant attributes. This part also consists of a code used for the construction of the Fuzzy

Inference System (FIS) to which training data is pushed. The second part is a code used for testing, i.e. for the evaluation of FIS with different testing data.

6.2.1. Data processing

Data analysis is currently performed only for symbolic attributes. This type of analysis is introduced in the code to get familiarized with data trends. The distributions of values for all symbolic attributes, that is the number of normal and attack connections in the original uncleaned dataset, are illustrated in Figure 9.

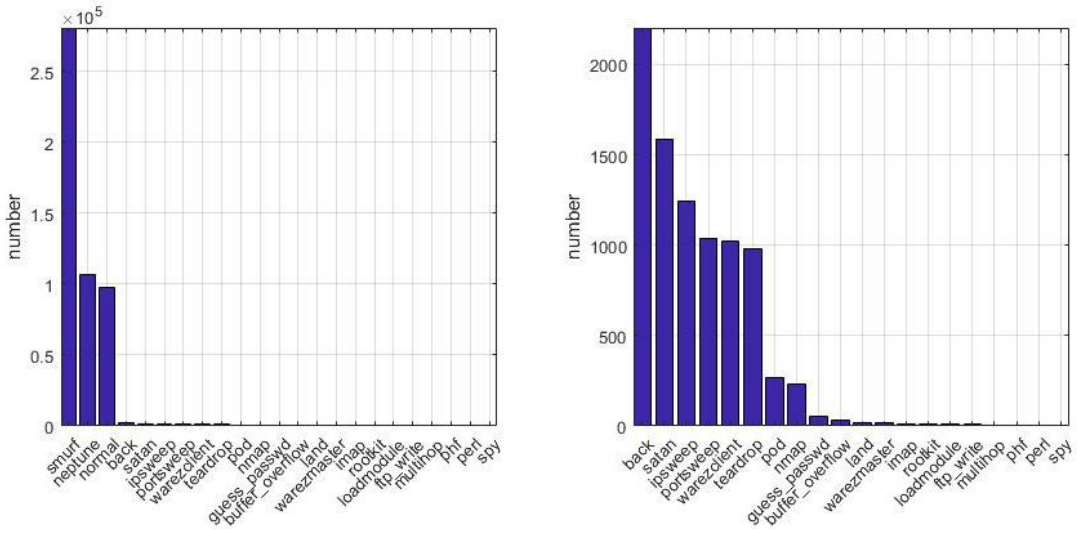

Figure 9 Number of normal connections and attacks in the original KDD CUP '99 10% dataset, very small values are enlarged in figure right

This analysis is not applied to the final selection and data cleaning, but it provides some useful information. It could be noticed that only 22 connections have defined attribute "land" (value one) while other 493999 have value zero. Additionally, the attribute "is_host_login" has value equal to zero in every record, which means that none of the connections in 10% database is from the host, so this parameter can be ignored in further analysis. The results obtained while using the KDD CUP '99 10% dataset for attributes "land" and "is_host_login" are presented in Figure 10. The other data printed by kddcup_analysis.m is given in Appendix A.

land - 1 if connection is from/to the same host/port 0 otherwise

0 493999
1 22

1 if the login belongs to the "host" list 0 otherwise

0 494021

Figure 10 Some results extracted from data printed by kddcup_analysis.m

The part of the model related to processing of data also includes a tool for the dataset cleaning. Importance of duplicates cleaning is obvious when comparing the output in Figure 9, where the number of normal/attack connections for the KDD CUP ’99 10% data is illustrated before cleaning of duplicates, and output in Figure 11, where the connections are presented after the cleaning is done.

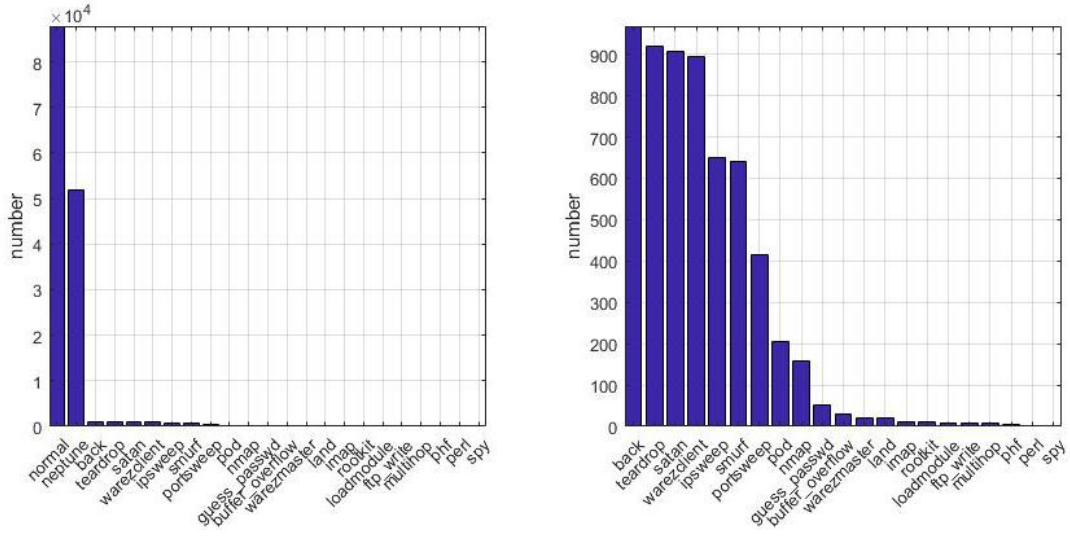

Figure 11 Number of normal connections and attacks in the cleaned KDD CUP ’99 10% dataset, very small values are enlarged in figure right

In the original KDD CUP ’99 10% dataset the number of “smurf” attack records is much larger than number of normal connections, “neptune” attack records, or any other attack group. After cleaning of duplicates, however, the normal connections are the most present group of data. This subject is discussed in detail in Chapter 7.

The classification of data is also performed in this part of the code. Data is classified into the group of normal connections or four major attack groups (Figure 12). The classification is necessary for clustering that takes place in the Fuzzy logic controller.

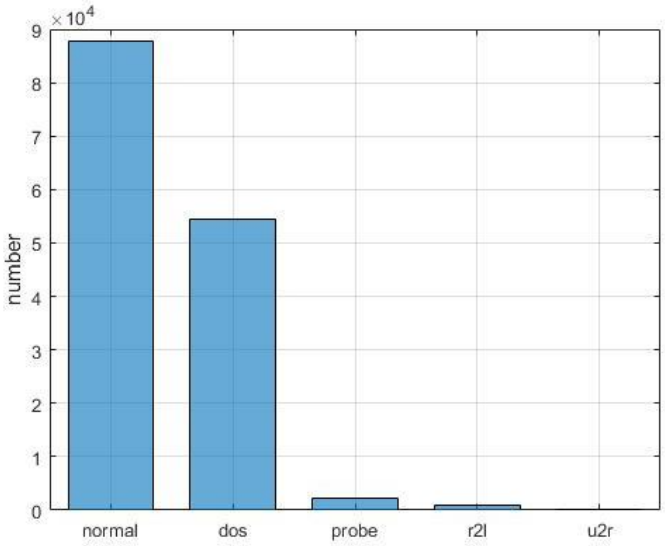

Figure 12 Classification of the KDD CUP '99 10% dataset (normal and four groups of attacks)

Cleaned and classified data is forwarded to the FIS since the efficiency of the FIS depends on the form of the data.

6.2.2. Fuzzy inference system

6.2.2.1. Data normalization

Data is normalized before applying clustering method. This process represents the fuzzification. Normalization is performed using Matlab function "mapstd". Function "mapstd" processes matrices by mapping each row's means to 0 and deviations to 1. [7] Syntax of the function is as follows

```
[Y,PS] = mapstd(X,ymean,ystd)
```

"where X is the matrices, ymean and ystd are optional parameters, Y is resulting matrices and PS is carrying process settings that allow consistent processing of values. Function "mapstd" has option "reverse" which is used afterwards to convert output data back to original units, which is known as defuzzification." [35]

6.2.2.2. Generation of FIS - genfis

Fuzzy C-Mean clustering is used as an option of Matlab Fuzzy Logic Toolbox function "genfis" – generate FIS. The function "genfis" generates Fuzzy Inference System from data. [7]

Synatx of "genfis" function is as follows

```
fis = genfis(inputData,outputData,options)
```

Training data is provided to “genfis” function as an input. Options field is used to select Fuzzy C-Mean (FCM) clustering (option “FCMClustering”) as a method to generate Fuzzy System. In case of FCM clustering, each input variable has one 'gaussmf' input membership function for each Fuzzy cluster. One rule is generated for each fuzzy cluster. Finally, each output variable has one output membership function for each fuzzy cluster. Options field is also defined by selecting Mamdani over Sugeno system. These two systems are most commonly used Direct Fuzzy inference methods. The difference between them lies in a way how an output is acquired. The membership function type of output variable is 'gaussmf' for Mamdani system. To generate the output of FIS the following steps are applied:

1.”determining a set of Fuzzy rules,
2. fuzzifying the inputs using the input membership functions,
3. combining the fuzzified inputs according to the Fuzzy rules to establish a rule strength,
4. finding the consequence of the rule by combining the rule strength and the output membership function,
5. combining the consequences to get an output distribution, and
6. defuzzifying the output distribution”. [40]

When defining options, it is possible to select a number of clusters for FCM clustering. If it is not defined, “genfis” estimates the number of clusters using subtractive clustering method.

6.2.2.3. Fuzzy logic designer

The Fuzzy logic designer can be used to design FIS. An example is given in Appendix B. Also, the designer can be used to see result of FIS generated by “genfis” and to make modifications, if necessary. In the model, the FIS is saved to a file that will be used afterwards for testing, but the inspection of FIS is enabled by starting the Fuzzy logic designer.

The syntax to start the designer is as follows

```
fuzzyLogicDesigner(fuzzySys)
```

6.2.3. Evaluation of FIS

Matlab function “evalfis” performs evaluation of the Fuzzy inference system and calculates the results by using input data that needs to be tested and the FIS constructed by “genfis”. The function is used in kddcup_fis.m code to evaluate training results and in kddcup_test.m code to calculate results for testing dataset. Two codes are given in Appendix B.

The syntax for “evalfis” is as follows

```
output= evalfis(input,fismat)
```

where `input` is data that needs to be evaluated using Fuzzy inference system `fismat`.

CHAPTER 7
TEST AND EVALUATION

The intrusion detection system is constructed using Fuzzy C-Mean clustering as a data mining technique. The clustering results are used by the Matlab to define membership functions and the rule matrix. The System is trained and tested with the KDD CUP ’99 dataset and other datasets derived from it.

7.1. SETUP AND VALIDATION OF FUZZY INFERENCE SYSTEM

The original KDD Cup ’99 10% dataset is used. In the beginning, the dataset contained 494021 samples, 280790 “smurf” attacks, 107201 “neptune” attacks, and 97278 normal connections. The number of other attacks is much lower. After normal connections, the most frequent type of data is “back” attack which appeared 2203 times (Figure 9, Chapter 6). The most numerous attacks in original 10% dataset belong to DoS group.

Having excluded symbolic attributes, the number of attributes reduces from 41 to 33. After cleaning of duplicates the dataset contains 145585 samples, which is around 30% of the original 10% dataset. The most numerous are normal connection. “Neptune” is now the most numerous attack group. When classified into four major groups, normal connections still occurred more often than then DoS, Probe, R2L, and U2R (Figure 11, Chapter 6).

For classification purpose, symbolic values of normal connections and names of four attack types are replaced by numbers from one to five. After cleaning and classification into normal and four major attack group, the data is used for construction of the FIS. There is a possibility in the written code to choose a number of clusters or to select a default value which will trigger a subclustering method to obtain the number of clusters. In the first series of calculations, the default value is selected. The first conclusion is that the subclustering method is very time and memory consuming, especially compared to the option when the number of clusters is

manually defined. The time spent for automatic subclustering was around 90 minutes. The difference in results for two options is also checked and is presented below.

The first notable results are recorded when 6 clusters are obtained by the subclastering method. The Resulting confusion matrix for cleaned and classified original 10% dataset is presented in Table 8.

Table 8 Confusion matrix for cleaned and classified the KDD CUP '99 10% dataset when subclustering option is used

		Truth					
		normal	**dos**	**probe**	**r2l**	**u2r**	**total**
Predicted	**normal**	80853	1330	314	949	44	83490
	dos	2351	52540	1500	10	6	56407
	probe	4628	702	316	40	2	5688
	r2l	0	0	0	0	0	0
	u2r	0	0	0	0	0	0
	total	87832	54572	2130	999	52	145585

Overall accuracy, calculated by summing the number of correctly classified values and dividing the sum by the total number of values, is 91.84%. Two types of attack, R2L and U2R, are not detected at all. Problem is that most of these attacks are detected as normal connections. 95% of R2L and 84.61% of U2R are treated as normal packets. When it comes to Probe attack, 85.23% are detected as attacks but 70.42% are detected as a wrong attack group (DoS).

When the number of clusters is selected in advance, the time needed for calculation and output of results lasts less than a minute. It is noticed that the increase of cluster number has very little influence on precision rate and it does not influence at all the number of detected attacks. The only improvement that can be observed is in the number of correctly detected normal connections (Figure 13).

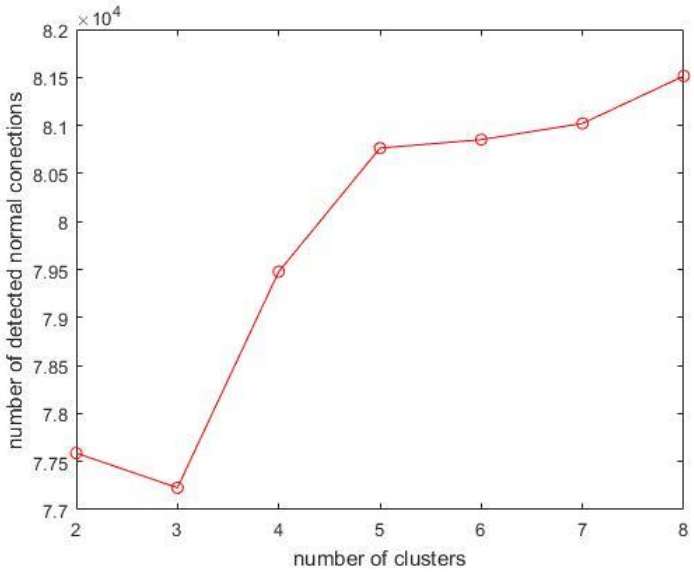

Figure 13 Number of detected normal connections with respect to number of clusters for the KDD CUP ’99 10% dataset

To confirm the assertion above, 16 clusters are selected alongside the same preprocessing rules as in the previous case. The Confusion matrix for this case is presented in Table 9. The number of correctly detected normal connections increased, as well as the number of correctly detected DoS and Probe attacks. The other two types of attacks are still not detected. The overall accuracy is now 94.28%. Normal packets are treated correctly in 96.46% of cases. Similar performance is achieved in case of DoS records (95.74%). It is obvious that normal and DoS connections are the most numerous sample groups.

Table 9 Confusion matrix for cleaned and classified the KDD CUP '99 10% dataset when 16 clusters are selected

		Truth					
		normal	**dos**	**Probe**	**r2l**	**u2r**	**total**
Predicted	**normal**	84725	1316	360	986	47	87434
	dos	1564	52252	1489	5	3	55313
	probe	1543	1004	281	8	2	2838
	r2l	0	0	0	0	0	0
	u2r	0	0	0	0	0	0
	total	87832	54572	2130	999	52	145585

When training data is classified into a normal connection group and an attack group (when only 2 groups are applied) and 4 clusters are selected, overall accuracy is 93.82%. the Confusion matrix for this case is presented in Table 9. This way of system setup produces a more consistent result since both normal and attack groups have a solid classification rate. 81612 of total 87832 normal records are treated as regular traffic and 54981 of total 57753 attack records are treated as malicious data.

The way of labeling data on a higher level, as normal or attack, provides more training data that belong to one single attack group. Like in previous cases, it is obvious that DoS are the most numerous attack group and thus mostly contribute to good overall accuracy related to the attack group. This could mean that the number of training data has a great influence on prediction rate. The same could be assumed for normal data since it is the most present in the dataset. Further on, additional analysis is performed.

Table 10 Confusion matrix for cleaned and classified the KDD CUP '99 10% dataset when 4 clusters are selected and only two groups normal and attack

		Truth		
		normal	**attack**	**total**
Predicted	**normal**	81612	2772	84384
	attack	6220	54981	61201
	total	87832	57753	145585

7.2. NSL-KDD DATASET

In the previous section the setup of the Fuzzy logic system based on Fuzzy C-Mean clustering and the first steps of training are presented. The training was conducted with the original KDD Cup '99 10% dataset with symbolic attributes excluded and data cleaned from duplicates. With the mentioned operations, the dataset is reduced to 30% of the previous number of samples but it is even smaller in size after symbolic attributes are deleted.

The result looks quite satisfactory according to the accuracy achieved (more than 90%). When attention is paid to details it is obvious that some types of attacks are not detected at all and attacks that are misclassified are mostly detected as normal connections.

The next dataset used for training of the system is one of the NSL-KDD datasets [38] with 125974 samples. The same type of cleaning and classification is applied. After processing, a few samples are discarded. The resulting number is 125941. Overall accuracy is 87.84%. The confusion matrix for this case is presented in Table 11.

Table 11 Confusion matrix for the cleaned and classified NSL-KDD datasets (16 clusters)

		Truth					
		normal	**dos**	**probe**	**r2l**	**u2r**	**Total**
Predicted	**normal**	61746	2173	951	916	44	65830
	dos	2242	41924	3729	39	3	47937
	probe	3355	1812	6962	40	5	12174
	r2l	0	0	0	0	0	0
	u2r	0	0	0	0	0	0
	total	67343	45909	11642	995	52	125941

The same problem appears as in case of usage of the original dataset. Good prediction accuracy is achieved for normal connections, DoS and probe. R2L and U2R are again not detected. R2L and U2R attacks, that are misclassified, are mostly detected as normal connections.

In the next iteration, symbolic values of attributes in the NSL-KDD database are replaced by numbers so that all attributes can be used in Fuzzy calculations. Resulting accuracy for the above dataset with all 41 attributes included is 86.36%. It is almost equivalent to the one achieved previously. The Confusion matrix for this case is illustrated in Table 12.

Table 12 Confusion matrix for the cleaned and classified NSL-KDD datasets (16 clusters) – symbolic attributes included in calculations

		Truth					
		normal	**dos**	**probe**	**r2l**	**u2r**	**Total**
Predicted	**normal**	59458	1402	537	863	44	62304
	dos	4386	44064	5864	86	6	54406
	probe	3499	443	5241	46	2	9231
	r2l	0	0	0	0	0	0
	u2r	0	0	0	0	0	0
	total	67343	45909	11642	995	52	125941

The advantage of this approach is that less misclassified attacks are detected as normal connections. When the NSL-KDD database is classified only into normal and

attack, the accuracy is increased to 92.05%. It turns out that the behavior of this database is the same as in the case of the original KDD CUP ’99 10% database.

7.3. ANALYSIS OF TWO SPECIFIC GROUPS OF ATTACKS

Normal connections and two major groups of attacks with large number of samples, DoS and Probe, are successfully detected in the previously presented calculations. Because of the large number of data, these groups raise the percentage of the accuracy of the result. The problem arises in two major attack groups, R2L and U2R, that have disproportional number of samples compared to the other groups. The problem arises for both datasets.

In order to process more training data related to these two attack groups, the original KDD CUP ’99 dataset is analyzed. The idea is to isolate all corresponding records and add them to the original KDD CUP ’99 10% dataset. It turns out that the original dataset has only 52 U2R records which are already presented in 10% dataset. The same is found out regarding R2L records. The original dataset contains 2183 records, but after cleaning of duplicates and incomplete data, the number is lowered to 999 records which are present in 10% dataset. An overview is presented in Table 13. This situation represents a limitation of input data since it is not possible to process additional data and try to increase prediction accuracy of the two specific types of attacks.

Table 13 Comparison of specific types of attack between the original KDD Cup ’99 and the 10% KDD Cup ’99 datasets

	Before preprocessing		**After preprocessing**	
	KDD Cup ’99	**10% KDD Cup ’99**	**KDD Cup ’99**	**10% KDD Cup ’99**
r2l	2183	1231	999	999
u2l	52	52	52	52

The solution for the problem of attacks’ detection coming from groups with small number of samples may be in the selection of appropriate continuous attributes. The analysis is carried out in such a way that the attributes belonging to a particular group are ejected from the calculation and the solution is tested. The goal is to obtain knowledge by each iteration. “Trial and error” learning method is repeated until improvement is reached. The confusion matrix from one of these tests is presented in Table 14.

Table 14 Confusion matrix for the cleaned and classified KDD CUP '99 10% dataset with 24 attributes are included in the calculation (attributes from

Table 4 excluded)

		Truth					
		normal	**dos**	**probe**	**r2l**	**u2r**	**total**
Predicted	**normal**	75432	13	50	96	9	75600
	dos	9227	52895	840	534	18	63514
	probe	3034	1664	1240	63	24	6025
	r2l	139	0	0	306	1	446
	u2r	0	0	0	0	0	0
	total	87832	54572	2130	999	52	145585

The attributes presented in

Table 4 are excluded from calculations. The reason for this lies in data analysis before it is pushed to the FIS but after all preprocessing is done. 75956 of total 125941 records have none of these attributes defined, which represents 60.31%. 46011 of total 125941 records have one of these attributes defined, which represents 36.53%. According to this information, it is assumed that this attribute group could be excluded in further steps. Table 15 shows the frequency of occurrence for each of 13 parameters of the group.

Table 15 Frequency of occurrence for attributes belonging to the group in Table 5

Number of defined attributes	Number of records	Share (%)
none	75956	60.311
1	46011	36.534
2	1572	1,248
3	2236	1.775
4	55	0.044
5	40	0.032
6	61	0.048
7	7	0.006
8	3	0.002
9	0	0.000
10	0	0.000
11	0	0.000
12	0	0.000
13	0	0.000

The overall accuracy of the test presented in Table 14 is 90.52%. The accuracy increases with the increase of cluster number. Once again, an increase in accuracy is achieved due to an increase of correctly detected groups with a large number of samples but, as illustrated in Table 14, approximately 30% of attacks belonging to the R2L group are correctly detected. Approximately 60% is detected as other groups of attacks and only 10% is detected as normal connections. The U2R group is not detected with this selection of attributes. This leads to the conclusion that more detailed attribute analysis may lead to an increase of classification rate of R2L and even U2R.

CONCLUSION

Network intrusion detection systems become important because the number of intrusion incidents has been increasing. Although current systems can offer a certain level of protection, they do show vulnerabilities in process of detecting novel attacks, which leads to an unacceptable level of false alarms rate. Therefore, the proposed framework based on Fuzzy decision-making module and Fuzzy C-Means algorithm represents an additional approach for detection of various types of network intrusions.

The records from the KDD Cup ‘99 dataset are used as a main source of data for training and testing. Additionally, in a phase of testing the NSL-KDD dataset is used.

The best achieved overall performance is 94.28%. The result is recorded when 16 clusters and 5 groups (normal and 4 types of attack) are defined. The conclusion is that normal and DoS records contribute most to the achieved result since prediction rate for both groups is above 95%. Two types of attacks, R2L and U2R, are not detected at all. Nevertheless, the limited number of samples in the dataset related to R2L and U2R should be taken into account.

The overall performance of 93.82% is achieved when 4 clusters and 2 groups (normal and attack) are defined. Similar to the previous case, normal and DoS records make the biggest contribution to the overall result.

After detailed testing and analysis of different test scenarios and setups, it can be concluded that the number of training records has a great influence on prediction accuracy. In other words, the more data is used as input, the more precise is prediction accuracy. Based on the given analysis, it could be said that this system can achieve a solid performance when working on big data.

It is important to emphasize that the attribute analysis process should be paid a great attention to because by filtering them, prediction accuracy can be drastically improved. This is proven when specific filtering of attributes is applied, which resulted in the improvement of prediction accuracy for R2L (approximately 30%).

To conclude, this thesis contributes to the topic of intrusion detection, while at the same time it opens certain new questions for further research. First of all, additional samples related to R2L and U2R should be simulated and added to the existing datasets. Besides that, the process of analysis of all attributes and determination of their correlation should be treated in more detail. It can lead to determination of the most relevant attributes which may result in significant improvement of classification rate. Last but not least, the classification rate may also be improved by tuning membership function and Fuzzy rules which may enable capturing new types of network incidents more precisely.

BIBLIOGRAPHY

[1] "DataBreaches: Information security," [Online]. Available: https://www.databreaches.net/category/commentaries-and-analyses/.

[2] "Data breaches," [Online]. Available: https://digitalguardian.com/blog/history-data-breaches.

[3] "Nato: Review Magazine," [Online]. Available: https://www.nato.int/docu/review/.

[4] "Forrester: Business Technographics," [Online]. Available: https://www.forrester.com/data/business/surveys.

[5] R. Shanmugavadivu and N. N, NETWORK INTRUSION DETECTION SYSTEM USING FUZZY, Indian Journal of Computer Science and Engineering (IJCSE).

[6] H. Tipton F and M. Krause, Information Security Management Handbook, CRC Press, p. 1978, 2000.

[7] A. Babak and A. Hamid, Emerging Trends in ICT Security, Morgan Kaufmann, p. 285, 2013.

[8] A. K. Jones and R. S. Sielken, Computer System Intrusion Detection: A Survey, University of Virginia.

[9] "Mathematical Introduction to Fuzzy logic," [Online]. Available: https://www.maplesoft.com/applications/view.aspx?SID=1409&view=html.

[10] "Calvin: Defining Fuzzy Sets," [Online]. Available: https://www.calvin.edu/~pribeiro/othrlnks/Fuzzy.

[11] J. P. Anderson, Computer Security Technology Planning Study, James P. Anderson Co., 1972.

[12] J. P. Anderson, Computer Security Threat Monitoring and Surveillance, James P. Anderson Co., 1980.

[13] D. E. Denning, An Intrusion-Detection Model, IEEE, 1986.

[14] G. Bruneau, The History and Evolution of Intrusion Detection, SANS Institute, 2001.

[15] "The Evolution of Firewalls: Past, Present & Future," [Online]. Available: https://www.informationweek.com/partner-perspectives/the-evolution-of-firewalls-past-present-and-future.

[16] "The Evolution of Intrusion Detection/Prevention," [Online]. Available: https://www.secureworks.com/blog/the-evolution-of-intrusion-detection-prevention.

[17] T. Lappas and K. Pelechrinis, Data Mining Techniques for (Network) Intrusion Detection Systems, UC Riverside.

[18] I. Guyon and E. A, An Introduction to Variable and Feature Selection, Journal of Machine Learning Research, p.1157-1182, 2003.

[19] H. Jiawei and M. Kamber, Data Mining: Concepts and Techniques, Morgan Kufmann, 2011.

[20] A. Husagic-Selman, Intrusion Detection System using Fuzzy Logic, Southeast Europe Journal of Soft Computing

[21] L. Jianxiong and B. S. M, Mining fuzzy association rules and fuzzy frequency eisodes for intrusion detection, 2000: p. 687-703.

[22] S.P. Thakare and M.S. Ali, NETWORK INTRUSION DETECTION SYSTEM & FUZZY LOGIC, BIOINFO Security Informatics,p. 23-27, 2012.

[23] A. F. A. Pinem and E. B. Setiawan, Implementation of Classification and Regression Tree and Fuzzy Logic Algorithm for IDS, 3rd International Conference on Information and Communication Technology, 2015.

[24] H. Mostaque Md. Morshedur, Network Intrusion Detection System Using Genetic Algorithm and Fuzzy Logic, International Journal of Innovative Research in Computer and Communication Engineering, 2013.

[25] B. Kavitha, S. Karthikeyan and P. Sheeba Maybell, Emerging Intuitionistic Fuzzy Classifiers for Intrusion Detection System, JOURNAL OF ADVANCES IN INFORMATION TECHNOLOGY, 2011.

[26] "Overview and Applications of Artificial Neural Networks," [Online]. Available: https://medium.com/@xenonstack/overview-of-artificial-neural-networks-and-its-applications-2525c1addff7.

[27] S. Singh and M. Bansal, mprovement of Intrusion Detection System in Data Mining using Neural Network, International Journal of Advanced Research in Computer Science and Software Engineering, 2013.

[28] U. Ahmed and A. Masood, Host based intrusion detection using RBF neural networks, Emerging Technologies, 2009. ICET 2009. International Conference, 2009.

[29] D. R. Jawale and B. V.K., A Novel Approach for Classification and Detection Network Intrusion Detection System Using ANN, International Journal of Advanced Research in Computer Science and Software Engineering, 2014.

[30] S. Devaraju and S. Ramakrishnan, Performance analysis of intrusion detection system using various neural network classifiers, Recent Trends in Information Technology (ICRTIT), 2011.

[31] S. O. Al-mamory and F. S. Jassim, On the designing of two grains levels network intrusion detection, Elsevier, p. 15-25, 2015.

[32] "Pipe and Filter Architecture," [Online]. Available: http://www.dossier-andreas.net/software_architecture/pipe_and_filter.html.

[33] "Instrusion Detection: Layered approach," [Online]. Available: https://www.ies.uni-kassel.de/intrusion_detection.

[34] J. Bezdek, "Pattern Recognition with Fuzzy Objective Function Algoritms," *Plenum Press, New York*, 1981.

[35] "Matlab documentation," [Online]. Available: https://www.mathworks.com/help/matlab/index.html.

[36] "KDD Cup 1999: Computer network intrusion detection," [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/task.html.

[37] M. L. Laboratory, "DARPA Intrusion Detection Evaluation," [Online]. Available: https://www.ll.mit.edu/ideval/data/1998data.html.

[38] S. Ranka, A. Banerjee, K. Kishore Biswas, S. Dua, P. Mishra, R. Moona, S.-H. Poon and C.-L. Wand, Contemporary Computing, Springer, p. 220, 2010.

[39] "NSL-KDD dataset," [Online]. Available: http://www.unb.ca/cic/datasets/nsl.html.

[40] "Mamdani's Fuzzy Inference Method," [Online], Available: http://www.cs.princeton.edu/cources/archive/fall07/cos436/HIDDEN/Knapp/fuzzy004.htm.

APPENDIX A
TESTING DOCUMENTATION

Below presented documentation is for KDD CUP ’99 10% dataset. For other datasets changes were applied to kddcup_analysis.m before execution.

Analysis of symbolic attributes of original data

Normal - attacks

back. 2203
buffer_overflow. 30
ftp_write. 8
guess_passwd. 53
imap. 12
ipsweep. 1247
land. 21
loadmodule. 9
multihop. 7
neptune. 107201
nmap. 231
normal. 97278
perl. 3
phf. 4
pod. 264
portsweep. 1040
rootkit. 10
satan. 1589
smurf. 280790
spy. 2
teardrop. 979
warezclient. 1020
warezmaster. 20

Type of the protocol, e.g. tcp, udp, etc.

icmp 283602
tcp 190065
udp 20354

Network service on the destination, e.g., http, telnet, etc.

IRC 43
X11 11
Z39_50 92
auth 328
bgp 106
courier 108
csnet_ns 126
ctf 97

daytime 103
discard 116
domain 116
domain_u 5863
echo 112
eco_i 1642
ecr_i 281400
efs 103
exec 99
finger 670
ftp 798
ftp_data 4721
gopher 117
hostnames 104
http 64293
http_443 99
imap4 117
iso_tsap 115
klogin 106
kshell 98
ldap 101
link 102
login 104
mtp 107
name 98
netbios_dgm 99
netbios_ns 102
netbios_ssn 107
netstat 95
nnsp 105
nntp 108
ntp_u 380
other 7237
pm_dump 1
pop_2 101
pop_3 202
printer 109
private 110893
red_i 1
remote_job 120
rje 111
shell 112
smtp 9723
sql_net 110
ssh 105
sunrpc 107
supdup 105
systat 115
telnet 513
tftp_u 1
tim_i 7
time 157
urh_i 14
urp_i 538
uucp 106
uucp_path 106
vmnet 106

whois 110

Normal or error status of the connection

OTH 8
REJ 26875
RSTO 579
RSTOS0 11
RSTR 903
S0 87007
S1 57
S2 24
S3 10
SF 378440
SH 107

land - 1 if connection is from/to the same host/port 0 otherwise

0 493999
1 22

1 if successfully logged in 0 otherwise

0 420784
1 73237

1 if root shell is obtained 0 otherwise

0 493966
1 55

1 if "su root" command attempted 0 otherwise

0 494009
1 6
2 6

1 if the login belongs to the "host" list 0 otherwise

0 494021

1 if the login is guest login 0 otherwise

0 493336
1 685

* Count how many times each group of attack appears in original data
and plot graph (values in descendant order)...

Analysis of symbolic attributes after duplicates were removed

Normal - attacks

back. 968
buffer_overflow. 30
ftp_write. 8
guess_passwd. 53
imap. 12
ipsweep. 651
land. 19
loadmodule. 9
multihop. 7
neptune. 51820

nmap. 158
normal. 87832
perl. 3
phf. 4
pod. 206
portsweep. 416
rootkit. 10
satan. 906
smurf. 641
spy. 2
teardrop. 918
warezclient. 893
warezmaster. 20

Type of the protocol, e.g. tcp, udp, etc.

icmp 2406
tcp 130913
udp 12267

Network service on the destination, e.g., http, telnet, etc.

IRC 43
X11 11
Z39_50 91
auth 328
bgp 104
courier 108
csnet_ns 126
ctf 97
daytime 103
discard 116
domain 114
domain_u 5425
echo 112
eco_i 916
ecr_i 1027
efs 101
exec 98
finger 668
ftp 798
ftp_data 4592
gopher 117
hostnames 103
http 62054
http_443 99
imap4 117
iso_tsap 115
klogin 106
kshell 98
ldap 101
link 102
login 103
mtp 107
name 98
netbios_dgm 98
netbios_ns 102
netbios_ssn 107

netstat 95
nnsp 105
nntp 108
ntp_u 290
other 4769
pm_dump 1
pop_2 101
pop_3 200
printer 108
private 49057
red_i 1
remote_job 120
rje 111
shell 111
smtp 9721
sql_net 110
ssh 105
sunrpc 107
supdup 105
systat 115
telnet 512
tftp_u 1
tim_i 5
time 139
urh_i 14
urp_i 443
uucp 105
uucp_path 106
vmnet 106
whois 110

Normal or error status of the connection

OTH 7
REJ 14712
RSTO 569
RSTOS0 11
RSTR 425
S0 42278
S1 57
S2 24
S3 10
SF 87459
SH 34

land - 1 if connection is from/to the same host/port 0 otherwise

0 145566
1 20

1 if successfully logged in 0 otherwise

0 74032
1 71554

1 if root shell is obtained 0 otherwise

0 145531
1 55

1 if "su root" command attempted 0 otherwise

0 145574
1 6
2 6

1 if the login belongs to the "host" list 0 otherwise

0 145586

1 if the login is guest login 0 otherwise

0 144901
1 685

* Count how many times each group of attack appears in data after cleaning
and plot graph (values in desendant order)...

Individual attacks compared to normal data

normal = 87832
attack = 57753

Generate fuzzy inference system (FIS)...
Iteration count = 1, obj. fcn = 472618.048943
Iteration count = 2, obj. fcn = 355638.303479
Iteration count = 3, obj. fcn = 355607.243265
Iteration count = 4, obj. fcn = 355398.905748
Iteration count = 5, obj. fcn = 354007.783208
Iteration count = 6, obj. fcn = 345187.694169
Iteration count = 7, obj. fcn = 303310.645801
Iteration count = 8, obj. fcn = 230586.177592
Iteration count = 9, obj. fcn = 202604.601890
Iteration count = 10, obj. fcn = 187861.145303
Iteration count = 11, obj. fcn = 173084.885252
Iteration count = 12, obj. fcn = 158127.450175
Iteration count = 13, obj. fcn = 150796.932600
Iteration count = 14, obj. fcn = 147799.246684
Iteration count = 15, obj. fcn = 145815.380628
Iteration count = 16, obj. fcn = 143599.500727
Iteration count = 17, obj. fcn = 139406.425212
Iteration count = 18, obj. fcn = 132455.630481
Iteration count = 19, obj. fcn = 124493.216022
Iteration count = 20, obj. fcn = 118235.245152
Iteration count = 21, obj. fcn = 114411.124282
Iteration count = 22, obj. fcn = 109075.032389
Iteration count = 23, obj. fcn = 105648.342882
Iteration count = 24, obj. fcn = 105366.125478
Iteration count = 25, obj. fcn = 105353.532887
Iteration count = 26, obj. fcn = 105351.090722
Iteration count = 27, obj. fcn = 105349.963688
Iteration count = 28, obj. fcn = 105349.321548
Iteration count = 29, obj. fcn = 105348.929933
Iteration count = 30, obj. fcn = 105348.684665
Iteration count = 31, obj. fcn = 105348.529400
Iteration count = 32, obj. fcn = 105348.430686
Iteration count = 33, obj. fcn = 105348.367820
Iteration count = 34, obj. fcn = 105348.327757
Iteration count = 35, obj. fcn = 105348.302220
Iteration count = 36, obj. fcn = 105348.285941

```
Iteration count = 37, obj. fcn = 105348.275565
Iteration count = 38, obj. fcn = 105348.268950
Iteration count = 39, obj. fcn = 105348.264734
Iteration count = 40, obj. fcn = 105348.262047
Iteration count = 41, obj. fcn = 105348.260334
Iteration count = 42, obj. fcn = 105348.259242
Iteration count = 43, obj. fcn = 105348.258547
Iteration count = 44, obj. fcn = 105348.258103
Iteration count = 45, obj. fcn = 105348.257821
Iteration count = 46, obj. fcn = 105348.257640
Iteration count = 47, obj. fcn = 105348.257526
Iteration count = 48, obj. fcn = 105348.257452
Iteration count = 49, obj. fcn = 105348.257406
Iteration count = 50, obj. fcn = 105348.257376
Iteration count = 51, obj. fcn = 105348.257357
Iteration count = 52, obj. fcn = 105348.257345
Iteration count = 53, obj. fcn = 105348.257337

Saving fuzzy C-means clustering results

Saving fuzzy inference system (FIS) to file
>>
```

APPENDIX B

SOURCE CODE

For kddcup_analysis.m dataset must be organized as original KDD CUP ’99 10% dataset. For other datasets changes must be applied to code or to dataset.

```
close all;
clear
clc

%% input/output
% name of input and output files
tableName = 'kddcup.data_10_percent_corrected.txt';
uniqueTableName = 'T_unique.dat';
fcmOut = 'out.dat';
fisOut = 'myfis.fis';

% number of clusters (0 if subclustering)
numClast = 8;

% if attacks are grouped in four groups group = 4 if in normal-attack group = 2
group = 4;

% selection of attributes - only continuous attributes
attributesName = {...
        'dur', ...
        'src_bytes', ...
        'dst_bytes', ...
        'wrong_fragment', ...
        'urgent', ...
        'hot', ...
        'num_failed_logins', ...
        'num_compromised', ...
        'root_shell', ...
        'num_root', ...
        'num_file_creations', ...
        'num_shells', ...
        'num_access_files', ...
        'num_outbound_cmds', ...
        'count', ...
        'srv_count', ...
        'serror_rate', ...
        'srv_serror_rate', ...
        'rerror_rate', ...
        'srv_rerror_rate', ...
```

```
            'same_srv_rate', ...
            'diff_srv_rate', ...
            'srv_diff_host_rate', ...
            'dst_host_count', ...
            'dst_host_srv_count', ...
            'dst_host_same_srv_rate', ...
            'dst_host_diff_srv_rate', ...
            'dst_host_same_src_port_rate', ...
            'dst_host_srv_diff_host_rate', ...
            'dst_host_serror_rate', ...
            'dst_host_srv_serror_rate', ...
            'dst_host_rerror_rate', ...
            'dst_host_srv_rerror_rate'...
            'label'};

%% analysis KDD cup data and generate cleaned table
kddcup_analysis(tableName, group, uniqueTableName);

%% Generate fuzzy inference system (FIS)
disp(' ')
disp('Generate fuzzy inference system (FIS)...')
[out, myfis] = kddcup_fis(uniqueTableName, attributesName, numClast);

% display results
disp(' ')
disp('Saving fuzzy C-means clustering results')
save(fcmOut,'out','-ascii');
disp(' ')
disp('Saving fuzzy inference system (FIS) to file')
writefis(myfis,fisOut);

% calculate and print confusion matrix
if group == 4
   conf(5);
else
   conf(2);
end


function [] = kddcup_analysis(filename, group, uniqueTableName)
% import data
T = readtable(filename);

%% analysis of symbolic attributes
disp(' ');
disp(' Analysis of symbolic attributes of original data');
disp('----------------------------------------------------------------');
n_a = categorical(T.normal_attack);
disp(' ');
disp(' Normal - attacks');
disp('----------------------------------------------------------------');
```

```
summary(n_a);
T.protocol_type = categorical(T.protocol_type);
disp(' ');
disp(' Type of the protocol, e.g. tcp, udp, etc.');
disp('---------------------------------------------------------------');
summary(T.protocol_type);
T.service = categorical(T.service);
disp(' ');
disp(' Network service on the destination, e.g., http, telnet, etc.');
disp('---------------------------------------------------------------');
summary(T.service);
T.flag = categorical(T.flag);
disp(' ');
disp(' Normal or error status of the connection');
disp('---------------------------------------------------------------');
summary(T.flag);
T.land = categorical(T.land);
disp(' ');
disp(' land - 1 if connection is from/to the same host/port 0 otherwise ');
disp('---------------------------------------------------------------');
summary(T.land);
T.logged_in = categorical(T.logged_in);
disp(' ');
disp(' 1 if successfully logged in 0 otherwise');
disp('---------------------------------------------------------------');
summary(T.logged_in);
T.root_shell = categorical(T.root_shell);
disp(' ');
disp(' 1 if root shell is obtained 0 otherwise');
disp('---------------------------------------------------------------');
summary(T.root_shell);
T.su_attempted = categorical(T.su_attempted);
disp(' ');
disp(' 1 if "su root" command attempted 0 otherwise');
disp('---------------------------------------------------------------');
summary(T.su_attempted);
T.is_host_login = categorical(T.is_host_login);
disp(' ');
disp(' 1 if the login belongs to the "host" list 0 otherwise');
disp('---------------------------------------------------------------');
summary(T.is_host_login);
T.is_guest_login = categorical(T.is_guest_login);
disp(' ');
disp(' 1 if the login is guest login 0 otherwise');
disp('---------------------------------------------------------------');
summary(T.is_guest_login);

%% Count how many times each group of attack appears in original data and plot graph
figure
subplot(1,2,1)
disp(' ')
```

```
disp('* Count how many times each group of attack appears in original data');
disp(' and plot graph (values in desendant order)...');
tmp(:,1) = unique(T.normal_attack,'stable');
tmp(:,2) = cellfun(@(x) sum(ismember(T.normal_attack,x)),tmp(:,1),'un',0);
tmp = sortrows(tmp, 2,'descend');
bar(1:23,cell2mat(tmp(:,2)));
set(gca,'TickLabelInterpreter', 'none');
set(gca,'yscale','linear')
ax = gca;
ax.XTick = 1:23;
ax.XTickLabels = {string(tmp(:,1))};
ax.XTickLabelRotation = 45;
ylabel('number');
axis tight;
grid;
subplot(1,2,2)
bar(4:23,cell2mat(tmp(4:23,2)));
set(gca,'TickLabelInterpreter', 'none');
set(gca,'yscale','linear')
ax = gca;
ax.XTick = 4:23;
ax.XTickLabels = {string(tmp(4:23,1))};
ax.XTickLabelRotation = 45;
ylabel('number');
axis tight;
grid;

%% cleaning duplicates I
T_unique = unique(T);

% analysis of symbolic attributes
disp(' ');
disp(' Analysis of symbolic attributes after duplicates were removed');
disp('-----------------------------------------------------------------');
n_a = categorical(T_unique.normal_attack);
disp(' ');
disp(' Normal - attacks');
disp('-----------------------------------------------------------------');
summary(n_a);
T_unique.protocol_type = categorical(T_unique.protocol_type);
disp(' ');
disp(' Type of the protocol, e.g. tcp, udp, etc.');
disp('-----------------------------------------------------------------');
summary(T_unique.protocol_type);
T_unique.service = categorical(T_unique.service);
disp(' ');
disp(' Network service on the destination, e.g., http, telnet, etc.');
disp('-----------------------------------------------------------------');
summary(T_unique.service);
T_unique.flag = categorical(T_unique.flag);
disp(' ');
```

```
disp(' Normal or error status of the connection');
disp('------------------------------------------------------------------');
summary(T_unique.flag);
T_unique.land = categorical(T_unique.land);
disp(' ');
disp(' land - 1 if connection is from/to the same host/port 0 otherwise ');
disp('------------------------------------------------------------------');
summary(T_unique.land);
T_unique.logged_in = categorical(T_unique.logged_in);
disp(' ');
disp(' 1 if successfully logged in 0 otherwise');
disp('------------------------------------------------------------------');
summary(T_unique.logged_in);
T_unique.root_shell = categorical(T_unique.root_shell);
disp(' ');
disp(' 1 if root shell is obtained 0 otherwise');
disp('------------------------------------------------------------------');
summary(T_unique.root_shell);
T_unique.su_attempted = categorical(T_unique.su_attempted);
disp(' ');
disp(' 1 if "su root" command attempted 0 otherwise');
disp('------------------------------------------------------------------');
summary(T_unique.su_attempted);
T_unique.is_host_login = categorical(T_unique.is_host_login);
disp(' ');
disp(' 1 if the login belongs to the "host" list 0 otherwise');
disp('------------------------------------------------------------------');
summary(T_unique.is_host_login);
T_unique.is_guest_login = categorical(T_unique.is_guest_login);
disp(' ');
disp(' 1 if the login is guest login 0 otherwise');
disp('------------------------------------------------------------------');
summary(T_unique.is_guest_login);

%% Count how many times each group of attack appears in data after cleaning and plot
graph
figure
subplot(1,2,1)
disp(' ')
disp('* Count how many times each group of attack appears in data after cleaning');
disp(' and plot graph (values in desendant order)...');
tmp(:,1) = unique(T_unique.normal_attack,'stable');
tmp(:,2) = cellfun(@(x) sum(ismember(T_unique.normal_attack,x)),tmp(:,1),'un',0);
tmp = sortrows(tmp, 2,'descend');
bar(1:23,cell2mat(tmp(:,2)));
set(gca,'TickLabelInterpreter', 'none');
set(gca,'yscale','linear')
ax = gca;
ax.XTick = 1:23;
ax.XTickLabels = {string(tmp(:,1))};
ax.XTickLabelRotation = 45;
```

```
ylabel('number');
axis tight;
grid;
subplot(1,2,2)
bar(3:23,cell2mat(tmp(3:23,2)));
set(gca,'TickLabelInterpreter', 'none');
set(gca,'yscale','linear')
ax = gca;
ax.XTick = 3:23;
ax.XTickLabels = {string(tmp(3:23,1))};
ax.XTickLabelRotation = 45;
ylabel('number');
axis tight;
grid;
%%
label = zeros(height(T_unique),1);
if (group == 4)
        for i = 1:height(T_unique)
                if(isequal(T_unique.normal_attack{i},'normal.'))
                        label(i,1) = 1;
                elseif(isequal(T_unique.normal_attack{i},'back.') || ...
                        isequal(T_unique.normal_attack{i},'land.') || ...
                        isequal(T_unique.normal_attack{i},'neptune.') || ...
                        isequal(T_unique.normal_attack{i},'pod.') || ...
                        isequal(T_unique.normal_attack{i},'smurf.') || ...
                        isequal(T_unique.normal_attack{i},'teardrop.'))
                        label(i,1) = 2;
                elseif(isequal(T_unique.normal_attack{i},'ipsweep.') || ...
                        isequal(T_unique.normal_attack{i},'nmap.') || ...
                        isequal(T_unique.normal_attack{i},'portsweep.') || ...
                        isequal(T_unique.normal_attack{i},'satan.'))
                        label(i,1) = 3;
                elseif(isequal(T_unique.normal_attack{i},'buffer_overflow.') || ...
                        isequal(T_unique.normal_attack{i},'loadmodule.') || ...
                        isequal(T_unique.normal_attack{i},'perl.') || ...
                        isequal(T_unique.normal_attack{i},'rootkit.'))
                        label(i,1) = 5;
                else
                        label(i,1) = 4;
                end
        end
        T1 = table(label);
        T_unique.normal_attack = [];
        T_unique = [T_unique T1];

        %% cleaning duplicates II
        T_unique2 = unique(T_unique);

        %% histogram normal - attacks
        figure
        C = categorical(T_unique2.label,[1 2 3 4 5],{'normal','dos','probe','r2l','u2r'});
```

```
        h = histogram(C,'BarWidth',0.7);
        ylabel('number');
        grid
        disp(' ');
        disp('Individual attacks compared to normal data');
        disp('-------------------------------------------------------------');
        disp(['normal = ', num2str(h.Values(1))]);
        disp(['dos = ', num2str(h.Values(2))]);
        disp(['probe = ', num2str(h.Values(3))]);
        disp(['r2l = ', num2str(h.Values(4))]);
        disp(['u2r = ', num2str(h.Values(5))]);
else
        for i = 1:height(T_unique)
                if(isequal(T_unique.normal_attack{i},'normal.'))
                        label(i,1) = 1;
                else
                        label(i,1) = 2;
                end
        end
        T1 = table(label);
        T_unique.normal_attack = [];
        T_unique = [T_unique T1];

        %% cleaning duplicates II
        T_unique2 = unique(T_unique);

        %% histogram normal - attacks
        figure
        C = categorical(T_unique2.label,[1 2],{'normal','attack'});
        h = histogram(C,'BarWidth',0.7);
        ylabel('number');
        grid
        disp(' ');
        disp('Individual attacks compared to normal data');
        disp('-------------------------------------------------------------');
        disp(['normal = ', num2str(h.Values(1))]);
        disp(['attack = ', num2str(h.Values(2))]);
end

writetable(T_unique2,uniqueTableName);
end



function [out, correl, myfis] = kddcup_fis(tablename, attribute_name, numClast)
% training data input
T = readtable(tablename);
train = T{:, attribute_name};
trainRes = train(:,end);
trainData = train(:,1:end-1);
```

```
% training data normalization
[trainDataMap, ~] = mapstd(trainData);
[trainResMap, trainRes_ps] = mapstd(trainRes);

%% generate fuzzy inference system structure from data with genfis

% FCM clastering - clastering type Mamdani
if (numClast ~= 0)
        opt = genfisOptions('FCMClustering','FISType','mamdani','NumClusters',numClast);
else
        opt = genfisOptions('FCMClustering','FISType','mamdani');
end
myfis = genfis(trainDataMap,trainResMap,opt);

% starting fuzzy logic designer
fuzzyLogicDesigner(myfis);

%% evaluation of fis
trainOut = evalfis(trainDataMap,myfis);

% convert training data output back into the original units
out11 = mapstd('reverse',trainOut,trainRes_ps);
out12 = mapstd('reverse',trainResMap,trainRes_ps);
tmp1 = 0; tmp2 = 0; tmp3 = 0;
for i = 1:length(out11)
        tmp1 = tmp1 + (out11(i,1) - mean(out11))*(out12(i,1) - mean(out12));
        tmp2 = tmp2 + (out11(i,1)-mean(out11))^2;
        tmp3 = tmp3 + (out12(i,1)-mean(out12))^2;
end
correl = tmp1 / sqrt(tmp2 * tmp3);

% training result and original data (out11 and out12)
out = [out11 out12];
end
```

```
function [out] = kddcup_test(tablename, attribute_name, myfisName)
% training data input
T = readtable(tablename);
test = T{:, attribute_name};
myfis = readfis(myfisName);
testRes = test(:,end);
testData = test(:,1:end-1);

% test data normalization
[testDataMap, ~] = mapstd(testData);
[testResMap, testRes_ps] = mapstd(testRes);

%% evaluation of fis with genfis results
```

```
testOut = evalfis(testDataMap,myfis);

% convert testing data output back into the original units
out11 = mapstd('reverse',testOut,testRes_ps);
out12 = mapstd('reverse',testResMap,testRes_ps);

% testing result and original data (out11 and out12)
out = [out11 out12];

end


function [] = conf(n)
load('out.dat');
outputs = round(out(:,1));
target = out(:,2);

% calculate confusion matrix
m = n + 1;
confm = zeros(m,m);
for i = 1:length(outputs)
   confm(outputs(i),target(i)) = confm(outputs(i),target(i)) + 1;
   confm(m,target(i)) = confm(m,target(i)) + 1;
end

% calculate totals and accuracy
accu = 0;
for i = 1:n
   for j = 1:n
      confm(i,m) = confm(i,m) + confm(i,j);
   end
   confm(m,m) = confm(m,m) + confm(i,m);
   accu = accu + confm(i,i);
end
accu = 100*accu/confm(m,m);

% display result for accuracy and confusion matrix
disp(' ')
disp(['overall accuracy = ' num2str(accu) '%'])
disp(' ')
disp('confusion matrix')
disp(' ')
disp(confm)
```