Ain Shams University
Faculty of Computer & Information Sciences

Implementation of Intelligent Techniques for Intrusion Detection Systems

A Thesis Submitted to
Department of Computer Science
In partial fulfillment of the requirements for
The degree of Master of Science

by

Sahar Selim Fouad
B.Sc. Computer & Information Sciences
(Computer Science Department)

Under Supervision of

Prof. Dr. Taymoor M. Nazmy
Professor in Computer Science department,
Vice Dean of Higher Studies and Researches,
Faculty of Computer & Information Sciences,
Ain Shams University

Prof. Dr. Mohamed Hashem
Professor in Information System department,
Vice Dean of Educational & Students’ Affairs,
Faculty of Computer & Information Sciences,
Ain Shams University

Cairo 2011
Acknowledgements

Praise is to Allah and gratitude is given where it is due most to Allah.

So with genuine humility, I acknowledge the aid of Allah.

My deepest gratitude and sincerest appreciation to Prof. Dr. Mohamed Hashem Abdel-Aziz, for his continuous encouragement and enlightening advice and helpful advices, that helped me throughout this work.

I feel greatly indebted to Prof. Dr. Taymoor M. Nazmy for his dynamic efforts, sincere guidance and continuous support without which this work would have never seen the light.

Finally, I cannot sufficiently express my gratitude to my family; my parents, my sisters, my dearest friend Manal Mohsen whose love, support, guidance, and encouragement through the years have thoroughly equipped me for life, and my son Abdel Rahman, who makes it all worthwhile.
Abstract

With the rapid expansion of computer networks during the past decade, security has become a crucial issue for computer systems. New security failures are discovered everyday and there are a growing number of bad-intentioned people trying to take advantage of such failures. Intrusion detection is a critical process in network security. Intrusion Detection Systems (IDS) aim at protecting networks and computers from malicious network-based or host-based attacks.

Different soft-computing based methods have been proposed in recent years for the development of intrusion detection systems. Most current approaches to intrusion detection involve the use of rule-based expert systems to identify indications of known attacks. Artificial neural networks and decision trees provide the potential to identify and classify network activity.

Most of the previous systems have some deficiencies. Some drawbacks of previous Intrusion detection systems (IDSs) are that they are unable to detect new attacks that are never seen before. Most of these systems don’t identify the attack type but only specify whether the given network data is normal or attack. One of the drawbacks of IDSs that are signature-based is that they can only detect known attacks while all new unknown attacks will go unnoticed until the system is updated to be able to detect them.

This thesis proposes a hybrid intelligent intrusion detection system to improve the detection rate for known and unknown attacks. The introduced system has the capability to learn fast, enhanced capability of detection of new unidentified attacks, and alarming the system administrator of these unseen before attacks. Unlike other systems that have one level of detection, the proposed system has three levels of detection. The first level is where the system classifies the network users to either normal or intruder. The second level is where system can identify four categories of intruders (DOS, Probe, R2L & U2R). The third level is the fine detection level, where the attack type can be identified.
The proposed model consists of multi-level based on hybrid neural network and decision tree. We examined different neural network and decision tree techniques. Each module in each level is implemented with the technique (Neural Network or Decision Tree) which gave best experimental results for this module.

From our experimental results with different network data, our model achieves correct classification rate of 93.64%, average detection rate about 98%; 99.8% for known attacks and 93.8% for new unknown attacks.

The advantages of the proposed system is its **high Detection Rate**, **scalability** (if new attacks of specific class are added to the dataset we don't have to train all the modules but only the modules affected by the new attack), **adaptive** (attacks that are misclassified by the IDS as normal activities or given wrong attack type will be relabeled by the network administrator. Training module can be retrained at any point of time which makes its implementation adaptive to any new environment or any new attacks in the network), **generalization ability** (the proposed system outperforms previous IDSs in detecting both known and new attacks which combines the advantages of signature-based and anomaly-based IDS). Also every module can be trained on separate computer in parallel which provides less training time.

2- Sahar Selim, Mohamed Hashem and Taymoor M. Nazmy, “Hybrid Multi-level Intrusion Detection System,” International Journal of Computer Science and Information Security (IJCSIS), pp. 23-29, Vol. 9, No. 5, May 2011, (ISSN 1947 5500, Pittsburgh, USA, Impact Factor 0.423, http://sites.google.com/site/ijcsis/) . This paper was selected for “IJCSIS Best Paper Award”.
# CONTENTS

<table>
<thead>
<tr>
<th>Acknowledgment</th>
<th>II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>III</td>
</tr>
<tr>
<td>Publications</td>
<td>V</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>VI</td>
</tr>
<tr>
<td>List of Figures</td>
<td>VI</td>
</tr>
<tr>
<td>List of Tables</td>
<td>X</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 What’s Intrusion?</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Firewall Evasion</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Classification of Intrusion Detection Systems</td>
<td>3</td>
</tr>
<tr>
<td>1.3.1 Classification Based on Data Source</td>
<td>3</td>
</tr>
<tr>
<td>1.3.2 Classification Based on Detection Approach</td>
<td>7</td>
</tr>
<tr>
<td>1.4 Difference between Intrusion Detection System and Intrusion Prevention System</td>
<td>8</td>
</tr>
<tr>
<td>1.5 Thesis Outline</td>
<td>8</td>
</tr>
<tr>
<td>2 Related Works for Intrusion Detection Systems</td>
<td>10</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>10</td>
</tr>
<tr>
<td>2.2 Intrusion Detection categorized according to techniques used</td>
<td>12</td>
</tr>
<tr>
<td>2.2.1 Intrusion Detection Systems based on Neural Network</td>
<td>12</td>
</tr>
<tr>
<td>2.2.2 Intrusion Detection Systems based on Decision Tree</td>
<td>18</td>
</tr>
<tr>
<td>2.2.3 Hybrid Intrusion Detection System</td>
<td>22</td>
</tr>
<tr>
<td>3 Theoretical Aspects</td>
<td>33</td>
</tr>
<tr>
<td>3.1 Artificial Intelligence and Intrusion Detection</td>
<td>33</td>
</tr>
<tr>
<td>3.2 Artificial Neural Networks Approach</td>
<td>33</td>
</tr>
<tr>
<td>3.2.1 Neural Network and Intrusion Detection</td>
<td>35</td>
</tr>
<tr>
<td>3.2.1.1 Difference between Supervised &amp; Unsupervised Learning</td>
<td>35</td>
</tr>
<tr>
<td>3.2.2 Application of Neural Networks in Misuse Detection</td>
<td>37</td>
</tr>
<tr>
<td>3.2.3 Advantages of Neural Network-based Misuse</td>
<td>38</td>
</tr>
</tbody>
</table>
Detection Systems ................................................................. 39
3.2.4 Disadvantages of Neural Network-based Misuse Detection Systems ........................................ 39
3.2.5 Multi-Layer Perceptron ................................................. 40
   3.2.5.1 Topology of MLP Network ...................................... 40
   3.2.5.2 Training MLP classifier .......................................... 42
3.2.6 Radial Base Functions .................................................. 43
   3.2.6.1 Topology of RBF Network ...................................... 45
   3.2.6.2 Training of RBF .................................................. 45
3.2.7 Exhaustive Prune .......................................................... 47
3.3 Decision trees ............................................................... 48
   3.3.1 Decision Trees Approach .......................................... 49
   3.3.2 C5.0 Decision Trees ................................................ 50
   3.3.3 Classification and Regression Trees .............................. 52
   3.3.4 Chi-squared Automatic Interaction Detector .............. 53
   3.3.5 Quick, Unbiased, Efficient Statistical Tree ................. 55
3.4 Selection of Neural Network & Decision Tree Classifier among Other classification techniques .......... 56
   3.4.1 Neural Network Classifiers ...................................... 56
   3.4.2 Decision Trees classifiers ........................................ 57
4 Proposed System Architecture ................................. 59
   4.1 Introduction .............................................................. 59
4.2 Proposed System Architecture ........................................ 60
   4.2.1 The Capture Module .............................................. 61
   4.2.2 The Preprocessing Module ...................................... 61
   4.2.3 The classification Module ...................................... 62
   4.2.4 The Decision Module ............................................. 63
4.3 Architectures Examined .................................................. 63
   4.3.1 Single Level Neural Network Intrusion Detection System .............................................. 64
   4.3.2 Multi-Level Neural Network Intrusion Detection System .............................................. 64
   4.3.3 Hybrid Multi-Level Intrusion Detection System .............. 68
   4.3.4 Enhanced Hybrid Multi-Level Intrusion Detection System .............................................. 70
4.4 Dataset ........................................................................ 72
   4.4.1 NSL-KDD data set description .................................. 73
5 Experimental Results ...................................................... 76
5.1 Definitions of Performance Measures ........................................ 76
5.2 Experiment 1 ........................................................................... 77
  5.2.1 Dataset used in Experiment 1 ............................................ 77
  5.2.2 The Over-fitting Problem .................................................. 78
  5.2.3 Single Level Neural Network ............................................. 79
    5.2.3.1 Training Single Level Neural Network ......................... 79
    5.2.3.2 Single Level Neural Network Testing Results ............ 79
  5.2.4 Multi-Level Neural Network ............................................. 79
    5.2.4.1 Training multi-level Neural Network ......................... 79
    5.2.4.2 Multi-Level Neural Network Testing Results .......... 80
  5.2.5 Discussion of Results of Experiment 1 ............................ 81
5.3 Experiment 2 ........................................................................... 83
  5.3.1 Dataset used in Experiment 2 ............................................ 83
  5.3.2 Training Hybrid Multi-Level Intrusion Detection System ........................................... 85
    5.3.2.1 Neural Networks Implementation ............................. 85
      5.3.2.1.1 Multi-Layer Perceptron Implementation .......... 85
      5.3.2.1.2 Radial Basis Function Implementation .......... 86
      5.3.2.1.3 Exhaustive Prune Implementation ................. 87
    5.3.2.2 Decision trees Implementation ................................. 88
      5.3.2.2.1 C5.0 Tree Implementation ............................. 88
      5.3.2.2.2 Classification and Regression Trees Implementation ................................................. 89
      5.3.2.2.3 Chi-squared Automatic Interaction Detector Implementation ........................................ 89
      5.3.2.2.4 Quick, Unbiased, Efficient Statistical Tree Implementation ........................................... 90
  5.3.3 Hybrid Multi-Level Testing Results .................................. 90
    5.3.3.1 Level 1 output ......................................................... 90
    5.3.3.2 Level 2 output ......................................................... 91
    5.3.3.3 Level 3 output ......................................................... 92
  5.3.4 Enhanced Hybrid Multi-Level Testing Results .................. 95
  5.3.5 Discussion of Results of Experiment 2 ............................ 97
6 Conclusion and Future Work ..................................................... 98
  6.1 Conclusion ........................................................................... 98
  6.2 Future Work ........................................................................ 101
References .................................................................................. 102
Appendix ...................................................................................... 108
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>Attack via tunnels in a firewall [1]</td>
<td>4</td>
</tr>
<tr>
<td>Figure 1.2</td>
<td>Shows how the two types of IDS may exist in a network environment [4]</td>
<td>5</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>Previous works categorization</td>
<td>11</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>System Architecture and Data Flow Diagram [16]</td>
<td>15</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Network intrusion detection using labeled data [42]</td>
<td>35</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>Network intrusion detection using unlabelled data [42]</td>
<td>36</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>MLP with one hidden layer [45]</td>
<td>41</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Learning Phase of Proposed System architecture</td>
<td>60</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>Working Phase Proposed System architecture</td>
<td>60</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>Single-Stage Single Layer Perceptron Network which Classify Normal and Attack type</td>
<td>64</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>Multi-Levels</td>
<td>65</td>
</tr>
<tr>
<td>Figure 4.5</td>
<td>First Level Network which differentiate between Normal and Attack</td>
<td>66</td>
</tr>
<tr>
<td>Figure 4.6</td>
<td>Single Layer Perceptron of Second Level Network which Classify the Attack Class DOS or Probe</td>
<td>66</td>
</tr>
<tr>
<td>Figure 4.7</td>
<td>Single Layer Perceptron of third Level Network which Classify Attack type of DOS category</td>
<td>67</td>
</tr>
<tr>
<td>Figure 4.8</td>
<td>Third Level Network Single Layer Perceptron which Classify Attack type of Probe category</td>
<td>68</td>
</tr>
<tr>
<td>Figure 4.9</td>
<td>Levels of Enhanced Hybrid System</td>
<td>70</td>
</tr>
<tr>
<td>Figure 4.10</td>
<td>Dual Protection Stages of Enhanced Multi-Level Intrusion Detection System</td>
<td>71</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>Comparison between Multi-Level and Single-Level</td>
<td>82</td>
</tr>
<tr>
<td>Figure 5.2</td>
<td>Detection and False Alarm Rate of Multi-Level and Single-Level</td>
<td>83</td>
</tr>
<tr>
<td>Figure 5.3</td>
<td>Level 1 Classification Rate</td>
<td>91</td>
</tr>
<tr>
<td>Figure 5.4</td>
<td>Level 2 Classification Rate</td>
<td>92</td>
</tr>
<tr>
<td>Figure 5.5</td>
<td>Summary of Results of Enhanced Hybrid Multi-Level System</td>
<td>95</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Network-Based vs. Host-Based Intrusion-Detection Systems [6]</td>
<td>6</td>
</tr>
<tr>
<td>2.1</td>
<td>Summarizes the previous work of Intrusion Detection</td>
<td>26</td>
</tr>
<tr>
<td>5.1</td>
<td>Single Level Classification Rate</td>
<td>79</td>
</tr>
<tr>
<td>5.2</td>
<td>Level 1 Classification Results</td>
<td>80</td>
</tr>
<tr>
<td>5.3</td>
<td>Level 2 Classification Rate</td>
<td>81</td>
</tr>
<tr>
<td>5.4</td>
<td>Level 3 Classification Rate</td>
<td>81</td>
</tr>
<tr>
<td>5.5</td>
<td>Classification Rate of Multi-Level and Single-Level</td>
<td>82</td>
</tr>
<tr>
<td>5.6</td>
<td>False Alarm Comparison</td>
<td>82</td>
</tr>
<tr>
<td>5.7</td>
<td>Dataset for training and testing</td>
<td>84</td>
</tr>
<tr>
<td>5.8</td>
<td>New Attacks used for testing</td>
<td>84</td>
</tr>
<tr>
<td>5.9</td>
<td>Correct Classification Rate for Level 1</td>
<td>90</td>
</tr>
<tr>
<td>5.10</td>
<td>Detection rate &amp; False alarm rate for level 1</td>
<td>91</td>
</tr>
<tr>
<td>5.11</td>
<td>Correct classification rate for level 2</td>
<td>92</td>
</tr>
<tr>
<td>5.12</td>
<td>DOS attacks Classification Rate</td>
<td>93</td>
</tr>
<tr>
<td>5.13</td>
<td>Probe attacks Classification Rate</td>
<td>93</td>
</tr>
<tr>
<td>5.14</td>
<td>R2L attacks Classification Rate</td>
<td>94</td>
</tr>
<tr>
<td>5.15</td>
<td>U2R attacks Classification Rate</td>
<td>94</td>
</tr>
</tbody>
</table>
Security of network system is becoming increasingly important as more sensitive information is being stored and manipulated online. It is difficult to prevent attacks only by passive security policies, firewall, or other mechanisms. Intrusion Detection Systems (IDS) have thus become a critical technology to help protect these systems as an active way. An IDS can collect system and network activity data, and analyze those gathered information to determine whether there is an attack [1].

Along with the conventionally used security tools like firewalls, intrusion detection systems (IDS) are becoming of supreme significance. Intrusion Detections Systems (IDS) is a new path of security systems, which provides efficient approaches to secure computer networks. Artificial Intelligence approaches have been used enormously to produce a lot of IDS. Some of these approaches rely on Neural Network to provide the network with an efficient classifier to recognize and detect intrusions actions.

The main objective of this work is to design and develop security architecture (an intrusion detection and prevention system) for computer networks. This proposed system should be positioned at the network server to monitor all passing data packets and determine suspicious connections. Therefore, it can inform the system administrator with the suspicious attack type. Moreover, the proposed system is adaptive by allowing new attack types to be defined.

We build the model to improve the detection rate for known and unknown attacks. First, we train and test our hybrid model on the normal and the known intrusion data. Then we test our system for unknown attacks by introducing new types of attacks that are never seen by the training module.
1.1 What’s Intrusion?

When a user of an information system takes an action that user was not legally allowed to take, it is called intrusion. The intruder may come from outside, or the intruder may be an insider, who exceeds his limited authority to take action. Whether or not the action is detrimental, it is of concern because it might be detrimental to the health of the system, or to the service provided by the system [2].

Intrusion detection involves determining that some entity, an intruder, has attempted to gain, or worse, has gained unauthorized access to the system. None of the automated detection approaches of which we are aware seeks to identify an intruder before that intruder initiates interaction with the system. Of course, system administrators routinely take actions to prevent intrusion. These can include requiring passwords to be submitted before a user can gain any access to the system, fixing known vulnerabilities that an intruder might try to exploit in order to gain unauthorized access, blocking some or all network access, as well as restricting physical access. Intrusion detection systems are used in addition to such preventative measures [2].

1.2 Firewall Evasion

Many people consider firewalls "the best solution for ensuring network security". However, although these solutions are rather efficient, they do not provide reliable protection against all types of attacks. Firewall system neither detects and nor locks attacks. A firewall is a device that at first prohibits everything, and then permits only those things that must be determined by the administrator. In other words, when the firewall is installed, all connections between the network being
protected and the external network are prohibited. After that, the administrator adds specific rules that enable specified traffic to pass through the firewall. A typical firewall configuration prohibits all incoming ICMP traffic, leaving only outgoing traffic enabled, along with some types of incoming traffic based on the UDP and TCP protocols (such as HTTP, DNS, SMTP, etc.). This configuration will allow employees to access the Internet and deny intruders access to internal resources of the network. Firewalls can not state for sure if an attack is present in the traffic. They can only inform the administrator whether or not the traffic satisfies the requirements established by the specific rules [3].

Firewalls prevent certain kinds of attacks, they protect communications between networks (typically internal network and the Internet), and offer no or little protection from attacks or misuse within local network. If network perimeter is breached, or if the misuse is internal to organization, a Firewall offers no help [3].

This fact is best explained by the following analogy. Consider the firewall to be a "fence" around your network, which simply limits access to specific points behind it, but can not detect if someone is trying to dig a tunnel under it [3].
1.3 Classification of Intrusion Detection Systems

Intrusion detection systems attempt to detect attacks against computer systems and networks. Traditionally, intrusion detection systems can be classified using two approaches, namely, data source and detection approach [4].

1.3.1 Classification Based on Data Source

Based on the source of input data, an Intrusion Detection System (IDS) can be classified as follows [4]:

1. **Host-based IDS:** Deployed on a host computer, the IDS monitors only the activity of that particular host. Information such as operating system audit trails, registry entries and file accesses is used to detect an intrusion.

2. **Network-based IDS:** It resides on a separate system that watches network traffic, looking for indications of attacks that traverse that portion of the network. It passively monitors the network activities of a particular host or a network of hosts.
3. **Hybrid/Hierarchical IDS:** A Hybrid/Hierarchical IDS combines the advantages of host-based and network-based IDSs. Improved intrusion detection capability is achieved through analysis of both host and network data.

![Diagram showing two types of IDS in a network environment](image)

**Figure 1.2** Shows how the two types of IDS may exist in a network environment.

The distinction is useful because network-based intrusion detection tools usually process completely different data sets and features than host-based intrusion detection. As a result, the types of attacks that are detected with network-based intrusion detection tools are usually different than host-based intrusion detection tools. Some attacks can be detected by both network-based and host-based IDSs, however, the "sweet spots", or the types of attacks each is best at detecting, are usually distinct. As a result, it is difficult to make direct comparisons between the performance of a network-based IDS and a host-based IDS. A useful corollary of distinct sweetspots, though, is that in combination both techniques are more powerful than either one by itself [5]. Table 1 shows some of the differences between a HIDS and a NIDS [6].
### Table 1.1: Network-Based vs. Host-Based Intrusion-Detection Systems

<table>
<thead>
<tr>
<th></th>
<th>HIDS</th>
<th>NIDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope</td>
<td>Broad in scope (watches all network activities)</td>
<td>Narrow in scope (watches only specific host activities)</td>
</tr>
<tr>
<td>Setup</td>
<td>Easier setup</td>
<td>More complex setup</td>
</tr>
<tr>
<td>Detection</td>
<td>Better for detecting attacks from the outside</td>
<td>Better for detecting attacks from the inside</td>
</tr>
<tr>
<td>Cost</td>
<td>Less expensive to implement</td>
<td>More expensive to implement</td>
</tr>
<tr>
<td>Detection Method</td>
<td>Detection is based on what can be recorded on the entire network</td>
<td>Detection is based on what any single host can record</td>
</tr>
<tr>
<td>Packet Analysis</td>
<td>Examines packet headers</td>
<td>Does not see packet headers</td>
</tr>
<tr>
<td>Response Time</td>
<td>Near real-time response</td>
<td>Usually only responds after a suspicious log entry has been made</td>
</tr>
<tr>
<td>OS Compatibility</td>
<td>OS-independent</td>
<td>OS-specific</td>
</tr>
<tr>
<td>Attack Detection</td>
<td>Detects network attacks as payload is analyzed</td>
<td>Detects local attacks before they hit the network</td>
</tr>
<tr>
<td></td>
<td>Detects unsuccessful attack attempts</td>
<td>Verifies success or failure of attacks</td>
</tr>
</tbody>
</table>
1.3.2 Classification Based on Detection Approach

Based on the approach used for intrusion detection, an IDS can be classified as follows:

1. **Misuse-based IDS:** Also known as signature-based detection, an IDS based on misuse detection maintains a database of “signatures”, unique network activity patterns, of well-known intrusions. A pattern matching program compares the observed activity pattern against the stored signatures, and raises an alert whenever a match is found. Although a misuse-based IDS detects known intrusions with fair amount of certainty, novel attacks go undetected [4].

   The main advantage of the misuse detection is the ability to alert for attempts of known attacks fast and effectively with a very low rate of false positives. It has drawbacks: firstly in most systems, all new attacks will go unnoticed until the system is updated, creating a window of opportunity for attackers to gain control of the system under attack. Secondly, only known attacks can be detected [7].

2. **Anomaly-based IDS (ABS):** Also known as profile-based detection. It builds statistical models that describe the normal behavior of the network, and flags any behavior that significantly deviates from the norm as an attack [7]. Although an anomaly-based IDS has the advantage of detecting never-before-seen attacks (i.e. few false-negatives), it is highly prone to false positives [4].

   ABS can extract information to detect attacks from different layers: packet headers, packet payload or both. Header information is mainly useful to recognize attacks aiming at vulnerabilities of the network stack implementation or probing the operating system to
identify active network services. On the other hand, payload information is most useful to identify attacks against vulnerable applications (since the connection that carries the attack is established in a normal way) [8].

1.4 Difference between Intrusion Detection System and Intrusion Prevention System

An intrusion detection system (IDS) is software that automates the intrusion detection process while an intrusion prevention system (IPS) is software that has all the capabilities of an intrusion detection system and can also attempt to stop possible incidents [6].

1.5 Thesis Outline

This thesis is organized as follows: Chapter 2 describes the intrusion detection categorization according to detection process and techniques used. It surveys intrusion detection systems using different Neural Network & Decision Tree techniques. Finally it surveys works that used hybrid systems.

Chapter 3 is about Artificial Intelligence and Intrusion Detection. It gives the required theoretical aspects for machine learning techniques used in our study. It gives a brief background for neural network Multilayer Perceptron (MLP), Radial Basis Functions (RBF) and Exhaustive Prune neural networks. It also gives brief background for decision trees approaches (C5, CHAID, CRT and QUEST).

Chapter 4 explains the problem definition and discusses in details the proposed system architecture. We examined four system architectures; Single Level Intrusion Detection System, Multi-Level Intrusion Detection System, Hybrid Multi-Level Intrusion Detection System and Enhanced Hybrid Multi-Level Intrusion Detection System. We compared between Single level
and Multi-Level Neural Network Intrusion Detection System. In this comparison we used two types of attacks DOS (Neptune and Smurf) and Probe (Satan and Portsweep). While Hybrid Multi-Level Intrusion detection system was examined using different machine learning techniques and larger dataset containing the four types of attacks (DOS, Probe, U2R & R2L). This chapter also handles the full description of the NSL-KDD dataset that was used in examining our system.

Chapter 5 analyzes the experimental results of this work. This Chapter overviewed our two experiments together with their results. The first experiment compares between Single Level and Multi-Level Neural Network System. It shows the training process Single Level Neural Network together with its testing results. The Training process of the Multi-Level Neural Network is described together with the testing results. The second experiment defines the values parameters of machine learning techniques used in the hybrid system. Then the discussion of the results of both the hybrid system and its enhancement is explained.

The final conclusion of this work is given in chapter 6, together with the possible future work that may be performed building onto the work of this thesis.
2.1 Introduction

An increasing amount of research has been conducted on for detecting network intrusions. The idea behind the application of soft computing techniques in implementing IDSs is to include an intelligent agent in the system that is capable of disclosing the latent patterns in abnormal and normal connection audit records, and to generalize the patterns to new (and slightly different) connection records of the same class.

There are researches implement an IDS using MLP which have the capability of detecting normal and attacks connection as in [9], [10], [11]. Reference [12] used MLP not only for detecting normal and attacks connection but also identify attack type.

Neural Network was also used for dimension reduction of features as in [13]. The SOM was also applied to perform the clustering of network traffic and to detect attacks in [14] and [15]. In [16], self-organizing maps was used for data clustering and MLP neural networks for detection.

Decision Tree (C4.5 Algorithm) was explored as intrusion detection models in [17] and [18].

Neural network and C4.5 have different classification capabilities for different intrusions. Therefore, Hybrid model improves the performance to detect intrusions. References [1, 19] compare the performance of Hybrid model, single BP network, and single C4.5 algorithm. A multi-classifier model, where a specific detection algorithm is associated with an attack category for which it is the most promising, was built in [20].
The proposed system is a hybrid multi-level system. It consists of three levels. Each level is built using the best classifier which gave best results. It has the ability to identify normal and attack records and also being able to detect attack type by the next levels. This approach has the advantage to flag for suspicious record even if attack type of this record wasn't identified correctly.

Related works which used the DARPA or KDDcup dataset can be categorized in different categories as shown in figure 2.1. One category is the detection process. Some works aimed for detecting whether the coming record is normal or attack without specifying attack type while others aimed at specifying either class category (DoS, Probe, R2L & U2R) or attack type. Another category is according to machine learning techniques used in detection process (Neural Network, Decision tree or hybrid system).

![Intrusion Detection Diagram](image)

**Figure 2.1** Previous works categorization


2.2 Intrusion Detection categorized according to techniques used

2.2.1 Intrusion Detection Systems based on Neural Network

Neural Network was used in intrusion detection. Cannady [9] used MLP to decide whether coming record is attack or normal record.

Ryan et al. [10] proposed a way of applying neural networks to detect intrusions. Instead of trying to optimize the selection of features (commands) for the input, they used a set of 100 most common commands in the logs, and let the network figure out what information was important and what superfluous. In order to introduce more overlap between input vectors, and therefore better generalization, the number of times a command was used was divided into intervals. There were 11 intervals, non-linearly spaced, so that the representation is more accurate at lower frequencies where it is most important. The first interval meant the command was never used; the second that it was used once or twice, and so on until the last interval where the command was used more than 500 times. The intervals were represented by values from 0.0 to 1.0 in 0.1 increments. These values, one for each command, were then concatenated into a 100-dimensional command distribution vector (also called user vector below) to be used as input to the neural network. The standard three-layer backpropagation architecture was chosen for the neural network. The idea was to get results on the most standard and general architecture so that the feasibility of the approach could be demonstrated and the results would be easily replicable. The input layer consisted of 100 units, representing the user vector;
the hidden layer had 30 units and the output layer 10 units, one for each user. The backpropagation neural network was trained in the identification task and tested experimentally on a system of 10 users. The system was 96% accurate in detecting unusual activity, with 7% false alarm rate.

Mukkamala [11] described approaches to intrusion detection and audit data reduction using support vector machines and neural networks. They demonstrate that efficient and highly accurate classifiers can be built using either support vector machines (SVMs) or neural networks for intrusion detection. They presented SVMs and neural networks that use only the (13 of 41) most significant features of the data and deliver only-slightly-lower detection accuracy in the binary attack/normal classification. They also compared the performance of neural networks and SVMs. Classifications were performed on the binary (attack / normal) basis. Both SVMs and neural networks deliver highly-accurate (99% and higher) performance, with SVMs showing slightly better results. Further, when a reduction is performed to reduce the 41 features to the 13 most significant, both SVMs and neural networks again were able to train to deliver accurate results. Even though SVMs are limited to making binary classifications, their superior properties of fast training, scalability and generalization capability give them an advantage in the intrusion detection application.

Moradi and Zulkernine [12] presented a neural network approach to intrusion detection. A Multi Layer Perceptron (MLP) is used for intrusion detection based on an off-line analysis approach. This research aims to solve a multi class problem in which the type of attack is also detected by the neural network. Different neural network structures are
analyzed to find the optimal neural network with regards to the number of hidden layers. An early stopping validation method is also applied in the training phase to increase the generalization capability of the neural network. The results show that the designed system is capable of classifying records with about 91% accuracy with two hidden layers of neurons in the neural network and 87% accuracy with one hidden layer.

Bouzida et al. [13] introduced a method that enables to significantly reduce the information quantity in the different datasets without loss of information before applying some machine learning algorithms. This method is based on Principal Component Analysis (PCA). They deal with, decision trees and nearest neighbor to improve the decision process. The main drawback which persists in combining decision trees or the nearest neighbor with PCA is the poor prediction ratio rate of the R2L class which is in most of the time classified as normal.

Bivens et al. [16] showed that evidence of many of these attacks can be found by a careful analysis of network data. They also illustrate that neural networks can efficiently detect this activity. They tested their systems against denial of service attacks, distributed denial of service attacks, and portscans. They explored network based intrusion detection using classifying, self-organizing maps for data clustering and MLP neural networks for detection. They implemented the system shown in figure 2.2.
They used SOM as a clustering method for MLP neural networks which is an efficient way of creating uniform, grouped input for detection when a dynamic number of inputs are present.

Gang et al. [21] proposed a new approach, called FC-ANN, based on ANN and fuzzy clustering, to solve the problem and help IDS achieve higher detection rate, less false positive rate and stronger stability. The general procedure of FC-ANN is firstly fuzzy clustering technique is used to generate different training subsets. Subsequently, based on different training subsets, different ANN models are trained to formulate different base models. Finally, a meta-learner, fuzzy aggregation module, is employed to aggregate these results. Experimental results on the KDD CUP 1999 dataset showed that their proposed approach, FC-ANN, outperforms Back Propagation Neural Network and other well-known methods such as decision tree, the naïve Bayes especially for low-
frequent attacks, i.e., R2L and U2R attacks in terms of detection precision and detection stability.

Ibrahim [22] introduced a hierarchical off-line anomaly network intrusion detection system based on Distributed Time-Delay Artificial Neural Network. His research aimed to solve a hierarchical multi class problem in which the type of attack (DoS, U2R, R2L and Probe attack) detected by dynamic neural network. The results indicate that dynamic neural nets (Distributed Time-Delay Artificial Neural Network) can achieve a high detection rate, where the overall accuracy classification rate average is equal to 97.24% (DoS (97.6%), U2R (96.2%), R2L (95.8%), Probe (98.2%) from normal one (Normal (98.4%)). Experiments on the KDD99 network intrusion dataset show that DTDNN are best suited due to their high speed and fast conversion rates as compared with other learning techniques and a DTDNN are more powerful than static networks because dynamic networks have memory, they can be trained to learn sequential or time varying patterns, and also show that our approach by using DTDNN obtains superior performance in comparison with other state-of-the-art detection methods.

Tran [23] proposed a Machine Learning algorithm, namely Boosted Subspace Probabilistic Neural Network (BSPNN), which integrates an adaptive boosting technique and a semi-parametric neural network to obtain good trade-off between accuracy and generality. As the result, learning bias and generalization variance was significantly minimized. This research was inspired by the need of a highly performing but low in computation classifier for applications in Network Security. Particularly, the Boosted Subspace Probabilistic Neural Network (BSPNN) is proposed which combines two
emerging algorithms, an adaptive boosting method and a probabilistic neural network to obtain good trade-off between accuracy and generality. **BSPNN** retains the semi parametric characteristics of Vector Quantized-Generalized Regression Neural Network (VQ-GRNN) and therefore obtains low generalization variance while receives accuracy boosting (low bias). Though **BSPNN** requires more processing power due to the effect of boosting, the increased computation is still lower than GRNN or other boosted algorithms. Substantial experiments on KDD-99 intrusion benchmark indicate that their model outperforms other state-of-the-art learning algorithms, with significantly improved detection accuracy, minimal false alarms and relatively small computational complexity.

Anyanwu et al. [24] proposes a scalable application-based model for detecting attacks in a communication network using recurrent neural network architecture. Its suitability for online real-time applications and its ability to self-adjust to changes in its input environment cannot be over-emphasized.

Ahmad et al. [25] compared different neural networks (NNs). They evaluated different neural networks such as Self-organizing map (SOM), Adaptive Resonance Theory (ART), Online Backpropagation (OBPROP), Resilient Backpropagation (RPROP) and Support Vector Machine (SVM) towards intrusion detection mechanisms using Multi-criteria Decision Making (MCDM) technique. The results indicate that in terms of performance, supervised NNs are better, while unsupervised NNs are better regarding training overhead and aptitude towards handling varied and coordinated intrusion. Consequently, the combined, that is, hybrid approach
of NNs is the optimal solution in the area of intrusion detection.

The evaluation is based on two types of criteria, that is, the main criteria and sub criteria. The main criteria consists of adaptable, minimum training, performance, maturity and aptitude, on the other hand, the sub criteria consists of detection rate, minimum false positive, minimum false negative, cost, time, handling coordinated and varied intrusion. They concluded that the combined (hybrid) approach using artificial neural network is a more suitable tactic among other approaches to tackle the present issues of intrusion detection systems such as regular updating, detection rate, false positive, false negative, and flexibility [25].

2.2.2 Intrusion Detection Systems based on Decision Tree

Decision Trees have been applied to the field of intrusion detection for more than a decade. Machine Learning techniques can learn normal and anomalous patterns from training data and generate classifiers that then are used to detect attacks on computer systems.

Stein et al. [26] used a genetic algorithm to select a subset of input features for decision tree classifiers, with a goal of increasing the detection rate and decreasing the false alarm rate in network intrusion detection. The experiments showed that the resulting decision trees can have better performance than those built with all available features.

Wang et al. [27] used C4.5 decision tree classification method to build an effective decision tree for intrusion detection to judge whether the new network behavior is normal
or abnormal. Experiments show that: the detection accuracy rate of intrusion detection algorithm based on C4.5 decision tree is over 90%.

Farid et al. [17] used decision tree algorithm to distinguish attacks from normal behaviors and identified different types of intrusions. Experimental results on the KDD99 benchmark network intrusion detection dataset demonstrate that the proposed learning algorithm achieved 98% detection rate (DR) in comparison with other existing methods.

Rajeswari and Arputharaj [18] proposed an active rule based enhancement to the C4.5 algorithm for network intrusion detection in order to detect misuse behaviors of internal attackers through effective classification and decision making in computer networks. This enhanced C4.5 algorithm derived a set of classification rules from network audit data and then the generated rules are used to detect network intrusions in a real-time environment. The main advantage of this proposed algorithm is that the generalization ability of enhanced C4.5 decision trees is better than that of C4.5 decision trees. They employed data from the third international knowledge discovery and machine learning tools competition (KDDcup’99) to train and test the feasibility of this proposed model. By applying the enhanced C4.5 algorithm an average detection rate of 93.28 percent and a false positive rate of 0.7 percent have respectively been obtained in this work, and it has a good performance in detection of unknown attacks, especially for PROBE, DOS and U2R attacks.

Chebrolu et al. [28] aimed to identify important input features in building an IDS that is computationally efficient and effective. They investigated the performance of two feature
Related Works

selection algorithms involving Bayesian networks (BN) and Classification and Regression Trees (CART) and an ensemble of BN and CART. Empirical results indicate that significant input feature selection is important to design an IDS that is lightweight, efficient and effective for real world detection systems. They proposed hybrid architecture for combining different feature selection algorithms for real world intrusion detection.

Syurahbil et al. [29] proposed a method to find intrusion characteristic for IDS using decision tree machine learning of machine learning technique. Method used to generate of rules is classification by ID3 algorithm of decision tree. Combination of IDS and firewall so-called the IPS, so that besides detecting the existence of intrusion also can execute by doing deny of intrusion as prevention.

In [30] Mulay et al. proposed the multiclass SVM algorithm for implementation of Intrusion Detection System. The integration of Decision tree model and SVM model gives better results than the individual models. Support Vector Machines (SVM) are classifiers which were originally designed for binary classification. The classification applications can solve multi-class problems. Decision-tree-based support vector machine which combines support vector machines and decision tree can be an effective way for solving multi-class problems. This method can decrease the training and testing time, increasing the efficiency of the system. The different ways to construct the binary trees divides the data set into two subsets from root to the leaf until every subset consists of only one class. The construction order of binary tree has great influence on the classification performance. In this paper they studied an
Related Works

algorithm, Tree structured multiclass SVM, which has been used for classifying data.

Gandhi et al. [31] evaluated the performance of a set of classifier algorithms of rules (JRIP, Decision Table, PART, and OneR) and trees (J48, RandomForest, REPTree, NBTree). Based on the evaluation results, best algorithms for each attack category is chosen and two classifier algorithm selection models are proposed. The empirical simulation result shows the comparison between the noticeable performance improvements. The classification models were trained using the data collected from Knowledge Discovery Databases (KDD) for Intrusion Detection. The results indicate that the C4.5 decision tree Classifier trees J48 outperforms in prediction than Rules. PART classifier, the Computational Performance differs significantly. As the nature of the application demands more accurate prediction than the learning time, it is suggested that the C4.5 the Decision Tree Classifier may be practically used by the Network Security Professional or the Administrators to assess the risk of the attacks.

Abbes et al. [32] proposed a combination of pattern matching and protocol analysis approaches. While the first method of detection relies on a multi-pattern matching strategy, the second one benefits from an efficient decision tree adaptive to the network traffic characteristics. Most intrusion detection systems rely on pattern matching operations to look for attack signatures. Nevertheless, the number of signatures is constantly growing following the increasing types and varied forms of attacks. The pattern matching becomes a computational expensive task. They showed that an arbitrary definition of the domain search for signatures causes the generation of false negatives. To overcome this problem, they rely on a protocol
analysis approach that leads to the construction of decision trees in the initial phase of the IDS deployment. The built tree is adaptive to the network traffic characteristics since the features chosen to split the tree ensure the highest reduction of entropy or the lowest Gini impurity. In addition, pattern matching operations are now integrated inside decision tree. They are triggered after achieving light verifications and benefit from a refined domain search of signatures.

2.2.3 Hybrid Intrusion Detection System

Neural network and C4.5 have different classification capabilities for different intrusions. Therefore, Hybrid model improves the performance to detect intrusions. Experimental results demonstrate that while neural networks are highly successful in detecting known attacks, decision trees are more interesting to detect new attacks. While the neural networks are very interesting for generalization and very poor for new attacks attack detection, the decision trees have proven their efficiency in both generalization and new attacks detection.

Pan et al. [1] presented an intrusion detection model based on hybrid neural network and C4.5. The key idea is to take advantage of different classification abilities of neural network and the C4.5 algorithm for different attacks. What is more, the model could also be updated by the C4.5 rules mined from the dataset after the event (intrusion). Neural network have high performance to DOS and Probing attacks rather than to R2L and U2R attacks. On the contrast, C4.5 can detect the R2L and U2R more accurately than neural network. Their model achieved more than 85 percent detection rate on average, and less than 19.7 percent false alarm rate for five typical types of attacks. Through the analysis after-the-event module, the
average detection rate of 93.28 percent and false positive rate of 0.2 percent can respectively be obtained.

Peddabachigari et al. [33] presented two hybrid approaches for modeling IDS. Decision trees (DT) and support vector machines (SVM) are combined as a hierarchical hybrid intelligent system model (DT–SVM) and an ensemble approach combining the base classifiers. The hybrid intrusion detection model combines the individual base classifiers and other hybrid machine learning paradigms to maximize detection accuracy and minimize computational complexity. They explored DT and SVM as intrusion detection models. Then they designed a hybrid DT–SVM model and an ensemble approach with DT, SVM and DT–SVM models as base classifiers. Empirical results reveal that DT gives better or equal accuracy for Normal, Probe, U2R and R2L classes. The hybrid DT–SVM approach improves or delivers equal performance for all the classes when compared to a direct SVM approach. The Ensemble approach gave the best performance for Probe and R2L classes. The ensemble approach gave 100% accuracy for Probe class.

Depren et al. [34] proposed a novel Intrusion Detection System (IDS) architecture utilizing both anomaly and misuse detection approaches. This hybrid Intrusion Detection System architecture consists of an anomaly detection module, a misuse detection module and a decision support system combining the results of these two detection modules. The proposed anomaly detection module uses a Self-Organizing Map (SOM) structure to model normal behavior. Deviation from the normal behavior is classified as an attack. The proposed misuse detection module uses J.48 decision tree algorithm to classify various types of attacks. A rule-based Decision Support System (DSS)
is also developed for interpreting the results of both anomaly and misuse detection modules. Simulation results of both anomaly and misuse detection modules based on the KDD 99 Data Set are given. It is observed that the proposed hybrid approach gives better performance over individual approaches.

Farid et al. [35] used a learning algorithm for adaptive network intrusion detection using naive Bayesian classifier and decision tree, which performed balance detections and kept false positives at acceptable level for different types of network attacks, and eliminated redundant attributes as well as contradictory examples from training data that make the detection model complex. The proposed algorithm also addressed some difficulties of machine learning such as handling continuous attribute, dealing with missing attribute values, and reducing noise in training data. Due to the large volumes of security audit data as well as the complex and dynamic properties of intrusion behaviors, several machine learning based intrusion detection techniques have been applied to network-based traffic data and host-based data in the last decades. The experimental results prove that the proposed hybrid algorithm achieved high detection rates (DR) and significant reduce false positives (FP) for different types of network intrusions using limited computational resources.

Sabhnani and Serpen [20] evaluated performance of a comprehensive set of pattern recognition and machine learning algorithms on four attack categories as found in the KDD 1999 Cup intrusion detection dataset. Results of simulation study implemented to that effect indicated that certain classification algorithms perform better for certain attack categories: a specific algorithm specialized for a given attack category. Consequently, a multi-classifier model that was built using
The proposed multi-expert classifier showed improvement in detection and false alarm rates for all attack categories as compared to the KDD 1999 Cup winner. Furthermore, reduction in cost per test example was also achieved using the multi-classifier model. However, none of the machine learning classifier algorithms evaluated was able to perform detection of user-to-root and remote-to-local attack categories significantly (no more than 30% detection for U2R and 10% for remote-to-local category). In conclusion, it is reasonable to assert that machine learning algorithms employed as classifiers for the KDD 1999 Cup data set do not offer much promise for detecting U2R and R2L attacks within the misuse detection context.

Table 2.1 summarizes the previous intrusion detection systems.
Table 2.1 Summarizes the previous work of Intrusion Detection Systems

<table>
<thead>
<tr>
<th>Author</th>
<th>Year</th>
<th>Technique</th>
<th>Description</th>
<th>Result</th>
<th>Identifies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ryan et al. [10]</td>
<td>1997</td>
<td>MLP</td>
<td>Proposed a way of applying neural networks to detect intrusions.</td>
<td>Detection Rate 96%, false alarm 7%.</td>
<td>Normal / Attack</td>
</tr>
<tr>
<td>Bivens et al. [16]</td>
<td>2002</td>
<td>SOM for clustering &amp; MLP for classification</td>
<td>Used SOM as a clustering method for MLP neural networks for detection.</td>
<td>Detection rate 62%</td>
<td>3 attacks</td>
</tr>
<tr>
<td>Pan et al [1]</td>
<td>2003</td>
<td>hybrid neural network and C4.5</td>
<td>Presented an intrusion detection model based on hybrid neural network and C4.5</td>
<td>Detection rate 93.28% and false alarm 0.2%</td>
<td>Attack type</td>
</tr>
<tr>
<td>Authors</td>
<td>Year</td>
<td>Methodologies</td>
<td>Performance</td>
<td>Classes</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>------</td>
<td>-------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Sabhnani and Serpen</td>
<td>2003</td>
<td>MLP for probing, K-Means for DoS &amp; U2R, and GAU for R2L</td>
<td>Evaluated performance of machine learning algorithms on four attack categories</td>
<td>Probe 88.7%, Dos 97.3%, U2R 29.8%, R2L 9.6%</td>
<td></td>
</tr>
<tr>
<td>Moradi and Zulkernine</td>
<td>2004</td>
<td>MLP</td>
<td>Aimed to detect type of attack by the neural network.</td>
<td>Detection rate 91%</td>
<td></td>
</tr>
<tr>
<td>Bouzida et al.</td>
<td>2004</td>
<td>PCA for information reduction &amp; DT - Nearest Neighbor for classification</td>
<td>Reduce the information quantity in the different datasets without loss of information</td>
<td>Detection rate 92.63%</td>
<td></td>
</tr>
<tr>
<td>Stein et al.</td>
<td>2005</td>
<td>GA for feature selection, DT for classification</td>
<td>Used a genetic algorithm to select a subset of input features for decision tree classifiers</td>
<td>Normal / Attack</td>
<td></td>
</tr>
<tr>
<td>Authors</td>
<td>Year</td>
<td>Methodology</td>
<td>Description</td>
<td>Detection Rate</td>
<td>False alarm</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Depren et al. [34]</td>
<td>2005</td>
<td>SOM for anomaly detection, J.48 for misuse detection module</td>
<td>Hybrid Intrusion Detection System consisting of anomaly detection module, misuse detection module and a decision support system combining the results of these two detection modules.</td>
<td>99.48%, 1.25%</td>
<td></td>
</tr>
<tr>
<td>Chebrolu et al [28]</td>
<td>2005</td>
<td>Ensemble of BN and CART</td>
<td>Aimed to identify important input features using Bayesian networks and Classification and Regression Trees.</td>
<td>Normal, Probe and DOS 100%, U2R 84%, and R2L 99.47%</td>
<td></td>
</tr>
<tr>
<td>Rajeswari and Arputharaj [18]</td>
<td>2008</td>
<td>Enhanced C4.5</td>
<td>Proposed an Enhanced C4.5 algorithm for intrusion detection.</td>
<td>93.28%, 0.7%</td>
<td></td>
</tr>
<tr>
<td>Authors</td>
<td>Year</td>
<td>Algorithm</td>
<td>Methodology</td>
<td>Detection Rate</td>
<td>False Alarm</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Wang et al. [27]</td>
<td>2009</td>
<td>C4.5</td>
<td>build an effective decision tree for intrusion detection to judge whether the new network behavior is normal or abnormal</td>
<td>Detection Rate 90%</td>
<td></td>
</tr>
<tr>
<td>Syurahbil et al [29]</td>
<td>2009</td>
<td>ID3</td>
<td>Combination of IDS and firewall (IPS) using decision tree</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tran [23]</td>
<td>2009</td>
<td>Boosted Subspace Probabilistic NN</td>
<td>integrates an adaptive boosting technique and a semi-parametric neural network to obtain good trade-off between accuracy and generality</td>
<td>Detection Rate 94.31%, False Alarm 1.12%</td>
<td></td>
</tr>
<tr>
<td>Authors</td>
<td>Year</td>
<td>Neural Networks</td>
<td>Evaluation</td>
<td>Detection Rate</td>
<td>Classes</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
<td>-----------------</td>
<td>------------</td>
<td>----------------</td>
<td>---------</td>
</tr>
<tr>
<td>Ahmad et al. [25]</td>
<td>2010</td>
<td>SOM, ART, SVM, OBPROP and RPROP</td>
<td>evaluation of different neural networks such as Self-organizing map (SOM), Adaptive Resonance Theory (ART), Online Backpropagation (OBPROP), Resilient Backpropagation (RPROP) and Support Vector Machine (SVM) towards intrusion detection mechanisms</td>
<td>Supervised NNs are better, while unsupervised NNs are better regarding training overhead and handling varied and coordinated intrusion.</td>
<td>Normal / Attack</td>
</tr>
<tr>
<td>Gang et al. [21]</td>
<td>2010</td>
<td>ANN and fuzzy clustering</td>
<td>Fuzzy clustering technique is used for clustering. ANN models are used for classification.</td>
<td>Detection Rate 96.71</td>
<td>Classes</td>
</tr>
<tr>
<td>Authors</td>
<td>Year</td>
<td>Methodology</td>
<td>Approach Description</td>
<td>Detection Rate</td>
<td>Classes</td>
</tr>
<tr>
<td>--------------------</td>
<td>------</td>
<td>----------------------------------</td>
<td>---------------------------------------------------------------------------------------</td>
<td>----------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Farid et al. [35]</td>
<td>2010</td>
<td>naive Bayesian classifier and decision tree</td>
<td>Used naive Bayesian classifier and decision tree for adaptive network intrusion detection.</td>
<td>99%</td>
<td></td>
</tr>
<tr>
<td>Farid et al [17]</td>
<td>2010</td>
<td>ID3 &amp; C4.5</td>
<td>Used decision tree algorithm to identify different types of intrusions.</td>
<td>98%</td>
<td></td>
</tr>
<tr>
<td>Ibrahim [22]</td>
<td>2010</td>
<td>MLP</td>
<td>Aimed to solve a hierarchical multi class problem in which the type of attack is defined.</td>
<td>97.24% (DoS 97.6 %, U2R 96.2%, R2L 95.8%, Probe 98.2%, Normal 98.4%)</td>
<td></td>
</tr>
<tr>
<td>Mulay et al. [30]</td>
<td>2010</td>
<td>SVM &amp; DT</td>
<td>Proposed the multiclass SVM algorithm together with decision tree.</td>
<td>Normal 99.7, Probe 100%, DOS 99.92%, U2R 68%, R2L 97.16%</td>
<td>Attack type</td>
</tr>
<tr>
<td>-------------------</td>
<td>------</td>
<td>----------</td>
<td>---------------------------------------------------------------</td>
<td>--------------------------------------------------</td>
<td>------------</td>
</tr>
</tbody>
</table>

Theoretical Aspects

3.1 Artificial Intelligence and Intrusion Detection

“There is no basic, simple, or agreed upon strict definition of artificial intelligence however as a general definition artificial intelligence is the science and engineering of making intelligent machines, especially intelligent computer programs” [36].

Human’s biological intelligence has inspired system security designers and researchers to build artificial intelligence system which emulates the defense mechanism of Human Immune Systems. Artificial Intelligence systems have been experienced and developed, which relies of the algorithms and intelligent techniques, combining the knowledge of past intrusions in improving the systems detection [36, 37, 38].

Regarding the Intrusion detection, researches use different types of Artificial Intelligence methods and techniques as Neural Network & Decision Trees.

3.2 ANN (Artificial Neural Networks) Approach

ANN learning techniques are mainly divided into supervised or unsupervised according to the learning method used. Supervised method should reach a desired output, if not; the mathematical algorithms built in ANN will perform some adjustments until it reaches the expected output. The unsupervised learning is the opposite way of the former method, i.e. it is given a set of inputs and no correct output. In case of IDS these learning techniques are used to increase the system intelligence in distinguishing between normal and intruder behaviors [39].
An artificial neural network consists of a collection of processing elements that are highly interconnected and transform a set of inputs to a set of desired outputs. The result of the transformation is determined by the characteristics of the elements and the weights associated with the interconnections among them. By modifying the connections between the nodes the network is able to adapt to the desired outputs [41, 42]. Unlike expert systems, which can provide the user with a definitive answer if the characteristics which are reviewed exactly match those which have been coded in the rule-base, a neural network conducts an analysis of the information and provides a probability estimate that the data matches the characteristics which it has been trained to recognize. While the probability of a match determined by a neural network can be 100%, the accuracy of its decisions relies totally on the experience the system gains in analyzing examples of the stated problem. The neural network gains the experience initially by training the system to correctly identify preselected examples of the problem. The response of the neural network is reviewed and the configuration of the system is refined until the neural network’s analysis of the training data reaches a satisfactory level. In addition to the initial training period, the neural network also gains experience over time as it conducts analyses on data related to the problem [9].
3.2.1 Neural Network and Intrusion Detection

3.2.1.1 Difference between Supervised & Unsupervised Learning

The data source can be labeled or unlabelled based on the learning algorithm used in intrusion detection system. Unsupervised algorithms can be applied to unlabelled data while supervised algorithms can only use labeled data.

In supervised learning, the training data must be labeled before they are presented to the training algorithm. Figure 3.1 shows the intrusion detection process using supervised learning algorithm. First, the original data must be analyzed and labeled as normal connections or attacks by human experts. After that, the learning algorithms generalize the rules from the training data. Finally, the classifier uses the generated rules to classify the new network connections. A difficulty of the supervised learning is labeling the data. If a large data set is used for training, the labeling duty could be very hard. If choosing a small portion as the training data, the selection of the training examples is crucial to the learning result [42].

![Network intrusion detection using labeled data](image)

**Figure 3.1** Network intrusion detection using labeled data
Unlike supervised learning algorithms, which can only use labeled data, unsupervised learning algorithms have the ability to learn from unlabelled data. The detection process using the unlabelled data is illustrated in Figure 3.2. First, the training data are clustered by the clustering algorithm. Second, the clustered weight vectors can be labeled by a labeling process. Various methods can be applied to this process. One approach to label a cluster center is to select a sample group of the data from this cluster randomly and label this cluster with the major type of the sample. Finally, the labeled weight vectors can be used to classify the network connections. The main difference between the two processes discussed above is the time and the number of examples for labeling. Unlike the first process, in which the data must be labeled before training, the second process has the ability to organize the unlabelled data and to provide the cluster centers for labeling. Therefore, the second process reduces the risk of selecting improper data as the training set [42].

**Figure 3.2** Network intrusion detection using unlabelled data

The timely and accurate detection of computer and network system intrusions has always been an elusive goal for system
administrators and information security researchers. The individual creativity of attackers, the wide range of computer hardware and operating systems, and the ever-changing nature of the overall threat to target systems have contributed to the difficulty in effectively identifying intrusions. While the complexities of host computers already made intrusion detection a difficult endeavor, the increasing prevalence of distributed network-based systems and insecure networks such as the Internet has greatly increased the need for intrusion detection [43].

3.2.2 Application of Neural Networks in Misuse Detection

While there is an increasing need for a system capable of accurately identifying instances of misuse on a network there is currently no applied alternative to rule-based intrusion detection systems. This method has been demonstrated to be relatively effective if the exact characteristics of the attack are known. However, network intrusions are constantly changing because of individual approaches taken by the attackers and regular changes in the software and hardware of the targeted systems. Because of the infinite variety of attacks and attackers even a dedicated effort to constantly update the rule base of an expert system can never hope to accurately identify the variety of intrusions.

The constantly changing nature of network attacks requires a flexible defensive system that is capable of analyzing the enormous amount of network traffic in a manner which is less structured than rule-based systems. A neural network-based misuse detection system could potentially address many of the problems that are found in rule-based systems [9].
3.2.3 Advantages of Neural Network-based Misuse Detection Systems

The first advantage in the utilization of a neural network in the detection of instances of misuse would be the flexibility that the network would provide. A neural network would be capable of analyzing the data from the network, even if the data is incomplete or distorted. Similarly, the network would possess the ability to conduct an analysis with data in a non-linear fashion. Both of these characteristics are important in a networked environment where the information which is received is subject to the random failings of the system. Further, because some attacks may be conducted against the network in a coordinated assault by multiple attackers, the ability to process data from a number of sources in a non-linear fashion is especially important.

The inherent speed of neural networks is another benefit of this approach. Because the protection of computing resources requires the timely identification of attacks, the processing speed of the neural network could enable intrusion responses to be conducted before irreparable damage occurs to the system.

Because the output of a neural network is expressed in the form of a probability the neural network provides a predictive capability to the detection of instances of misuse. A neural network-based misuse detection system would identify the probability that a particular event, or series of events, was indicative of an attack against the system. As the neural network gains experience it will improve its ability to determine where these events are likely to occur in the attack process. This information could then be used to generate a series of events that should occur if this is in fact an intrusion
Theoretical Aspects

attempt. By tracking the subsequent occurrence of these events the system would be capable of improving the analysis of the events and possibly conducting defensive measures before the attack is successful.

However, the most important advantage of neural networks in misuse detection is the ability of the neural network to "learn" the characteristics of misuse attacks and identify instances that are unlike any which have been observed before by the network. A neural network might be trained to recognize known suspicious events with a high degree of accuracy. While this would be a very valuable ability, since attackers often emulate the "successes" of others, the network would also gain the ability to apply this knowledge to identify instances of attacks which did not match the exact characteristics of previous intrusions [9].

3.2.4 Disadvantages of Neural Network-based Misuse Detection Systems

There appear to be two primary reasons why neural networks have not been applied to the problem of misuse detection in the past. The first reason relates to the training requirements of the neural network. Because the ability of the artificial neural network to identify indications of an intrusion is completely dependent on the accurate training of the system, the training data and the training methods that are used are critical. The training routine requires a very large amount of data to ensure that the results are statistically accurate. The training of a neural network for misuse detection purposes may require thousands of individual attacks sequences, and this quantity of sensitive information is difficult to obtain.
However, the most significant disadvantage of applying neural networks to intrusion detection is the "black box" nature of the neural network. Unlike expert systems which have hard-coded rules for the analysis of events, neural networks adapt their analysis of data in response to the training which is conducted on the network. The connection weights and transfer functions of the various network nodes are usually frozen after the network has achieved an acceptable level of success in the identification of events. While the network analysis is achieving a sufficient probability of success, the basis for this level of accuracy is not often known. The "Black Box Problem" has plagued neural networks in a number of applications [44]. This is an on-going area of neural network research.

3.2.5 Multi-Layer Perceptron (MLP)

MLP are feed-forward neural networks trained with the standard backpropagation algorithm. They are supervised networks so they require a desired response to be trained. They learn how to transform input data into a desired response, so they are widely used for pattern classification. With one or two hidden layers, they can approximate virtually any input–output map. The most popular static network is the MLP [45, 46].

3.2.5.1 Topology of MLP Network

This network distinguishes itself by presence of one or more hidden layer (figure 3.3) whose computation nodes are correspondingly called hidden neurons. The function of the hidden neurons is to intervene between the external input and network output in some useful manner. By this way, the
network is enabled to extract higher-order statistics and solve difficult and diverse problems [45].

![MLP with one hidden layer](image)

**Figure 3.3** MLP with one hidden layer

MLP is characterized by having non-linear differentiable activation function (e.g., sigmoid function) and this nonlinearity is biologically motivated since it attempts to account for the refractory phase of real neurons. MLP networks have been applied successfully to solve difficult and diverse problems, by training them in a supervised manner with highly popular algorithm known as *Error Back-Propagation Algorithm*.

Termination condition usually will be one of the following:
- Terminates when a maximum number of epochs has been exceeded.
- Terminates when the Mean Squared Error (MSE) on the training set is below than a small specified value.
- Terminates when the change in MSE from one iteration to the next is less than a predefined threshold.
3.2.5.2 Training MLP classifier

MLP is trained by using back-propagation algorithm mentioned in the previous section. The training process is terminated when the MSE for the training set is below than a small specified value. So first we need to adjust this small value. Minimizing the MSE does not guarantee increasing in the recognition accuracy. As the MSE decreases, the network becomes more knowledgeable about the training data but when the MSE tends to zero, the network may memorize the training data (exactly fit training data) and provides poor recognition accuracy. This phenomena is called over-training (over-fitting). Therefore, the MSE target value will be adjusted by trail and error and is controlled by the recognition accuracy, as the accuracy increased the MSE will be decreased until the accuracy begins to decrease. Therefore MSE will be adjusted to the value that achieves the best recognition accuracy.

There is no rule for adjusting the MLP parameters: The learning rate, momentum and number of hidden nodes, so they are also adjusted by trail and error according to the behavior of the learning curve (which shows how the mean square error evolves with the training iterations).

1. **Learning rate:** When the learning curve is flat, this means that the learning process is very slow so the learning rate should be increased to speed it up. On the other hand when the learning curve oscillates up and down, this means that the network diverges and becomes unstable due to large changes in its synaptic weights so the learning rate should be decreased. Therefore, the learning rate will be adjusted to a value that allows the learning curve to smoothly converge to the specified MSE value at minimum training time.
2. **Momentum:** when the learning curve continues to oscillate even with decreasing the learning rate, a momentum term is added to control the weights update in each iteration. The momentum value is adjusted to a value near to 1 (ex. 0.9) [45] and decreases until the network is trained stably.

3. **Hidden nodes:** When the learning curve stabilizes after much iteration at an error level that is not acceptable, it is time to increase the number of hidden nodes. Increasing number of hidden nodes is also controlled by recognition accuracy. More hidden nodes means more freedom to the network, so its output can vary more quickly in response to a change in the input and this may lead to over-training phenomena. Therefore, the optimum number of nodes is the minimum number that achieves the best accuracy.

### 3.2.6 Radial Base Functions (RBF)

Radial basis functions were first introduced by Powell to solve the real multivariate interpolation problem [47]. This problem is currently one of the principal fields of research in numerical analysis. In the field of neural networks, radial basis functions were first used by Broomhead and Lowe [48].

The design of a RBFN in its most basic form consists of three separate layers. The input layer is the set of source nodes (sensory units). The second layer is a hidden layer of high dimension. The output layer gives the response of the network to the activation patterns applied to the input layer. The transformation from the input space to the hidden-unit space is *nonlinear*. On the other hand, the transformation from the hidden space to the output space is linear [45].
In RBF an approach is taken by viewing the design of a neural network as a curve-fitting (approximation) problem in a high-dimensional space. According to this viewpoint, learning is equivalent to finding a surface in a multidimensional space that provides a best fit to the training data, with the criterion for "best fit" being measured in some statistical sense [45].

Correspondingly, recognition is equivalent to the use of this multidimensional surface to interpolate the test data. This viewpoint is the real motivation behind the RBF method in the sense that it draws upon research work on traditional strict interpolations in a multidimensional space. In a neural network, the hidden units form a set of “functions” that compose a random “basis” for the input patterns (vectors). These functions are called \textit{radial basis functions} [45]

In the context of a neural network, the hidden units provide a set of "functions" that constitute an arbitrary basis for the input patterns (vectors) when they are expanded in the hidden space; these functions are called radial-basis functions. RBF networks have been successfully applied to a large diversity of applications [49] including interpolation, chaotic time-series modeling, pattern recognition, etc.

RBF networks have a static Gaussian function as the non-linearity for the hidden layer processing elements. The Gaussian function responds only to a small region of the input space where the Gaussian is centered. The key to a successful implementation of these networks is to find suitable centers for the Gaussian functions. This can be done with supervised learning, but an unsupervised approach usually produces better results. The advantage of the radial basis function network is that it finds the input to output map using local approximators.
Usually the supervised segment is simply a linear combination of the approximators. Since linear combiners have few weights, these networks train extremely fast and require fewer training samples [45, 46].

### 3.2.6.1 Topology of RBF Network

The RBF is basically composed of three different layers: the input layer in which the number of nodes is equal to the dimension of input vector. In the hidden layer, the input vector is transformed by a radial basis activation function. For pattern recognition, Gaussian function is preferred [49] as activation function.

### 3.2.6.2 Training of RBF

In order to use RBF network we need to specify the number of hidden units (no. of centers). Fix the weight vectors (centers) of the hidden layer, the spread constant of the Gaussian functions and the weights of the outer layer.

The weights of the outer layer are optimized in order to fit the network desired outputs to the given inputs. The fit is evaluated by means of a cost function, usually assumed to be the mean MSE or SSE (Sum Squared Error). Since the outer layer is linear, these weights can be determined using Least Squares (LS) method [45]. As discussed in section there is no rule for defining the optimum MSE where the network reaches its best recognition performance, minimum MSE does not guarantee the best recognition performance for the network, since over-training phenomena can be occurs and the network loses its ability to generalize. Therefore, the optimum MSE is found by trail and error. The case is the same for the gaussian
spread constant which has to be large enough so that the active input regions of the gaussian neurons overlap enough so that several gaussian neurons always have fairly large outputs at any given moment. This makes the network function smoother and results in better recognition for new input vectors occurring between input vectors used in the design. However, spread should not be so large that each neuron is effectively responding in the same, large, area of the input space and this will lead to poor recognition performance. Therefore we conclude that decreasing the MSE and increasing the spread constant parameters are controlled by the recognition accuracy, when the accuracy begins to decrease the parameters will be fixed to the values that achieve the maximum accuracy.

The performance of the RBF critically depends upon the centers of the hidden layer. So the key question is therefore how to fix the centers. The most commonly used algorithms are as follows:

1. In the initial approaches, to each data sample was assigned a center. This solution proved to be expensive in terms of memory requirement and in the number of parameters. On the other hand, exact fit to the training data (over-training) may cause bad recognition and also in case of large training sets, this method will be rarely practical [49].

2. Centers are often chosen to be a subset of the data. Although researchers are well aware that fixed centers should suitably sample the input domain, most published results simply assume that the centers are arbitrarily selected from data points. Such solution is clearly an unsatisfactory method for building RBF networks. The resulting RBF networks often either perform poorly or have a large size. Furthermore numerical ill-conditioning
frequently occurs owing to the near dependency caused by, for example, some centers being too close [50].

3. Using K-mean algorithm [45, 49] to choose centers that sample the training data. The limitation of this algorithm is that it can only achieve a local optimum solution that depends on the initial choice of cluster centers.

4. Training the hidden layer using unsupervised competitive learning process known as self-organizing map [49]. The centers of the radial (gaussian) functions are initialized randomly and optimum number of centers is found through experiments. The outer layer is trained using back-propagation algorithm. This method also achieve a local optimum solution since it depend on the random initialization of the centers and also the back-propagation algorithm can get stuck into a local minima of the cost function (MSE).

3.2.7 Exhaustive Prune

This method is related to the prune method. It starts with a large network and prunes the weakest units in the hidden and input layers as training proceeds. Exhaustive pruning is similar to stepwise regression, in that only the diversity measures (i.e., independent variables) considered significant in predicting performance are retained in the network when it is trained. With exhaustive prune, network training parameters are chosen to ensure a very thorough search of the space of possible models to find the best one. This method is usually the slowest but yields to good results [51].

Each neuron is pruned at each stage. When pruning neuron i, we remove its connection to the readout before training.
Once trained, the network runs in free-run mode in order to compute the pruning validation error corresponding to pruning neuron \( i \).

The pruning is applied as follows to a dataset of size \( N \):

1. The validation error of neuron \( i \) is computed for \( i = 1, \ldots, N \).
2. The readout connection \( k \) corresponding to the neuron \( k \) with the minimal pruning validation error is pruned.
3. \( N := N - 1 \). If \( N = 1 \), stop; otherwise, go back to step 1.

The algorithm can be stopped earlier, if for instance the error increases significantly with the pruning.

One has to note that this requires to re-train and then re-run (in free-run) the whole network at each step and for each neuron (in order to compute the pruning validation error of each neuron). This algorithm is thus computationally demanding [7].

### 3.3 Decision trees

The decision tree (DT) is very powerful and popular machine learning algorithm for decision-making and classification problems. It has been using in many real life applications like medical diagnosis, radar signal classification, weather prediction, credit approval, and fraud detection etc [52]. The decision tree is a simple if then else rules but it is a very powerful classifier and proved to have a high detection rate. They are used to classify data with common attributes. Each decision tree represents a rule which categorizes data according to these attributes [53].
3.3.1 Decision Trees Approach

Decision tree can be constructed from large volume of dataset with many attributes, because the tree size is independent of the dataset size. A decision tree has three main components: nodes, leaves, and edges. Each node is labeled with a feature attribute which is most informative among the attributes not yet considered in the path from the root, each arc out of a node is labeled with a feature value for the node’s feature and each leaf is labeled with a category or class. Each node has a number of edges, which are labeled according to possible values of the attribute. An edge connects either two nodes or a node and a leaf. Leaves are labeled with a decision value for categorization of the data. Decision trees classifiers are based on the “divide and conquer” strategy to construct an appropriate tree from a given learning set containing a finite and not empty set of labeled instances. To make a decision using a decision Tree, start at the root node and follow the tree down the branches until a leaf node representing the class is reached. Each decision tree represents a rule set, which categorizes data according to the attributes of dataset. The DT building algorithms may initially build the tree and then prune it for more effective classification. With pruning technique, portions of the tree may be removed or combined to reduce the overall size of the tree. The time and space complexity of constructing a decision tree depends on the size of the data set, the number of attributes in the data set, and the shape of the resulting tree. Decision trees are used to classify data with common attributes [54]. The amount of information associated with an attribute value is related to the probability of occurrence. The concept used to quantify information is called entropy, which is used to measure the amount of randomness from a data set. When all data in a set belong to a single class, there is no uncertainty,
and then the entropy is zero. The objective of decision tree classification is to iteratively partition the given data set into subsets where all elements in each final subset belong to the same class.

Besides the construction and classification steps, many decision trees algorithms use another optional step. This step consists in removing some edges that are considered useless for improving the performance of the tree in the classification step. Pruning trees simplifies the tree since many useless edges are removed rendering complex trees more comprehensive for interpretation. In addition, a tree that is already built is pruned only when it gives better classification results than before pruning [52].

The decision tree is constructed during the learning phase, it is then used to predict the classes of new instances. Most of the decision trees algorithms use a top down strategy; i.e from the root to the leaves. Two main processes are necessary to use the decision tree: the building process and the classification process [54].

### 3.3.2 C5.0 Decision Trees

C5.0 is an extension of C4.5 which is one of the most popular inductive learning tools originally proposed by J.R.Quinlan [53]. C4.5 is a classic decision tree algorithm which is an extension of ID3 attribute-based machine learning system (Quinlan, 1993). The ID3 algorithm operation involves inducing classification rules for a given set of objects that are described by fixed set of attributes. The set of objects is known as a training set. It is assumed that it is known whether each object in the training set is a positive instance (P) or a negative instance (N). The system then applies an algorithm to induce a
rule which would correctly classify the objects in the training set. ID3 accounts for unavailable values, continuous attribute value ranges, pruning of decision trees and rule derivation. Reduced-error pruning examines trees that have already been created by running a decision tree algorithm on a set of training instances, makes adjustments to the tree and then tests them on validation instances. The algorithm keeps changes only if the pruned tree performs better than the original tree.

To develop a training set, the algorithm can use part of the original instance set to create a decision tree and then use the rest as a validation set. The use of a validation set helps to eliminate over fitting data by dealing with inconsistencies and coincidences in the training set. Reduced-error pruning involves selecting internal nodes, re-labeling them with a classification and then discarding their descendent nodes. Rule post-pruning changes the paths to leaf-nodes along the decision tree to logical rules. Once these rules have been created, their antecedents are refined in an attempt to limit over fitting. The algorithm classifies instances by applying the rules in a sorted order (sorted by estimated accuracy).

C5.0 handles continuous-valued functions by dividing them into a set of discrete valued functions. This can be repeated at each step of the algorithm to make the divisions that yield the largest information gain.

C5.0 can deal with missing attributes in several ways. One is to give the missing attribute the value that is most common for other instances at the same node. Or, the algorithm could make probabilistic calculations based on other instances to assign the value. C5.0 does not solely select which attribute to test on predicted information gain.
Basing attribute selection on the gain ratio avoids selecting attributes that classify entities arbitrarily.

C5.0 supports boosting of decision trees. Boosting is a technique for generating and combining multiple classifiers to give improved predictive accuracy. By this process error rate is reduced on some datasets. C5.0 incorporates variable misclassification costs.

Algorithm allows a separate cost to be defined for each predicted/actual class pair; if this option is used, C5.0 then constructs classifiers to minimize expected misclassification costs rather than error rates.

Automatic winnowing of attributes can be used in C5.0, discarding those that appear to be only marginally relevant before a classifier is constructed. Winnowing can lead to smaller classifiers and higher predictive accuracy.

3.3.3 Classification and Regression Trees (CRT or CART)

CART is a recursive partitioning method to be used both for regression and classification. The key elements of CART analysis are a set of rules for splitting each node in a tree; deciding when tree is complete and assigning a class outcome to each terminal node. CART is constructed by splitting subsets of the data set using all predictor variables to create two child nodes repeatedly, beginning with the entire data set [55].

CART splits the data into segments that are as homogeneous as possible with respect to the dependent variable. A terminal node in which all cases have the same value for the dependent variable is a homogeneous, "pure" node.
It is characterized by the fact that it constructs binary trees, namely each internal node has exactly two outgoing edges. The splits are selected using the twoing criteria and the obtained tree is pruned by cost-complexity Pruning. When provided, CART can consider misclassification costs in the tree induction. It also enables users to provide prior probability distribution. An important feature of CART is its ability to generate regression trees. Regression trees are trees where their leaves predict a real number and not a class. In case of regression, CART looks for splits that minimize the prediction squared error (the least-squared deviation). The prediction in each leaf is based on the weighted mean for node.

3.3.4 Chi-squared Automatic Interaction Detector (CHAID)

Starting from the early seventies, researchers in applied statistics developed procedures for generating decision trees. CHAID (Chi-square-Automatic-Interaction-Detection) was originally designed to handle nominal attributes only.

CHAID method is based on the chi-square test of association. A CHAID tree is a decision tree that is constructed by repeatedly splitting subsets of the space into two or more child nodes, beginning with the entire data set [56]. To determine the best split at any node, any allowable pair of categories of the predictor variables is merged until there is no statistically significant difference within the pair with respect to the target variable. This CHIAD method naturally deals with interactions between the independent variables that are directly available from an examination of the tree. The final nodes identify subgroups defined by different sets of independent variables [57].
For each input attribute $a_i$, CHAID finds the pair of values in $V_i$ that is least significantly different with respect to the target attribute. The significant difference is measured by the p value obtained from a statistical test. The statistical test used depends on the type of target attribute. If the target attribute is continuous, an F test is used. If it is nominal, then a Pearson chi-squared test is used. If it is ordinal, then a likelihood-ratio test is used. For each selected pair, CHAID checks if the p value obtained is greater than a certain merge threshold. If the answer is positive, it merges the values and searches for an additional potential pair to be merged. The process is repeated until no significant pairs are found. The best input attribute to be used for splitting the current node is then selected, such that each child node is made of a group of homogeneous values of the selected attribute. Note that no split is performed if the adjusted p value of the best input attribute is not less than a certain split threshold. This procedure also stops when one of the following conditions is fulfilled:

1. Maximum tree depth is reached.
2. Minimum number of cases in node for being a parent is reached, so it can not be split any further.
3. Minimum number of cases in node for being a child node is reached.

CHAID handles missing values by treating them all as a single valid category. CHAD does not perform pruning.

ECHAID is a modified version of CHAID, which exhaustively computes classification accuracies for all of possible combinations of tree architectures and then picks up the best one. In general, an ECHAID method obtains better classification result but spends more computational time than a CHAID method.
3.3.5 Quick, Unbiased, Efficient Statistical Tree (QUEST)

QUEST is a binary-split decision tree algorithm for classification and machine learning. QUEST can be used with univariate or linear combination splits. A unique feature is that its attribute selection method has negligible bias. If all the attributes are uninformative with respect to the class attribute, then each has approximately the same change of being selected to split a node [58].

For each split, the association between each input attribute and the target attribute is computed using the ANOVA F-test or Levene's test (for ordinal and continuous attributes) or Pearson's chi-square (for nominal attributes). If the target attribute is multinomial, two-means clustering is used to create two super classes. The attribute that obtains the highest association with the target attribute is selected for splitting. Quadratic Discriminant Analysis (QDA) is applied to find the optimal splitting point for the input attribute. QUEST has negligible bias and it yields binary decision trees. Ten-fold cross-validation is used to prune the trees.
3.4 Selection of Neural Network & Decision Tree Classifier among Other classification techniques

3.4.1 Neural Network Classifiers

An artificial neural network is a system based on the operation of biological neural networks, in other words, is an emulation of biological neural system. Although computing these days is truly advanced, there are certain tasks that a program made for a common microprocessor is unable to perform; even so a software implementation of a neural network can be made with their advantages and disadvantages.

Advantages:
1. A neural network can perform tasks that a linear program can not.
2. When an element of the neural network fails, it can continue without any problem by their parallel nature.
3. A neural network learns and does not need to be reprogrammed.
4. It can be implemented in any application.
5. It can be implemented without any problem.

Disadvantages:
1. The neural network needs training to operate.
2. The architecture of a neural network is different from the architecture of microprocessors therefore needs to be emulated.
3. Requires high processing time for large neural networks.
4. The "black box" nature of the neural network.
3.4.2 Decision Trees classifiers

Decision trees Classifier chosen for building our classifier that it deals with both integer and real numbers. The decision tree is a simple if then else rules that is easy to be implemented. It is a very powerful classifier and proved to have a high detection rate as will be shown in Experimental Results section.

Advantages

1. Decision trees are self-explanatory and when compacted they are also easy to follow. In other words if the decision tree has a reasonable number of leaves, it can be grasped by non-professional users. Furthermore decision trees can be converted to a set of rules. Thus, this representation is considered as comprehensible.
2. Decision trees can handle both nominal and numeric input attributes.
3. Decision tree representation is rich enough to represent any discrete value classifier.
4. Decision trees are capable of handling datasets that may have errors.
5. Decision trees are capable of handling datasets that may have missing values.
6. Decision trees are considered to be a nonparametric method. This means that decision trees have no assumptions about the space distribution and the classifier structure. [59]
Theoretical Aspects

Disadvantages

1. Most of the algorithms (like ID3 and C4.5) require that the target attribute will have only discrete values.

2. As decision trees use the "divide and conquer" method, they tend to perform well if a few highly relevant attributes exist, but less so if many complex interactions are present. One of the reasons for this is that other classifiers can compactly describe a classifier that would be very challenging to represent using a decision tree. A simple illustration of this phenomenon is the replication problem of decision trees [60]. Since most decision trees divide the instance space into mutually exclusive regions to represent a concept, in some cases the tree should contain several duplications of the same sub-tree in order to represent the classifier.

3. The greedy characteristic of decision trees leads to another disadvantage that should be pointed out. This is its over-sensitivity to the training set, to irrelevant attributes and to noise [53].
4.1 Introduction

Timely detection of computer system intrusion is a problem that is receiving increasing attention. While firewalls and access control are being used to keep intruder from breaking into a system, 80% of intrusions or attacks are from within an organization (Cerias website, 2001). Compare with outside intruders, these insiders know the system structure and where to find valuable information, thus they potentially pose bigger threat to the system security. However, no technique can prevent legitimate user from abusing their rights in a computer system. The only solution to the problem is to equip a computer system with an effective intrusion detection mechanism. There are two main approaches to build up the intrusion detection system. One is called misuse detection and the other is called anomaly detection. In misuse detection, one tries to detect intrusion by recognizing well defined attacks patterns in user’s data. In anomaly detection, usually a user profile is first built using the user’s normal activity data. Any significant deviation of current user data pattern from the profile indicates possible intrusion [61].

The main motivation for using machine learning techniques for the development of advanced IDS is their generalization capability, which may support the recognition of intrusions that have not been seen previously and have no previously described patterns. This formulation of intrusion detection problem combines the advantages of signature-based and anomaly-based IDS. Therefore we use different machine learning techniques in the implementation of our proposed intrusion detection system. We aim to develop an intelligent adaptive system that is capable of detecting known and unknown attacks with high detection rate and less time consuming.
4.2 Proposed System Architecture

Our system is a modular network-based intrusion detection system that analyzes Tcpdump data using machine learning techniques to classify the network records to not only normal and attack but also identify attack type. It exploits the historical user behaviors of the target system’s audit trails to train its artificial training module with the most dominant features of these audit trails to identify the different types of normal and intruder profiles. The system components of the learning phase are shown in Figure 4.1.

![Figure 4.1 Learning phase of Proposed System architecture](image)

Figure 4.1 Learning phase of Proposed System architecture

![Figure 4.2 Working Phase of Proposed System architecture](image)

Figure 4.2 Working Phase of Proposed System architecture
4.2.1 The Capture Module

Raw data of the network are captured and stored using the network adapter. It utilizes the capabilities of the TCP dump capture utility for Windows to gather historical network packets. It is an architecture that adds to the operating systems of the Win32 family the ability to capture the data of a network (dump traffic on a network) using the network adapter of the machine. Moreover, it provides to the applications a high level Application Programming Interface (API) that makes the use of its low-level capabilities simpler [62, 63].

4.2.2 The Preprocessing Module

The input data are preprocessed. The data must be of uniform representation to be processed by the classification module. The preprocessing module is responsible for reading, processing, and filtering the audit data to be used by the classification module. The preprocessing module handles Numerical Representation, Normalization and Features selection of raw input data. The preprocessing module maps the raw packets captured from the network by the TCP dump capture utility to a set of patterns of the most Effective Selected Feature. These dominant features are then used as inputs to the training module.

The preprocessing module consists of three phases:

A. Numerical Representation: Converts non-numeric features into a standardized numeric representation. This process involved the creation of relational tables for each of the data type and assigning number to each unique type of element. (e.g. protocol_type feature is encoded according to IP protocol field: TCP=0, UDP=1, ICMP=2). This is achieved by creating a transformation
table containing each text/string feature and its corresponding numeric value.

**B. Normalization:** The ranges of the features were different and this made them incomparable. Some of the features had binary values where some others had a continuous numerical range (such as duration of connection). As a result, inputs to the classification module should be scaled to fall between zero and one [0, 1] range for each feature.

**C. Dimension reduction:** reduce the dimensionality of input features of the classification module. Reducing the input dimensionality will reduce the complexity of the classification module, and hence the training time.

The final output of the pre-processing module components is a subset of the captured data (packets) that will be used as inputs to the classification module.

### 4.2.3 The classification Module

The classification module has two phases of operation. The learning phase (which uses the captured preprocessed historical data set) and the detection phase (which uses the current captured preprocessed input pattern).

**A. The Learning Phase**

In the learning phase, the classifier uses the pre-processed captured network user profiles as input training patterns. This phase continues until a satisfactory correct classification rate is obtained.

**B. The Detection Phase**

Once the classifier is learned, its capability of generalization to correctly identify the different types of users should be
utilized to detect intruder. This detection process can be viewed as a classification of input patterns to either normal or attack.

4.2.4 The Decision Module

Once an attack is detected by the classification module, the decision module will trigger an alarm with the suitable action to the system administrator. The basic responsibility of the decision module is to transmit alert to the system administrator informing him of coming attack. This gives the system administrator the ability to monitor the progress of the detection module. The set of actions is determined based on the system capability and its management’s policy.

4.3 Architectures Examined

We examined four system architectures:
1- Single Level Intrusion Detection System
2- Multi-Level Intrusion Detection System
3- Hybrid Multi-Level Intrusion Detection System
4- Enhanced Hybrid Multi-Level Intrusion Detection System

We compared between Single level and Multi-Level Neural Network Intrusion Detection System. In this comparison we used two types of attacks DOS (Neptune and Smurf) and Probe (Satan and Portsweep). While Hybrid Multi-Level Intrusion detection system was examined using different machine learning techniques and larger dataset containing the four types of attacks (DOS, Probe, U2R & R2L). The results produced by the hybrid system encouraged us to develop the Enhanced Hybrid Multi-Level Intrusion detection system.
4.3.1 Single Level Neural Network Intrusion Detection System

In this experiment we examined the use of the neural network for classifying normal and attack type, which means that we input the record and let the Single Layer Perceptron Neural Network identify the normal and specify the attack name as shown in Figure 4.3.

![Fully Connected Network](image)

**Figure 4.3** Single-Stage Single Layer Perceptron Network which Classify Normal and Attack type

4.3.2 Multi-Level Neural Network Intrusion Detection System

Attacks of the same class have a defined signature which differentiates between attacks of every class/category from others, i.e. DOS attacks have similar characteristics which identifies them from attacks of Probing. That's why there's often misclassification between attacks of the same class. For that reason, we thought of making a multi-stage neural network consisting of three levels as shown in Figure 4.4:
Proposed System Architecture

- **Level 1**: is a Neural Network that identifies attacks from normal
- **Level 2**: is a Neural Network that identifies classes
- **Level 3**: is a neural network that specify attack type

![Diagram of Proposed System Architecture]

**Figure 4.4 Multi-Levels.**

The data is input in the first level which identifies if this record is a normal record or attack without exhausting the network to identify the attack name. If the record is identified as an attack then the module would raise a flag to the administrator that the coming record is an attack then the module inputs this record to the second level which identifies the class of the coming attack. If record was classified by network II to be DOS then it would be entered to the DOS network of the third level that identify attacks' type of DOS otherwise it would be introduced to the Probe network. The idea is that if ever the attack name of the third level is misclassified then at least the admin was identified that this record is suspicious after the first level network. Finally the admin would be alerted of the suspected attack type to guide him for the suitable attack response.
1. **Level 1 Architecture**: Neural Network that identifies attacks from normal as shown in Figure 4.5.

![Diagram of Level 1 Architecture](image)

**Figure 4.5** First Level Network which differentiate between Normal and Attack.

2. **Level 2 Architecture**: If the output of Level 1 is identified as attack then this output is fed as input to second level which identifies whether the coming attack is DOS or Probe as shown in Figure 4.6.

![Diagram of Level 2 Architecture](image)

**Figure 4.6** Single Layer Perceptron of Second Level Network which Classify the Attack Class DOS or Probe
3. **Level 3 Architecture**: Neural network that specify attack type. This level consists of 2 networks. If the output of network 2 was identified as DOS then it's fed to the DOS neural network that specify whether the coming attack is neptune or smurf as shown in figure 4.7. Otherwise if the output of network 2 was identified as Probe attack then it's fed as input for the Probe Neural Network which identifies whether the coming probe attack is Satan or Portsweep as shown in figure 4.8.

**Attack type of DOS class whether Neptune or Smurf as shown in Figure 4.7.**

![Figure 4.7](image)

**Figure 4.7** Single Layer Perceptron of third Level Network which Classify Attack type of DOS category.
**Proposed System Architecture**

**Attack type of Probe class whether Satan or Portsweep as shown in figure 4.8.**

![Fully Connected Network](image)

**Figure 4.8** Third Level Network Single Layer Perceptron which classify Attack type of Probe category.

### 4.3.3 Hybrid Multi-Level Intrusion Detection System

The main characteristics of our proposed system:

- **Multilevel**: has the capability of classifying network intruders into a set of different levels. The first level classifies the network records to either normal or attack. The second level can identify four categories/classes. The third level where the attack type of each class can be identified.

Attacks of the same class have a defined signature which differentiates between attacks of every class/category from others, i.e. DOS attacks have similar characteristics which identifies them from attacks of Probing, R2L and U2R. That's why there's often misclassification between attacks of the same class, which gave the importance of making a multi-stage system consisting of three levels.
The data is input in the first level which identifies if this record is a normal record or attack. If the record is identified as an attack then the module would raise a flag to the administrator that the coming record is an attack then the module inputs this record to the second level which identifies the class of the coming attack. Level 2 module pass each attack record according to its class type to level 3 modules. Level 3 consists of 4 modules one for each class type (DOS, Probe, R2L, U2R). Each module is responsible for identifying the attack type of coming record.

The idea is that if ever the attack name of the third level is misclassified then at least the admin was identified that this record is suspicious after the first level network. Finally the admin would be alerted of the suspected attack type to guide him for the suitable attack response.

• **Hybrid:** Modules of each level can use different machine learning techniques. We made a comparative study examining several machine learning techniques to find the best classifier for each level. Neural network and decision trees have different classifying abilities for different intrusions. Neural network have high performance to DOS and Probing attacks while decision trees can detect the R2L more accurately than neural network. Therefore, Hybrid model will improve the performance to detect intrusions.

• **Adaptive:** Attacks that are misclassified by the IDS as normal activities or given wrong attack type will be relabeled by the network administrator. The training module can be retrained at any point of time which makes its implementation adaptive to any new environment and/or any new attacks in the network.
4.3.4 Enhanced Hybrid Multi-Level Intrusion Detection System

The enhanced system has a dual protection phases for Level 1 to increase the attacks detection rate. The levels of the enhanced system are shown in Figure 4.9. The first stage of level 1 is passing the input data through C5 Model, and then the records that are classified as normal by the C5 are passed to second stage which is MLP Model to detect some attacks that are bypassed by the C5 Model.

**Figure 4.9** Levels of Enhanced Hybrid System

1- First Stage of Level 1

The input data are preprocessed then entered to the C5 Model of Level 1. If the input record is classified as attack then the admin would be alarmed of the coming attack & the attack record would be input to Level 2 Model to specify its attack category.
2- Second Stage of Level 1

If the record is classified by the first phase to be normal then it is passed to the second protection phase of Level 1 which is an MLP Model of Level 1. If the record is classified as normal by the MLP Model then it's allowed to internal network. Otherwise if it's classified as attack the admin would be alarmed and the record would be input to the second level module to identify its attack category.

The enhanced system has the advantage of higher detection rate as MLP can detect some attacks that are bypassed by the C5 Model. Meanwhile it has the disadvantage that it produce higher false alarm rate. The two stages of Level 1 of the enhanced system are shown in Figure 4.10.

![Figure 4.10 Dual Protection Stages of Enhanced Multi-Level Intrusion Detection System](image-url)
4.4 Dataset

During the last decade, anomaly detection has attracted the attention of many researchers to overcome the weakness of signature-based IDSs in detecting novel attacks. KDDCUP’99 [64] is the mostly widely used data set for the evaluation of these systems. The KDD Cup 1999 uses a version of the data on which the 1998 DARPA Intrusion Detection Evaluation Program by MIT Lincoln Labs [65] was performed. Lincoln labs acquired nine weeks of raw TCP dump data. They set up an environment to acquire raw TCP/IP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN. NSL-KDD, which consists of selected records of the complete KDD data set and does not suffer from any of shortcomings.

The first important deficiency in the KDD data set is the huge number of redundant records. Analyzing KDD train and test sets, it was found that about 78% and 75% of the records are duplicated in the train and test set, respectively. This large amount of redundant records in the train set will cause learning algorithms to be biased towards the more frequent records, and thus prevent it from learning unfrequent records which are usually more harmful to networks such as U2R attacks. The existence of these repeated records in the test set, on the other hand, will cause the evaluation results to be biased by the methods which have better detection rates on the frequent records [66].

The new version of KDD data set is the NSL-KDD [67] dataset which consists of selected records of the complete KDD data set has the following advantages over the original KDD data set:
Proposed System Architecture

- It does not include redundant records in the train set, so the classifiers will not be biased towards more frequent records.
- There are no duplicate records in the proposed test sets; therefore, the performances of the learners are not biased by the methods which have better detection rates on the frequent records.
- The number of records in the train and test sets are reasonable, which makes it affordable to run the experiments on the complete set without the need to randomly select a small portion.

Consequently, evaluation results of different research works will be consistent and comparable. Although, the proposed data set still suffers from some of the problems discussed by McHugh [68] and may not be a perfect representative of existing real networks, because of the lack of public data sets for network-based IDSs, it is still believed that it can be applied as an effective benchmark data set to help researchers compare different intrusion detection methods [66].

4.4.1 NSL-KDD data set description

The simulated attacks fall in one of the following four categories:

1) **Denial of Service Attack (DoS):** is an attack in which the attacker makes some computing or memory resource too busy or too full to handle legitimate requests, or denies legitimate users access to a machine. Examples are Apache2, Back, Land, Mailbomb, SYN Flood, Ping of death, Process table, Smurf, Teardrop.

2) **User to Root Attack (U2R):** is a class of exploit in which the 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. Examples are Eject, Loadmodule, Ps, Xterm, Perl, Fdformat.

3) **Remote to Local Attack (R2L):** occurs when an attacker who has the ability to send packets to a machine over a network but who does not have an account on that machine exploits some vulnerability to gain local access as a user of that machine. Examples are Dictionary, Ftp_write, Guest, Imap, Named, Phf, Sendmail, Xlock.

4) **Probing Attack:** an attacker scans a network of computers to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network can use this information to look for exploits. Examples are Ipsweep, Mscan, Saint, Satan, Nmap.

   It is important to note that the test data is not from the same probability distribution as the training data, and it includes specific attack types not in the training data which make the task more realistic. Some intrusion experts believe that most novel attacks are variants of known attacks and the signature of known attacks can be sufficient to catch novel variants.

   The datasets contain a total number of 24 training attack types, with an additional 14 types in the test data only. The name and detail description of the training attack types are listed in [65].

   NSL-KDD features can be classified into three groups shown in Appendix 1:

   1) **Basic features:** this category encapsulates all the attributes that can be extracted from a TCP/IP connection.
Most of these features leading to an implicit delay in detection.

2) **Traffic features:** this category includes features that are computed with respect to a window interval and is divided into two groups:

   a) **“same host” features:** examine only the connections in the past 2 seconds that have the same destination host as the current connection, and calculate statistics related to protocol behavior, service, etc.

   b) **“same service” features:** examine only the connections in the past 2 seconds that have the same service as the current connection.

The two aforementioned types of “traffic” features are called time-based. However, there are several slow probing attacks that scan the hosts (or ports) using a much larger time interval than 2 seconds, for example, one in every minute. As a result, these attacks do not produce intrusion patterns with a time window of 2 seconds. To solve this problem, the “same host” and “same service” features are re-calculated but based on the connection window of 100 connections rather than a time window of 2 seconds. These features are called connection-based traffic features.

3) **Content features:** unlike most of the DoS and Probing attacks, the R2L and U2R attacks don’t have any intrusion frequent sequential patterns. This is because the DoS and Probing attacks involve many connections to some host(s) in a very short period of time; however the R2L and U2R attacks are embedded in the data portions of the packets, and normally involves only a single connection. To detect these kinds of attacks, we need some features to be able to look for suspicious behavior in the data portion, e.g., number of failed login attempts. These features are called content features.
Experimental Results

We performed two experiments. Experiment 1 was a comparison between using single level neural network and multi-level neural network intrusion detection system. The Single Level Neural Network examined the use of the neural network for classifying normal and attack type. The Multi-Level Neural Network consists of three detection levels. The first level differentiates between normal and attack. The second level specifies whether this attack is DOS or probe. The third detection level identifies attacks of denial of service and probe attacks.

Experiment 2 was to develop a hybrid multi-level intrusion detection system. This experiment was a comparative study between different machine learning models. We compared the results of using seven machine learning algorithms (MLP Neural Network, RBF Neural Network, Exhaustive Prune Neural Network, C5 Decision Tree, CHAID, CRT and QUEST). The hybrid system uses multi-level as explained in experiment 1. Each Level in the system is built with the machine learning algorithm that produced higher result for this level. The Hybrid Multi-Level was evolved to an Enhanced Hybrid Multi-Level Intrusion Detection System which has a dual protection phases in level 1.

The next section describes the definitions of performance measures that are used to evaluate the results of our proposed system.

5.1 Definitions of Performance Measures

To evaluate our system we used two major indices of performance. We calculate the detection rate and the false alarm rate according to [69] the following assumptions:
Experimental Results

- False Positive (FP): the total number of normal records that are classified as anomalous
- False Negative (FN): the total number of anomalous records that are classified as normal
- Total Normal (TN): the total number of normal records
- Total Attack (TA): the total number of attack records
- Detection Rate = \[(TA-FN) / TA]\*100
- False Alarm Rate = [FP/TN]\*100
- Correct Classification Rate = Number of Records Correctly Classified / Total Number of records in the used dataset

5.2 Experiment 1

This experiment aims to examine the difference between a multi-level MLP and single-level MLP. One of the objectives of the present experiment was to evaluate the possibility of achieving the same results with this less complicated neural network structure. Using a less complicated neural network is more computationally efficient. Also it would decrease the training time. Therefore we use a single layer perceptron with no hidden layers for all the networks in this experiment. For each network 20% of the training data were set for cross validation. Early stopping criterion for validation set was applied to stop the training process to prevent over-fitting.

5.2.1 Dataset used in Experiment 1

In this experiment we use neural network to check the ability of the intrusion detection system to identify attacks from different categories. We use two attacks from each DOS and Probe classes. The sample dataset contains 20000 record for training (10000 normal and 2500 for each attack type) and 1200 for testing (600 normal and 150 for each attack type).
5.2.2 The Over-fitting Problem

One problem that can occur during neural network training is over-fitting. In an over fitted ANN, the error (number of incorrectly classified patterns) on the training set is driven to a very small value, however, when new data is presented, the error is large. In these cases, the ANN has memorized the training examples; however, it has not learnt to generalize the solution to new situations. One possible solution for the over-fitting problem is to find the suitable number of training epochs by trial and error which isn't reasonable for cases that which takes too much time in training. A more reasonable method for improving generalization is called early stopping. In this technique, the available data is divided into three subsets. The first subset is the training set, which is used for training and updating the ANN parameters. The second subset is the validation set. The error on the validation set is monitored during the training process. The validation error will normally decrease during the initial phase of training similar to the training set error. However, when the ANN begins to over-fit the data, the error on the validation set will typically begin to rise. When the validation error increases for a specified number of iterations, the training is stopped, and the weights that produced the minimum error on the validation set are retrieved [70]. In the present study, this training-validation strategy was used in order to maximize the generalization capability of the ANN.
5.2.3 Single Level Neural Network

5.2.3.1 Training Single Level Neural Network

This network is a single layer feed-forward networks with activation function SoftMax. The output layer of this network consists of 5 neurons (normal, Neptune, Smurf, Satan, Portsweep). The training process was terminated with mean square error equal to 0.00034 at 12078 epochs.

5.2.3.2 Single Level Neural Network Testing Results

The testing phase resulted in success rate 98.8% with error rate 1.2%. Table 5.1 shows the Correct Classification Rate for each of the 5 classes and the total average classification accuracy of the single level neural network.

Table 5.1 Single Level Classification Rate

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Training Set</th>
<th>Testing Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>99.4</td>
<td>99.3</td>
</tr>
<tr>
<td>Neptune</td>
<td>100</td>
<td>99.3</td>
</tr>
<tr>
<td>Smurf</td>
<td>99.8</td>
<td>100</td>
</tr>
<tr>
<td>Satan</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Portsweep</td>
<td>99.9</td>
<td>94.7</td>
</tr>
<tr>
<td><strong>Average Success Rate</strong></td>
<td><strong>99.8</strong></td>
<td><strong>98.7</strong></td>
</tr>
<tr>
<td><strong>Error Rate</strong></td>
<td><strong>0.19</strong></td>
<td><strong>1.2</strong></td>
</tr>
</tbody>
</table>

5.2.4 Multi-Level Neural Network

5.2.4.1 Training multi-level Neural Network

All the 3 levels are a single layer perceptron feed-forward networks (which is the output layer as the input layer contains no processing so it's not considered a layer) with softmax
activation function which output results of summation equal to one.

The output layer of first level consists of two neurons one for normal and other for attack. The training process was stopped with mean square error equal 0.0015 at 10000 epochs.

The output layer of second level consists of two neurons one for DOS and other for Probe. The training process was stopped with mean square error equal to 0.000672 at 7914 epochs.

There are two networks in level three. The first one contains two neurons one for Neptune and the other for smurf. The training process is stopped with mean square error equal to 0.000001 at 1574 epochs.

The second network of level three consists of 2 neurons one for satan and the other for portsweep. The training process was terminated with performance 0.00233 at 5838 epochs.

5.2.4.2 Multi-Level Neural Network Testing Results

A. Level 1 Testing

The testing phase resulted in success rate 99.83 with error rate 0.167. Table 5.2 shows Correct Classification Rate for each of the 2 classes (Attack-Normal) and the total average classification accuracy.

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Training Set</th>
<th>Testing Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>99.5</td>
<td>99.7</td>
</tr>
<tr>
<td>Attack</td>
<td>99.9</td>
<td>100</td>
</tr>
<tr>
<td><strong>Average Success Rate</strong></td>
<td>99.74</td>
<td>99.8</td>
</tr>
<tr>
<td><strong>Error Rate</strong></td>
<td>0.27</td>
<td>0.17</td>
</tr>
</tbody>
</table>


**Experimental Results**

**B. Level 2 Testing**

The testing phase resulted in success rate 100. Table 5.3 shows the Correct Classification Rate for each of the 2 classes of Level 2 and the total average classification accuracy.

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Training Set</th>
<th>Testing Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOS</td>
<td>99.95</td>
<td>100</td>
</tr>
<tr>
<td>Probe</td>
<td>99.8</td>
<td>100</td>
</tr>
<tr>
<td><strong>Average Success Rate</strong></td>
<td><strong>99.9</strong></td>
<td><strong>100</strong></td>
</tr>
<tr>
<td><strong>Error Rate</strong></td>
<td><strong>0.14</strong></td>
<td><strong>0</strong></td>
</tr>
</tbody>
</table>

**C. Level 3 Testing**

The testing phase resulted in success rate 99.5 with error rate 0.5. Table 5.4 shows the Correct Classification Rate for each of the 4 classes and the total average classification accuracy.

<table>
<thead>
<tr>
<th>Level 3 Networks</th>
<th>Class Name</th>
<th>Correct Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Training Set</td>
</tr>
<tr>
<td><strong>DOS Network</strong></td>
<td>Neptune</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Smurf</td>
<td>100</td>
</tr>
<tr>
<td><strong>Probe Network</strong></td>
<td>Satan</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Portsweep</td>
<td>100</td>
</tr>
<tr>
<td><strong>Average Success Rate</strong></td>
<td><strong>100</strong></td>
<td><strong>99.5</strong></td>
</tr>
<tr>
<td><strong>Error Rate</strong></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

**5.2.5 Discussion of Results of Experiment 1**

Building all the networks with a single layer perceptron with no hidden layers gave the advantage of less computation time and less complicated network. The experimental results show that using a multi-level neural network is more promising than
Experimental Results

single-level network as shown in following tables and figures. Table 5.5 shows the Correct Classification Rate of testing dataset for each of the 5 classes for both Multi-level and single-level.

**Table 5.5 Classification Rate of Multi-Level and Single-Level**

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Multi-Level</th>
<th>Single-Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>99.7</td>
<td>99.3</td>
</tr>
<tr>
<td>Neptune</td>
<td>100</td>
<td>99.3</td>
</tr>
<tr>
<td>Smurf</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Satan</td>
<td>98.7</td>
<td>100</td>
</tr>
<tr>
<td>Portsweep</td>
<td>99.3</td>
<td>94.7</td>
</tr>
</tbody>
</table>

**Figure 5.1 Comparison between Multi-Level and Single-Level**

**Table 5.6 False Alarm Comparison**

<table>
<thead>
<tr>
<th>Method</th>
<th>Multi-Level</th>
<th>Single-Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>FN</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>TN</td>
<td>600</td>
<td>600</td>
</tr>
<tr>
<td>TA</td>
<td>600</td>
<td>600</td>
</tr>
</tbody>
</table>

**Detection Rate**

100

**False Alarm Rate**

0.3
5.3 Experiment 2

The second experiment was developing a hybrid multi-level intrusion detection system. We applied seven distinct pattern recognition and machine learning algorithms on the NSL-KDD dataset. This study was implemented using SPSS Clementine and Neuro Solution. The Hybrid Multi-Level was evolved to an Enhanced Hybrid Multi-Level Intrusion Detection System which has a dual protection phases in level 1.

5.3.1 Dataset used in Experiment 2

In this study we examine using attacks from the four classes to check the ability of the intrusion detection system to identify attacks from different categories. The sample dataset contains 83655 record for training (40000 normal and 43655 for attacks) and 16592 for testing (9657 normal, 6935 for known attacks and 3202 for unknown attacks) as shown in table 5.7 & table 5.8.
# Experimental Results

## Table 5.7 Dataset for training and testing

<table>
<thead>
<tr>
<th>Class</th>
<th>Attacks Name</th>
<th>Number of Records</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Training</td>
</tr>
<tr>
<td>Normal</td>
<td></td>
<td>40000</td>
</tr>
<tr>
<td>DOS</td>
<td>Neptune</td>
<td>25000</td>
</tr>
<tr>
<td></td>
<td>Smurf</td>
<td>2646</td>
</tr>
<tr>
<td></td>
<td>Teardrop</td>
<td>892</td>
</tr>
<tr>
<td></td>
<td>Back</td>
<td>956</td>
</tr>
<tr>
<td>Probe</td>
<td>Ipsweep</td>
<td>3591</td>
</tr>
<tr>
<td></td>
<td>Portsweep</td>
<td>2919</td>
</tr>
<tr>
<td></td>
<td>Satan</td>
<td>3632</td>
</tr>
<tr>
<td></td>
<td>Nmap</td>
<td>1485</td>
</tr>
<tr>
<td>R2L</td>
<td>guess_passwd</td>
<td>1231</td>
</tr>
<tr>
<td></td>
<td>warezmaster</td>
<td>944</td>
</tr>
<tr>
<td></td>
<td>Snmpguess</td>
<td>300</td>
</tr>
<tr>
<td>U2R</td>
<td>buffer_overflow</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Rootkit</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Loadmodule</td>
<td>9</td>
</tr>
</tbody>
</table>

## Table 5.8 New Attacks used for testing

<table>
<thead>
<tr>
<th>Class</th>
<th>Attacks Name</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOS</td>
<td>apache2</td>
<td>737</td>
</tr>
<tr>
<td></td>
<td>processtable</td>
<td>685</td>
</tr>
<tr>
<td></td>
<td>Mailbomb</td>
<td>293</td>
</tr>
<tr>
<td>Probe</td>
<td>Saint</td>
<td>316</td>
</tr>
<tr>
<td></td>
<td>Mscan</td>
<td>996</td>
</tr>
<tr>
<td>R2L</td>
<td>Httpunnel</td>
<td>133</td>
</tr>
<tr>
<td></td>
<td>Sendmail</td>
<td>14</td>
</tr>
<tr>
<td>U2R</td>
<td>Ps</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Xterm</td>
<td>13</td>
</tr>
</tbody>
</table>
5.3.2 Training Hybrid Multi-Level Intrusion Detection System

The hybrid system uses algorithms in the fields of neural networks and decision trees. As for Neural Network we examined the use of MLP, RBF and Exhaustive prune Neural Networks, while for decision trees we used C5, CHAID, CRT and QUEST.

5.3.2.1 Neural Networks Implementation

The neural network gains the experience initially by training the system to correctly identify pre-selected examples of the problem. The response of the neural network is reviewed and the configuration of the system is refined until the neural network’s analysis of the training data reaches a satisfactory level. In addition to the initial training period, the neural network also gains experience over time as it conducts analysis on data related to the problem [9].

5.3.2.1.1 Multi-Layer Perceptron (MLP) Implementation

MLP are feed-forward neural networks trained with the standard backpropagation algorithm. They learn how to transform input data into a desired response, so they are widely used for pattern classification [45].

The architecture used for the MLP during simulations consisted of a three layer feed-forward neural network: one input, two hidden, and one output layers. Sigmoid transfer functions were used for each neuron in both the hidden layers and softmax in the output layers. The network was set to train
until the desired mean square error of 0.001 was met or 10000 epochs was reached.

For the first level there were 31 neurons in the input layer (31-feature input pattern) after feature selection, 22 neurons in first hidden layer, 18 neurons in second hidden layer and 2 neurons (one for normal and the other for attack) in the output layer. During the training process, the mean square error is 0.0157 at 10000 epochs. For the second level 38 in input layer, 12 in first hidden layer, 10 in second hidden layer and 4 neurons in the output layer (DOS, Probe, R2L and U2R). During the training process, the mean square error is 0.0114 at 10000 epochs. We've four networks in the third level. DOS network has layers of 28-2-2-7 feed-forward neural network. (i.e. 28 in input layer, 2 in the 1\textsuperscript{st} hidden layer, 2 in the 2\textsuperscript{nd} hidden layer and 7 in the output layer). During the training process, the mean square error is 0 at 1574 epochs. Probe network has layers of 24-22-14-6 feed-forward network with mean square error 0.05 at 10000 epochs. R2L network has layers of 26-17-10-5 feed-forward network with mean square error 0 at 5838 epochs. U2R network has layers of 11-9-7-5 feed-forward network with mean square error 2.33 at 10000 epochs.

5.3.2.1.2 Radial Basis Function (RBF) Implementation

RBF networks have a static Gaussian function as the non-linearity for the hidden layer processing elements. The Gaussian function responds only to a small region of the input space where the Gaussian is centered. The key to a successful implementation of these networks is to find suitable centers for the Gaussian functions. This can be done with supervised
learning, but an unsupervised approach usually produces better results. The advantage of the radial basis function network is that it finds the input to output map using local approximators. Usually the supervised segment is simply a linear combination of the approximators [45].

The RBF layer uses gaussian transfer functions. The learning rate was set to 0.1 for the hidden layer and 0.01 for the output layer. The alpha was set to 0.75. For the first level there were 31 neurons in the input layer, 10 neurons in hidden layer and 2 neurons (one for normal and the other for attack) in the output layer. Estimated accuracy of training was 94.4%. The second level has 37 in input layer, 10 in hidden layer and 4 neurons in the output layer (DOS, Probe, R2L and U2R) with estimated accuracy of 93.5%. We've four networks in the third level. DOS RBF network has layers of 28-20-7. (i.e. 28 in input layer, 20 in hidden layer and 7 in the output layer) with estimated accuracy 100%. Probe network has layers of 24-20-6 network with estimated accuracy 98.3%. R2L RBF network has layers of 26-20-5 with estimated accuracy 98.3%. U2R network has layers of 11-20-5 with estimated accuracy 75%.

5.3.2.1.3 Exhaustive Prune Implementation

This method is related to the prune method. It starts with a large network and prunes the weakest units in the hidden and input layers as training proceeds. With exhaustive prune, network training parameters are chosen to ensure a very thorough search of the space of possible models to find the best one. This method is usually the slowest but often yields the best results [51].
Experimental Results

The first level there consists of 13 neurons in the input layer, 22 neurons in first hidden layer, 7 neurons in second hidden layer and 2 neurons (one for normal and the other for attack) in the output layer with estimated accuracy of training 99.8%. The second level consists of 25 in input layer, 9 in first hidden layer, 5 in second hidden layer and 4 neurons in the output layer (DOS, Probe, R2L and U2R) with accuracy of training 99.9%. We've four networks in the third level. DOS network has layers of 3-19-17-7 network with accuracy of training 100%. Probe network has layers of 10-12-5-6 network with estimated accuracy of 99.6%. R2L network has layers of 14-3-2-5 network with estimated accuracy of 100%. U2R network has layers of 1-3-2-5 network with estimated accuracy of training 81.5%.

5.3.2.2 Decision trees Implementation

The decision tree is a simple if then else rules but it is a very powerful classifier and proved to have a high detection rate. They are used to classify data with common attributes. Each decision tree represents a rule which categorizes data according to these attributes. A decision tree consists of nodes, leaves, and edges. A node of a decision tree specifies an attribute by which the data is to be partitioned. Each node has a number of edges which are labeled according to a possible value of the attribute in the parent node. An edge connects either two nodes or a node and a leaf. Leaves are labeled with a decision value for categorization of the data [53].

5.3.2.2.1 C5.0 Tree Implementation

C5.0 decision tree is one of the most popular inductive learning tools originally proposed by J.R.Quinlan as C4.5 algorithm (Quinlan, 1993) [11]. Single C5 acquires pruned
decision tree with pruning severity 75% and winnowing attributes. First level consists of 121 nodes on train data and 20 tree depth and standard error 0.01%. Second level consists of 113 nodes and tree depth of 12 with standard error 0.05%. Third level DOS tree consists of 6 nodes and tree depth of 4 levels with standard error 0%. Probe tree consists of 69 nodes and tree depth of 10 levels with standard error 0.4%. R2L tree consists of 7 nodes and tree depth of 4 levels with standard error 0%. U2R tree consists of 9 nodes and tree depth of 4 levels with standard error 8.33%.

5.3.2.2.2 Classification and Regression Trees (CRT or CART) Implementation

CRT was set of maximum surrogates 10, minimum change in impurity 0.0 and Gini impurity measure for categorical targets. First level consists of 15 nodes and of depth 4. Second level consists of 15 nodes of tree depth 4. Third level DOS consists of 7 nodes of tree depth = 3. Probe consists of 13 nodes of tree depth 5. R2L consists of 7 nodes of tree depth 4. U2R consists of 17 nodes of tree depth 6.

5.3.2.2.3 Chi-squared Automatic Interaction Detector (CHAID) Implementation

CHAID was adjusted of Alpha splitting 0.05, alpha for merging 0.05, epsilon for convergence 0.001, using pearson chi-square method. First level consists of 35 nodes and of depth 5. Second level consists of 28 nodes of tree depth 4. Third level DOS consists of 6 nodes of tree depth 3. Probe consists of 49 nodes of tree depth 6. R2L consists of 7 nodes of tree depth 3. U2R consists of 12 nodes of tree depth 5.
5.3.2.2.4 Quick, Unbiased, Efficient Statistical Tree (QUEST) Implementation

QUEST was adjusted of maximum surrogates 5, and alpha for splitting 0.05. First Level consists of 15 nodes and of 4 tree depth. Third level DOS consists of 11 nodes of tree depth 6. Probe consists of 17 nodes of tree depth 6. R2L consists of 9 nodes of tree depth 5. U2R consists of 13 nodes of tree depth 6.

5.3.3 Hybrid Multi-Level Testing Results
5.3.3.1 Level 1 output

Level 1 duty is to classify whether coming record is normal or attack. It is observed that MLP best classifies normal records while C5 is more efficient in detecting known and unknown attacks. The results of Level 1 are shown in table 5.9 & 5.10.

**Table 5.9 Correct Classification Rate for Level 1**

<table>
<thead>
<tr>
<th>Percentage</th>
<th>Normal</th>
<th>Attacks</th>
<th>New Attacks</th>
<th>Correct Classification Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>95.1</td>
<td>97.2</td>
<td>78.7</td>
<td>93.2</td>
</tr>
<tr>
<td>RBF</td>
<td>90.4</td>
<td>93.1</td>
<td>45.5</td>
<td>84.1</td>
</tr>
<tr>
<td>Exhaustive</td>
<td>89.7</td>
<td>97.3</td>
<td>86.2</td>
<td>91.8</td>
</tr>
<tr>
<td>C5</td>
<td>90.6</td>
<td><strong>99.5</strong></td>
<td>97</td>
<td><strong>93.2</strong></td>
</tr>
<tr>
<td>CRT</td>
<td>93.3</td>
<td><strong>98.9</strong></td>
<td>45.4</td>
<td>87.5</td>
</tr>
<tr>
<td>QUEST</td>
<td>85.5</td>
<td>98</td>
<td>67.1</td>
<td>86.9</td>
</tr>
<tr>
<td>CHAID</td>
<td>89.6</td>
<td>97.1</td>
<td>59.2</td>
<td>87.3</td>
</tr>
</tbody>
</table>
Figure 5.3 Level 1 Classification Rate

Table 5.10 Detection rate & False alarm rate for level 1

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Detection Rate</th>
<th>False Alarm Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>91.4</td>
<td>5</td>
</tr>
<tr>
<td>RBF</td>
<td>78.1</td>
<td>9.6</td>
</tr>
<tr>
<td>Exhaustive</td>
<td>91.8</td>
<td>10.3</td>
</tr>
<tr>
<td>C5</td>
<td><strong>95.6</strong></td>
<td>9.4</td>
</tr>
<tr>
<td>CRT</td>
<td>82</td>
<td>15.8</td>
</tr>
<tr>
<td>QUEST</td>
<td>88</td>
<td>14.5</td>
</tr>
<tr>
<td>CHAID</td>
<td>85</td>
<td>10.4</td>
</tr>
</tbody>
</table>

For normal and known attacks most of the techniques are comparable. C5 has a significant detection rate for known and unknown attacks but it produces a higher false alarm rate compared to MLP.

5.3.3.2 Level 2 output

Records classified as attacks by the first level are introduced to second level which is responsible for classifying coming attack to one of the four classes (DOS, Probe, R2L & U2R). Testing results showed that C5 & CRT (decision trees) produced best correct classification rate for second level as shown in table 5.11 & Figure 5.4.
Experimental Results

**Table 5.11 Correct classification rate for level 2**

<table>
<thead>
<tr>
<th>Level 2 Classifiers</th>
<th>Known Attacks</th>
<th>New Attacks</th>
<th>Correct Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>95.1</td>
<td>56.3</td>
<td>82.8</td>
</tr>
<tr>
<td>RBF</td>
<td>86</td>
<td>50.8</td>
<td>74.8</td>
</tr>
<tr>
<td>Exhaustive</td>
<td>92.8</td>
<td>49.9</td>
<td>79.2</td>
</tr>
<tr>
<td>C5</td>
<td>98.4</td>
<td>59.3</td>
<td>86</td>
</tr>
<tr>
<td>CRT</td>
<td>96.5</td>
<td>62.7</td>
<td>85.8</td>
</tr>
<tr>
<td>CHAID</td>
<td>97.2</td>
<td>38.8</td>
<td>78.8</td>
</tr>
</tbody>
</table>

**Figure 5.4 Level 2 Classification Rate**

**5.3.3.3 Level 3 output**

The third level consists of four modules; a module for each class. For example records that were classified by the second level to be DOS attack are sent to the DOS module of the 3\textsuperscript{rd} level & so on.

Results of Denial of service modules showed that DOS attacks are easy to be correctly classified by many classifiers either neural network or decision trees as shown in table 5.12.
### Table 5.12 DOS attacks Classification Rate

<table>
<thead>
<tr>
<th>DOS Classifier</th>
<th>Correct Classification Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>100</td>
</tr>
<tr>
<td>RBF</td>
<td>99.4</td>
</tr>
<tr>
<td>Exhaustive</td>
<td>99.9</td>
</tr>
<tr>
<td>C5</td>
<td>100</td>
</tr>
<tr>
<td>CRT</td>
<td>100</td>
</tr>
<tr>
<td>QUEST</td>
<td>99.9</td>
</tr>
<tr>
<td>CHAID</td>
<td>100</td>
</tr>
</tbody>
</table>

Results of Probe module showed that C5 & MLP are most efficient for detecting this type of attacks as shown in table 5.13.

### Table 5.13 Probe attacks Classification Rate

<table>
<thead>
<tr>
<th>Probe Classifier</th>
<th>Correct Classification Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>99.3</td>
</tr>
<tr>
<td>RBF</td>
<td>97.8</td>
</tr>
<tr>
<td>Exhaustive</td>
<td>97</td>
</tr>
<tr>
<td>C5</td>
<td>98.6</td>
</tr>
<tr>
<td>CRT</td>
<td>92.6</td>
</tr>
<tr>
<td>QUEST</td>
<td>94.1</td>
</tr>
<tr>
<td>CHAID</td>
<td>95.5</td>
</tr>
</tbody>
</table>

Results of R2L module showed that C5 are most efficient for detecting this type of attacks significantly as shown in table 5.14.
Experimental Results

Table 5.14 R2L attacks Classification Rate

<table>
<thead>
<tr>
<th>R2L Classifier</th>
<th>Correct Classification Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>91</td>
</tr>
<tr>
<td>RBF</td>
<td>93</td>
</tr>
<tr>
<td>Exhaustive</td>
<td>91</td>
</tr>
<tr>
<td>C5</td>
<td>100</td>
</tr>
<tr>
<td>CRT</td>
<td>97</td>
</tr>
<tr>
<td>QUEST</td>
<td>96</td>
</tr>
<tr>
<td>CHAID</td>
<td>97</td>
</tr>
</tbody>
</table>

U2R attacks have a very low classification rate compared to other classes. Results showed that Exhaustive prune is better than other classifiers for detecting attacks of this class as shown in table 5.15.

Table 5.15 U2R attacks Classification Rate

<table>
<thead>
<tr>
<th>U2R Classifier</th>
<th>Correct Classification Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>48.2</td>
</tr>
<tr>
<td>RBF</td>
<td>43.1</td>
</tr>
<tr>
<td>Exhaustive</td>
<td>54.4</td>
</tr>
<tr>
<td>C5</td>
<td>44.1</td>
</tr>
<tr>
<td>CRT</td>
<td>44.1</td>
</tr>
<tr>
<td>QUEST</td>
<td>35.3</td>
</tr>
<tr>
<td>CHAID</td>
<td>41.2</td>
</tr>
</tbody>
</table>
5.3.4 Enhanced Hybrid Multi-Level Testing Results

The Enhanced Hybrid Multi-Level System contains a dual protection phases in level 1. The input data are input to the first protection phase which is C5 Model. Records that are classified as normal are passed to the second phase of protection which is a MLP model. Thus we obtain a higher protection rate as some attacks that are bypassed by C5 are detected by the MLP whose nature differs from C5. The Statistics of Enhanced Hybrid Multi-Level System is shown in figure 5.5.

![Diagram of Enhanced Hybrid Multi-Level System]

**Figure 5.5** Summary of Results of Enhanced Hybrid Multi-Level System

Known attacks detected by C5 = 6903
Known attacks detected by MLP = 22
Detection Rate of known attacks = \((6903 + 22) / 6935 = 99.8\%\)
Experimental Results

Unknown attacks detected by C5 = 2784
Unknown attacks detected by MLP = 219
Detection Rate of Unknown attacks = (2784 + 219) / 3201 = 93.8 %

Total Detection Rate of Level 1 = (9687 + 241) / 10136 (Total No of Normal Records) = 97.95%

Total Correct Classification Rate of Level 1 = (8598[Normal] + 9687[C5 detected attacks] + 241[MLP detected attacks]) / (10136[TA] + 9647[TN]) = 93.65%

Total False Alarm Rate of Level 1 = (902 + 147) / 9647 (Total Normal) = 10.87%
5.3.5 Discussion of Results of Experiment 2

Simulation results demonstrated that for a given attack category certain classifier algorithms performed better. Consequently, a multi-classifier model that was built using most promising classifiers for a given attack category was evaluated for probing, denial-of-service, user-to-root, and remote-to-local attack categories. While the neural networks are very interesting for generalization and very poor for new attacks detection, the decision trees have proven their efficiency in both generalization and new attacks detection. Besides the C5 has less training time than the MLP. However, none of the machine learning classifier algorithms evaluated was able to perform detection of user-to-root attack categories significantly (no more than 54% detection for U2R category).

The advantage of the proposed multi-level system is not only higher accuracy but also the parallelism as every module can be trained on separate computer which provides less training time. Also the multi-level powers the system with scalability because if new attacks of specific class are added to the dataset we don't have to train all the modules but only the module affected by the new attack. Attacks that are misclassified by the IDS as normal activities or given wrong attack type will be relabeled by the network administrator. Training module can be retrained at any point of time which makes its implementation adaptive to any new environment or any new attacks in the network. The dual protection phases of first level in the Enhanced Hybrid Multi-Level Intrusion Detection System gave higher detection rate. Meanwhile it increases the false alarm rate of the proposed system.
6.1 Conclusion

In this work an Enhanced Hybrid Multi-Level Intrusion Detection System was developed. The proposed system consists of three detection levels. The network data are introduced to the module of the first level which aims to differentiate between normal and attack. The first level has dual protection phase. In the first phase of level one the data is passed through C5 Model which identifies whether the coming record is normal or attack. Records that are classified as normal are passed to an MLP Model which reclassifies these records. Thus some attacks that are bypassed by C5 Decision Tree model are detected by the MLP Model consequently achieving higher detection level. If the input record was identified as an attack then the administrator would be alarmed that the coming record is suspicious and then this suspicious record would be introduced to the second level which specifies the class of this attack (DOS, probe, R2L or U2R). The third detection level consists of four modules one module for each class type to identify attacks of this class. Finally the administrator would be alarmed of the expected attack type.

We examined each module using different machine learning models (MLP, RBF, C5, CRT, QUEST & Exhaustive Prune). Each module is implemented with the most promising classifier that gave highest correct classification rate. Therefore, Hybrid model will improve the performance to detect intrusions.

The experimental results showed that the designed multi-level system has detection rate equal to 98% for both (known and unknown attacks). The first level is implemented by C5 decision tree & MLP Neural Network which showed significant detection rate for both known and unknown attacks. The
drawback of using C5 decision tree is the high false alarm rate that it produces. The second level is implemented by C5. As for the third level DOS & Probe modules are implemented by MLP, R2L module is implemented by C5 decision tree and U2R module is implemented by Exhaustive prune Neural Network.

While the neural networks are very interesting for generalization and very poor for new attacks attack detection, the decision trees have proven their efficiency in both generalization and new attacks detection. Besides the C5 has less training time than the MLP. However, none of the machine learning classifier algorithms evaluated was able to perform detection of user-to-root attack categories significantly (no more than 54% detection for U2R category).
Advantages of proposed system:

1- **Dual Protection:** The dual protection phases of first level in the Enhanced Hybrid Multi-Level Intrusion Detection System gave higher detection rate as some of the attacks that were bypassed by the C5 Model of the first phase of level 1 were detected by the MLP Model of the second phase of level 1.

2- **High Detection Rate:** The Enhanced Proposed System achieved almost 98 % Detection Rate compared to other systems that used the KDD dataset.

3- **Scalability:** because if new attacks of specific class are added to the dataset we don't have to train all the modules but only the module affected by the new attack

4- **Adaptive:** Attacks that are misclassified by the IDS as normal activities or given wrong attack type will be relabeled by the network administrator. Training module can be retrained at any point of time which makes its implementation adaptive to any new environment or any new attacks in the network.

5- **Generalization Ability:** The proposed system outperforms previous work in detecting both known and new attacks which combines the advantages of signature-based and anomaly-based IDS.

6- Every module can be trained on separate computer in parallel which provides less training time.

Disadvantages:

1- High false alarm rate.

2- Models that are implemented by MLP Neural Network take long time during training phase.
6.2 Future Work

1. The detection of U2R attack is more difficult because of their close resemblance with the normal connections. Our future research will be directed towards developing more accurate base classifiers particularly for the detection of U2R type of attacks.
2. Also finding ways to produce less false alarm rate for the C5 Decision tree.
3. Applying this architecture to different datasets.
References


References


References

References

[70] MATLAB online support:
www.mathworks.com/access/helpdesk/help/techdoc/matlab.shtml
### Appendix 1. Description of KDD 99 Intrusion Detection Dataset Features

#### Table A.1. List of features with their descriptions and data types [64]

<table>
<thead>
<tr>
<th>Index</th>
<th>Feature</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>duration</td>
<td>Duration of the connection.</td>
<td>Cont.</td>
</tr>
<tr>
<td>2</td>
<td>protocol type</td>
<td>Connection protocol (e.g. tcp, udp)</td>
<td>Disc.</td>
</tr>
<tr>
<td>3</td>
<td>service</td>
<td>Destination service (e.g. telnet, ftp)</td>
<td>Disc.</td>
</tr>
<tr>
<td>4</td>
<td>flag</td>
<td>Status flag of the connection</td>
<td>Disc.</td>
</tr>
<tr>
<td>5</td>
<td>source bytes</td>
<td>Bytes sent from source to destination</td>
<td>Cont.</td>
</tr>
<tr>
<td>6</td>
<td>destination bytes</td>
<td>Bytes sent from destination to source</td>
<td>Cont.</td>
</tr>
<tr>
<td>7</td>
<td>land</td>
<td>1 if connection is from/to the same host/port; 0 otherwise</td>
<td>Disc.</td>
</tr>
<tr>
<td>8</td>
<td>wrong fragment</td>
<td>number of wrong fragments</td>
<td>Cont.</td>
</tr>
<tr>
<td>9</td>
<td>urgent</td>
<td>number of urgent packets</td>
<td>Cont.</td>
</tr>
<tr>
<td>10</td>
<td>hot</td>
<td>number of &quot;hot&quot; indicators</td>
<td>Cont.</td>
</tr>
<tr>
<td>11</td>
<td>failed logins</td>
<td>number of failed logins</td>
<td>Cont.</td>
</tr>
<tr>
<td>12</td>
<td>logged in</td>
<td>1 if successfully logged in; 0 otherwise</td>
<td>Disc.</td>
</tr>
<tr>
<td>13</td>
<td># compromised</td>
<td>number of compromised conditions</td>
<td>Cont.</td>
</tr>
<tr>
<td>14</td>
<td>root shell</td>
<td>1 if root shell is obtained; 0 otherwise</td>
<td>Cont.</td>
</tr>
<tr>
<td>15</td>
<td>su attempted</td>
<td>1 if &quot;su root&quot; command attempted; 0 otherwise</td>
<td>Cont.</td>
</tr>
<tr>
<td>16</td>
<td># root</td>
<td>number of &quot;root&quot; accesses</td>
<td>Cont.</td>
</tr>
<tr>
<td>17</td>
<td># file creations</td>
<td>number of file creation operations</td>
<td>Cont.</td>
</tr>
<tr>
<td>18</td>
<td># shells</td>
<td>number of shell prompts</td>
<td>Cont.</td>
</tr>
<tr>
<td>19</td>
<td># access files</td>
<td>number of operations on access control files</td>
<td>Cont.</td>
</tr>
<tr>
<td>20</td>
<td># outbound cmds</td>
<td>number of outbound commands in an ftp session</td>
<td>Cont.</td>
</tr>
<tr>
<td>21</td>
<td>is hot login</td>
<td>1 if the login belongs to the &quot;hot&quot; list; 0 otherwise</td>
<td>Disc.</td>
</tr>
<tr>
<td>22</td>
<td>is guest login</td>
<td>1 if the login is a &quot;guest&quot; login; 0 otherwise</td>
<td>Disc.</td>
</tr>
<tr>
<td>23</td>
<td>Count</td>
<td>number of connections to the same host as the current connection in the past two seconds</td>
<td>Cont.</td>
</tr>
<tr>
<td>24</td>
<td>srv count</td>
<td>number of connections to the same service as the current connection in the</td>
<td>Cont.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>----------------</td>
<td>----------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>serror rate</td>
<td>% of connections that have &quot;SYN&quot; errors</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>srv serror rate</td>
<td>% of connections that have &quot;SYN&quot; errors</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>reror rate</td>
<td>% of connections that have &quot;REJ&quot; errors</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>srv reror rate</td>
<td>% of connections that have &quot;REJ&quot; errors</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>same srv rate</td>
<td>% of connections to the same service</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>diff srv rate</td>
<td>% of connections to different services</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>srv diff host rate</td>
<td>% of connections to different hosts</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>dst host count</td>
<td>count of connections having the same destination host</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>dst host srv count</td>
<td>count of connections having the same destination host and using the same service</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>dst host same srv rate</td>
<td>% of connections having the same destination host and using the same service</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>dst host diff srv rate</td>
<td>% of different services on the current host</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>dst host same src port rate</td>
<td>% of connections to the current host having the same src port</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>dst host srv diff host rate</td>
<td>% of connections to the same service coming from different hosts</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>dst host serror rate</td>
<td>% of connections to the current host that have an S0 error</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>dst host srv serror rate</td>
<td>% of connections to the current host and specified service that have an S0 error</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>dst host reror rate</td>
<td>% of connections to the current host that have an RST error</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>dst host srv reror rate</td>
<td>% of connections to the current host and specified service that have an RST error</td>
<td></td>
</tr>
</tbody>
</table>

Appendix
جامعة عين شمس
كلية الحاسبات والمعلومات

استخدام التقنيات الذكية في أنظمة كشف التسلل

الرسالة مقدمة للحصول على درجة الماجستير في الحاسبات والمعلومات

إعداد
سحر سليم فؤاد يوسف

بكالوريوس الحاسبات والمعلومات
جامعة عين شمس

تحت إشراف
أ.د. تيمور محمد نظمي
أساتذة بقسم علوم الحاسب ونظم
ووكيل الكلية لشئون الدراسات العليا والبحوث
كلية الحاسبات والمعلومات، جامعة عين شمس

أ.د. محمد هاشم
أساتذة بقسم نظم المعلومات
ووكيل الكلية لشئون التعليم والطلاب
كلية الحاسبات والمعلومات، جامعة عين شمس

المؤهله 2011
ملخص الرسالة

مع التوسع السريع في شبكات الحاسوب خلال العقد الماضي، أصبح الأمن مسألة بالغة الأهمية لنظم الحاسوب. الثقافات الأمنية الجديدة يتم اكتشافها كل يوم، وهناك عدد متزايد من أصaab النوايا السيئة من الناس الذين يحاولون الاستفادة من هذه الاختلاطات للتفوّل على شبكات. لذا فقد أصبح كشف التسلل عملية هامة في أمن الشبكات. أنظمة كشف التسلل (Intrusion Detection Systems) تهدف إلى حماية الشبكات وإجهزة الكمبيوتر من الهجمات الضارة.

وقد اقتربت في السنوات الأخيرة أساليب مختلفة لتطوير أنظمة كشف التسلل على أساس الحوسبة اللينة (soft-computing techniques). أكثر الأساليب الحالية للكشف التسلل تعتمد لتحديد (rule-based expert systems) على استخدام النظم الخبيرة القائمة على قواعد مؤشرات الهجمات المعروفة. الشبكات العصبية الاصطناعية (Artificial neural networks) تتبع إمكانية تطبيقية وتصنيف نشاط الشبكات (decision trees) وشجرة القرار، وتعد مثال لاستخدام النظم الخبيرة في المجالات الأمنية.

معظم الأنظمة السابقة، من بعض أوجه القصور. بعض سلبيات أنظمة كشف التسلل السابقة هي أنها غير قادرة على اكتشاف الهجمات الجديدة الغير معروفة التي لم تسبق للنظام التعامل لها. معظم هذه النظم لا تحدد نوع الهجوم ولكن فقط تحدد ما إذا كانت البيانات المطلوبة للشبكة عادية أم هجومية. إحدى سلبيات نظام كشف التسلل المبنية على بصمة الهجوم (Signature-Based Intrusion Detection Systems) أنها يمكنها الكشف عن الهجمات المعروفة فقط، وجميع الهجمات الجديدة غير معروفة سوف تمر مرور الكرام حتى يتم تحديث النظام ليتمكن من التعرف عليها.

هذه الرسالة تقترح نظام مهجين ذكي لكشف التسلل لتحسين معدل اكتشاف الهجمات المعروفة والغير معروفة. المرايا الرئيسية لنظام المقترح هي قدرته على سرعة التعلم، وتحسين قدرته على التعزيم، وتلبية مسؤول النظام إذا تعرض النظام لهجمات غير معروفة من قبل. على عكس الأنظمة الأخرى التي لديها مستوى واحد من الكشف، فإن النظام المقترح يتكون من ثلاثة مستويات من الكشف. المستوى الأول حيث يصنف مستخدمي الشبكة إما عادي أو متطفل، المستوى الثاني حيث يمكن للنظام تحديد أربع فئات من التشذيبات (DOS, Probe, R2L & U2R)، المستوى الثالث هو مستوى الذي يتم فيه تحديد نوع الهجوم.

النموذج المقترح يتكون من مستويات متعددة مؤلفة من هجين الشبكات العصبية وشجرة القرار. قمنا بدراسة التقنيات المختلفة للشبكات العصبية وشجرة القرار على كل وحدة في كل مستوى، ويتم تنفيذ كل وحدة من كل مستوى باستخدام التقنية (شبكات العصبية أو شجرة القرار) التي أعطت أفضل النتائج التجريبية.

من خلال النتائج التجريبية على بيانات مختلفة للشبكة، فإن النموذج المقترح يحقق معدل تصنيف صحيح 93.64 %، ونطاق معدل اكتشاف ما يقرب من 98 %: 99.8 % للهجمات المعروفة و93.8 % للهجمات الجديدة الغير معروفة.
مميزات النظام المقترح هي القدرة على الكشف عن التسلل بمعدل مرتفع، بالإضافة إلى أنه إذا أضيفت هجمات جديدة من فئة معينة لقاعدة بيانات الهجمات فإنها ليس من الضروري إعادة تدريب كل وحدة النظام بل فقط الوحدة المتتأثرة بهذه الهجمات الجديدة، كما أن الهجمات التي أساء النظام تصنيفها يتم إعادة تصنيفها من قبل مسؤول الشبكة ثم يتم إعادة تدريب النظام، ويمكن تدريب الوحدة في أي وقت مما يجعل النظام قادر على التكيف مع أي بيئة جديدة أو أي هجمات جديدة في الشبكة، كما يتميز النظام بالقدرة على التعليم فهو يتفوق على الأنظمة السابقة بقدرته على الكشف عن الهجمات المعروفة والهجمات الجديدة التي لم يسبق للنظام التعرض لها، بالإضافة فإن كل وحدة يمكن تدريبها على جهاز مفصل في نفس الوقت مما يجعل وقت التدريب أقل.
Chapter 1

Introduction
Chapter 2

Related Works for Intrusion Detection Systems
Chapter 3

Theoretical Aspects of the Implemented System
Chapter 4

Proposed System Architecture
Chapter 5

Experimental Results
Chapter 6

Conclusion & Future Work
Appendix
Publications
Arabic Summary