Adaptive Real-time Anomaly-based Intrusion Detection using Data Mining and Machine Learning Techniques

Dissertation
in partial fulfillment of the requirements for the degree

Doktor der Ingenieurwissenschaften (Dr. –Ing.)

for submission to the

Faculty of Electrical Engineering / Computer Science
University of Kassel, Germany

Oral Examination date: 12 August, 2014

M.Sc. –Ing. Maher Salem

Advisors:
Prof. Dr. Kurt Geihs
Prof. Dr. Ulrich Bühler

Signed: Maher Salem                                Date: 20 August, 2014
DEDICATION

To my mother and father for your unwavering trust and invaluable support.

To my wife and son for their moral assistance.
ACKNOWLEDGMENTS

Indeed, it is not easy to convey my warm regards to all who have supported and encouraged me, but some of them deserve my sincere and heartfelt thanks.

I want to gratefully express my thanks to my adviser Prof. Dr. Ulrich Buehler (University of Applied Sciences Fulda); for his enthusiasm and diligence to accomplish this work and his important instructions that made this work a distinct and successful in every sense of the word. He has been supportive since the day I began working on the project SecMonet. In addition to his scientific support, I learned from him the true meaning of team spirit and creativity at work.

I would like to deeply thank my adviser Prof. Dr. Kurt Geihs (University of Kassel); for his valuable guidance and precious advices. I proudly learned from him discipline and perfectionism which could guide my steps toward the completion of work. Honestly, I will forthwith love and never forget once I meet them.

I also want to deliver my appreciation to my brother Essa for his cooperation in doing language check for the thesis as well as the rest of my brothers for their emotional support.

I will always be thankful for my colleagues in the group of network and data security for their help and I also thank everyone helps me to finish this journey.

Finally, I would like to extend my thanks to the German federal ministry of education and research for their financial support and JUMO GmbH for their technical support.
ABSTRACT

Interconnections between various networks help the emergence of several shortcomings such as generating voluminous data flow, intimidating services to be vulnerable, and increasing the amount of suspicious connections rapidly. In addition, malware solutions and standard security gateway such as the firewall system or the URL blocker have become partially untrustworthy due to the complexity of network traffic and the increase of vulnerabilities and attacks. These problems in network management are unrestrained and threaten the overall system components. Hence, one of the most key aspects in securing computer and communication networks nowadays is the ability to uncover malicious connections (or the so-called zero-day-attack) effectively, accurately and within sufficient detection time. On the other hand, current intrusion detection systems are most likely signature-based applications, which operate in the offline operational mode and, thusly, unable to detect new attacks. Accordingly, the fundamental problem of current Intrusion Detection System (IDS) can be summarized into two points. The first one has to do with the difficulty of processing the massive data flows and the second one is proposing an adaptive intrusion detection model which operates in real-time and efficiently reveals anomaly. In this dissertation, a comprehensive IDS framework has been proposed to overcome these shortcomings. The framework consists of two main parts. The first part, known as OptiFilter, is in charge of aggregating massive data flow, deploying a dynamic queuing concept to process the data flows, constructing sequential connection vectors accordingly, and exporting datasets in an appropriate format. On the other hand, the second part is an adaptive classifier that includes a classifier Model based on the Enhanced Growing Hierarchical Self Organizing Map (EGHSOM), a Normal Network Behavior model (NNB), and update models to keep the proposed framework adaptive in real-time. In OptiFilter, the tcpdump and SNMP traps have been exploited to aggregate the network packets and hosts events continuously. They have also been subject to further analysis, in that they have been converted to connection vectors, which are constructed based on several valuable and important features in the area of intrusion detection. Regarding the adaptive classifier, the intelligent artificial neural network model GHSOM has been intensively investigated and improved upon in order to be effective in classifying the constructed connection vectors into normal, anomaly or unknown during the online operational mode in real-time. In the current study, the original GHSOM approach has been enhanced with several contributions. Namely, it has offered classification-confidence margin threshold to uncover the unknown malicious connections, a stability of the growth topology by an expressive initialization process for weight vectors and reinforcing the final winner units, and a self-adaptive process to update the model constantly. Moreover, the main task of the NNB model is to further investigate the detected unknown connections from the EGHSOM and examine if they belong to the normal behavior model or not. However, during the online classification in real-time, network traffic keeps changing based on the concept drift which in turn leads to generate non-stationary data flows. Thus, in this dissertation, this phenomenon has been controlled by the proposed update models which use benefit of detected anomaly and normal connections to adapt the current EGHSOM and
NNB models. Hence, the updated EGHSOM model can detect new anomaly even if they appear in different structure and the updated NNB model can accommodate the changes in data flows of the computer network.

In the experimental study, the performance evaluation of the proposed framework shows very satisfactory results. The first experiment has evaluated the framework in the offline operational mode. In this regard, OptiFilter has been evaluated by available, synthetic and realistic data flows. In contrast, 10-fold cross-validation has performed on the adaptive classifier to estimate the overall accuracy using the above mentioned data flows. In the second experiment, the framework has been evaluated in a real 1 to 10 GB computer network, i.e. in an online operational mode in real-time. OptiFilter and the adaptive classifier have accurately performed in the sense that the first part constructs continuous connections from the massive data flow and the second part classifies them precisely. The final comparison study between the proposed framework and other well-known IDS approaches shows that the proposed IDS framework outperforms all approaches, especially by the following major points: handling the massive data flow, achieving the best performance metrics (such as the overall accuracy), uncovering unknown connections, and proposing an adaptive technique.
TABLE OF CONTENTS

Dedication ................................................................................................................................. iii
Acknowledgments ...................................................................................................................... iv
Abstract ..................................................................................................................................... v
Zusammenfassung ..................................................................................................................... vii
Table of Contents ...................................................................................................................... ix
List of Figures .......................................................................................................................... xii
List of Tables .......................................................................................................................... xiv
List of Algorithms ..................................................................................................................... xv
Acronyms ................................................................................................................................ xvi

CHAPTER 1. INTRODUCTION ......................................................................................... 1
  1.1. Background and Motivation ............................................................................................... 1
  1.2. Statement of the Problem .................................................................................................. 5
  1.3. Research Questions .......................................................................................................... 6
  1.4. Thesis Organization ......................................................................................................... 8

CHAPTER 2. INTRUSION DETECTION SYSTEMS ................................................. 9
  2.1. Overview .......................................................................................................................... 9
  2.2. IDS Categories based on Monitoring Target ................................................................. 11
    2.2.1. Host-based IDS ........................................................................................................ 11
    2.2.2. Network-based IDS .................................................................................................. 12
  2.3. IDS Categories based on Detection Method ................................................................. 14
    2.3.1. Signature-based IDS ............................................................................................... 14
    2.3.2. Anomaly-based IDS ............................................................................................... 15
    2.3.3. Specification-based IDS .......................................................................................... 16
  2.4. IDS Categories based on Superior Techniques ............................................................ 17
    2.4.1. Statistical Techniques ............................................................................................ 17
    2.4.2. Machine Learning Techniques ............................................................................... 19
    2.4.3. Artificial Neural Networks ..................................................................................... 25
    2.4.4. Clustering Techniques ............................................................................................ 34
  2.5. Shortcomings of Current Intrusion Detection ............................................................... 37

CHAPTER 3. RELATED WORK ...................................................................................... 39
  3.1. Related Contributions in Data Aggregation ................................................................. 40
  3.2. Data Preprocessing Methods ......................................................................................... 41
3.3. Distinguished Classification Achievements ................................................................. 44
3.4. Other Approaches in IDS ............................................................................................ 47

CHAPTER 4. PROPOSED METHODOLOGIES ................................................................. 49
4.1. Overview ...................................................................................................................... 49
4.2. Data Aggregation ........................................................................................................ 50
  4.2.1. Dataset .................................................................................................................. 50
  4.2.2. Proposed Aggregation Method ............................................................................. 54
4.3. Feature Selection and Extraction ................................................................................ 59
  4.3.1. Hybrid Selection Method: Theoretical Approach .................................................. 60
4.4. Preprocessing .............................................................................................................. 64
  4.4.1. Conversion of Nominal Features ......................................................................... 65
  4.4.2. Normalization Methods ....................................................................................... 66

CHAPTER 5. CLASSIFICATION ................................................................................... 68
5.1. Growing Hierarchical Self Organizing Map ............................................................... 68
5.2. Standard GHSOM Algorithm ..................................................................................... 70
  5.2.1. Fundamental Definitions ...................................................................................... 70
  5.2.2. Algorithm Main Steps ......................................................................................... 71
5.3. Proposed Theoretical Modifications on GHSOM ....................................................... 82
  5.3.1. Enhancement 1: Dynamic Classification-Confidence ............................................. 83
  5.3.2. Enhancement 2: Reinforcing Final Best Matching Units ....................................... 86
  5.3.3. Enhancement 3: Embedded Conversion and Normalization ............................... 90
  5.3.4. Enhancement 4: Meaningful Map Initialization .................................................... 91
  5.3.5. Enhancement 5: Topology Stabilization by Node Splitting ................................... 93
  5.3.6. Enhancement 6: Normal Network Behavior Model ............................................. 96
  5.3.7. Enhancement 7: Updating EGHSOM and NNB Models in Real-time ................... 98
5.4. Proposed Prevention Method ...................................................................................... 105
  5.4.2. Alarm Information Content ............................................................................... 105
  5.4.3. Proposed Alarm Categories ................................................................................ 106
5.5. Online IDS Framework .............................................................................................. 107
  5.5.1. Overview Online Framework .............................................................................. 107
  5.5.2. Part 1: OptiFilter ............................................................................................... 109
  5.5.3. Part 2: Online Adaptive Classification ............................................................... 110

CHAPTER 6. EXPERIMENTAL STUDY ........................................................................ 116
6.1. Preparation for Evaluation ......................................................................................... 116
LIST OF FIGURES

Figure 1-1: Number of weeks required to identify infected file ................................................................. 2
Figure 1-2: The growing cost of cyber crime................................................................................................... 2
Figure 1-3: Security status in Germany 2011 .................................................................................................. 3
Figure 1-4: Main steps of knowledge discovery in IDS ................................................................................. 4
Figure 1-5: Dissertation structure .................................................................................................................. 8
Figure 2-1: Work plan of major and minor steps .............................................................................................. 10
Figure 2-2: Proposed IDS taxonomy .............................................................................................................. 11
Figure 2-3: Scenario of HIDS over computer network .................................................................................. 12
Figure 2-4: Scenario of centralized NIDS over computer network ............................................................. 13
Figure 2-5: Scenario of decentralized NIDS over computer network .......................................................... 13
Figure 2-6: General diagram of signature-based IDS .................................................................................... 15
Figure 2-7: Example of anomaly connections ............................................................................................... 16
Figure 2-8: General diagram of anomaly-based IDS ..................................................................................... 16
Figure 2-9: Flowchart of supervised machine learning .............................................................................. 20
Figure 2-10: Optimum hyperplane separation in linear SVM ................................................................. 21
Figure 2-11: Flowchart of unsupervised machine learning ............................................................................ 23
Figure 2-12: General flowchart of semi-supervised machine learning ........................................................... 24
Figure 2-13: Basic illustration of the neuron in the human brain ................................................................. 25
Figure 2-14: Artificial neuron ....................................................................................................................... 26
Figure 2-15: sigmoid function with different values of $\alpha$ ...................................................................... 28
Figure 2-16: Artificial neural network architectures ..................................................................................... 29
Figure 2-17: Artificial neural network classification with respect to learning algorithms ...................... 31
Figure 2-18: Simple ANN numerical example .............................................................................................. 33
Figure 2-19: Example of hierarchical clustering algorithm .......................................................................... 35
Figure 4-1: An example of a dataset based on KDD features ...................................................................... 53
Figure 4-2: General overview of the proposed aggregation method ............................................................. 54
Figure 4-3: Internal block diagram of the queuing concept .......................................................................... 55
Figure 4-4: General diagram of the proposed feature selection method ..................................................... 61
Figure 4-5: Flowchart of SBS main steps ..................................................................................................... 62
Figure 4-6: Example of equal frequency discretization ................................................................................. 63
Figure 4-7: Flowchart of IG main steps ........................................................................................................ 64
Figure 5-1: Visual overview of SOM .......................................................................................................... 69
Figure 5-2: Cluster representation of a BMU and mapped input vectors .................................................... 69
Figure 5-3: Most used learning rate in the neighborhood area .................................................................... 74
Figure 5-4: Gaussian neighborhood function for the BMU and the neurons in the vicinity ...................... 75
Figure 5-5: neighborhood topology in a single SOM grid .......................................................................... 75
Figure 5-6: Gaussian and Bubble neighborhood functions ........................................................................ 76
Figure 5-7: Horizontal growth by adding row or column ............................................................................ 78
Figure 5-8: Example of a final GHSOM topology ...................................................................................... 79
Figure 5-9: Flow chart of the GHSOM algorithm main steps .................................................................. 80
Figure 5-10: illustration example of the average distance threshold ......................................................... 84
Figure 6-23: Number of processed packets by OptiFilter in real-time – Second scenario ..................................... 146
Figure 6-22: Most used services in real-time– Second scenario ............................................................................ 146
Figure 6-21: Attraction value of each BMU in real-time – First scenario ............................................................. 144
Figure 6-20: Active BMUs in real-time classification – First scenario .................................................................. 144
Figure 6-19: Distances of all connections to the EGHSOM model in real-time – First scenario ........................... 142
Figure 6-18: Classification-confidence margin threshold in real-time – First scenario ........................................... 142
Figure 6-17: Processing time of OptiFilter in real-time – First scenario ............................................................... 141
Figure 6-16: Number of constructed connections by OptiFilter in real-time – First scenario ................................ 141
Figure 6-15: Number of processed packets by OptiFilter in real-time – First scenario ......................................... 140
Figure 6-14: Training time after each fold in the offline cross-validation ............................................................. 137
Figure 6-13: BMUs number before and after merging in the offline cross-validation ............................................ 137
Figure 6-12: Total number of maps for each fold in the offline cross-validation .................................................. 136
Figure 6-11: Total number of detected unknown connections during the offline cross-validation ......................... 136
Figure 6-10: Total number of maps for each fold in the offline cross-validation .................................................. 136
Figure 6-9: Offline evaluation of OptiFilter for number of packets– Second scenario ........................................... 130
Figure 6-8: Offline evaluation of OptiFilter for number of connections– Second scenario .................................... 131
Figure 6-7: Offline evaluation of OptiFilter for number packet drops– Second scenario ....................................... 131
Figure 6-6: TPR results of SBS using the NN classifier ...................................................................................... 122
Figure 6-5: Offline evaluation of OptiFilter for processing time– Second scenario ............................................. 130
Figure 6-4: Offline evaluation of OptiFilter for number of packets– Second scenario ........................................... 130
Figure 6-3: Offline evaluation of OptiFilter – First scenario results ............................................................... 130
Figure 6-2: Venn diagram for all classifiers ......................................................................................................... 123
Figure 6-1: TPR results of SBS using the NN classifier ...................................................................................... 122
Figure 6-24: Number of constructed connections by OptiFilter in real-time – Second scenario ......................................................... 147
Figure 6-25: Processing time of OptiFilter in real-time – Second scenario ......................................................................................... 147
Figure 6-26: Classification-confidence margin threshold in real-time – Second scenario ................................................................. 148
Figure 6-27: Distances of all connections to the EGHSOM model in real-time – Second scenario ..................................................... 149
Figure 6-28: Active BMUs in real-time classification – Second scenario .......................................................................................... 150
Figure 6-29: Attraction value of each BMU in real-time – Second scenario ...................................................................................... 150
Figure 6-30: Adapted radiuses of the NNB model in real-time – Second scenario ........................................................................... 151
Figure 6-31: Adapted Classification-confidence margin in real-time – Second scenario ................................................................. 151

LIST OF TABLES

Table 4-1: Configuration parameters of the proposed aggregation method .................................................................................. 55
Table 4-2: Considered timeout values ................................................................................................................................................. 57
Table 4-3: Direction of established connection ................................................................................................................................. 58
Table 4-4: Considered flag values ......................................................................................................................................................... 58
Table 4-5: Widely used normalization methods ................................................................................................................................. 67
Table 5-1: Final GHSOM Model .......................................................................................................................................................... 82
Table 6-1: Confusion matrix ................................................................................................................................................................. 118
Table 6-2: TPR and FPR of neural network – multilayer perceptron ................................................................................................. 120
Table 6-3: Final feature sets of SBS for all classifiers .......................................................................................................................... 122
Table 6-4: Information gain of all features in the MVF set ................................................................................................................... 124
Table 6-5: Information gain of all features in the MVRF set ............................................................................................................. 124
Table 6-6: Final network and host features ........................................................................................................................................... 126
Table 6-7: Offline evaluation of OptiFilter – First scenario .............................................................................................................. 127
Table 6-8: Offline evaluation of OptiFilter – First scenario comparison ............................................................................................ 129
Table 6-9: Offline evaluation of OptiFilter – Second scenario ......................................................................................................... 129
Table 6-10: Offline evaluation of OptiFilter – Second scenario comparison ...................................................................................... 132
Table 6-11: Datasets characteristics for offline evaluation of EGHSOM .......................................................................................... 133
Table 6-12: EGHSOM configuration parameters for offline evaluation ............................................................................................ 133
Table 6-13: Confusion matrix from the 1-fold of the dataset SecMonet+anomaly ................................................................................ 134
Table 6-14: Performance evaluation result of the adaptive classifier using 10-fold cross-validation ................................................ 134
Table 6-15: Configuration parameters for the first scenario in real-time ........................................................................................ 139
Table 6-16: Confusion matrix for the adaptive classifier in real-time – First scenario ................................................................. 143
Table 6-17: Classified connections in real-time – First scenario ...................................................................................................... 144
Table 6-18: Configuration parameters for the second scenario in real-time .................................................................................... 145
Table 6-19: Confusion matrix of the adaptive classifier in real-time – Second scenario ................................................................. 149
Table 6-20: Classified connections in real-time – Second scenario ................................................................................................. 149
Table 6-21: Final comparison between proposed IDS approaches .................................................................................................... 153
LIST OF ALGORITHMS

Algorithm 4-1: Proposed Sequential backward search algorithm ................................................................. 61
Algorithm 5-1: GHSOM Algorithm main steps................................................................................................... 81
Algorithm 5-2: Addressing coherent BMUs ...................................................................................................... 89
Algorithm 5-3: Embedded conversion and normalization algorithm ................................................................. 90
Algorithm 5-4: Labeling unknown connections using NNB model................................................................. 98
Algorithm 5-5: Major steps of updating the EGHSOM model ....................................................................... 103
Algorithm 5-6: Major steps of updating the NNB model ............................................................................... 104
## ACRONYMS

<table>
<thead>
<tr>
<th>ACC</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>Artificial Neural Networks</td>
</tr>
<tr>
<td>ARP</td>
<td>Address Resolution Protocol</td>
</tr>
<tr>
<td>BMU</td>
<td>Best Matching Unit</td>
</tr>
<tr>
<td>BP</td>
<td>Back Propagation</td>
</tr>
<tr>
<td>CSV</td>
<td>Comma Separated Value</td>
</tr>
<tr>
<td>DARPA</td>
<td>Defense Advanced Research Projects Agency</td>
</tr>
<tr>
<td>DM</td>
<td>Data Mining</td>
</tr>
<tr>
<td>DMZ</td>
<td>Demilitarized Zone</td>
</tr>
<tr>
<td>DoS</td>
<td>Denial of Service</td>
</tr>
<tr>
<td>DR</td>
<td>Detection Rate</td>
</tr>
<tr>
<td>FPR</td>
<td>False Positive Rate</td>
</tr>
<tr>
<td>GHSOM</td>
<td>Growing Hierarchical Self Organizing Map</td>
</tr>
<tr>
<td>GPRS</td>
<td>General Packet Radio Service</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>HIDS</td>
<td>Host-based Intrusion Detection System</td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Markov Model</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>IANA</td>
<td>Internet Assigned Numbers Authority</td>
</tr>
<tr>
<td>ICMP</td>
<td>Internet Control Message Protocol</td>
</tr>
<tr>
<td>IDS</td>
<td>Intrusion Detection System</td>
</tr>
<tr>
<td>IG</td>
<td>Information Gain</td>
</tr>
<tr>
<td>IPS</td>
<td>Intrusion Prevention System</td>
</tr>
<tr>
<td>KDD</td>
<td>Knowledge Discovery in Databases</td>
</tr>
<tr>
<td>ML</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>MQE</td>
<td>Mean Quantization Error</td>
</tr>
<tr>
<td>NIC</td>
<td>Network Interface Card</td>
</tr>
<tr>
<td>NIDS</td>
<td>Network Intrusion Detection Systems</td>
</tr>
<tr>
<td>NMS</td>
<td>Network Management System</td>
</tr>
<tr>
<td>NNB</td>
<td>Normal Network Behavior</td>
</tr>
<tr>
<td>Pcap</td>
<td>Packet Capture</td>
</tr>
<tr>
<td>qe</td>
<td>Quantization error</td>
</tr>
<tr>
<td>RF</td>
<td>Receptive Field</td>
</tr>
<tr>
<td>SBS</td>
<td>Sequential Backward Search</td>
</tr>
<tr>
<td>SCADA</td>
<td>Supervisory Control and Data Acquisition</td>
</tr>
<tr>
<td>SFS</td>
<td>Sequential Forward Search</td>
</tr>
<tr>
<td>SNMP</td>
<td>Simple Network Management Protocol</td>
</tr>
<tr>
<td>SOM</td>
<td>Self Organizing Map</td>
</tr>
<tr>
<td>SPADE</td>
<td>Statistical Packet Anomaly Detection Engine</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TPR</td>
<td>True Positive Rate</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
</tbody>
</table>

xvi
This page intentionally left blank
Chapter 1. Introduction

1.1. BACKGROUND AND MOTIVATION

Information technology inflation and the expansion of heterogeneity in computer systems from industry to home entertainment lead networks to be more vulnerable, because they are interconnected through the Internet, which unfortunately makes them an exposed victim to cyber attacks. Furthermore, using recent technologies in small networks such as the wireless home automation networks has also become a daily activity, which, in consequence, will be targets for new security threats[1].

Cyber criminals are spreading and threatening networks’ infrastructure and causing a real disaster like the worm Stuxnet, the malware Flame, and the worm Duqu [2], [3]. In the last few years, most famous social networks such as Facebook, Google, and Twitter became victims to attackers due to their broader connectivity. Moreover, hackers used to embezzle passwords and private users’ information from well-known online services such as Evernote platform [4]. In addition, in 2009 the armed forces network in France was infected by the worm Conficker [5]. Concerning these issues, several Network Management Systems (NMS) have been modified to preserve network metrics like availability, reliability, resilience, and functionality.

Gradually, NMS modifications become more and more complex and inconvenient. Note that the more inconvenient the modification, the worse the security will become. Therefore, the need for a specific solution that detects and prevents attacks has become vital. One of the most known solutions is installing a firewall system to protect the network by filtering and blocking unwanted traffic. However, the firewall solution suffers from different drawbacks such as the inability to detect attacks from an internal network, and it is not capable of protecting the network against cyber-attacks, especially zero-day-attacks [6]. Hence, antimalware applications overcome firewall drawbacks by revealing
several types of attacks such as worm, Malware, Trojan, Denial-of-Service, etc. Basically, antimalware applications store signatures (or patterns) of known attacks in a database to detect them once they appear in the future. However, these databases need to be updated with the new attack patterns, which, in turn, make these antimalware applications unfavorable for the consumers. Statistical reports and analysis results show the tradeoff between using the antimalware applications and the amount of detected attacks, as well as other related metrics. In this regard, the security firm Imperva has revealed that less than 5% of the top 40 antimalware applications are able to detect new attacks. They have tested more than 80 new attacks on these antimalware applications and have shown that many applications need more than a month to detect these attacks [7]. This firm has performed the comparison between known antimalware applications and the comparison was about the number of weeks required from the antimalware to detect new attacks. Figure 1-1 shows the results of the comparison.

![Figure 1-1: Number of weeks required to identify infected file](image1)

Similarly, HP confirms this result and shows that it is very clear that cyber attacks are still growing. Growth is seen in cost, attack’s type, time to discover attack, and successful attacks as shown in figure 1-2 [8].

![Figure 1-2: The growing cost of cyber crime](image2)
1.1. Background and Motivation

Other resources in this concern are available in Symantec security response, which offers continuously updated white papers relating to Internet security [9]. Regarding Germany, the German federal office for information security (BSI) addresses the most common threats and their spread throughout Germany [10]. It emphasizes that uncovering attacks become a serious challenge as shown in figure 1-3.

![Development of spam in Germany](image)

![Intensity of DDoS attacks](image)

![Spam distribution by country](image)

Figure 1-3: Security status in Germany 2011

Obviously, securing computer networks has become an essential problem especially by the expansion of the Internet and the increase of interconnection in social networks. The shortages of firewall systems and antimalware applications, and the necessity to establish a secure network, lead to data mining and machine learning techniques to present the intrusion detection system. These kinds of securing systems can predict the anomalous traffic beforehand and defend accordingly. Intrusion Detection Systems (IDS) are commonly categorized based on the detection method (signature-based, anomaly-based, and specification-based). Briefly, the signature-based IDS compare the traffic to
stored signatures or patterns of known attacks and raise an alarm once a match is found. In contrast, anomaly-based IDS depend on a predefined normal network behavior profile and hence they predict any deviation from the profile as anomaly. Finally, specification-based IDS determine a system profile from several states, parameters, and characteristics of the monitored environment and predict any activity out of this profile as anomaly. However, anomaly-based is the detection method that attracts researchers’ interest more than any other methods.

Data Mining (DM) is considered as a process of knowledge discovery [11], which is concerned with uncovering patterns, associations, anomalies, and events in data. It can not only help us in knowledge discovery, that is, the identification of new phenomena, but it is also useful in enhancing our understanding of known phenomena. It is widely used in information processing and in statistical computation such as the mutual information method, information gain ratio, and clustering. Figure 1-4 shows the main steps of knowledge discovery in IDS.

Intuitively, researchers exploit DM in image processing and pattern recognition to predict patterns. Indeed, this idea has inspired the security researchers to build an IDS model that is able to detect anomalous traffic in computer and communication networks. In general, any IDS model gets the benefit of the aggregated data, preprocesses them to present a suitable data format, and then, utilizes the DM to build a model that is able to predict anomalous traffic. Machine learning techniques are considered specific techniques in DM. They are categorized in supervised learning, unsupervised learning, and semi-supervised learning techniques. The most applied techniques are the unsupervised and semi-supervised such as Artificial Neural Networks (ANN), where the network data are modeled in a form that is similar to the neural system of the human brain [12],[13]. Hence, ANN becomes the most used application in network IDS [14]. One of the special approaches of ANN is the Kohonen Self Organizing Map (SOM) [15],[16],[17],[18]. SOM can discover knowledge in datasets, create abstracts relations of high-dimensional data, and map these data into a two-dimensional representation space. These characteristics in SOM have shown efficiency in detecting anomaly in computer networks. However, it has several shortages such as: the static architecture; highly computationally expensive; and the random initialization of all parameters. GHSOM solved these problems by structuring several SOMs in hierarchical growing form [19].
On the other hand, several pioneer companies have produced antimalware applications to defend anomaly in networks. The most known open source application is Snort [20],[21]. It is considered as a signature-based detector that is capable of detecting anomaly in networks based on configured rules and a complex statistical engine. This application is widely used and therefore has become the de facto standard for intrusion detection. The disadvantages can be summarized by the inability to detect new attacks and unknown ones, and by its incapability to be self-adaptive. Several plug-ins have been presented based on Snort, such as the Statistical Packet Anomaly Detection Engine (SPADE) that employs statistics in network traffic to detect anomaly [22]. This platform has achieved a degree of accuracy but unfortunately only for known attacks. Other intrusion detections, which are considered as signature-based, are Prevx (Webroot) [23], Suricata [24], and Airsnare [25]. Unfortunately, all open sources or commercial antimalware applications in IDS could not be listed in this thesis, but most are suffering from the above mentioned disadvantage or they are focusing only on a certain host instead of the entire range of computer networks. In addition, others are specialized only in Internet traffic and others only in certain protocol, such as file transfer protocol transactions over network. In this thesis, different proposed methods and approaches will be presented to overcome previously mentioned drawbacks and shortcomings. These approaches aim to propose a significant feature selection method, a sufficient data aggregation, a normal network behavior model, and an accurate classification model. All of these methods will operate online and they will be implemented to be a real-time framework that secures computer networks, constricts vulnerabilities, remedies drawbacks of IDS, and detects new and unknown anomalies on networks in the best sufficient time.

1.2. STATEMENT OF THE PROBLEM

The growth of information technology in several branches, and the linkage between several communication networks, serve to increase system vulnerabilities and offer new attacks the possibility to explode rapidly [26]. Therefore, antimalware applications cannot always detect new attacks sufficiently. Moreover, the problem of being a victim becomes profound since authorized users misuse their privileges or normal users are not aware about new techniques of attacking such as backdoors or driven download. In addition, big data, as a counterpart to the voluminous data flow, rapidly compound the amount of data flow and then degrade the overall performance of the IDS model [27],[28]. Therefore, capturing the online data flow and preprocessing them to an adequate format for the IDS model becomes a serious challenge in network management. On the other hand, IDS models either
machine learning-based or data mining-based are not able to accommodate the huge amount of unknown flows and hence classify them often incorrectly, which in turn increases the false positive alarm rate. Moreover, these models still need to be adaptive to avoid any detected suspicious connection once it reappears on the concerned network. Thus an enhanced and adaptive detection engine that is able to effectively uncover anomaly and classify the unknown flows in real-time is becoming an essential demand in the area of IDS.

Therefore, IDS are considered as the main defender against anomaly in addition to the firewall system. Intrusion detection is the act of detecting anomaly that attempt to expose the confidentiality, integrity or privacy of a resource [29]. It must also be efficient enough to handle large amounts of network data without affecting the performance, i.e. mitigate the overall overhead on the network. Intuitively, an intrusion detection system discovers any abnormal behavior on the network when this behavior deviates from a predefined Normal Network Behavior model (NNB). Hence, defining a NNB for heterogeneous and non-stationary data flow is a serious challenge, which, rather than complicated, needs to be adaptive and reconfigurable. This is to imply that NNB is an essential component in IDS although it is practically infeasible, due to the non-stationary traffic and the expansion of information technology. In summary, this thesis addresses the following challenges to build an adaptive real-time IDS:

- Aggregate network data and host events and construct continuous connection vectors online in sufficient time.
- Select the most significant network and host features (attributes) which enhance the accuracy and reduce data dimensionality.
- Preprocess network data and host events to prepare a proper format to the intrusion detection model.
- Define a normal network behavior that supports the IDS by detecting or labeling unknown anomaly.
- Remedy the shortages Growing Hierarchical SOM (GHSOM) and present an enhanced model EGHSOM to detect anomaly and unknown anomaly in data flow.
- Develop an online IDS prototype

In addition, several minor drawbacks will be also treated tacitly like the alarm information content.

### 1.3. Research Questions

According to the addressed challenges in the previous section, this thesis resolves several research questions and opens issues in the area of IDS:
1.3. Research Questions

**Research Question 1**: Which network features are valuable and relevant in the research area of intrusion detection systems?

**Research Question 2**: What is the optimal way to aggregate network traffic in real-time and construct connection vectors continuously?

**Research Question 3**: What is the best possible method to define a normal network behavior? How is it possible to mitigate the impact of concept drift?

**Research Question 4**: Which enhancements are necessary to improve the GHSOM model?

**Research Question 5**: What are the possibilities to achieve an adaptive intrusion detection framework that reduces overhead and improves performance?

These research questions are addressed based on intensive study on the recent achievements in science and praxis. Although several proposals achieve a high degree of preciseness, after performing an intensive literature review it was found they still suffer from various salient problems. Anomaly-based detection systems is a relatively new research field that aims to achieve a high detection rate and low false positive rate. However, achieving a superior detection model requires accurate investigation and robust preparation. Therefore, this thesis has outlined the main questions in the IDS area. The objective of research question 1 is to extract only relevant network and hosts’ features for IDS, which are necessary to achieve a high detection rate and minimum false positive rate accordingly.

On the other hand, research question 2 aims to capture network and host data continuously in real time. Although, aggregating live stream is infeasible the objective is to gather network traffic and host traps as they appear to the best possible performance.

Research question 3 investigates the possibility of achieving an adaptive normal network behavior that is able to predict normal or abnormal connections from unknown data flow and mitigate the problem of concept drift.

The aim of research question 4 is to intelligently enhance the classifier model and remove all possible drawbacks which affect the performance. Therefore, this research question focuses on enhancing the GHSOM to overcome its shortages and drawbacks.

Research question 5 examines the possibility of achieving an adaptive framework that is able to accommodate the massive data flow and adapts the models and parameters continuously.
1.4. Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 focuses on a brief and delicate clarification about intrusion detection systems. This chapter illustrates main types of IDS and the most applied methods or techniques in this regard. Furthermore, some major shortcomings in the current IDS applications are also invoked. Chapter 3 presents an intensive and comprehensively related work in the following domains: data aggregation, feature selection and extraction, preprocessing, and classification. In addition, it presents recent scientific and industrial achievements in intrusion detection systems. Moreover, other minor topics will be mentioned such as concept drift. In this regard, Chapter 4 precisely illustrates the proposed methodologies in this thesis and explains each methodology individually. Particularly, chapter 5 concentrates on the GHSOM classifier model due to its importance. This chapter presents all enhancements on GHSOM, a short explanation about the prevention techniques and a suggested categorization for alarms and finally it clearly describes the proposed online IDS framework.

The IDS model undergoes several test scenarios in the offline and online operational mode and the results of the performance evaluation and various comparison studies are depicted and discussed in Chapter 6. Finally, Chapter 7 concludes this work, suggests other improvements for future work, and indicates some open issues. Additionally, some appendices are affixed to the end of this documentation.

In a few words, this thesis addresses the most common shortages of the recent IDS and treats each shortage individually to offer an adaptive and superior model that overcomes these shortages. Figure 1-5 illustrates the structure of this dissertation.

![Figure 1-5: Dissertation structure](image)
Chapter 2. Intrusion Detection Systems

2.1. Overview

Intrusion detection systems are considered one of the fundamental applications in combating cyber attacks. They aim to enhance the network performance and reduce vulnerabilities by building a model that learns system data. Generally, intrusion detection consists of four main steps, which are data aggregation, feature selection and preprocessing such as discretization and normalization, classification and prevention.

This chapter briefly illustrates our proposed taxonomy of IDS based on:

- The installation area: i.e. on a single host or an entire network.
- The detection approaches: detecting suspicious traffic based on prior known patterns, based on a predefined system model, or detecting threats based on certain specifications.
- Superior technologies: building the IDS model using various technologies such as statistical techniques, machine learning, artificial neural networks, or clustering.

In the last decade, building reliable IDS models has become a serious interest from security experts and researchers. The models analyze system data and defend against anomaly to optimize the network performance. In addition, IDS models have been showing promising achievements against cyber attacks by using superior technologies such as artificial neural networks or clustering approaches.

According to the proposed taxonomy, major and minor steps of the work plan in this thesis have been summarized in figure 2-1.
Forthcoming sections of this chapter explain the proposed taxonomy of IDS in this thesis and present a short example for each category individually.

The proposed taxonomy of IDS considers the following factors: diversity, accordance and vitality. The first factor examines the detection methods, which are differentiated between signature-based, anomaly-based and specification-based. Whereas the second factor explores the monitored target such as a single host or an entire network. The last factor indicates the most vital technologies to build an effective detection model. Although, this thesis focuses on network data and host events to monitor the entire network, i.e. network-based IDS, a short overview about host-based IDS will be presented. Similarly, the main target of this work is to achieve an anomaly-based detection model, however, other detection methods like signature-based and specification-based will also be explained. Finally, known technologies in building IDS models in network security will be expressly presented. Figure 2-2 demonstrates the proposed taxonomy of IDS.
2.2. IDS Categories based on Monitoring Target

2.2.1. Host-based IDS

The detection model that concerns only the data of a single host (such as a computer or smartphone) is called host-based IDS, also known as HIDS. Basically, HIDS performs most probably the same tasks as many virus scanners. It analyzes several resources of the operating system to detect malicious content or intrusion that breaches from the outside. Basically, HIDS examines log files or events and compares them against an internal database of known signature patterns. As under UNIX systems, HIDS can also verify the data integrity of important files and executables. It checks a database of sensitive files and creates a checksum of each file with a message-file digest utility, such as md5sum or sha1sum. It then periodically compares the file checksums against the stored values. If any of the file checksums do not match, the HIDS informs the administrator by email or cellular pager [30].

This type of IDS uses pattern-based or anomaly-based detection methods to prevent suspicious activities. J. Hu et al. have addressed the host-based anomaly detection by exploiting an efficient Hidden Markov Model (HMM) [31]. They have proposed a simple data preprocessing approach to speed up a HMM training for system calls up to 50 percent. On the other hand, L. Ying et al. have designed and implemented a HIDS that combines log file analysis and back-propagation neural networks together [32]. Their IDS can sufficiently monitor various log files such as: firewall log, router log, web server log and other logs. Similar work on Windows operating systems has also been provided [33]. Industrial products, beside the scientific achievements, provide a HIDS as

![Figure 2-2: Proposed IDS taxonomy](image-url)

Brackets in the above figure show the adopted methodologies in this thesis. Note that, other methods (such as clustering) are also considered in this thesis because they partially support building the proposed IDS model.
an open source or commercial solution such as the known host-based OSSEC [34], Swatch [35] and LIDS [36]. As a result, the HIDS has the following advantages:

- Monitor specific system activities, e.g. monitoring only file accesses.
- Detection of some attacks better than network-based IDS, e.g. buffer overflow.
- Required no further hardware installation.
- Less expensive than network-based.

Figure 2-3 illustrates a common scenario of host-based intrusion detection.

![Figure 2-3: Scenario of HIDS over computer network](image)

The previous figure shows general and simple network architecture, where the HIDS is installed only on the red hosts.

### 2.2.2. Network-based IDS

Network-based intrusion detection, also known as NIDS, sniffs raw network packets at the router or even at host level to detect anomaly. Particularly, NIDS scans the headers of network packets (or even the complete packet with payload) and compares them with an attack-patterns database. Once a match is found, it treats the suspicious packets and assigns an alarm via several means such as email or informative messaging. For current large scale and heterogeneous networks, the necessity to deploy an NIDS becomes essential. Mainly, deployment scenarios on computer networks are centralized and decentralized. In the centralized scenario, a single NIDS component is installed in a position that stands for a center of the entire network. This scenario is considered easy, inexpensive, and fast configurable. However, it degrades the overall network performance and causes packet loss when the amount of devices is large enough. Hence, this scenario is not preferable by firms and it is advisable only for evidently small networks. A scenario of centralized NIDS is shown in figure 2-4.
2.2. IDS Categories based on Monitoring Target

The second scenario is more sufficient and practical, but it has several installation concepts which make it, in comparison to the centralized option, a real challenge for the administrator or the IT Architect. Conceptually, the decentralized NIDS handles network segments and reacts upon each segment data individually; however, other installation approaches have mixed the centralized and decentralized due to the growth and expansion in information technology and due to the current complex network infrastructure. Figure 2-5 demonstrates a general installation concept of decentralized NIDS.

In addition to the previous two scenarios, there are several scenarios of NIDS, but they are varying in concept and rely on the network infrastructure. Therefore, an optimal installation of NIDS is considered a very critical issue in computer and communication networks. In short, NIDS’s are achieving an efficient performance but they still have various limitations [37]. For instance,
packets drop at high speed networks is considered one of the serious challenges for NIDS’s especially the centralized scenario. In this regard, several solutions have been conducted to resolve this issue [38],[39],[40], and others. On the other hand, decentralized NIDS’s still suffer from some limitations such as the complexity of communicating between distributed IDS’s components. This issue has been partially resolved using a collaborative approach between IDS’s [41],[42]. Similarly, The idea of developing distributed IDS has been also examined by D. Fisch et al.[43]. They have proposed a framework for large-scale collaborative intrusion detection agents (IDA). The framework includes no central control unit, but rather distributed and self-organized agents which operate in four separate layers: sensor layer, detection layer, alert correlation layer, and reaction layer. These agents work in a decentralized manner by data acquisition, data analysis and data communication. The framework focuses on large-scale environment where agents can communicate and synchronize information. However, agent solutions sometimes are not preferable due to time consuming in communication and may fail by data exchange. In addition, in detection layer, the used classifiers are taken from the tool RapidMiner without any further investigation. Therefore, D. Fisch, F. Kastl, and B. Sick have proposed an organic computing technique for attack recognition based on probabilistic rule modeling [44]. The work lays out the IDA methodology that recognizes new attacks and reacts by creating new rules and exchanging them with other agents. Although the proposed technique has been evaluated by the offline DARPA traffic, it achieves a self-adaptive and promising result, especially in the area of organic computing.

2.3. IDS Categories based on Detection Method

2.3.1. Signature-based IDS

Signature-based or misuse IDS’s are well-known and very effective against known attacks. Examining misuse detection started with the report from Anderson in 1980 [45], which states that any Interference is uncovered by matching the current behavior with known malicious patterns, but, it is impossible to detect unknown behavior due to non-existence of a matching pattern. Principally, signature-based IDS consists of main three components: traffic aggregation (such as network packets and system calls); profiler that builds a model from the monitored system and from patterns of known attacks; and a misuse detection engine that detects suspicious connections based on the system model. Once the misuse detection finds a match, it will react by a certain prevention technique. Note that the profiler should define an accurate system model to reduce false alarms. However, suspicious patterns are
2.3. IDS Categories based on Detection Method

constantly changing, which makes signature-based IDS insufficient. Figure 2-6 shows a general diagram of signature-based IDS.

![General diagram of signature-based IDS](image)

Data collection sends an understandable format for the misuse IDS engine that adheres to the model sent from the profiler. Moreover, the profiler builds a model based on one of the following techniques to detect intrusions: pattern matching, rule-based, or state-based. This is to imply that the detection engine uses the same technique chosen from the profiler to detect malicious behavior.

Snort is considered one of the most used signature-based IDS’s [20],[21]. It is a misuse detector that uses the rule-based technique to detect malicious activity. Its major disadvantage is the high false positive rate. Another widely used signature-based IDS is the Bro IDS [46]. It has several modifications superior to Snort such as flexibility, efficiency, and adaptability. However, it still suffers from the inability to reveal unknown anomaly.

2.3.2. Anomaly-based IDS

In contrast to misuse detection, anomaly-based IDS’s are able to detect unknown suspicious connections. D. Denning was the first researcher who proposed a real-time intrusion detection model in 1987 that uncovers abnormal behavior in computer networks [47]. Since that time, the research in this field has increased enormously [48]. Fundamentally, anomaly-based IDS’s reveal threats using two main steps, which are creating a normal profile from system behaviors and activities, and then detecting any deviation from this normal profile as anomalous. Anomaly-based IDS uses three main techniques to define the normal profile or to learn the IDS model, supervised, unsupervised, and semi-supervised learning, which are discussed in 2.4.2. For an observed computer network, it is necessary to know the attributes of this network to be able to define an NNB. It is worth mentioning that defining an NNB is almost infeasible due to concept drift dilemma and to the fact that computer networks are considered as non-stationary environments. Several articles have examined these shortcomings and provided a proper solution. For example, some researchers tend to build a model
from both normal and anomalous data to overcome the previous shortcomings [49],[50]. Anomaly-based IDS’s mainly include four components: the data collector, anomaly model, normal profiler, and response. Normally, anomaly instances are the outermost activities from other instances (also known as outliers) as shown in figure 2-7.

![Figure 2-7: Example of anomaly connections](image)

The general model of anomaly-based IDS can be structured from only labeled normal traffic or mixture of data. Normal instances can be used to build the normal profiler (or normal network behavior). Then incoming connections will be used to evaluate the model if it is able to detect anomalous instances. Figure 2-8 shows a general diagram for anomaly-based IDS.

![Figure 2-8: General diagram of anomaly-based IDS](image)

The model totally depends on the normal profiler, which leads then to incorrect detection in case of any flaw in building the normal profiler. Examples of industrial products in anomaly-based IDS are Juniper [51] and Enterasys [52].

### 2.3.3. Specification-based IDS

In contrast to signature and anomaly-based IDS’s, specification-based IDS’s abstract several specifications from system behavior to detect exploitation of
vulnerabilities in security-critical programs. This idea was first introduced by C.Ko et al. in 1996 [53], in which programs behaviors are exploited to describe a security specification approach that is able to detect attacks gradually. In this kind of IDS's there is no learning process or a model from data. They build a specification for a system based on experts' knowledge. Accordingly, these specifications are describing the allowed system behavior, which is used to classify incoming events as attacks if they deviate from the allowed behavior. The most applicable area of specification-based IDS is the current technology of smart metering [54]. In this area, researchers try to define the most allowable behavior to combat suspicious behaviors in smart grid. Regrettably, this type of IDS is not as effective as the other two types, because of the complexity of defining all allowed system behaviors. Therefore, the use of specification-based IDS is still limited and not widely deployed.

2.4. IDS Categories based on Superior Techniques

Generally, building the IDS model follows the steps in figure 1-4 to achieve an effective detection. IDS models can be association analysis, classification and prediction, clustering analysis, outlier analysis, or evaluation analysis. On the other hand, data mining performs a predictive task to examine an investigation on data in order to make predictions [55]. This means, IDS (as the case with most other applications) uses DM in the following:

- Abstract association rules for a given set of data.
- Preprocess a certain amount of data to reconstruct an adequate format. Some preprocessing methods are harvesting, purgation, discretization, transformation, normalization.
- Building models for classification and pattern recognition. These models are basically generated from training data and evaluated by test data.
- Clustering analysis. Grouping mixture of data in several groups, where each group shares the same label.
- Evaluation of models.

The IDS model in this thesis focuses on classification and prediction. The technologies used to achieve such a model are: statistical, machine learning, artificial neural networks, and clustering techniques. The next sections present a glance at each category and propose an example for illustration.

2.4.1. Statistical Techniques

Several contributions consider statistical techniques as a main category of DM, and others are still against this categorization [56]. Statistical techniques explore data and reveal some characteristics and performance studies to enhance the
concerned system and improve the performance accordingly. Therefore, this thesis considers statistical techniques as a DM category.

Statistical techniques in classification and prediction use mathematical computation to ease understanding of the data. They have an explicitly underlying probability model that provides certainty about instances to which label or class they belong to. Likewise, statistical techniques can define a profile for the presented data to uncover uncertainty in the investigated system. Hence, using statistics to classify new instances and then predict a proper class (that most probably related to the instance) is considered an active area in DM. Nonetheless, DM is considered often as an extension to the statistics. General techniques in statistics are: linear discriminant analysis, maximum entropy, and Bayesian networks [57].

**Naive Bayes Classifier**

In this thesis, a short overview about naïve Bayes classifier will be introduced because it is the most well-known statistical technique in classification and prediction.

Principally, Bayesian classifiers can predict class membership probabilities, such as the probability that a given sample belongs to a particular class. Naive Bayesian classifiers assume that each feature is conditionally independent of every other feature in a given class. This assumption is called class conditional independence, and in this sense, is considered “naive”. Naïve Bayes classifier is based on the following Bayes theorem

\[
P(A | B) = \frac{P(B | A) \cdot P(A)}{P(B)}. \tag{2-1}
\]

Where \(P(A | B)\) is the probability that \(B\) belong to class \(A\), \(P(B | A)\) is the probability of occurring instance \(B\) given class \(A\), \(P(A)\) is the probability of occurrence of class \(A\), and \(P(B)\) is the probability of instance \(B\) happening.

Let \(D\) be a training set with samples and each sample \(X = \{x_1, x_2, ..., x_n\}\) has its class label from \(C = \{C_1, C_2, ..., C_k\}\). For a given sample \(X\), naïve Bayes classifier will predict that \(X\) belongs to the class \(C_i\) having the maximum posteriori probability. Therefore, we need to find the class that maximizes \(P(C_i | X)\), which is called the maximum posteriori hypothesis by interpreting it in Bayes theorem

\[
P(C_i | X) = \frac{P(X | C_i) \cdot P(C_i)}{P(X)}. \tag{2-1}
\]

To reduce the computation of \(P(X | C_i) \cdot P(C_i)\) the naïve assumption is considered here, which postulates that the values of the features are conditionally independent. Formally, that yields to
\[ P(X | C_i) \approx \prod_{j=1}^{n} P(x_j | C_i) \]  

If the feature is discrete (categorical), then \( P(x_j | C_i) \) is the number of samples of class \( C_i \) in \( D \) having the value \( x_j \) of this feature, divided by number of class \( C_i \) in \( D \). However, if the feature is continuous, then we assume the values have a Gaussian distribution shape with mean \( \mu \) and standard deviation \( \sigma \) as

\[ g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \]  

so that \( P(x_j | C_i) = g(x_j, \mu_{C_i}, \sigma_{C_i}) \). Finally, the classifier predicts that the class label of \( X \) is \( C_i \) if and only if it is the class that maximizes \( P(X | C_i) \cdot P(C_i) \) [58].

### 2.4.2. Machine Learning Techniques

Machine Learning (ML) can be defined as a combination of mathematical descriptions and practical applications to construct a computer program, which learns from system data and serves to enhance this system. Although the relation between DM and ML is still obfuscated, they are very bonded together. ML applications use several mathematical formulations to build a model that is able to predict in real-time an incoming object or event in the concerned system. Hence, the major idea is to mine and handle data to achieve an accurate prediction or classification. Therefore, DM methods are considered complementary to the ML techniques based on relevancy and originality. Compactly, this section addresses the main three types of ML, which are supervised learning, unsupervised learning, and semi-supervised learning.

#### 2.4.2.1. Supervised Machine Learning

Basically, the supervised learning requires a training dataset which has labeled connections for normal and anomaly class to build the classifier model. New connections or the test dataset will be presented to the classifier model which predicts the class label it belongs to. In short, the machine is given desired labels and its goal is to learn to produce the correct output for the given new input [59]. Figure 2-9 illustrates a flowchart of main steps in supervised ML.
20

2.4. IDS Categories based on Superior Techniques

Hence, the basic concept of supervised ML is receiving labeled input samples $X$ and generating the output $Y$. Generally, the training algorithm receives labeled instances as $\{(x_1, y_1), \ldots, (x_M, y_M)\}$, where $x_i \in X, y_i \in Y$, then outputs a hypothesis $g : X \rightarrow Y$ so that it can predict the label of the new input $g(x_i)$.

There are some issues in supervised ML which affect the final model, such as choosing the optimal size of the training dataset in comparison to the test dataset. If the test dataset is too small while the training dataset is too large, it will lead to the overfitting issue, which leads to a misclassification in detection. Several algorithms use labeled datasets to build a classifier model and the most effective one in this regard is the support vector machine. Therefore, this thesis briefly presents it as an example of supervised ML.

Support Vector Machine

The basic idea of SVM is to find the largest margin between input instances to separate data classes by an optimal separating hyperplane. Hence, maximizing the margin ensures finding the largest possible distance between the hyperplane and the input instances [59],[60]. An exquisite survey and tutorials can be found in [61],[62]. Let $\{x_i, y_i\}$ be our training data that contains $M$ instances and each instance $x_i$ has $n$ attributes and one class label $y_i = -1 \lor y_i = +1$ such that $x_i \in \mathbb{R}^n, i = 1, \ldots, M, y_i \in \{-1, +1\}$. Accordingly, if the training data are linearly separable then this is a linear SVM, and we can separate them by a line when $n = 2$ or by a hyperplane when $n > 2$. The hyperplane can be expressed as $(w \cdot x) + b = 0$, where $w$ is the normal to the hyperplane and $b$ is the bias. The
Figure 2-10 illustrates a linear SVM where the points on the lines are the
support vectors.

![Image of SVM with support vectors](image)

**Figure 2-10: Optimum hyperplane separation in linear SVM**

Training data on the figure can be described as

\[
\begin{align*}
(w \cdot x_i) + b &\geq +1, \quad y_i = +1 \\
(w \cdot x_i) + b &\leq -1, \quad y_i = -1
\end{align*}
\]  

These equations lead to

\[y_i \cdot ((w \cdot x_i) + b) - 1 \geq 0\]  

(2-4)

According to a simple geometry, the margin separates the support vectors is the
difference between \(H_2\) to the origin and \(H_1\) to the origin equals to \(\frac{2}{\|w\|}\). As
mentioned before, maximizing the margin will lead to an optimum separation,
which is equivalent to obtaining \(\min \|w\|\) such that \(y_i \cdot ((w \cdot x_i) + b) - 1 \geq 0\). In
other words, the optimum hyperplane separation can be found by minimizing the
squared norm of the separation hyperplane, so we need to find

\[
\min \frac{1}{2} \|w\|^2 \quad s.t. \quad y_i \cdot ((w \cdot x_i) + b) - 1 \geq 0
\]  

(2-5)

This problem is difficult to solve due to the complex constraints. Therefore, we
use the Lagrangian duality theory (Lagrangian multipliers \(\alpha\)) to simplify this
problem and solve it [63] which gives

\[
L_p = \frac{1}{2} \|w\|^2 - \sum_{i=1}^{M} \alpha_i y_i \cdot ((w \cdot x_i) + b) + \sum_{i=1}^{M} \alpha_i
\]  

(2-6)
Based on the equation, we need to find $w$ and $b$ which minimizes, and $\alpha_i$ which maximizes such that $\alpha_i \geq 0 \ \forall i$. Consequently, we differentiate $L_p$ w.r.t. $w$ and $b$ and equaling it with zero to get

$$w = \sum_{i=1}^{M} \alpha_i y_i x_i \sum_{i=1}^{M} \alpha_i y_i = 0$$  \hspace{1cm} (2-7)$$

then substitute (2-7) into (2-6) to become a formula $L_D$ depends only on $\alpha_i$. So by minimizing $L_p$ and maximizing $L_D$ we become a convex quadratic programming problem that can be solved using a quadratic programming solver to return $\alpha$ and $w$. Hence, any point satisfying the condition $\alpha_i > 0$ is called a support vector. For instance, in figure 2-10, support vectors are bold circles on $H_2$ and $H_1$ hyperplanes. Finally, by using the support vectors we can determine $b$ for each support vector as

$$b = y_s - \left( \sum_{m \in S} \alpha_m y_m x_m \right) \cdot x_s$$  \hspace{1cm} (2-8)$$

Where $S$ is the set of all support vectors $m$.

As a result, each point can be classified as

$$\text{class}(x_i) \begin{cases} +1 & \text{if } (w \cdot x_i) + b > 0 \\ -1 & \text{if } (w \cdot x_i) + b < 0 \end{cases}$$  \hspace{1cm} (2-9)$$

In contrast, if the training data cannot be linearly separated then this is a nonlinear SVM. Hence, we can separate them by a hyperplane but not the same as linear SVM. Accordingly, the optimum hyperplane can be found by presenting a positive slack variable $\xi_i, i = 1, \ldots, M$ such that the data can be described as

$$\begin{cases} (w \cdot x_i) + b \geq +1 - \xi_i, y_i = +1 \\ (w \cdot x_i) + b \leq -1 + \xi_i, y_i = -1 \end{cases}, \xi_i \geq 0  \hspace{1cm} (2-10)\)$$

In this case we need to find

$$\min \frac{1}{2} ||w||^2 + C \sum_{i=1}^{M} \xi_i \text{ for } y_i \cdot ((w \cdot x_i) + b) - 1 + \xi_i \geq 0$$  \hspace{1cm} (2-11)$$

where $C$ controls the trade-off between the slack and the size of the margin. However, in nonlinear SVM as a real world problem, there will be no hyperplane in existence which can successfully separate the training data. Accordingly, by
repeating all procedures as in linear SVM, we can find $w$, $b$, and $C$. Moreover, the solution of separating data points in nonlinear SVM is to map the data into a higher dimension space, this space is called transformed feature space. So to do such a transformation we use a kernel function that allows inner products to be calculated directly in feature space without examining a mapping function. Hence, once the hyperplane has been created, the kernel function is used to map new points into the feature space for classification. More explanation about the mathematical formulas and derivations can be found in [63],[64].

**2.4.2.2. Unsupervised Machine Learning**

Unlike the supervised ML, unsupervised ML techniques don’t require labeled training data. It assumes that the normal instances are more frequent than anomalous instances so it can easily label output groups as normal. Basically, unsupervised ML often aims to find groups of instances which share the same characteristics and similarities based on similarity measure. However, these techniques aim to build a model that can be used for reasoning, decision making, or prediction. Two very simple classic examples of unsupervised learning are clustering and dimensionality reduction.

Figure 2-11 shows general flowchart of unsupervised ML.

![Flowchart of unsupervised machine learning](image)

Unlike supervised ML, the unsupervised ML handles unlabeled datasets and assembles correlated instances together in a cluster and most probably labels the cluster as normal. Therefore, the most used application of unsupervised ML is clustering, which is explained in section 2.4.4.
2.4.2.3. Semi-Supervised Machine Learning

Obtaining labeled instances is difficult and time consuming. On the other hand, unlabeled instances can be examined by certain methods which need further efforts to understand the data. Therefore, semi-supervised ML techniques use the benefit of combining labeled and unlabeled data of both supervised and unsupervised ML. One approach is training the algorithm by mixed data of labeled and unlabeled instances and evaluating it using unlabeled data. Another approach to train the algorithm by large amounts of normal labeled instances and small amounts of anomaly labeled instances, then evaluate the model using unlabeled data. One more approach is using only normal labeled instances to train the algorithm and then using the model to identify anomaly in the test dataset. Note that the use of each approach depends on the phenomenon at hand. Accordingly, semi-supervised ML achieves a high accuracy and less human effort such that it becomes an individual interest in IDS theory and practice. In short, the machine is given a desired output just in the training process and then it builds a model that can be used to predict the label of a new unlabeled input. The flowchart in figure 2-12 describes a general overview of the main steps in semi-supervised ML.

The switch on the flowchart of semi-supervised ML chooses between labeled, unlabeled, or both instances to train the algorithm. Moreover, evaluating the model can be examined by either labeled or unlabeled instances such as online live traffic. Any method that follows the steps in figure 2-12 or uses almost the same technique can be considered as a semi-supervised method, such as clustering labeled and unlabeled data, and predicting the label of new instance.
based on a classifier model of a semi-supervised learning method. Therefore, no individual example is demonstrated in this section.

2.4.3. Artificial Neural Networks

Artificial Neural networks (ANN) are computer models inspired by the human brain. They are self-learning and can determine their own rules to be able to classify the output (if any). The main idea of ANN has been evolved from the connectivity and functionality of the human brain. Generally, constituents of the human brain are nerve cells neurons, which are connected by synapses, hence, it is a highly complex and nonlinear system. Therefore, ANN is a computational model that looks like the brain in acquiring and storing knowledge. Figure 2-13 depicts a basic illustration of the neuron in the human brain [65],[66].

![Figure 2-13: Basic illustration of the neuron in the human brain](image)

The neuron receives signals through synapses located on the dendrites or membrane of the neuron. If the received signal is strong and fulfills a certain threshold, the neuron is activated and emanates a signal through the axon to another synapse terminal. Note that the human brain consists of $10^{11}$ neurons which are highly connected with around $10^{15}$ connections [67]. This complex process in the natural neuron is literally adopted to build a model for the artificial neuron. Basically, an artificial neuron consists of inputs (synapses), weights (signal strength), activation function, and the output (emanated signal). Consequently, we can imagine the artificial neuron as a decision maker that determines the output based on input data received. A general overview of the artificial neuron from McCulloch-Pitts is depicted in figure 2-14.
2.4. IDS Categories based on Superior Techniques

The activation function (transfer function or squash function) performs a mathematical operation on the output signal \( v_i \) according to the vitality level of the input that should meet a specific threshold value \( \theta_i \) (or the so-called sensibility threshold or magnitude offset). Moreover, the activation function limits the allowable range of output to finite value, i.e. it normalizes the output into the range \([0,1]\) or other finite values. The bias value \( w_{i0} = b_i \) is optional here but it controls the net input of the activation function. Mathematically, this type of artificial neuron can be described as follows:

\[
v_i = \sum_{j=0}^{n} w_{ij} x_j - \theta_i \equiv [w_{i0}, w_{i1}, \ldots, w_{in}] \times [x_0, x_1, \ldots, x_n]^T - \theta_i \tag{2-12}
\]

\[
y_i = \psi(v_i) \tag{2-13}
\]

Hereby \( w_{ij}, x_j, v_i, \psi(.) \) are synaptic weights, input data, weighted sum, and activation function respectively.

**Definition**: The perceptron is the very simple form of ANN that is used in classifying special types of vectors, which are linearly separated. A perceptron consists of a single McCulloch-Pitts neuron with adjustable weight and a bias.

### 2.4.3.1. Common Activation Functions

If we consider the weighted sum of the neuron by \( v_i \) then the activation function \( \psi(.) \) can be one of the following simple functions:

**Identity Function**

In this form, the activation of the neuron is the same as the weighted sum of the input signals into the neuron. This type of activation is used often in the output.
2.4. IDS Categories based on Superior Techniques

\[
y_i = \psi(v_i) = v_i
\]  

(2-14)

**Binary Step Function**

This function is also known as threshold function or Heaviside function. The activation has a binary value (0 or 1) only based on a threshold value \( \theta \). This type of activation function is used often in a single layer network.

\[
y_i = \psi(v_i) = \begin{cases} 
1 & \text{if } v_i \geq \theta \\
0 & \text{if } v_i < \theta 
\end{cases}
\]  

(2-15)

**Bipolar Step Function**

The same as a binary step function with threshold except that the output has bipolar values (-1 or 1).

\[
y_i = \psi(v_i) = \begin{cases} 
1 & \text{if } v_i \geq \theta \\
-1 & \text{if } v_i < \theta 
\end{cases}
\]  

(2-16)

**Piecewise Linear Function**

It is also known as a saturating linear function. This activation function requires more iterations, but it saves on CPU processing time. Moreover, it can have either a binary or bipolar range for the saturation limits of the output.

\[
y_i = \psi(v_i) = \begin{cases} 
1 & \text{if } v_i \geq \theta \\
-1 & \text{if } -1 \geq v_i \geq 1 \\
-1 & \text{if } v_i < \theta 
\end{cases}
\]  

(2-17)

Note that the limit range is -1 and 1.

**Binary Sigmoid Function**

This is the most common type of activation, which is usually used in back-propagation ANN. It is an advantageous type because it is smooth and differentiable which in turn dramatically reduces the burden of computation.

\[
y_i = \psi(v_i) = \frac{1}{1 + e^{-\alpha v_i}}
\]  

(2-18)
where $\alpha$ is a positive parameter that gives different shapes of the function (as shown in figure 2-15). Note that the derivative of (2-18) is 

$$y_i = \psi(v_i) = \alpha \psi(v_i)[1 - \psi(v_i)]$$

which ease the computation clearly.

![Figure 2-15: sigmoid function with different values of $\alpha$](image)

It is clear that for large negative input values the output is $\approx 0$ and for the large positive input values the output is $\approx 1$.

**Bipolar Sigmoid Function**

Same as the binary sigmoid function but it is scaled to have values in range (-1 to 1).

$$y_i = \psi(v_i) = \frac{1 - e^{-\alpha v_i}}{1 + e^{-\alpha v_i}} \quad (2-19)$$

There are other activation functions which are rarely used such as non-saturating activation function or softmax activation function. Non-stationary activation function may be better when the piecewise linear function is not beneficial.

**2.4.3.2. Neural Network Architectures**

The interconnection of the artificial neuron in figure 2-14 with other artificial neurons builds an artificial neural network. Therefore, we can simply define the ANN as a complex connectivity of several artificial neurons that uses a mathematical or computational model for information processing which exchanges signals to yield a proper net reaction. ANN consists of several neurons
which form various layers, i.e. input layer, hidden layer and output layer. Principally, if the input layer is connected directly to the output layer then this is a single ANN, but if the input layer is connected to the output through several hidden layers, then this is a multilayer ANN. Moreover, the output of each neuron in one layer is considered an input to neurons in the next layer. Furthermore, in some ANN topologies there is no direct connection between neurons of the same layer, whereas in others they are fully connected. In addition, there are different learning processes of the ANN, which lead to different types of neural nets. These learning algorithms are supervised, unsupervised and hybrids, which are presented in the next sub-section.

Intuitively, the way neurons are connected forms the network architecture. This is to imply that, ANN can be adequate for several types of problems such as the problem of detecting intrusion in networks. The architecture determines the function of a neural network as discussed in activation functions. Therefore, ANN can be categorized based on connection patterns into two categories:

- Feed-forward networks, which have no loops. In this category neurons are organized into layers that have unidirectional connections between them. Furthermore, topologies of this category produce one set of output from the given input and the response to an input is independent of the previous network state (also known as static networks).

- Recurrent (feedback) networks, which have loops. This category, on the other hand, is a dynamic network. Hence, it is a bidirectional due to the feedback paths.

Figure 2-16 presents known ANN architectures [68].

![Artificial neural network architectures](image)

Figure 2-16: Artificial neural network architectures
2.4.3.3. Neural Network Learning Algorithms

Each topology (architecture) in neural networks requires an appropriate learning algorithm. Principally, the process of adjusting the weights to make the network learn the relationship between the inputs and targets is called learning, or training. Hence, learning in ANN can be defined as the process that updates network architecture and weights so that the network can perform a specific task. In addition, training patterns must be available so that the network can learn and improve its performance automatically instead of following specific rules. This is considered as the salient advantage of ANN over other systems.

Basically, there are three type of learning paradigms: supervised learning, unsupervised learning, and hybrid learning (see section 2.2.4).

Learning paradigms in ANN are further categorized, based on learning rules:
- Error correction: the basic idea is to use the error signal to adjust the weights to reduce the error gradually.
- Gradient descent: The basic idea is based on minimization of error, that is, the updates of weights are dependent on the gradient of the error.
- Boltzmann: it is a stochastic learning rule derived from the information theory and thermodynamic principle. The basic idea is to adjust the connection weights so that the state of a neuron fulfills a particular probability distribution.
- Hebbian: the most used learning rule. The basic idea is that the strength of a neuron is adjusted based on the neighboring neurons. That means, the change on connection weight of a neuron depends on the activity of both neurons connected to it.
- Competitive learning: using this rule of learning, unlike the Hebbian, neurons compete amongst themselves for activation, only one output is activated at a time and its neuron is called winner-take-all.

Learning algorithms must address three fundamental issues from learning the data: capacity, sample complexity, and computational complexity. The capacity is the amount of patterns that can be stored during the training. Sample complexity is the required number of training inputs to train the network, because too few inputs may cause overfitting, e.g. large training datasets and very small test datasets. The last issue is the computational complexity which is simply the time required for learning the algorithm and estimating a solution. The flow chart in figure 2-17 demonstrates the taxonomy of ANN learning algorithms and in which topology they are used.
There are other learning rules in the ANN research area such as the Delta learning rule which is a variation of the Hebbian rule. But we will briefly explain the Hebbian learning rule as the most used rule in neural networks.

**Hebbian Learning Rule**

When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A’s efficiency, as one of the cells firing B, is increased (Donald Hebb, 1949).

In ANN, if we consider the connected weight between two neurons \(i\) and \(j\) is denoted by \(w_{ij}\) and the output of \(i\) is \(y_i\) and the activation of \(j\) is \(x_j\) like the following demonstration then

\[
\Delta w_{ij} = \eta x_j y_i
\]  

(2-20)

where \(\eta\) is a control value or often known as a learning rate. Hence, Hebbian learning updates the weights for \(n\) number of iterations according to the following formula

\[
w_{ij}(n + 1) = w_{ij}(n) + \eta x_j(n)y_i(n)
\]  

(2-21)

There are other variations of defining the Hebbian rule which are available in [69], [70].
2.4.3.4. Objective Function in Neural Networks

Objective function (or cost function) measures how well the neural network performs. Selection of an objective function is very important because the function represents the design goals and decides what training algorithm can be taken. The sum of squares error function is very commonly used in research. Considering certain ANN architecture, error over a given training pattern is commonly expressed in terms of the total sum of squares error, which is simply equal to the sum of all squared errors of all output nodes and all training patterns. Let \( y_j^k, j = 1, \ldots, m, k = 1, \ldots, l \) are the actual output values for the input vector \( x^k = (x_1^k, x_2^k, \ldots, x_n^k) \), then the error of the given vector \( x^k \) can be defined as the sum of the square of deviating the actual output from the desired output values \( \hat{y}^k = (\hat{y}_1^k, \hat{y}_2^k, \ldots, \hat{y}_m^k) \) as

\[
E^k = \frac{1}{2} \sum_{j=1}^{m} (\hat{y}_j^k - y_j^k)^2
\]  

(2-22)

hence, for the following \( l \) training pairs \((x^1, \hat{y}^1), (x^2, \hat{y}^2), \ldots, (x^l, \hat{y}^l)\) \( \in \mathbb{R}^n \times \mathbb{R}^m \) such as \( f(x^k) = \hat{y}^k, k = 1, \ldots, l \) the total error is defined as

\[
E := \sum_{k=1}^{l} E^k = \frac{1}{2lm} \sum_{k=1}^{l} \sum_{j=1}^{m} (\hat{y}_j^k - y_j^k)^2
\]  

(2-23)

To keep the total error small as much as possible all weights will be adjusted after each iteration such as

\[
w_i^{k+1} := w_i^k + \Delta w_i^k, k = 1, 2, \ldots
\]  

(2-24)

The adjustments of weights are made in accordance with the respective errors computed for each sample presented to the network. the delta learning rule (Widrow-Hoff rule) is the simplest rule used to determine the delta as

\[
\Delta w_i^k = \gamma \cdot y_j (\hat{y}_j^k - y_j^k)
\]  

(2-25)

where \( \gamma \) is the proportionality factor. Similarly, the Mean Square Method uses the gradient descent to find the weights which minimize the error in equation (2-22) to be
2.4. IDS Categories based on Superior Techniques

\[ \Delta w_p^k = \gamma \cdot \delta_p^k \cdot x_i^k, \quad \delta_p^k = \frac{\partial E^k}{\partial \psi_p^k} \]  

(2-26)

where \( \psi \) is the activation function and \( \delta_p^k \) is the local gradient. As a result, weight changes of a unit \( p \) in the network can be determined once the \( \delta_p^k \) is known.

2.4.3.5. Simple Artificial Neural Network Example

Assume we have a feed-forward network with 2 inputs, 2 neurons in the hidden layer and one output in the output layer as demonstrated in figure 2-18.

![Figure 2-18: Simple ANN numerical example](image)

What is the final output if the activation function considered is a binary sigmoid function? Note that binary sigmoid is mentioned in equation (2-18)

1. \( \text{Input}(\Sigma_1) := (0.2 \times 0.1) + (0.4 \times 0.9) = 0.38 \rightarrow \text{Output}(\Sigma_1) := \frac{1}{1 + e^{-0.38}} = 0.594 \)
2. \( \text{Input}(\Sigma_2) := (0.2 \times 0.5) + (0.4 \times 0.8) = 0.42 \rightarrow \text{Output}(\Sigma_2) := \frac{1}{1 + e^{-0.42}} = 0.603 \)
3. \( \text{Input}(\Sigma \Sigma) := (0.594 \times 0.7) + (0.603 \times 0.5) = 0.717 \rightarrow \text{Output}(\Sigma \Sigma) := \frac{1}{1 + e^{-0.717}} = 0.672 \)

2.4.3.6. Summary

ANN is considered one of the oldest methods to handle large-scale problems and derive a solution that can address a complex system. ANN has several architectures and uses several learning rules to train the architecture and derive the output of neurons, as well as update the weights accordingly. This simple form can be applied in several categories such as: classification or pattern recognition; function approximation; and prediction systems. Intrusion detection systems can be considered on the application area of ANN in classification and recognition via learning a system and using the model to detect attacks, as well as zero-day-attack.

However, using ANN provides the following capabilities:

- Input-Output mapping: reading the input vectors and its desired output in a certain number of iterations to reduce the difference between the desired
and the actual output offers a mapping issue between the input and the output. Moreover, this mapping is also achievable for nonlinear systems.

- Adaptivity: it is clear that the ANN can adapt the weights according to the around environment characteristics. Furthermore, the ANN can be also designed to change its weight in real-time even in a non-stationary environment.
- Useful for software and hardware solutions: ANN can be implemented using several programming languages and even using a hardware approach such as VLSI.

Although computational complexity and overfitting are considered major shortcomings, ANN is considered as outstanding in the IDS research area.

2.4.4. Clustering Techniques

Clustering can be defined as finding intrinsic structure in a collection of data in such a way that similar objects or instances group together and stay separated from other dissimilar objects. Hence, a group of instances which have similarity between them is called a cluster. For instance, figure 2-7 depicts two clusters marked by circles. Moreover, clustering as a data analysis technique performs an indispensable turn in uncovering knowledge on unlabeled data. The idea of eliciting information using clustering to understand various incidents has been applied in IDS in order to distinguish between normal and anomaly instances. Of course, normal instances should gather in a cluster and stay apart from anomaly cluster.

In this thesis, we will just mention the most commonly used clustering algorithms in various application areas [71],[72]. These are summarized into hierarchical and partitional clustering. Note that, clustering algorithms follow certain clustering methods, which are: agglomerative vs. divisive; deterministic vs. stochastic; and incremental vs. non-incremental.

2.4.4.1. Hierarchical Clustering Algorithm

Hierarchical clustering arranges data into a hierarchical structure according to the proximity matrix. The result of arranging data is represented by a binary tree of nodes or dendrogram. The root node represents the whole dataset, the intermediate nodes describe how instances converge on each other, and the height of the tree demonstrates the distance between instances. Figure 2-19 illustrates an example of hierarchical clustering algorithm.
To achieve a more distinguished result, the tree can be cut into several levels to explore more information and relationships. Hierarchical algorithms can be classified as agglomerative methods and divisive methods. In a few words, agglomerative methods are of bottom-up structure and divisive methods proceed in the opposite way. Single link and complete link are examples of hierarchical cluster algorithms.

### 2.4.4.2. Partitional clustering algorithm

Partitional clustering assigns instances into a certain amount of clusters without building top-down or bottom-up structures. However, this type of clustering seeks often, by means of special criteria, to achieve the optimal partition, i.e. the optimal number of clusters and the number of instances belonging to a specific cluster. Hence, this type of clustering requires the number of clusters $K$ before performing the clustering task on the data. It is worth mentioning that several approaches have been proposed to determine the optimal number of clusters instead of choosing an arbitrary number [73],[74].

One of the most used criteria in this type of clustering is the sum of square error function. Therefore, some algorithms under this type of clustering are called square-error partitional clustering algorithm, such as the well-known $K$-means algorithm. Other partitional clustering algorithms are graph theory-based clustering and expectation maximization.

Due to the importance and the broader application of partitional clustering over hierarchical clustering, this thesis presents a detailed explanation about $K$-means clustering in the next section as an example of clustering.
2.4. IDS Categories based on Superior Techniques

**K-means clustering algorithm**

*K*-means clustering algorithm accumulates objects based on their feature values into *K* clusters. Hence, the goal is to partition input vectors into *K* clusters \([75],[76]\). So, if we define a set of instances \(x_j \in \mathbb{R}^n, j = 1, \ldots, M\) and \(C\) as the total number of clusters \(C := \{C_1, C_2, \ldots, C_K\}\) and that the Euclidean distance is used as a measure of similarity, then *K*-means aims to minimize the within-cluster sum of squares for the following squared error criteria:

\[
W = \sum_{i=1}^{K} \sum_{j=1}^{M} d(x_j, m_i)^2
\]  

(2-27)

where \(m_i\) denotes the centroid of cluster \(C_i\) and \(d\) for distance measure. The following steps summarize the K-means algorithm:

1. Determine the number of clusters *K*.
2. Initialize cluster centroids \(m := \{m_1, m_2, \ldots, m_K\}\). Normally, arbitrary initialized.
3. Iterate over all instances and assign each instance \(x_j\) to the closest cluster \(C_q\) if the following criteria is fulfilled:
   \[x_j \in C_q, \quad \|x_j - m_q\| < \|x_j - m_i\|, \quad i \neq q.\]
4. Recalculate the centroids based on the mean of all mapped instances to it, i.e. if \(C_q\) is the closest cluster of the instance \(x_j\) then the new centroid \(m_q(t+1)\) of the cluster \(C_q\) is simply the summation of all instances in the cluster divided by the total number of instances. Note that the cluster centroid \(m_q(t)\) is included.
5. Repeat steps 3&4 until the centroids do not change anymore, i.e. convergence.

Most of distance measurement used in *K*-means is the Euclidean distance. The mathematical formulation of the distance between two vectors \(x, y \in \mathbb{R}^n\) in the Euclidean space is given by:

\[
d(x, y) := \sqrt{(x_1 - y_1)^2 + \ldots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \equiv \|x - y\|
\]  

(2-28)

The *K*-means algorithm has been used often in clustering the aggregated network data to find out the suspicious connection or to define a normal network behavior model by exploring extra details about the network \([77],[78]\).
2.5. **Shortcomings of Current Intrusion Detection**

Recently, IDS models have achieved a significant level of superiority either host-based or network-based. However, this section attempts to clarify the major and minor shortcomings of current IDS models and disclose them briefly.

1. **Diversity**: most of detection models focus on a certain group of attacks or a certain target. This will devolve to reinforce other types of attacks to spread rapidly and to infiltrate into the system of the victim without any notice of the IDS model. Obviously, the reasons of this dilemma refer to the purpose of designing the IDS model. That means if the purpose of the design at the beginning focused on certain types of attacks, then the IDS model will not be able to uncover other types of new or unknown attacks. Thus, the more coverage the IDS model has, the more strength and capacity it achieves.

2. **Comprehensiveness**: if we dig deep into any IDS model we notice how limited it is. In other words, several IDS models can’t efficiently perform for both host events and network data. Available offline datasets, especially the KDDCup99 dataset, offer researchers the possibility to train and evaluate their models. This offer could be considered both as an advantage and disadvantage at the same time. The advantage is that there are always datasets to train and evaluate our models, but the disadvantage is that we build an inefficient IDS model for our heterogeneous and large-scale networks. Therefore, considering all related attributes, any other parameter will lead to reveal a large number of attacks, even new ones. Because this issue is very important scientists have declared a new research area called “Big Data”, which is focused on data understanding, processing and analysis [27].

3. **Labeling**: I admit this is a very crucial point in unsupervised IDS models. The recent approaches offer various methods to label the instances once they have been detected, but a reliable method is not presented yet. This is analogous to the increase of false positive alarms, because raising an incorrect alarm means the label for the detected instance is incorrectly assigned. This shortcoming will be intricate, because attacks appear often in various forms and sometimes in new patterns. One solution can mitigate this shortcoming, which is modeling a plausible and adaptive normal network behavior. The latter can differentiate between anomaly and normal connections and then assign a correct label for the instance.
4. Survivability: in my point of view, this is a shortcoming in current IDS models. Once the model shows a large amount of alarms and fails to uncover anomaly, it slowly forfeits interest, even from its owner. In this case, it is very worthy to analyze the model and overcome the main problem that causes the flaw by updating the model and issuing a new release. Moreover, even if the model performs well, it should be updated and improved continuously.

5. Other shortcomings are also mentioned in [79] such as the complex configuration of recent IDS models, delay for signature update, rising bandwidth and data volume, and encrypted network connections.
Chapter 3. Related Work

This chapter provides a spacious literature review about intrusion detection systems and discusses the major achievements in this area. In particular, the salient scientific approaches, in each step of IDS, are investigated and discussed step by step. Hence, this chapter consists of four parts, which summarize these achievements. The first part examines distinct contributions in the data aggregation phase of IDS. On the other hand, the second part shows the known preprocessing methods in the area of intrusion detection. The third part, in turn, studies the most distinguished achievements in classification. The last part presents some special cases, such as IPv6 and defense against detected anomaly. However, at the beginning it is worth mentioning the most proposed surveys on IDS taxonomies, frameworks, and products.

Several researchers have provided an overview of IDS methods and techniques in such a survey or taxonomy. These taxonomies and overviews differ from the viewpoint of detection, data source, reaction, or other perspectives as well. A distinguished survey in IDS has been presented by V. Chandola et al. [59]. They have explored the most relevant literatures in the area of anomaly-based IDS and show an intensive commentary on some assumptions in this regard. On the other hand, a small but efficient survey on IDS has been proposed by A.K. Gulve and D.G. Vyawahare [80]. They have provided a short overview of the common IDS methods and presented a comparative study between 17 different proposed techniques according to the performance metrics of IDS. Almost the same overview as the previous one is presented from H.O. Alanazi et al. [81] but without a comparative study. In addition, some surveys focus on a certain topic in IDS such as [82], which described only the network IDS and divided it into four categories with examples: statistical, classifier based, machine learning, and finite state machine. On the other hand, several approaches offer a sufficient overview with a suggested taxonomy of IDS. The article from P.Garcia et al. starts with a review of the most well-known techniques and available platforms in IDS. In addition, it proposes taxonomy of IDS and outlines some challenges
and efforts [83]. In contrast, F. Sabahi and A. Movaghar have provided a comprehensive taxonomy and survey in their work [84]. More surveys and overviews of IDS are available [85], [86], [87]. But one of the very valuable dated works in this area was presented by B. Mukherjee et al. [86]. They addressed both host-based and network-based IDS and explained several major points that must be considered by installing NIDS.

Regarding IDS frameworks and products, Vigna and R. Kemmerer proposed an NIDS approach called NetSTAT [88]. They exploit state transition analysis to discover intrusions on network. They use a formal model of both network and attacks to be able to determine which network events should be monitored and analyzed by a security officer in the network. In contrast, H. Han et al. take advantage of data mining to reveal attack signatures in computer networks [89]. They have used an algorithm called signature Apriori to detect and generate signatures. Most probably, their work is considered one of the first achievements to use data mining to discover attacks. Most closely, N. Wattanapongsakorn et al. have adopted a well-known machine learning technique called C4.5 decision tree to build an NIDS model that is able to detect known attacks such as DoS and combat it accordingly [90]. There are other common and wide literature works in the field of network-based, which are available in [91], [41], [92]. Similarly, there are well known network-based IDS products, including but not limited to AXENT (Symantec) [93], Cisco [94], CyberSafe [95], and ISS from IBM [96].

### 3.1. Related Contributions in Data Aggregation

Several frameworks have been proposed to ease the possibility of aggregating network and host data, which, in turn, improve IDS to perform better in real-time. However, a plausible framework that aggregates and generates datasets online within a significant time has not been proposed yet. Moreover, current available datasets are outdated and insufficient, which leads to building an imprudent IDS model that has a high false positive rate and a flaw in detecting unknown attacks. U. Fayyad et al. have presented the main steps of KDD where the data mining is employed [97], which is represented as a step toward a common framework and illustration of the main goals and methods in KDD. Accordingly, the first known framework for building an IDS model was introduced by W. Lee, S. J. Stolfo, and K. W. Mok [98], [99]. The main idea was to mine audit data of programs and user behaviors and compute from these data association rules to be a guide for the feature selection process. These data have been used to evaluate IDS classifiers and then to detect anomalies. However, one shortage is that, the data are based only on system calls and a
behavior model from association rules. Another shortage was that the framework
doesn’t consider network packet data.
A more detailed and improved work was proposed in the wake of the JAM
project [100]. In this work, data mining has been exploited to uncover features
from TCP packets and abstract other features by statistical computations. But
this improvement can be examined only offline, and it suffers from consuming
memory resources. In this regard, the first evaluation environment to dump
network packet headers and system logs, to offer the researchers an offline data
flows so they can evaluate their IDS models, is proposed in [101]. S. Mukkamala
et al. have utilized these data flows and reported a detailed analysis of these
dump data [102]. However, a complete dataset, called KDDCup99, based on the
dump data flows has been proposed in [103]. This dataset has been used more
than a decade to evaluate the IDS models and secure network computing.
Unfortunately, it is now very old, has several redundant records, and has
insufficient features that affect the IDS performance.
In this concern, several works were examined to refine the original KDDcup99
and elicit a better format with no redundancy such as in [104],[105],[106]. Other
works have even used this benchmark dataset to evaluate their models in high
speed networks such as [107]. In contrast, J. Song et al. have presented a
similar dataset to KDDCup99 with even extra features named Kyoto 2006+
[108],[109]. It was built on 3 years of honeypot data and consists of 14
statistical features derived from KDDCup99 dataset as well as 10 additional
features which are used for further analysis and evaluation of IDS’s. Another
similar dataset based on KDDCup99 is presented in [110],[111], which is the
GureKDD. But this dataset likewise is outdated and considered as offline
dataset. Note that there are other datasets to evaluate IDS models such as
Auckland IV [112], LBNL/ICSI [113] and iCTF [114].
However, the current datasets or the corresponding frameworks are not viable in
real-time, i.e. they are not able to continuously aggregate network data and
hosts’ events and generate datasets sufficiently.
In this thesis, the idea presented by S. Babu and J. Widom is meaningfully
adopted. They propose a general and flexible architecture for processing
continuous queries in the presence of the data stream and present a prototype
that manages data stream accordingly [115].

3.2. DATA PREPROCESSING METHODS

Most artificial intelligent models in network security rely on certain network
features, which innervate security by enhancing the performance of detection
models. However, these models handle only normalized numeric data to reduce
3.2. Data Preprocessing Methods

model computations and avoid data dominancy [116]. These steps can be identified as feature selection, discretization and normalization [117],[118],[119]. Collected data from the various resources implicitly contain several features or attributes. Choosing the optimal feature set from the collected data plays an important role in the area of IDS [120]. Therefore, data mining methods exploit the collected data to reduce the huge dimensionality and elicit an appropriate and significant feature set. The most known features in the research area of IDS have been determined from the cooperative program between MIT and DARPA [103],[101]. Consequently, researchers have proposed different techniques and methods to determine the optimal feature set that improve the performance of the IDS model. A new fast feature reduction method is proposed by S. Parsazad et al. in [121]. They have used similarity measure to extract the effective features and then they evaluate their method using K-nearest neighbor and Bayes classifiers. Furthermore, they have compared their reduction method with other three methods, namely correlation coefficient, least square regression error and maximal information compression index. Although their proposed method did not deliver promising result, it had less of a computational cost.

Likewise, P. Jeya et al. have proposed a feature reduction method that uses the linear correlation coefficient to abstract the relevant features [122]. In addition, this method converts the symbolic features into numeric by replacing them with arbitrary integer values. However, we cannot simply convert a symbolic feature from its space into a real number space without using a suitable conversion or transformation function that scientifically explains the conversion process. Hence, the conversion in their article is considered questionable.

A conceptual work in feature selection is proposed by S. Mukkamala and A. Sung in [120]. They have used neural networks and SVM to categorize the 41 KDD features into important, secondary, and unimportant. They have shown that important features have performed the best performance in the evaluation. In contrast, floating search methods show a positive impact on the feature selection and extraction in IDS. Y. Chan et al. have studied the sequential backward search and sequential forward search methods with the localized generalization error as threshold criteria and radial basis function as a classifier [123]. The sequential backward search shows better performance when the number of features is large enough. An advanced version of sequential search method with backtracking, performed on the high dimensional data, has been developed by P. Pudil et al. [124]. However, their approach required a high computational cost, especially the forward search algorithm. Therefore, P.Somol et al. have tried to reduce the computational cost and find an optimal solution [125]. Similarly, K. Z. Mao has introduced the same idea of selecting feature subset by deriving an orthogonal forward selection and orthogonal backward elimination in cooperation with Gram-Schmidt orthogonal transforms to select
3.2. Data Preprocessing Methods

the feature subset [126]. In contrast, A. Zainal et al. have extracted the optimum features by demonstrating rough set theory [127]. This method was compared with other three methods and showed that it can be adequate for IDS.

A new feature selection algorithm for distributed cyber attack detection is proposed from H. Nguyen and Q. Cheng [128]. According to their proposed algorithm, two feature subsets are proposed, the unique feature subset and the pair wise feature subset. The latter was efficient in selecting best features for classification problems.

Further scientific works to extract features directly from the online stream has been proposed by X. Wu et al. [129]. They have noticed that this method could be feasible in some networks but insufficient in the large scale networks which have non-stationary and heterogeneous traffic [49]. Additional articles in feature selection are also available in [130],[131],[132],[133].

It is worth mentioning that, some researchers have exploited discretization to select the best feature set using the information gain method. For instance, C.S. Dhir et al. have exploited information gain with independent component analysis to achieve better detection rate especially in face recognition [134]. Whereas, they have confirmed that the information gain helps in improving the overall system accuracy.

Moreover, U. M. Fayyad and K. D. Irani have presented a method to discretize continuous values in a dataset using the information entropy minimization [135]. Similarly, C. Ratanamahatana has presented a comparison study between various discretization methods, namely equal width discretizer, proportional k-interval method, entropy minimization heuristic, and their proposed method CloNI [136]. The method CloNI was optimal for discretization in a large dataset and performed better than other methods.

Based on the previous works, feature selection and discretization provide reduced datasets to evaluate the classification models. However, some classifier models handle only normalized numeric datasets especially the artificial neural networks [116].

Thus, most recent IDS models use the reduced datasets, preprocess them and construct the required format [137]. Most preprocessing methods in the area of IDS are summarized by conversion and normalization. The first one is to convert all symbolic values (also known as nominal, string, or text observations) values into numeric ones. The second one is to scale all features into the same domain and avoid data dominancy.

Yu Liping et al. have evaluated several normalization methods in multi-attributes datasets [138]. They have proven that, evaluation target commonly determines
the used normalization methods. The method in [139] is considered sufficient in normalizing offline dataset but lacks the feasibility of normalizing the traffic in real-time. Moreover, the principle of normalizing incoming traffic is not clear and showed no enhancement to the IDS, as well.

In contrast, [117] and [118] have presented intensive and comparative studies between several normalization methods, which are minimum-maximum, statistical, decimal, and algorithmic normalization. They have evaluated the performance of various classifiers based on these methods, and have shown that the classifiers’ performance improved when the dataset was normalized, especially by minimum-maximum or statistical normalization. Oh et al. have presented an unsupervised method to detect attacks using SOM [140]. To achieve this target, they have changed each nominal value in features “protocol_type, service” by its decimal number, as defined in IANA protocol numbers assignment or port numbers, and then they normalized the dataset using minimum-maximum normalization. Converting a string to decimal number without an appropriate conversion function will affect the normalization method, and therefore, the classifier results. In addition, replacing each service is time consuming due to the large number of services in computer networks.

Briefly, the preprocessing method is very fundamental in IDS. Selection of the optimal features set depends on several factors such as the system, the detection method used, and the target of the model. Discretization supports by selecting the optimal feature set, as well. In addition, normalization provides the classifier model by numeric and normalized datasets, so it is very essential in the preprocessing step. In contrast, the proposed conversion methods are debatable because they are not convincing and infeasible in real-time. Therefore, this thesis will exploit some preprocessing methods such as normalization or discretization, but it will propose novel hybrid feature selection and conversion methods in preprocessing data.

3.3. Distinguished Classification Achievements

This section addresses the most significant and distinguished classifications’ achievements in IDS. Detection methods in IDS principally rely on four techniques, which are: statistical-based [141]; machine learning [142],[143]; neural networks [144],[145]; and clustering [71]. ANN become the most efficient IDS method in network security [14], whereas, the most successful applications of neural networks are classification or categorization and pattern recognition [13].

In this regard, M. Sato et al. have extended the work proposed by Song et al. [146] to detect unknown attacks using features extraction from anomaly-based
3.3. Distinguished Classification Achievements

IDS alerts [147]. Major steps of their proposed method are summarized by collecting the known attacks from Kyoto2006+ dataset, then extracting 10 features from the alert and retraining them using one-class SVM. This approach can handle only detected attacks by Kyoto and hence any new attacks will not be uncovered or even recognized. R. Karthick et al. have proposed a different adaptive IDS, which combined Naïve Bayes with HMM [148]. The first classifier monitors the online traffic and flags it as suspicious, and the second classifier isolates the flagged connections and classifies them as normal or attack. This approach is adaptive in a way that victim IPs are isolated and secured, but not in a way that the model updates itself constantly. Similar to the latter work, H. Nguyen and K. Franke have exploited the online learning framework [149] to enhance their classifier [150]. However, the result was examined only on offline datasets and not on a real traffic.

On the other hand, Y. Yu and H. Wu have used data mining techniques to analyze system call sequences of privileged processes [151]. The frequency of each call is considered the basic for the Naïve Bayes classifier and the ratio of the probabilities of a sequence is considered as the input to a fuzzy system. Accordingly, the fuzzy logic classifies each sequence as normal or abnormal. This approach is performing well for system calls, but it is still insufficient while network packets are not involved.

The same concept is also proposed by J. Kang and S. Oh [152]. However, the concept focuses on dynamic splitting of a cluster into two clusters or merging two adjacent clusters into one cluster.

Another known technique in classification is clustering, which is mostly used in IDS to label network data such as in [153], [154]. The latter proposes a hybrid method that combines K-means clustering and SOM, where SOM trains the network data and K-Means clusters SOMs output to label the connections into normal or anomaly. This work is considered sufficient in the area of the IDS. Due the huge amount of proposed works in the area of IDS, this thesis could not address all achievements; though for further reading refer to [155],[156],[157],[158],[159],[160].

As mentioned, ANN is considered as a vibrant machine learning technique in data mining. The most successful application of neural network is classification or categorization and pattern recognition [13]. A special approach of ANN is the GHSOM, which is a modified approach from the Kohonen Self Organizing Map [161], [15],[16]. Recently, researchers focus on improving the GHSOM due to the preciseness on exploring data and the vitality in the area of IDS. M. Zolotukhin, T. Hamalainen abd A. Juvonen have applied an n-gram model to HTTP request from network logs to form metric features, which are further analyzed by the GHSOM to detect the anomalous HTTP requests [162].
3.3. Distinguished Classification Achievements

Obviously, the main goal in the previous work of using GHSOM model is to detect new suspicious requests and does not focus on the GHSOM model. A noticeable improvement on the GHSOM has been presented by A. Ortiz et al. [163]. They have presented a new labeling or relabeling technique for the final best matching units of the GHSOM model to correctly label attacks. In contrast, T. Ichimura and T. Yamaguchi have proposed more advanced techniques in stabilizing the GHSOM [164]. They have defined a unit generation and unit elimination strategy that enhance the GHSOM. On the other hand, Y. Yang, D. Jiang and M. Xia have proposed new metrics for numerical and nominal data and use new growth threshold Tension and Mapping Ratio (TMR) to control the topology [116]. The same work is further improved from E. Palomo et al. by using only one threshold value to control the growth topology [165]. Finally, D. Ippoliti and X. Zhou have presented an Adaptive GHSOM (A-GHSOM) model that has four enhancements, which have boosted the GHSOM and enhanced the detection rate [139]. These enhancements are: enhanced threshold-based training; dynamic input normalization; feedback-based quantization error threshold adaptation; and prediction confidence filtering and forwarding. However, the threshold-based training is basically a quantization error and the dynamic input normalization is infeasible in real-time 1 to 10 GB networks. Furthermore, it has unstable topology and large maps.

Other approaches have used the SOM or GHSOM for visualization such as in image recognition. Y. Cao, H. He and H. Man have proposed kernel density estimation over data stream, named SOMKE, by building a series of SOMs over the data streams via two operations. The first operation creates SOMs from windows of data, and the second one merges the SOM sequences based on Kullback-Leibler divergence measure [166]. The evaluation process shows that SOMKE is effective for stationary and non-stationary data stream. Although the approach using SOM has performed very well, there are several parameters like growth control that are not mentioned or investigated. E. Palomo has also proposed another new probabilistic self-organizing model based on the stochastic approximation framework to successfully visualize high dimensional dataset [167].

Further methods based on GHSOM can be found in [168],[169],[170]. The previous investigations of GHSOM have mainly focused on improving the topology to detect anomaly in offline operational mode. In addition, other approaches have used the GHSOM to achieve different targets like the visualization of data stream. But, they did not examine all shortcomings and drawbacks of GHSOM and proper solutions to handle them. Therefore, this thesis will unveil the most shortages and challenges in GHSOM and propose an appropriate solution for each shortage respectively.
3.4. Other Approaches in IDS

This part addresses other approaches in IDS such as the IPv6, smart grid, and the prevention detection system. Most of proposed works and achievements had investigated IDS on IPv4 networking and ignored the IPv6, because the latter is not fully implemented and widely deployed. However, K. Yun and Z. Mei have addressed this issue in [171]. Their article has illustrated the coexistence of IPv4 and IPv6 and the related security issues, as well. Moreover, they have proposed a host-based and endpoint defense system, which outlines only the problem and investigates shallowly a hybrid model for a single host with IPv6 address. Further special issues in IDS/IPS are the usage of agent approach, which is the most applicable approach in paralleling the IDS model over an entire network. The issue of using an agent-based approach will be addressed in this thesis due to its popularity, although it is not the most preferable approach by real-time firms. M. Shouman et al. have proposed a multi-agent server framework that contains an NIP/HIP framework, which combats attacks on networks and hosts, and works in parallel [172]. The Network Intrusion Prevention (NIP) component focuses on network and packet behaviors by inspecting network traffic, whereas Host Intrusion Prevention (HIP) component focuses on the operating system and applications behavior. The framework did not investigate deep inspection of network connections and transactions but focused only on attacks' signatures databases. In addition, the NIP framework consists of 8 agents and the HIP consists of 11 agents, which obviously leads to overload on the network and hence degrades its performance. In contrast, C. Kavitha and M. Suresh have addressed a malicious countermeasure approach named Time-sensitive Stream Mined Access to Network (TISMAN) to prevent attacks on computer networks [155],[173]. They have deployed, in the first layer of TISMAN, a centralized detector on the monitored server in the DMZ to collect information. The second layer of TISMAN includes the protection mechanism that is located inside the Intranet. Specifically, TISMAN exploited stream processing in the second layer by using a sliding-window model, as a given window size $w$, so that only the latest $w$ transactions are utilized for mining. Moreover, an additional TISMAN authority module behind the firewall was installed to initiate the defense strategy to protect the information system. This kind of prevention adheres to the network topology, not to the data in real-time especially in high speed networks, so it is not plausible.

C. Gu and D. Gu have presented a research proposal of IPS for high speed networks [174]. They have briefly addressed the difference between IDS and IPS, and the advantages and disadvantages of IPS over high speed networks.
Integrating IT systems into physical environments has exposed several vulnerabilities, which are exploited by attackers to gain private information or to perform devastation to the surroundings. In this regard, security researchers have employed organic computing techniques to combat such security threats on the cyber-physical systems. J. Haehner et al. proposed a concept to secure the cyber-physical systems with the organic computing techniques[175]. The cyber-physical system consists of two main parts: the system under observation and control (SuOC) such as a network which is observed by an IDS. The second is their Organic Security System (OSS) which observes the SuOC. They have proposed a novel layered OSS architectural concept which includes reaction, cognition and social layers. Furthermore, they have explained some attacker models from inside and outside the OSS domain and defined a research agenda for the proposed concept, as well.

Another key aspect in the area of cyber-physical systems is the SCADA system. Y. Yang et al. have provided an overview of vulnerabilities of SCADA in smart grid [176]. Their article has presented a test-bed in the smart grid cyber-security that contains SCADA software and communication infrastructure. It has investigated an ARP poisoning-based man-in-the-middle attack on SCADA and found out that malicious attacks can negatively impact secure operations of SCADA. More IDS approaches are available in [177],[178].

As a summary, there are large numbers of contributions in the area of IDS/IPS, which address diverse challenges and treat them step-by-step. In addition, there is no available framework that covers all challenges and proposes a proper solution. Therefore, this thesis focuses on a certain spot in IDS and tries to propose a fundamental and adaptive framework in anomaly-based IDS, which remedies major counterparts and illustrates significances of each step individually.
4.1. Overview

Chapter 4. Proposed Methodologies

4.1. Overview

In the last two decades, many challenges and issues in data mining and machine learning have been identified. One of the critical issues is proposing an adaptive system that can accommodate the massive data flow or the so-called big data [179]. Accordingly, building an adaptive prediction system has become an essential demand from the network security perspective. However, several major critical points must be considered and identified to achieve a proper and adaptive IDS framework. They are discussed intensively in [179]:

- Making adaptive systems scalable.
- Dealing with realistic data.
- Improving usability and trust.
- Integrating expert knowledge.
- Taking into account various application needs.
- Moving from adaptive algorithm towards adaptive tools.

In this thesis, the desired IDS framework can handle the voluminous data flow generated from offline, synthetic, and realistic network data. Another point in this thesis is the consideration of system demands while building the framework, hence, in this thesis uncovering malicious and suspicious connections in sufficient time is one of the most important demands. Therefore, this chapter addresses the major and minor proposed methodologies of the adaptive system and from the industrial demands point of view as well. These methods are: data aggregation; feature selection and extraction; and preprocessing.
4.2. DATA AGGREGATION

The online operation mode of IDS has become a real challenge since the amount of heterogeneous and non-stationary data and the interconnection between communication networks are increasing rapidly [180],[181],[182]. Substantially, the model of a network IDS is trained often using an offline dataset, and evaluated offline using a small part of the training dataset. Accordingly, IDS models operate effectively and achieve a high-performance over computer networks in the offline mode [83]. In contrast, recent IDS models have been improved so that they can classify network traffic in online operation mode, but only for limited purposes such as detecting anomalous traffic in http request [183] or detecting only DoS attacks [184]. However, these models are not able to detect anomaly in massive data flow which becomes very complicated and non-stationary. Moreover, detection models suffer from the inability to detect new attacks or attacks with updated signatures in real-time, because these models adhere to the original offline training dataset, and they are not superior in recognizing unknown data flow. In addition, most IDS’s focus only on network packets and ignore hosts events, which cause a degrading in IDS performance. According to these reasons, most recent IDSs still struggle to adapt to the current heterogeneous networks. According to the previous problems, current IDS models should be professionally improved so that they adapt to the realistic massive data stream and provide preciseness in detecting unknown anomalous traffic. More specifically, they should be competent in converting massive data flow to connection vectors, which, in turn, are used for training and testing in real-time. Hence, the purpose of the proposed aggregation method in this thesis is to convert the online data flow to datasets effectively. Therefore, the next section provides a valuable introduction about datasets and the most used one in an IDS environment.

4.2.1. Dataset

Generally, input data for IDS are aggregated instances (also known as connections, patterns, vectors, samples, observations) [185], where each data instance is constructed based on a set of attributes (also known as features or characteristics). The attributes in general can be of discrete or continuous type and have either numeric or nominal values (also known as string, symbolic, or text). Hence, a single data instance might be constructed from one attribute, i.e. univariate, or multiple attributes, i.e. multivariate, which have the same type or mixture types [59]. Therefore, a dataset can be formally defined as the set of features and the constructed data instances based on these features.

In 1998 DARPA realized the need to an evaluation benchmark resource for systems that perform anomaly detection. Thus, they initiated a collaborative...
program with MIT’s Lincoln Labs to generate an evaluation dataset that can be used in evaluating the IDSs.

4.2.1.1. DARPA 1998/1999 Dataset

Subsequently MIT’s Lincoln Labs, as well as the Air Force Research Laboratory, built a computer network, intended to simulate a regular U.S. Air Force base LAN, to generate and capture realistic network traffic. The simulated network, consisting of about fifty nodes running different operating systems, was divided into two segments representing the inside and outside of a local area network. The simulation performed the following: producing real network traffic, injecting attacks at predefined points artificially using software tools, capturing connection and file system information from the network devices and computers, and aggregating the traffic from certain points in the system. The primary services used in the simulated network include HTTP, FTP, SMTP, POP, DNS, X, IRC, SWL/Telnet, SNMP, Time and Finger, which cover most of the frequently used services in real network. Injected attacks into the network can be divided into four categories:

- **Denial of Service (DoS):** type of attacks, where an attacker attempts to make a computer resource unavailable for users.
- **User to Root (U2R):** is a class of exploit, in which an attacker who is allowed to log in as a normal user is able to exploit some vulnerability to gain root access to the system.
- **Remote to Local (R2L):** type of attacks that occur when a remote user who has no user account to a computer system is able to exploit some vulnerability to get local access to that system.
- **Probing:** type of attacks that attempt to gather information about a network or computer system for the purpose of exploring security flaws.

Appendix A contains a complete list of these attacks.

The simulated network operated for nine weeks. Seven weeks of network based attacks in the midst of normal background data was collected for training. The two weeks of testing data contain some attacks that do not exist in the training data. In the experiments, two types of data have been collected: 1) tcpdump data captured from the network link and; 2) system audit data including Sun Basic Security Module (BSM) audit data from one UNIX Solaris host and file system dump. As a result, captured training data resulted in about four gigabytes of compressed binary TCP dump records that were processed into about five million connection records. Another two million connection records were captured in the two weeks of test operation [186]. For the generation of a dataset that can be used in IDS evaluation, 41 relevant attributes have been selected as relevant for the detection of anomalies; they can be distinguished into three categories:
4.2. Data Aggregation

- Basic features of individual TCP connections, such as the protocol type or service.
- Traffic features computed using a two second time window or a hundred-connection window, such as the number of connections to the same host or the percentage of these connections that have SYN errors.
- Content features within a connection suggested by domain knowledge, such as the number of failed login attempts or the number of shell prompts.

Appendix B contains a complete list of the 41 features.

The feedback that was provided by the researchers, who have evaluated their systems using the DARPA 1998 (also known as DARPA 98) dataset, has resulted in introducing the DARPA 1999 (also known as DARPA 99) dataset.

The primary goal of DARPA 1999 evaluation was to measure the ability of intrusion detection system to detect novel attacks since this was a major problem discovered in 98 evaluations. The other major change is the addition of inside attacks. Moreover, the evaluation of IDSs is more comprehensive since both attack detection and attack identification are evaluated.

Finally, in the wake of the DARPA 1999 evaluation program, KDD Cup99 [103] has emerged. It has been the most widely used dataset for the evaluation of network-based anomaly detection methods. This dataset was prepared by Stolfo et al. [98],[99],[100] and is built based on the data captured in DARPA’98 IDS evaluation program. The KDD training dataset consists of approximately 4,900,000 single connection vectors, each of which contains 41 features and is labeled as either normal or an attack with a specific attack type, while the test set contains about 300,000 samples with a total number of 24 training attack types, plus an additional 14 types in the test set only.

4.2.1.2. NSL-KDD Dataset

Researchers at the University of New Brunswick and the Institute for Information Technology in Canada have analyzed the KDD99 dataset and demonstrated that there are two important issues in the dataset which highly affect the performance of evaluated systems, and result in a very poor evaluation of anomaly detection approaches. One important issue is the huge number of redundant records caused by the synthetic generation of the network traffic, which will cause learning algorithms to be biased towards the more frequent records, and thus prevent it from learning unfrequent records [104],[105]. Therefore, they have generated a new dataset called NSL-KDD dataset [106]. For generating the NSL-KDD dataset, the researchers first removed all redundant records from the KDD99 train and test dataset. Then, they generated random subsets of the KDD99 dataset to get a homogenous distribution of the attacks. Also, the number of connection records in the NSL-KDD dataset with
4.2. Data Aggregation

about 126,000 training and 23,000 test records is more reasonable for making it affordable to run the experiments on the complete dataset without the need to randomly select a small portion.

These two datasets, namely KDDCup99 and NSL-KDD, are the most used datasets in evaluating the IDS models. However, there are several other datasets also available such as DARPA2000 [101], Kyoto2006+ [108],[109], DEFCON dataset [114] and GureKDD [111], but they haven’t gained researchers’ attention as much as the two-mentioned datasets. Figure 4-1 illustrates a simple dataset example.

Note that there are some criticisms from [104] and [105] on the procedure used in generating the dataset of KDDCup99, whereas the main critique is the characteristics of the synthetic data don’t contain background noise such as packet storms or invalid packets, which would be found in a normal computer network. Hence, the synthetic datasets from the simulation network are considered too simple in comparison to the real network traffic. On the other hand, other researchers have observed by dividing the dataset to small subsets that the distribution of attacks is disparate. Some subsets contain only one attack and others contain a high number of attacks in contrast. This issue leads to complicate the cross-validation learning method and hence distrust the classification model.

According to the previous introduction about datasets in the IDS research area, it is very important to train and evaluate the current IDS models by a novel dataset that represents realistic network traffic. However, the challenge is how to capture massive network traffic, perform an intensive analysis and processing steps, and then construct connection vectors based on selected features. The
following section will explain the proposed aggregation method in this thesis from capturing the traffic to generating datasets.

4.2.2. Proposed Aggregation Method

Basically, most IDS approaches aggregate network traffic and generate datasets to train the IDS model in the offline mode. Therefore, most detection models operate effectively in offline mode but fail to fully monitor the network and uncover anomaly in online operational mode [181], which leads to emerging IDS online problems. On the other hand, providing IDS models by continuous datasets for training or testing purposes will optimize the performance and enhance the detection rate. Thus, a professional and novel method that handles and treats massive data flow in computer networks has become an area of extreme focus in network security. This section tackles these challenges by proposing a state-of-the-art method that converts data flow to continuous connection vectors (datasets) to train and evaluate IDS models online in real-time. This method has been given the name "OptiFilter", which is derived from optimal filtering. In any large scale and heterogeneous IPv4 network, the proposed method captures network packets and receives hosts’ events, processes and analyzes them in a queue concept with a dynamic time slot window, and constructs connection vectors accordingly. Figure 4-2 demonstrates a general overview of the proposed method.

The proposed method consists of a queue as a container for determined time slot windows. Specifically, the queue in our method can contain up to $n$ windows and each window $w$ is a time slot of $t$ seconds, e.g. 5 seconds. Figure 4-3 illustrates a general overview of our proposed method with a 5 second time slot window. Principally, network packets and hosts’ events will be constantly captured. Accordingly, for the first 5 seconds, correlated hosts’ events and network packets will be constructed as connection vectors and pushed forward in the queue and the next 5 seconds window will take place, so that a queue
with 10 seconds is occupied. The entire analysis and processing methods are illustrated in figure 4-3 and in the next subsections respectively.

The previous figure shows the exact processes applied on each window inside the queue. That means, while the window is still inside the queue, all connections within this window will be constantly modified until the window reaches the end of the queue and the connections get exported as a dataset or directly in a pipe. The advantage of this proposed method over other methods is that, all parameters are configurable and can be adjusted upon the network environment. Moreover, the output of this method can be stored as a huge dataset, several small datasets, or written directly to a pipe based on First-in-First-out (FIFO) concept. The benefit of the pipe concept will be more clearly clarified in the next chapter when introducing the online IDS prototype. The configuration parameters are illustrated in Table 4-1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>Read from offline file or directly from network traffic</td>
</tr>
<tr>
<td>Windows inside the queue</td>
<td>Unlimited (3 – 10 is preferable due to performance issue)</td>
</tr>
<tr>
<td>Window length</td>
<td>Variable</td>
</tr>
<tr>
<td>Backlog</td>
<td>Start from 100 connections</td>
</tr>
<tr>
<td>Output</td>
<td>CSV file (complete dataset) or directly in a pipe (online)</td>
</tr>
<tr>
<td>Considered Protocols</td>
<td>Any protocol</td>
</tr>
<tr>
<td>Considered Services</td>
<td>Any service (based on destination port)</td>
</tr>
</tbody>
</table>

The following subsections illustrate the operational steps of the proposed method based on figure 4-3. Note that after several empirical studies, the
4.2. Data Aggregation

Window length has been used for 5 seconds and the backlog for 1000 connections. Some features can be calculated only after popping the window from the queue and using the previous exported connections in calculation. These exported connections are stored internally as a log file and therefore it is called backlog. This is to imply that the method captures traffic and events constantly and the queue reads always the last captured 5 seconds in the first window. It then pushes the window forward and reads the next captured 5 seconds in the next window so that the queue will be occupied by two windows with a length of 10 seconds, and so on. For instance, if we determine the number of windows inside the queue by 3, then the queue will be occupied for 15 seconds and the first window will export connection vectors after 15 seconds from the beginning of capturing, and by taking in consideration the status of the last 1000 connections.

4.2.2.1. Proposed capturing method

Our method starts by capturing tcpdump from the involved computer network using the following command:

```
tcpdump -i <Interface> -nN -w <Filename> -B 10240 -s 128
[-U] 'arp or icmp or tcp or udp',
```

where \(-i\) listens on a certain interface, \(-nN\) prevents tcpdump from converting addresses to host names, \(-B\) is to increase the operating systems capture buffer size e.g. 10 MB, and \(-s\) limits the packet length to 128 Byte, which reduces the storage usage by dropping all packet payload and, in turn, reduces packet loss accordingly. The \(-w\) option advises tcpdump to write the raw packet output to a file instead of parsing and printing it in a human readable format. In online mode, the parameter \(-U\) is needed to prevent the output being buffered.

The PcapFileManager permanently reads the pcap input and handles only network packets that have only IP/TCP, IP/UDP, ICMP, or ARP protocols. Note that, our proposed method can be configured to consider more protocols (see Table 4-1) but the most used protocols were selected based on expertise knowledge from the industry and the university computer center, as well.

Concurrently, hosts' events will be sent via SNMP traps directly from the operating system to the intended server, so that we avoid any agent solution. In a Windows operating system, we exploit the Windows Management Instrumentation (WMI) that is able to filter events and send them over SNMP-traps via the WMI-SNMP-Provider to a central SNMP server.

In contrast, we exploit in a Linux operating system the syslog daemon to generate SNMP traps using the NetSNMP agent and send them using a TCP connection to the SNMP server as well. The HostFileManager on the server side
uses an intelligent script and filters to retrieve information from SNMP traps and then derive only the considered host features, in turn assigning a unique ID for each event. Note that there are several methods to send SNMP traps from any operating system, but the benefits of Windows WMI and Linux NetSNMP were used because they send the host activity immediately as soon as it appears.

4.2.2.2. Connection and event analysis

Basically, the proposed method treats captured live packets and received host SNMP-traps similar to the first input first output (FIFO) principle in a configurable time slot window \( w \), e.g. 5 seconds. Captured network packets and received hosts’ traps get identified by a unique-ID \{timestamp, protocol_type, source_ip, source_port, destination_ip, destination_port\}. Consequently, TCP or UDP packets and hosts’ traps are correlated once they have the same unique-ID, which leads to construct a connection vector in the current window. However, there is no correlation required for ARP and ICMP packets, because they have no port and service at the destination, which means each ICMP or ARP network packet is considered as a single connection. Note that, the correlation process between network TCP- or UDP-packets and host SNMP-traps is performed for each window. On the other hand, all uncorrelated TCP or UDP packets, as well as uncorrelated hosts’ events will be stored in temporal databases for a certain timeout for the purpose of monitoring and updating. However, connections that exceed the timeout will be removed from the database and inserted into the dataset with a timeout flag. The timeout is determined in this thesis as shown in Table 4-2.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Connection status</th>
<th>Timeout</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP: -</td>
<td></td>
<td>180 sec</td>
<td>Snort [20][21]</td>
</tr>
<tr>
<td>TCP: Handshake</td>
<td>20 sec</td>
<td>Our method</td>
<td></td>
</tr>
<tr>
<td>TCP: established connection / trusted network in cold start</td>
<td>720 sec</td>
<td>[187][188]</td>
<td></td>
</tr>
<tr>
<td>TCP: connection termination</td>
<td>675 sec</td>
<td>[188]</td>
<td></td>
</tr>
<tr>
<td>TCP: connection closed / trusted network in cold start</td>
<td>240 sec</td>
<td>[188][189]</td>
<td></td>
</tr>
</tbody>
</table>

To reach a high degree of correctness by correlation, TCP and UDP packets have become further analyzed to determine the connection they belong to. Therefore, the following analysis steps are examined on TCP and UDP packets:

1. Identify each connection with a unique ID.
2. Identify the direction of the connection based on the initially dumped packet: this step is essential to determine the correct value of some features (such as src_byte); otherwise incorrect value will be obtained. In the insight of that, we defined certain conditions, in addition to these
defined in Snort, to identify the direction of established connection in Table 4-3.

<table>
<thead>
<tr>
<th>Status of initial Packet</th>
<th>Consideration</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYN-Flag is appeared</td>
<td>we consider it as initial TCP-SYN from initiator to responder</td>
</tr>
<tr>
<td>SYN- and ACK-Flag are appeared</td>
<td>we consider it a reply from responder to initiator</td>
</tr>
<tr>
<td>ACK-Flag without payload appeared</td>
<td>we consider it the last packet of 3-way-handshake, i.e. from initiator to responder</td>
</tr>
<tr>
<td>none of the previous conditions, but payload is transferred</td>
<td>the initiator has the larger port number</td>
</tr>
</tbody>
</table>

3. Intensive analysis of connection status: TCP or UDP connections are varying in time, so we run a modified powerful monitoring procedure to these in [187]. It tracks the connection and considers its flag SYN, ACK, FIN, and RST. Moreover, we define additional flags in the proposed method to cover all possible appearances in connections. Considered flags in our method are explained in Table 4-4.

<table>
<thead>
<tr>
<th>Flag</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>Connection attempt seen, no reply.</td>
</tr>
<tr>
<td>S1</td>
<td>Connection established, not terminated.</td>
</tr>
<tr>
<td>S2</td>
<td>Connection established and close attempt by originator seen, but no reply from responder.</td>
</tr>
<tr>
<td>S3</td>
<td>Connection established and close attempt by responder seen, but no reply from originator.</td>
</tr>
<tr>
<td>SF</td>
<td>The connection was normally established and terminated.</td>
</tr>
<tr>
<td>REJ</td>
<td>Connection attempt rejected.</td>
</tr>
<tr>
<td>RSTO</td>
<td>Connection established, originator aborted by sending a RST.</td>
</tr>
<tr>
<td>RSTR</td>
<td>Connection established, responder aborted by sending a RST.</td>
</tr>
<tr>
<td>RSTOS0</td>
<td>Originator sent a SYN followed by RST, we never saw a SYN ACK from the responder.</td>
</tr>
<tr>
<td>RSTRH</td>
<td>Responder sent a SYN ACK flowed by a RST, we never saw a SYN from the originator.</td>
</tr>
<tr>
<td>SH</td>
<td>Originator sent a SYN followed by a FIN, we never saw a SYN ACK from the responder.</td>
</tr>
<tr>
<td>SHR</td>
<td>Responder sent a SYN ACK followed by a FIN, we never saw a SYN from the originator.</td>
</tr>
<tr>
<td>OTH</td>
<td>Midstream traffic, we never saw a SYN.</td>
</tr>
</tbody>
</table>

In addition, in connection and events analysis block, we reassemble network packets and check the correctness of packets flow to determine the value of the feature wrong_fragment. As a result, analysis steps keep examine all network packets and hosts’ traps continuously, correlate them to construct connection vectors, monitor all databases and updates their contents, and at the last step,
they calculate all features for all connections in the current window to export a dataset accordingly.

Our queuing concept uses a database (DB) to save all constructed connections from network packets and correlate them with the hosts' events as long as the window is still inside the queue. Moreover, temporal databases (Temp. DB) are necessary to store uncorrelated connections or events, and if a closed connection is established again before reaching predefined timeout then it will be stored here for a certain time in such a way they become correlated or removed later. However, calculating of some features do not rely on the time slot window $w$, but on the number of same or different packets in the last connections, e.g. the last 100 connections as defined in KDDCup99. In our proposed method, we keep the last 1000 connections always as backlog to precisely calculate the features count, srv_count, dst_host_count and all other features derived from them are also calculated. Accordingly, we adjust backlog to accommodate up to 1000 connections. Finally, all windows $w_i, w_{i+1}, \ldots, w_n$ inside the queue will contain a certain number of instances $Dataset_i, Dataset_{i+1}, \ldots, Dataset_n$ from the correlated connections and events.

4.2.2.3. Export continuous datasets

At the end of the queue, each window contains a certain number of connections which are constructed based on the selected features (see section 6.1.2). The proposed method can export all connections to form a dataset in a Comma Separated Value (CSV) format or into a pipe concept based on FIFO principle (see figure 4-3). Exporting a complete dataset for all time slot windows has the benefit of providing researchers with a huge real-time dataset to train their model in the online or offline modes. In contrast, exporting the current dataset directly into a pipe is very effective for online anomaly detection. Therefore, the proposed method is adequate for the training or testing purposes in the online or offline operational modes.

4.3. Feature Selection and Extraction

Usually before collecting data, relevant features should firstly be determined. The main challenge of feature selection can be described as follows. If $F$ is the original set of features, then finding the optimal subset of relevant features $F'$ such as $F' \subset F$ is considered as the main challenge in the IDS. Advantages of feature selection can be summarized, but not limited to, in:

- Reducing the dimensionality of the input space.
- Disposal of irrelevant, redundant, and noisy features.
- Enhancing the data quality. Selecting only the relevant features provides an intensive exploration of data and hence better understanding them.
- Gaining more accuracy of the model

The criterion function and a subset searching method with a given criterion function are the concerned elements when choosing the best features set. The criterion function is used to measure the discriminating power of a feature subset. The subset searching method is an algorithm that explores the feature subset space in order to identify the best subset of features that optimizes the given criterion function [59]. In addition, feature selection techniques can be divided into three categories: wrapper; filter; and embedded methods.

**Wrapper methods** consider the classifier as a black box. They score the subsets of features based on their predictive power. Wrapper methods use a backward feature elimination scheme to recursively remove insignificant features from subsets of features. In each recursive step, the scheme ranks the features based on the amount of reduction in the objective function. It then eliminates the bottom ranked feature from the results.

**Filter Methods** select features based on discriminating criteria. They are relatively independent of classification (such as correlation coefficients). Filter Methods are considered faster than wrappers and embedded methods because they use heuristic algorithms.

**Embedded Methods** perform feature selection as a part of the training process and they adhere to the classification method.

Most of feature selection approaches have used the filter method Information Gain (IG) either as an individual step or combined with another methods to select the best feature subset. IG alone fails to identify discriminatory features [190]. Therefore, a hybrid feature selection method that combines both wrapper and filter is proposed in this thesis.

### 4.3.1. Hybrid Selection Method: Theoretical Approach

The most widely used wrapper methods in feature selection are the floating search methods. In this thesis, the Sequential Backward Search (SBS) has been adopted because it is very efficient in selecting features in large datasets. Hence, the predetermined features from KDD in intrusion detection (see appendix B) are considered to be evaluated using the SBS method. Features which positively affect the classifier performance (i.e. increase the detection rate and decrease the false positive rate) will be selected using our SBS method. In the next step, selected features from SBS will be ranked based on their uncertainty to the class label (i.e. normal or anomaly) using the IG method to abstract only features with high amount of information. Figure 4-4 depicts the proposed hybrid method.
4.3. Feature Selection and Extraction

4.3.1. Proposed wrapper method

The SBS method is considered one of the floating search methods. It shows a promising achievement on large datasets. In comparison to the Sequential Forward Search (SFS), SBS verifies the impact of a single feature on the classifier’s performance and decides if the feature is important or not. However, SFS examines the classifier’s performance by the presence of all features (i.e. it depends on the relations and linkage between features). The SFS method is infeasible in large datasets since the correlation (or the relation) between features obfuscate the individual impact of each feature. Therefore, in this thesis, a slight modification on the SBS method has been performed so that the impact of each feature can be individually examined. Algorithm 4-1 illustrates the major steps of the proposed SBS wrapper methods.

Algorithm 4-1: Proposed Sequential backward search algorithm

The previous algorithm uses a complete dataset, which is formed from \( F \) features and \( M \) instances. The first step is to remove a single feature and the corresponding values, i.e. deleting the entire column of this feature from the dataset. Then, evaluate an arbitrary classifier model using the dataset (with absence of the deleted feature) and notice the overall performance metrics (such as detection rate, false positive rate, and accuracy). In the next step, remove the
next feature and repeat the evaluation process again until no more features are available in the dataset. Finally, select all features which have positive impact on the classifier. For instance, if by removing the feature `protocol_type`, the performance has increased, then that means this feature has negatively affected the detection rate and the false positive rate and it must be ignored. Otherwise, the feature is important and should be considered as a valuable feature. Figure 4-5 illustrates the same algorithm from another perspective.

As a result, if we have 20 selected features in \( F_{\text{Plus}} \) from 50 features then the new dataset will be containing 20 columns instead of 50. This is to imply that the SBS has successfully reduced the dimensionality of the input space.

### 4.3.1.2. Proposed Filter Method

The selected features from the wrapper method can be used directly in the IDS research area, but in this thesis, these features will be further ranked according to the anomaly class label in the dataset. This step is essential to uncover features which apparently support in detecting anomaly. The instances in the reduced dataset from the SBS algorithm are labeled as normal or anomaly, hence, the filter method (by using the IG method) will find out which feature delivers the largest amount of information quantity with respect to the class label. However, IG in the proposed hybrid method handles only discrete values. Therefore, features with continuous value in the reduced dataset should firstly be discretized. Discretization techniques are used to preprocess amount of values.

![Figure 4-5: Flowchart of SBS main steps](image)
of a given continuous attribute, by dividing the range of the attribute into intervals. Most used discretization techniques are: binning methods or smoothing methods. There are three types of smoothing: smoothing by bin mean value, where each value in the bin is replaced by the mean value of the bin. The second one is smoothing by bin medians, where each bin value is replaced by the bin median. The last type is smoothing by bin boundaries, where the minimum and maximum values in a given bin are the bin boundaries. Accordingly, each bin value is then replaced by the closest boundary value. Another binning method is the equal width discretization, where each bin has a constant interval range, but it may produce unequal distribution of occurrences in the groups. The last discretization method is the equal frequency discretization. It divides the interval into \( k \) groups (Bins), where each group approximately contains the same number of occurrences. Equal frequency discretizer is simple and easy to implement. In this thesis, the equal frequency discretizer has been adopted to ensure fair grouping of the instances [42].

Let \( f \) be a continuous attribute and \( M \) be the number of instances associated with \( f \), then we can define \( n \) bins such that \( 1 \leq n \leq M \), \( n \) is user-defined parameter. The number of instances in each group will then equal \( M/n \). An example of equal frequency discretizer is shown in figure 4-6.

After discretizing the continuous features in the dataset, the IG can be used to rank the features.

Quantity of information or information gain uses the entropy to measure the uncertainty of attribute \( x \) associated with the value of the random variable \( X \) (also called Shannon entropy). If \( X \) and \( Y \) are discrete random variables and \( IG(X,Y) \) is the information gain of a given attribute \( X \) with respect to the class attribute \( Y \) (\( Y \) and \( X \) take values in \( \{y_1, y_2, \ldots, y_k\} \) and \( \{x_1, x_2, \ldots, x_k\} \)) with the probability distribution function \( P(X) \), then the entropy of \( X \) is given by:

\[
H(X) := -\sum_{i=1}^{M} P(X = x_i) \log_b(P(X = x_i))
\]  

(4-1)

Accordingly, the proposed IG of feature \( f \) in the dataset \( D \) is defined as:
4.4. Preprocessing

\[ IG(D,f) := H(D) - \sum_{\text{Attr} \in \text{value}(f)} \left[ \frac{|D_{\text{Attr}}|}{|D|} \cdot H(D_{\text{Attr}}) \right] \]

Where \( \text{value}(f) \) is the set of possible values of \( f \), \( D_{\text{Attr}} \) is the subset of \( D \) where \( f \) has the value \( \text{Attr} \), \( H(D) \) = entropy of feature class. For example, let \( f \) be the feature \text{protocol_type}, then \( \text{value}(f) := \{\text{TCP,UDP,ICMP,ARP}\} \), where \( D_{\text{ICMP}} \) is the subset that contains ICMP packets of feature \text{protocol_type}, and so forth.

Figure 4-7 shows the major steps of IG including the discretization.

![Diagram of IG main steps](image)

The proposed hybrid feature selection method uses benefit of both wrapper and filter methods to mine the most valuable and relevant feature set.

4.4. Preprocessing

Realistic network data, which come from heterogeneous and complex environments, are often noisy and redundant. Based on this fact, generated datasets should contain various feature types such as numeric and string features (see figure 4-1). In addition, some features will have different scales (i.e. one feature has small values and another feature has large values).

For instance, the feature \text{protocol_type} always has string values while the feature \text{count} only has numeric values. Likewise, the feature \text{logged_in} has small values and the feature \text{source_byte} often has large values. Due to these differences in feature types, data in the IDS models must be preprocessed so that all features have the same characteristics. Preprocessing is a very important data mining process in the KDD steps.

Converting string values into numeric ones is considered one of the challenges in preprocessing the data. In addition, different scales affect the classification task, which means, features with large scale values dominate features with small scale values, and hence mitigate their positive impact. Therefore, normalizing all features in the same scale is an essential preprocessing step in the IDS. In this
section, a significant explanation of the processing methods conversion and normalization will be presented.

4.4.1. Conversion of Nominal Features

Nominal features, such as protocol_type or service, are very important for classification methods especially by neural networks. Therefore, if symbolic values are not transferred into real values, the classifier will ignore them. Consequently, it will affect the performance of the classification, make the network more vulnerable, and lead to increase the anomaly.

Generally, for a predetermined time slot window \( w \), computer network traffic can be represented by a dataset \( D \) that consists of connection vectors. These connections are formally described as feature vectors \( x_1, x_2, \ldots, x_M \) where each vector consists of \( n \) attributes. Let \( x_i = (x_{i_1}, x_{i_2}, \ldots, x_{i_n}) \in \Omega_1 \times \ldots \times \Omega_n \), \( i = 1, \ldots, M \) be a feature vector such that \( x_{i_1} = \text{protocol\_type}, x_{i_2} = \text{service}, \ldots \) and \( M \) instances of the feature vector \( x_i \). Therefore, the dataset \( D \) can be formally described as a matrix \( X_M \) in the input space \( \Omega_M \) of the computer network as

\[
\Omega_M : X_M := \begin{pmatrix}
x_1 \\
\vdots \\
x_M
\end{pmatrix} = \begin{pmatrix}
x_{11} & \cdots & x_{1n} \\
\vdots & \ddots & \vdots \\
x_{M1} & \cdots & x_{Mn}
\end{pmatrix}
\] (4-3)

If we consider each column \( x_j = (x_{j1}, x_{j2}, \ldots, x_{jn})^T, j = 1, 2, \ldots, n \) in the matrix as \( M \) realizations of the \( j^{th} \) attribute, then we can interpret each feature as a realization of a random variable \( X_j \) of the space \( \Omega_j \).

Hence, a dataset \( D \) consists of \( M \cdot n \) values of the \( n \)-dimensional discrete random variables \( X = (X_1, X_2, \ldots, X_n) \) at \( \Omega_M := \Omega_1 \times \Omega_2 \times \ldots \times \Omega_n \).

Considering equation 4-3, network traffic can be described as \( M \) feature vectors of finite dimension \( n \in \mathbb{N} \), where each element represents a specification of a discrete random variable \( X_j \) with values from the probability space \( \Omega_j \). Let then \( X_j \) be a random variable with nominal values and \( x_{ij}, x_{i2}, \ldots, x_{ijn} \) are samples with \( K_j \) different nominal types \( \text{nom}^j_1, \ldots, \text{nom}^j_{K_j} \). We obtain the absolute frequency \( r_{kj} \) of the nominal type \( \text{nom}^j_k, k = 1, \ldots, K_j \) in \( X_j \) as

\[
r_{kj} = | \{ i \in \mathbb{N} \mid x_{ij} = \text{nom}^j_k, i = 1, \ldots, M \} |. \quad (4-4)
\]

Then we have, \( \sum_{k=1}^{K_j} r_{kj} = M \) and \( 0 \leq \frac{r_{kj}}{M} \leq 1 \). Hence,
4.4. Preprocessing

\[ f_{k_j} := \frac{r_{k_j}}{M}, k = 1, \ldots, K_j \tag{4-5} \]

is called the relative frequency of occurring the nominal feature type \( nom_k^j \) in \( X_j \). Finally, by using these relative frequencies we define the mapping \( pmf: \Omega_j \rightarrow [0,1] \) that transfers each nominal feature \( x_{k_j} \in \Omega_j \) into a real number as \( pmf(x_{k_j}) = f_{k_j} \).

For example, let \( X_i \) be a nominal feature \( protocol\_type \) with the following nominal values \{TCP, UDP, ARP, ICMP, ICMP, TCP\}, then we have \( M=6 \) and \( K_i=4 \) different types in the feature. By using the proposed conversion method, the relative frequencies that matches each feature type in the real space are

\[ pmf(TCP) = pmf(ICMP) = \frac{2}{6} = 0.33 \]
\[ pmf(UDP) = pmf(ARP) = \frac{1}{6} = 0.166. \]

Thus, nominal features are transferred to

\[ pmf(X_i) = \{0.33, 0.166, 0.166, 0.33, 0.33, 0.33\}. \]

4.4.2. Normalization Methods

The Normalization method removes the dominancy impact of various scales by bringing all values into the same domain scale. Most classifiers in the IDS handle only numeric values, but they still suffer from the different scales. Therefore, to eliminate the impact of dominancy between features, a normalization process must be performed on all features so that they will have the same scale (e.g. between 0 and 1). In this section, the most used normalization methods will be explained. In addition, section 5.3.3 will explain the use of normalization method in the online operational mode to generate normalized datasets from the real-time network traffic.

Let \( f: I \rightarrow [a,b], I = [\min, \max] \) be the normalization function and \( v \in \mathbb{R} \) is the numerical value of the feature \( X_j \), then the normalized value \( nv \) using three different normalization methods is determined as shown in Table 4-5.
# Preprocessing

Table 4-5: Widely used normalization methods

<table>
<thead>
<tr>
<th>Normalization Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal Normalization.</td>
<td>( nv := f_1(v) = \frac{v}{10^e} ), where ( e ) is the minimum number of positions such that maximum value drop into [0, 1].</td>
</tr>
<tr>
<td>Minimum-Maximum Normalization.</td>
<td>( nv := f_2(v) = \frac{v - \min(t)}{\max(t) - \min(t)} \cdot (b - a) + a ), where ( \min(v) ) and ( \max(v) ) are the minimum and maximum values of feature ( v ) and ( [a, b] ) is the new scale, e.g. ( a=0 ) and ( b=1 ).</td>
</tr>
<tr>
<td>Statistical (z-score) Normalization.</td>
<td>( nv := f_3(v) = \frac{v - \mu}{\sigma} ) where ( \mu ) and ( \sigma ) are the mean value and standard deviation of feature ( X_j ), respectively.</td>
</tr>
</tbody>
</table>

In the decimal normalization, the decimal point of \( v \) will be moved by \( e \) positions such that \( e \) is the minimum number of positions moved so that absolute maximum value falls in [0,1]. In contrast, statistical normalization or standard score computes the distance between the value and the mean value in unit of the standard deviation. This distance is the normalized value of the original value in the vector. The purpose of statistical normalization is to convert a data derived from any normal distribution into normal distribution with zero mean and variance = 1. On the other hand, minimum-maximum method sets the minimum value of the feature to 0 and the maximum value to 1 and normalizes all other values between 0 and 1.

In this thesis, the minimum-maximum normalization method has been selected due to its simple implementation. It achieves a very low CPU usage and does not consume memory resources as other methods. In addition, minimum-maximum normalization is more feasible than other methods in the online operational mode. It performs the normalization by only determining the minimum and maximum values of each feature. In contrast, other methods need to calculate some values such as the value \( e \) in decimal normalization or the mean value in statistical normalization.
Chapter 5. Classification

5.1. Growing Hierarchical Self Organizing Map

The most successful and popular model of ANN is the Growing Hierarchical Self Organizing Map (GHSOM). It is a modified approach from the Kohonen Self Organizing Map (SOM). SOM algorithm is based on unsupervised competitive learning. It can discover knowledge in data and abstract relations of high-dimensional data onto map units of two-dimensional plane. SOM has the property of topology preserving, which means that the mapping preserves the relative distance between the vectors. Vectors that are near each other in the input space are mapped to nearby map units in the SOM. Therefore, SOM can serve as a cluster analyzing tool of high-dimensional data. In addition, SOM network can recognize or characterize inputs that have never encountered before [191].

These characteristics in SOM show efficiency in detecting anomaly in computer networks. However, SOM has several shortages such as static architecture, expensive computation, and the random initialization of all parameters (e.g. the learning rate or the neighborhood radius). GHSOM solves these problems by structuring several SOMs in a hierarchical growing form [19].

Basically, SOM is a single-layer Feed-Forward neural network. SOM grid consists of \( N \) units (neurons or nodes), where each unit \( i \) is associated by a weight vector (also known as codebook) \( w_i = (w_{i1}, \ldots, w_{in}) \in \mathbb{R}^n, i = 1, \ldots, N \). Accordingly, the set \( \{w_i\}_{i \in \mathbb{N}} \) is called the codebook of the SOM grid. The number of units \( N \) on the grid is initialized randomly (e.g. 8x12) and all corresponding weight vectors, as well. Considering the input space \( \Omega_M \) in equation 4-3, one of the main tasks of SOM’s algorithm is to cluster all input vectors so that similar input vectors are grouped on the same unit on the grid. Figure 5-1 presents a visual overview for mapping \( M \) input vectors on the SOM.
5.1. Growing Hierarchical Self Organizing Map

All input vectors will be mapped onto a 2-dimensional grid that contains the final neurons which represent these input vectors. Each neuron can be represented as a center for the corresponding mapped input vectors and then form a cluster’s visualization. Therefore, SOM is well-known as an algorithm for clustering input data. Figure 5-2 demonstrates an example of a neuron and the corresponding mapped input vectors in a cluster form on the SOM grid.

As a result, the final SOM model can be defined by the final BMUs and their weight vectors. The label of each BMU is commonly determined from the majority of the mapped input vectors.

Based on the previous short introduction about SOM algorithm, the GHSOM can be considered as an amended version of SOM and it is able to discover supplementary details by structuring several maps in a horizontal and vertical growth. The next section will delve into the GHSOM algorithm and tacitly include the SOM algorithm.
5.2. **STANDARD GHSOM ALGORITHM**

This section will explain the main steps of GHSOM and tacitly the SOM algorithms. At the beginning a few fundamental definitions about the standard GHSOM will be presented to ease understanding the algorithm.

### 5.2.1. Fundamental Definitions

The standard GHSOM algorithm relies on some fundamentals, which are used during the training and building the final model. These are:

**The Projection Process**

The projection process maps input vectors from a high dimensional input space $\Omega_M$ on a two dimensional space grid $\mathbb{N}$. Specifically, the (SOM-) projection $\Phi: \Omega_M \rightarrow \mathbb{N}$ of the input space $\Omega_M$ on the SOM grid $\mathbb{N}$ provides clusters on the grid, so that each input vector $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ is mapped to the best weight vector $w_c \in \mathbb{R}^n$ of the neuron $c \in \mathbb{N}$. The best matching can be determined based on the minimal distance measure, i.e. the input vector is always mapped to the closest neuron, which is called the *winner*, the *Best Matching Unit* (BMU), or *winner-takes-all* unit as

$$
\Phi(x) := \arg \min_{i \in \mathbb{N}} \{d(x, w_i)\} = c
$$

(5-1)

In this thesis, the neuron $c$ will be called the BMU.

The amount of all vectors $x_l = (x_{l1}, x_{l2}, \ldots, x_{ln}) \in \mathbb{R}^n$ for $l = 1, \ldots, M$, which have been mapped on the same neuron $i$, is called the Receptive Field (RF), see figure 5-2, so

$$
RF_i = \{x \in \mathbb{R}^n | \Phi(x) = i, x \in \Omega_M\}.
$$

(5-2)

Based on the projection process, the final SOM grid will apparently have some neurons with plenty of input vectors, other neurons with a few amounts of input vectors, and some with no input vectors. Therefore, the total number of input vectors in the RF (i.e. $|RF|$ or is called later $n_{RF}$) can be considered as an "Attraction value" of the neuron.

**Distance Measurement**

As mentioned in equation 2-28, the Euclidean distance is one of the most used distance measure in machine learning. In addition, other distance measures can be also used, such as taxicab geometry (also is well-known as Manhattan.
distance or block distance). The block distance between any two vectors \( x, y \in \mathbb{R}^n \) in an n-dimensional space can be formally described as

\[
d(x, y) = |x_1 - y_1| + |x_2 - y_2| + \ldots + |x_n - y_n| \equiv \sum_{i=1}^{n} |x_i - y_i|
\]

(5-3)

More distance measurements are also available in [71],[192].

The Grid

The grid (also known as map or net) is a geometric representation for all units in each layer, which forms the final hierarchical topology. In this thesis, the notation \( \mathcal{N} = \mathcal{N}(N, w) \) means that a grid of \( N \) neurons and their corresponding weight vectors \( w = (w_1, w_2, \ldots, w_N), w_i \in \mathbb{R}^n \). The GHSOM is merely a structuring of several SOMs in a hierarchical growing form. Accordingly, the grid notation can be also used for GHSOM such as \( \text{GHSOM} : \mathcal{N}(N, w) \rightarrow \mathcal{N}_{rs}(N_{rs}, w_{rs}) \) where \( r \) is the layer number and \( s \) is the \( s^{th} \) grid that has \( N_{rs} \) neurons of \( w_{rs} \) weight vectors. For example \( \mathcal{N}_{44}(N_{44}, w_{44}) \) represents the fourth grid in the second layer.

5.2.2. Algorithm Main Steps

The main algorithm of GHSOM can be described in the following steps:

1- Initialization

The first step in GHSOM starts by preparing the training dataset \( X_M \) (the aggregated network traffic as described in equation 4-3) and accordingly initializing a single layer \( L_0 \) with a single map \( \mathcal{N}_0 = \mathcal{N}(N_0, w_0) \) that has only one single node \( (N_0=1) \) called the root with a weight vector \( w_0 = (w_{01}, w_{02}, \ldots, w_{0n}) \in \mathbb{R}^n \). Usually, the weight vector of the corresponding root node is initialized randomly or by the mean value of the input dataset. Referring to equation 4-3, the weight vector can be then calculated as

\[
w_0 = \frac{1}{M} \sum_{i=1}^{M} x_i
\]

(5-4)

Therefore, the entire dataset is considered as the initial receptive field \( RF_0 \). The second step is determining the quantization error \( (q_{e_p}) \) and the Mean Quantization Error (MQE_p) for the root node (as a parent node \( p=0 \)).

\[
q_{e_p} = \sum_{i=1}^{M} \left| x_i - w_0 \right|, \quad \text{MQE}_p = \frac{1}{M} \cdot q_{e_p}
\]

(5-5)
5.2. Standard GHSOM Algorithm

The qe measures the error of conformity between the input connections and the initialized weight vector to reinforce the overall topology. Therefore, initializing the weight vector meaningfully affects positively on the topology. These values will be used later in the training process. The third and the final step is initializing the first layer $L_1$ with a single map $\mathbb{N}_1 = \mathbb{N}(N_1, (w_{11}, w_{12}, w_{21}, w_{22}))$ that has normally $N_1 = 4$ neurons (2x2). Indeed, these neurons are initialized randomly. Normally, the number of iterations $\lambda$ is determined in the initialization step. However, sometime it is determined in the training process.

2- Training

The training (also known as learning or knowledge exploration) is a process of exploring behavior or knowledge between the input vectors after certain number of iterations. The training causes to expand the map horizontally by adding further units on the original map (i.e. the 2x2 map) or to extend the topology vertically by adding new layers. Hence, from the artificial neural network prospective, it is deemed as an important process in updating the weight vectors. The training steps are the same for all maps but each map is trained individually.

Note that, there is always a parent node for each map in the topology. For instance, the root node in $\mathbb{N}_0 = \mathbb{N}(N_0, w_{0})$ is the parent node for $\mathbb{N}_1$ in the first layer.

In training the GHSOM, there are two complementary processes: the competition and the cooperation.

2-a: Competition process

This is the first process in training the GHSOM, where each neuron competes with other neurons to win the current projected input connection. Formally, the competition process starts by choosing a random input connection from the input dataset $X_M$. The neuron that has the minimal Euclidean distance is called the winner or the BMU as described in the projection process (equation 5-1).

2-b: Cooperation process

This is the complementary process for the competition, whereas the weight vector of the winner and the neighbor neurons $N_c$ (neuron in the vicinity) are cooperated together by adapting their weights a bit toward the input vector based on the following rule
5.2. Standard GHSOM Algorithm

\[ w_i(t+1) := \begin{cases} w_i(t) + \alpha(t) \cdot h_{ci}(t) \cdot [x(t) - w_i(t)], & i \in N_c \\ w_i(t), & i \notin N_c \end{cases} \quad (5-6) \]

where \( \alpha(t) \) is the learning rate that guides the weight vector movement and decreases in time. \( h_{ci}(t) \) is the neighborhood function that determines the adapted units in the vicinity of the winner neuron, which are also decreasing in time. Examining this adaption process (after each iteration) will group the input connections, that are similar in the input space, together on the same winner and build a so-called cluster form [19]. The relationship between the learning rate (or the neighborhood function) and the neighbor neurons of the winner is proportional; that is, decreasing the learning rate with time will minimize the amount of neighbor neurons of the winner.

**Learning rate**

The learning rate is responsible for decreasing the distance between the input connection and the weight vectors of winner and neighbor neurons. There are three known learning rate functions which are commonly used in the GHSOM:

1- Exponential learning rate

\[ \alpha(t) := \alpha_0 \cdot e^{-\frac{n}{\alpha_1}}, n = 0,1,2,\ldots \quad (5-7) \]

where \( \alpha_1 \) is a time constant (usually is the number of iterations).

2- Linear learning rate

\[ \alpha(t) := \alpha_0 \cdot \left(1 - \frac{n}{A}\right), n = 0,1,2,\ldots \quad (5-8) \]

where \( A \) is a constant (usually is the number of iterations).

3- Inverse time learning rate

\[ \alpha(t) := \alpha_0 \cdot \left(\frac{A}{B + n}\right), n = 0,1,2,\ldots \quad (5-9) \]

where \( A \) and \( B \) are constant.

In all cases, initial learning rate \( \alpha_0 \) must satisfy the following condition

\[ 0 < \alpha(t) < 1. \]

Example: if we consider \( \alpha_0 = 0.3 \), number of iterations = 400, figure 5-3 shows the corresponding learning rate functions.
Although the above learning rate functions are different in action but they decrease in time. Thus, in this thesis, it is possible to train the model by any one of them.

**Neighborhood function**

The neighborhood function in the training process serves as a selection function, which means it is responsible for determining the nearest neurons (those should be affected by the adaption process) around the winner based on certain radius \( \sigma \). During the training, the radius of the neighborhood function decreases in time which leads in turn to decrease the neighborhood area around the winner.

Gaussian neighborhood function is the most used function in GHSOM training, which is defined as

\[
h_{ci} = e^{-\frac{d(r_i,r_j)^2}{2\sigma(t)^2}} \tag{5-10}
\]

where \( d(r_i,r_j) \) is the distance between the position of the cooperated neuron \( r_j \) and the position of the winning neuron \( r_i \). As mentioned before, \( \sigma(t) \) is the effective width of the topological neighborhood. It measures the degree of participation, from cooperated neurons in the vicinity of the winner, in the training step. The most popular choice of radius function is the exponential decay function such as

\[
\sigma(t) = \sigma_0 \cdot e^{-\frac{n}{\sigma_i}}, n = 0,1,2,\ldots, \sigma \tag{5-11}
\]

where \( \sigma_0 \) is a constant value between 0 and 1, \( \sigma_i \) is the total number of instances, and \( n \) represents the iteration number. Linear or inverse functions can be used here too but they are not popular as exponential.
5.2. Standard GHSOM Algorithm

A general representation of a BMU and the neighborhood area using Gaussian function is depicted in figure 5-4.

Figure 5-4: Gaussian neighborhood function for the BMU and the neurons in the vicinity

Based on the previous Gaussian diagram, the top is representing the BMU and all other neighbor neurons are located in the vicinity. The neighborhood topology in a single SOM grid can be rectangular which is the common form or hexagonal as illustrated in figure 5-5.

Figure 5-5: neighborhood topology in a single SOM grid

Notice that, the neighborhood radius decreases in time until reaching the winner neuron (the black dot inside).

Another commonly used neighborhood function is the bubble neighborhood function. A brief comparison between both neighborhood functions is presented in figure 5-6.
5.2. Standard GHSOM Algorithm

- Bubble neighborhood is faster to compute, so that it saves time in training process. But Gaussian neighborhood is more biologically appropriated than bubble one.
- Moreover, Gaussian topology decreases monotonically (consistence) with increasing distance between winning neuron and cooperated neurons, whereas bubble does not.
- Gaussian neighborhood function makes the SOM algorithm converge more quickly than bubble neighborhood function would.

As a result, in the cooperative process, the learning rate function, radius function, and neighborhood function should be determined before the training starts, and then the update rule in equation 5-6 will be examined on the winner and its neighbors. When the update is finished the following condition should be satisfied $h_{ci}(t + 1) < h_{ci}(t)$ otherwise the competition and cooperation processes (2-a and 2-b) must be repeated on another projected input connection. In case that the condition is satisfied, the algorithm picks a new input connection $x_j$ from $X_h$ and examines the competition and cooperation processes again and so forth until the end of iterations.

Notice: the previous two steps (competition and cooperation) are considered the main steps in SOM algorithm. Obviously, the grid size is static and should be determined at the beginning. Moreover, only the winner weight vector (and the neighbors as well) will be adapted with the input vectors and other parameters, such as the topology, will be ignored.

To overcome these shortcomings, the GHSOM has presented a growth strategy to build a hierarchical topology and to update not only the weight vectors but also the grid topological parameters.
3- Growth Strategy

The strategy of growing is to control the topology in horizontal and vertical directions. To achieve this goal, the GHSOM uses two threshold values. These are the horizontal threshold $\tau_1$ and the vertical threshold $\tau_2$, so that $0 < \tau_2 < \tau_1 < 1$. The former examines if more neurons should be added in the current grid, while the latter is to decide if new layers with child grids should be added from the current grid. Note that, each grid on the topology has a parent neuron $p$ in the previous layer $l - 1$ e.g. the root node is the parent of the first grid $L_1$.

After finishing the training step on the current map, i.e. the algorithm has examined the competition and cooperation process for $\lambda$ iterations, the two growth conditions will be examined on each grid as the following:

3-a Horizontal Growth

In horizontal growth, we firstly calculate the mean quantization error $mqe_i$ of each neuron $i$ on the current map (mostly we calculate the quantization error $qe_i$) by using equation 5-2

$$mqe_i = \frac{1}{|RF_i|} \sum_{x \in RF_i} \|x - w_i\|.$$  \hspace{1cm} (5-12)

Secondly, we calculate the mean quantization error of the grid. Let $BMU_{map} := \{i \in map \mid RF_i \neq \phi\}$ be the set of all neurons on the grid, which have been selected as winners during the training of the grid. Thus we can define the mean quantization error for the grid as

$$MQE_{map} = \frac{1}{|BMU_{map}|} \sum_{i \in map} mqe_i.$$ \hspace{1cm} (5-13)

Based on equations 5-12 and 5-13, the horizontal growth examines the following condition

$$MQE_{map} \geq \tau_1 \cdot qe_p \text{ (or \ } mqe_p\text{)}.$$  \hspace{1cm} (5-14)

where $p$ is the parent node for the current grid.

As long as the condition in equation 5-14 is satisfied, determine the node $e_{map}$ that has the highest mean quantization error (also called the Highest Error Node) and its most dissimilar node $i_d$ on the map as
5.2. Standard GHSOM Algorithm

\[ e_{map} = \arg \max_{e \in \text{map}} \{mqe, \ldots, mqe_{BMU} \}, \quad i_d = \arg \max_{i \in N_e} \|w_e - w_i\| \]  \hspace{1cm} (5-15)

where \( N_e \) are the neighbor neurons of the node \( e_{map} \), \( w_e \) and \( w_i \) are the weight vectors of the nodes \( e_{map} \) and \( i_d \), respectively. Next, add a new row or column between the nodes \( e_{map} \) and \( i_d \) based on the location of the most dissimilar node in the grid. Finally, the weight vector of each new node will be initialized from the mean value of the weight vectors of its neighbor nodes. Figure 5-7 demonstrates a node insertion for both cases.

![Figure 5-7: Horizontal growth by adding row or column](image)

The horizontal growth of any grid should be limited e.g. 50x50 can be the maximum growth for each grid in the topology.

On the other hand, if the condition in equation 5-14 is not satisfied then the second condition of the vertical growth should be examined.

3-b Vertical Growth

If the horizontal condition is not valid anymore, then the mean quantization error of each node on the current grid will be examined by the following condition

\[ mqe_i > \tau \cdot mqe_{BMU}, \quad i = 1, 2, \ldots, |BMU_{map}| \]  \hspace{1cm} (5-16)

As long as this condition is satisfied, we add a new layer \( l+1 \) from corresponding units and initialize a new grid 2x2 on this layer based on the first step of GHSOM algorithm (i.e. Initialization) and repeat the second step (i.e. training) on the new grid with the same number of iterations and so forth. In this case, if the node has satisfied the condition in equation 5-16, it will be considered as a parent node and its receptive field will be considered as an input dataset for the new grid.
Once the condition is no longer valid for any node on the grid then the algorithm stops and this will be the final step. As a result, the topology will consist of several layers and each layer has a certain number of grids. The winners are scattering on these grids so that each winner has a final weight vector, receptive field and a label that is determined from the majority of input vectors’ labels in the receptive field. Figure 5-8 illustrates an example of a GHSOM final hierarchy.

Figure 5-8: Example of a final GHSOM topology

In the previous figure, as an example, layer 3 includes 3 grids and 7 BMUs. In this thesis, a novel and expressive flow diagram that illustrates the main steps of GHSOM algorithm, is proposed in the figure 5-9.
5.2. Standard GHSOM Algorithm

Hence, the GHSOM algorithm can be summarized as follows:

1. Input dataset $X := (x_1, ..., x_M)^T$; $M$ = number of input vectors
   For all input vectors initialize layer 0 with a single root node and
   
   $w_0 := \frac{1}{M} \sum_{m=1}^{M} x_m$, $mqe_0 := \frac{1}{M} \sum_{m=1}^{M} \| x_m - w_m \|$, $q_{e0} := \sum_{m=1}^{M} \| x_m - w_m \| q_{e0} := q_{e0}$

2. Add layer 1 with a 2x2 map, note that SOM-Grid $ℵ$ consists of 4 neurons.

3. Initialize weight vectors randomly

Figure 5-9: Flow chart of the GHSOM algorithm main steps
5.2. Standard GHSOM Algorithm

Start Training
4: Initialize number of iterations randomly, e.g. $\lambda = 200$
5: **Competition process**
6: Select random input $x(t)$
7: Find the *winner* neuron, i.e. Best Matching Unit (BMU)
   
   \[
   c := \arg \min_{w_i} \left\| x(t) - w_i(t) \right\|
   \]
8: **Cooperative process**
9: Update the *winner* $c$ and its neighborhood neurons $N_c$
   
   \[
   w_c(t+1) := w_c(t) + h_c(t) \cdot \left[ x(t) - w_c(t) \right], \quad i \in N_c
   \]
   
   where
   \[
   h_c(t) := \alpha(t) \cdot e^{-\sigma^2(r_c^2)}
   \]
10: Check that $h_c(t+1) < h_c(t)$
11: repeat steps 4 – 10 until $\lambda = 0$

Growth Control
12: determine growth conditions $0 < \tau_i < \tau_i < 1$
13: Calculate the mean quantization error of the map $m$, $MQE_m$.
   
   \[
   MQE_m := \frac{1}{n} \sum_{i = 1}^{n} qe_i, \quad n \text{ is the number of neurons in map } m
   \]
   
   Note that for a neuron $s$, $qe_i := \sum_{x \in RF_i} \left\| x - w \right\|, \quad i \in N_m$, $RF_i$ is the receptive field
14: Examine the horizontal growth condition $MQE_m > \tau_i \cdot qe_p$
   
   where $p$ is the parent node, e.g. map on layer 1 has the parent node $qe_0$ on layer 0
15: If the condition is satisfied
   Calculate the highest error node $e$ on the map $m$ and the most dissimilar node $i_d$ of node $e$.
   
   \[
   e = \arg \max_{e \in N_m} \{ mqe_e, \ldots, mqe_{BMU(e)} \}, \quad i_d = \arg \max_{i \in N_e} \left( \left\| w_e - w_i \right\| \right)
   \]
16: Insert new column or row between $e$ and $i_d$ and initialize the weight vectors of the new column or row as the mean of neighborhood nodes
17: If the condition in step 14 is not satisfied.
18: Calculate $mqe_i$ of each neuron on the map $m$.
   
   \[
   mqe_i = \frac{1}{\left| RF_i \right|} \sum_{x \in RF_i} \left\| x - w \right\|, \quad i \text{ is the total number of BMUs on the map } m
   \]
19: For each neuron, examine a vertical growth condition $mqe_i > \tau_i \cdot mqe_p$ ($p$ is parent)
20: If the condition is satisfied, calculate $qe_p$ for the expanded node
   
   Add new layer with a 2x2 grid
   Initialize weight vectors randomly and repeat steps 4 – 10
21: If the condition is not satisfied then STOP training.

Algorithm 5-1: GHSOM Algorithm main steps
Thus, the GHSOM model can be finally defined as
\[
\left( C, w_C, RF_C, \text{Labels} \right)
\] (5-17)

Based on the algorithm, table 5-1 represents the GHSOM.

Table 5-1: Final GHSOM Model

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMUs (Winners)</td>
<td>( C := {c_1, c_2, \ldots, c_m}, m = {</td>
</tr>
<tr>
<td>Corresponding Receptive Fields</td>
<td>( RF_i := {x \in \mathbb{R}^n \mid \Phi(x) = i, x \in \Omega_M}, i = 1, \ldots, m )</td>
</tr>
<tr>
<td>Corresponding Weight Vectors</td>
<td>( w_C := {w_1, w_2, \ldots, w_m} )</td>
</tr>
<tr>
<td>Labels</td>
<td>{Normal, Anomaly}</td>
</tr>
</tbody>
</table>

Hereby, \( \Phi(x), RF, \mathbb{N}, m, M \) are the projection function, the receptive field of a BMU \( c \), the grid or the map, number of final BMUs, and the number of input vectors in the dataset, respectively.

5.3. Proposed Theoretical Modifications on GHSOM

The final model of the standard GHSOM has been exploited in the area of network security to uncover suspicious behavior on the network. It is able to classify online connections to normal or abnormal based on the closest BMU to the input connection, but it is not able to uncover unknown or new anomaly connections. The final model of the standard GHSOM classifies a new connection \( \hat{x} = (\hat{x}_1, \ldots, \hat{x}_n) \) by presenting it to the final GHSOM model \( (C, w_c, RF_c, Label) \) and determining the smallest distance to all BMUs as \( \hat{d} := \min\{d(\hat{x}, w_c) \mid c \in C\} \) and the corresponding BMU \( \hat{c} := \arg\min\{d(\hat{x}, w_c) \mid c \in C\} \). Thus, the new connection \( \hat{x} \) is assigned on \( \mathbb{N} \) with the pair \((\hat{c}, \hat{d})\), and it can be labeled according to the following two cases:

Case 1: If \( \hat{d} = 0 \) then \( \hat{c} = c \in C \) with a weight vector \( w_c \). In this case the input connection \( \hat{x} \in \mathbb{R}^n \) will be labeled the same as the neuron \( \hat{c} \).
5.3. Proposed Theoretical Modifications on GHSOM

Case 2: if \( d > 0 \) find the BMU \( \hat{c} \) that has the smallest distance to \( \hat{x} \) such that \( \hat{c} = \arg\min_{c \in C} \{d(\hat{x}, w_c)\} \) and, hence, the input connection \( \hat{x} \in \mathbb{R}^n \) will be labeled the same as the neuron \( \hat{c} \).

Apparently, the standard GHSOM suffers from the inability to detect new or unknown anomaly connections. In addition, it has a static final model that remains inadaptable in real-time. Therefore, the final model still experiences various shortages:
- Shallowly construction due to imperfect training parameters.
- Insignificant initialization method.
- Weak topology control. Only weight vectors are affected by the adaption processes in training and the topology is ignored.
- Classification-confidence threshold that is able to recognize the new or unknown connections.
- Real-time classification.
- Missing the Adaptivity option in real-time.

Note that, current research achievements used to employ the standard GHSOM in intrusion detection, which lead to insufficient results of current IDS systems.

In this section, several new enhancements are presented to handle these shortages. Therefore, we call our new model, i.e. after applying the following enhancements, Enhanced-GHSOM (EGHSOM).

5.3.1. Enhancement 1: Dynamic Classification-Confidence

Principally, the final model of GHSOM reveals malicious activities based on the smallest distance and the label of the concerned BMU. Accordingly, network connections will always be classified to normal and abnormal, and hence unable to detect unknown connections. This property in GHSOM is considered as a major drawback.

Therefore, there must be a threshold boundary that indicates if the connection should be mapped to one of the BMUs in the GHSOM model. This threshold is responsible for uncovering unknown connections. In this thesis, we propose two dynamic threshold criteria to be able to detect the new or unknown anomaly in the real-time.

Classification-Confidence using Average Distance Threshold

This proposed threshold criterion is abstracted from the idea of the mean quantization error, which can be used to determine the distance between vectors. Based on figure 5-2, we can determine the average distance of the BMU \( c_i \) to all corresponding vectors in the \( RF_i \) using the mean quantization error. Consequently, we determine the average distance \( \theta \) from all mean quantization errors (the average of the average distances in \( RF_i \)) then we defined a threshold value that is able to differentiate between known and
unknown connections. By using the equation 5-12, the average distance threshold value $\theta$ can be obtained as

$$\theta := \frac{1}{m} \sum_{c_i \in C} m_{qs_i}, m = |C|$$  \hspace{1cm} (5-18)

Hence, to classify a new connection $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_{n})$, we calculate the pair $(\hat{c}, \hat{d})$ so that $\hat{d} := \min\{d(\hat{x}, w_c) \mid c \in C\}$ and $\hat{c} := \arg\min_{c \in C}\{d(\hat{x}, w_c) \mid c \in C\}$ then examine the cases:

**Case 1:** if $\hat{d} \leq \theta$ then $\hat{x} \in RF_{\hat{c}}$ and $\hat{c}$ is the mapped BMU to the new connection $\hat{x}$. Accordingly, the connection $\hat{x}$ will be labeled the same as $\hat{c}$. Moreover, the $mqe_{\hat{c}}$ will be adapted and the average distance threshold $\theta$ consequently. That means $\theta$ will be dynamically adapted in real-time.

**Case 2:** if $\hat{d} > \theta$ then $\hat{x} \notin RF_{\hat{c}}$ and that means $\hat{x}$ matches no BMU so it is classified as *unknown anomaly*.

Figure 5-10 shows an example that illustrates the classification process using the average distance threshold.

<table>
<thead>
<tr>
<th>BMUs = C</th>
<th>RF vectors = RF</th>
<th>Distances for each BMU</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1 = (1,0,1,2)$</td>
<td>$x_1 = (1,3,2,0)$</td>
<td>3.74</td>
</tr>
<tr>
<td>Normal</td>
<td>$x_2 = (1,0,2,1)$</td>
<td>1.41</td>
</tr>
<tr>
<td></td>
<td>$x_3 = (1,1,1,1)$</td>
<td>1.41</td>
</tr>
<tr>
<td></td>
<td>Sum</td>
<td>6.56</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>2.186666667</td>
</tr>
<tr>
<td>$c_2 = (1,3,2,1)$</td>
<td>$x_1 = (1,4,1,0)$</td>
<td>1.73</td>
</tr>
<tr>
<td>Anomaly</td>
<td>$x_2 = (1,0,1,1)$</td>
<td>3.16</td>
</tr>
<tr>
<td></td>
<td>$x_3 = (1,4,3,1)$</td>
<td>1.41</td>
</tr>
<tr>
<td></td>
<td>$x_4 = (0,0,1,2)$</td>
<td>3.46</td>
</tr>
<tr>
<td></td>
<td>Sum</td>
<td>9.76</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>2.44</td>
</tr>
</tbody>
</table>

Figure 5-10: illustration example of the average distance threshold

In the above example, $c_1$ is a BMU labeled as normal and $c_2$ as anomaly. The first BMU has 3 input vectors in its RF and $c_2$ has 4 vectors. If we consider the average distance threshold equation 5-18, we find that $\theta = 2.313$. Let’s assume that the incoming input $\hat{x} = (3,9,0,2)$ so, $\|\hat{x} - c_1\| = 9.3$ and $\|\hat{x} - c_2\| = 6.7$. That means $\hat{x}$ is closer to $c_2$ than $c_1$ so it should be classified as anomaly. But this is not a confidant classification because we do not know if the incoming input related to the neighbor of the selected BMU and hence it is incorrect. Although
5.3. Proposed Theoretical Modifications on GHSOM

\( \hat{x} \) closer to \( c_2 \), it is not even close to one of its neighbors, i.e. \( \hat{x} \notin RF_{c_1} \). But by using the dynamic threshold \( \theta \), it is obvious that \( \hat{x} \) is very far from \( c_2 \) while \( \| \hat{x} - c_2 \| > \theta \), so it must be classified as unknown anomaly. This example is merely an illustration and cannot be considered valid for a general approach.

**Classification-Confidence using Margin Distance Threshold**

On the other hand, in the second threshold criterion, upper bound and lower bound have been abstracted from the final BMUs and the corresponding RFs and used as a threshold margin \( \delta \) to detect unknown connections. For each BMU \( c \in C \) we determine the minimum and maximum distances with the corresponding RF as

\[
\begin{align*}
    d_{\min} &:= \min\{d(x, w_i) \mid x \in RF_i, i = 1, 2, \ldots, m \} \\
    d_{\max} &:= \max\{d(x, w_i) \mid x \in RF_i, i = 1, 2, \ldots, m \}
\end{align*}
\]  

(5-19)

Hence, the minimum and maximum distances for all BMUs can be determined as

\[
\begin{align*}
    d_{\min} &:= \min\{d_{\min}, \ldots, d_{\min} \} \\
    d_{\max} &:= \max\{d_{\max}, \ldots, d_{\max} \}
\end{align*}
\]

(5-20)

Thus, the threshold margin is defined as

\[ \delta := [d_{\min}, d_{\max}] \]  

(5-21)

In real-time, let \( \hat{x} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n) \) be the incoming connection, then the smallest distance to all BMUs in the model can be determined as \( \hat{d} := \min\{d(\hat{x}, w_c) \mid c \in C \} \). According to the proposed threshold margin, the connection \( \hat{x} \rightarrow (\hat{c}, \hat{d}) \) can be classified into one of the following three cases:

**Case 1**: if \( \hat{d} < d_{\min} \) then a BMU \( \hat{c} \) is the closest neuron to \( \hat{x} \). Accordingly, the connection \( \hat{x} \) will be labeled the same as \( \hat{c} \). Moreover, the margin distance will be updated such that \( \delta := [\hat{d}, d_{\max}] \).

**Case 2**: if \( d_{\min} < \hat{d} < d_{\max} \) then \( \hat{x} \in RF_{\hat{c}} \) and it is mapped to the BMU \( \hat{c} \). The label of \( \hat{x} \) is same as the label of \( \hat{c} \) and the threshold margin will not be updated.

**Case 3**: if \( \hat{d} > d_{\max} \) then \( \hat{x} \notin RF_{\hat{c}} \) and it is classified as unknown-Anomaly. However, if \( \hat{x} \) later classified as “Normal” or even “Anomaly” then a new
neuron is added to the BMUs with a weight vector the same as \( \hat{x} \) and the margin threshold will be then updated accordingly. Obviously, using one of the above two threshold criteria will detect directly the unknown traffic and deliver an accurate classification as well.

5.3.2. Enhancement 2: Reinforcing Final Best Matching Units

The BMUs play a dominant role in classifying the incoming connections. In other words, if the attraction value \( n_{RF} \) (the amount of input vectors in the RF based on equation 5-2) of a BMU during the training is large enough the BMU will be active in the online classification mode, otherwise it will be inactive. Thus, reducing the number of BMUs in the final GHSOM model will accelerate the classification process especially in real-time. Therefore, the BMUs should be further examined to boost the topology and avoid such weakness. Some weaknesses on the final BMUs are:

- A receptive field of BMU \( c_i \in C \) has obviously very few instances \(|RF_i| \leq n_{RF_i}\) (i.e. it was not attractive during the training), this is a weak BMU.
- If a RF of a certain BMU is crossing another BMU’s RF area. Let \( d_{\text{max}} := \max \{d(x, w_i) \mid x \in RF_i\} \) is the maximum distance between the BMU \( c_i \) and its outermost input \( x \) in the \( RF_i \), and \( d_{ij} := d(w_i, w_j) \) is the distance between the BMUs \( c_i, c_j \), where \( c_j \) is the closest neuron to \( c_i \). So if \( d_{ij} < d_{\text{max}} \) the BMUs are considered coherent. These are coherent BMUs.
- If a BMU always misclassifies vectors (could be active, weak or coherent).

Such BMUs degrade the accuracy and increase the false positive alarm. To solve these problems, individual solutions are presented in this thesis to remedy these BMUs. For instance, healing a weak BMU that still has input vectors in its RF is more effective than deleting it. Figure 5-11 shows a visual illustration of a weak BMU and two coherent BMUs, where the instances in the receptive fields are marked as circles.

![Figure 5-11: Illustration of a weak BMU and coherent BMUs](image-url)
Merging a Weak BMU

The competition process some BMUs in the final model were less attractive for the input. A BMU is considered weak if the number of input vectors mapped on it is less than the attraction value (i.e. the amount of mapped inputs on this neuron ≤ n_{RF}). In this regard, we propose a remedy technique that merges this BMU with the most similar BMU that has the same label. Formally, if the BMU \( c_i \) has \( k \) input vectors such that \( k = |RF_i| \) and \( k \leq n_{RF} \), then this BMU is not attracting the input vectors. Therefore, it must be merged with the strongest similar BMU, at the same grid, that has the same label. If we assume that the neuron \((c_1, w_1, RF_1)\) is considered as a weak BMU, i.e. the condition \( k \leq n_{RF} \) is fulfilled, and the most similar BMU is \((c_2, w_2, RF_2)\), where similarity \( r_{ij} \) between two BMUs is:

\[
 r_{ij}(c_1, c_2) = \frac{\sum_{t=1}^{n} w_{1t} \cdot w_{2t}}{\sqrt{\sum_{t=1}^{n} w_{1t}^2 \cdot \sum_{t=1}^{n} w_{2t}^2}}, \quad w_1, w_2 \in \mathbb{R}^n . \tag{5-22}
\]

Hereby, find \( \max\{r_{ij}(c_1, c) \mid c \in \mathbb{R}_i \} \geq 0.5 \), where \( i \) is the map number (if there are many BMUs, take the BMU with the largest similarity). Therefore, these BMUs and their receptive fields must be combined together. The combination is simply as the following: a new neuron on the grid \((c_{new}, w_{new}, RF_{new})\) is created so that the \( c_{new} \) has a weight vector \( w_{new} = \frac{w_1 + w_2}{2} \) as a mean value from both. All instances in both receptive fields will be combined such that \( RF_{new} = RF_1 \cup RF_2 \).

Hence, merging the weak BMU will modify the grid, reduce the number of BMUs, and accelerate the classification in real time.

If a BMU is weak, it is then not sufficiently representing the entire input vectors. Thus, this BMU will most probably classify the input vectors in the online classification incorrectly. That leads to degrade the accuracy of the detection model. In the sense, the normal input vector will be classified as anomaly and vise versa. In general, the accuracy is the percentage of correctly detected connections to the total number of connections.

The tradeoff curve between the attraction value \( n_{RF} \) and the accuracy is shown in figure 5-12.
5.3. Proposed Theoretical Modifications on GHSOM

The curve shows that the number of input vectors in the RF should be between 150 and 450 vectors to consider it a weak BMU, which indeed leads to increase the accuracy.

**Addressing the coherent BMUs**

The triple $BMU_c : (c, w_c, RF_c)$ can be considered as a cluster form as demonstrated in figure 5-10. Optimal clustering solution is fulfilled when inter-cluster distances are maximized and intra-cluster distances are minimized [193]. Hereby, inter-cluster is the distance between two BMUs and intra-cluster is the distance between the BMU and the corresponding input vectors in the receptive field. In this concern, if two BMUs are very close together the classification performance will be negatively impacted. This is to imply that one BMU is in the neighborhood of the second one (see figure 5-11). In such case, these BMUs are called coherent. The proposed method will prune coherent BMUs by updating their weight vectors using $K$-means algorithm [75],[76] such that a sufficient separation between them is performed. Moreover, this proposed method aims to accurately reassign input vectors of both receptive fields on the correct BMU.

Suppose $c_1, c_2$ are coherent BMUs and have the same label. In addition, let $\theta_1, \theta_2$ are the average distance (the $mqe$ as shown in equation 5-12) of $c_1, c_2$ respectively. Thus, the following condition should be fulfilled to consider these BMU coherent ones:

$$d(w_1, w_2) < \theta_1 \quad or \quad d(w_1, w_2) < \theta_2 \quad (5-23)$$

where $d$ is the distance.
5.3. Proposed Theoretical Modifications on GHSOM

Once this condition is fulfilled, a K-means algorithm is examined on these BMUs to update their weight vectors and to correctly reassign input vectors on both of them. Algorithm 5-2 describes the steps of addressing coherent BMUs.

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>For $i = 1 : m$ // $m$ is the number of BMUs</td>
</tr>
<tr>
<td>2:</td>
<td>// calculate the mqe of each BMU, i.e. intra-distance</td>
</tr>
<tr>
<td>3:</td>
<td>$mqe_i$, $mqe_j$ // see equation 5-12</td>
</tr>
<tr>
<td>4:</td>
<td>End:</td>
</tr>
<tr>
<td>5:</td>
<td>For $i = 1 : m$</td>
</tr>
<tr>
<td>6:</td>
<td>For $j = 1 : m / j \neq i$</td>
</tr>
<tr>
<td>7:</td>
<td>Find $d(w_i, w_j)$</td>
</tr>
<tr>
<td>8:</td>
<td>IF $d(w_i, w_j) &lt; mqe_i \lor d(w_i, w_j) &lt; mqe_j$</td>
</tr>
<tr>
<td>9:</td>
<td>Then $(e_i, e_j)$ are coherent</td>
</tr>
<tr>
<td>10:</td>
<td>Prune($w_i, w_j, RF_i, RF_j$)</td>
</tr>
<tr>
<td>11:</td>
<td>End:</td>
</tr>
<tr>
<td>12:</td>
<td>End:</td>
</tr>
<tr>
<td>13:</td>
<td>Function Prune($w_i, w_j, RF_i, RF_j$)</td>
</tr>
<tr>
<td>14:</td>
<td>define: centroid$_i$ = $w_i$, centroid$_j$ = $w_j$, Set = $RF_i \cup RF_j$</td>
</tr>
<tr>
<td>15:</td>
<td>Run K-Means(2, centroid$_i$, centroid$_j$, Set) // see section 2.4.4.2. for K-Means</td>
</tr>
<tr>
<td>16:</td>
<td>return new BMUs and new RFs // separation and reassignment</td>
</tr>
</tbody>
</table>

Algorithm 5-2: Addressing coherent BMUs

After pruning these BMUs the final GHSOM model has the same number of BMUs but coherent BMUs are improved and their receptive fields are better separated so that interferences between them are avoided.

**Addressing BMUs with a misclassification weakness**

This weakness appears in real time when the IDS classifies the online traffic assuming that the final GHSOM grid has a BMU that was not well trained; this unit leads to misclassify the new connections which lead to increase the false alarm and degrade the performance. In this regard, we propose two solutions:

- Delete the BMU and its receptive field from the grid and retrain the GHSOM. Let $D_i := \{w_i, RF_i\}$ be the BMU and the corresponding RF, then retrain the GHSOM by considering $D_i$ is the input dataset. Certainly, after the retraining the final GHSOM will have $m-1$ BMUs.
- The worst case is to completely delete the BMU from the final model without retraining.
5.3.3. Enhancement 3: Embedded Conversion and Normalization

The GHSOM algorithm handles only numeric attributes. However, some features are string or boolean ones, which mean the GHSOM will not be able to handle these features.

If we define the conversion process in section 4.4.1 by “Converter”, algorithm 5-3 describes the steps of embedded conversion and normalization processes inside OptiFilter in real-time.

```
// assume windows every 5 sec in OptiFilter
1:   For ; ; // continuous loop
2:     Read $D_i$;  // receive the constructed connections in the current window
3:     conn = size($D_i$);  // number of connections in the current window
//_____Start Conversion_______//
4:     For $i = 1 : n$;  // n = number of features
5:         IF ($X_i$ is nominal)  // $X$ is represented a feature in D
6:             Run Converter($X_i$);
7:         Else
8:             Return $X_i$;
9:       End;
10:   End;
//_____Start Normalization_______//
11:   For $j = 1 : n$;  // n = number of features
12:     For $k = 1 : conn$
13:         $nv_{jk} := \frac{v - \min(X_i)}{\max(X_i) - \min(X_i)} \cdot (b - a) + a$
14:         $v_{jk} \leftarrow nv_{jk}$
15:       End;
16:   End;
17:   export ($\bar{D}_i$)    // $\bar{D}_i$ is a normalized dataset for the current window
18: End;
```

Algorithm 5-3: Embedded conversion and normalization algorithm

In this enhancement, we exploit the proposed preprocessing methods (see section 4.4.), i.e. conversion and normalization processes, and embed them in OptiFilter to export normalized datasets in a single window inside the queue. For instance, let’s assume that the dynamic queue in section 4.2.2 is configured to have 3 windows ($w_1$, $w_2$, $w_3$), each window is a 5 seconds time slot. OptiFilter exports these windows based on FIFO principle every 5 seconds without any further processing. In this enhancement, before exporting constructed
connections in the current window, each string feature (like protocol_type or flag) will firstly be converted using the proposed conversion in section 4.4.1 and secondly be normalized using the proposed min-max normalization method in section 4.4.2. Finally, numeric (normalized) connections in each window will be exported from OptiFilter and so forth.

Thus, the first step is to aggregate a sufficient dataset, the second one is to convert string values to numeric ones, and the third one is to normalize all features. This enhancement is very useful in the online operational mode of GHSOM, because it converts and normalizes every window directly inside the queue and presents a proper connection to the classifier.

5.3.4. Enhancement 4: Meaningful Map Initialization

Initialization process in GHSOM occurs at the beginning of training or after adding a new map as explained in section 5.2.2. Moreover, the weight vectors of the neurons on the new map are randomly initialized, which leads to project input vectors incorrectly on the grid and hence increase the error that weakens the final topology. In this regard, a new meaningful initialization process of weight vectors based on minimum and maximum boundaries of the input dataset has been proposed. To map input instances on the correct neuron during the training of GHSOM we initialize the first map after the root node by four neurons. Their weight vectors are initialized as the following: first and second vectors from the normal instances and third and fourth ones from the anomalous instances. Similarly, any new generated map will be also initialized by two neurons from the RF (i.e. the RF will deem here as a dataset for the new map) of the parent node, and trained by it as well.

Assume $X_M = (x_1, x_2, ..., x_n)$, $x_j = (x_{j_1}, x_{j_2}, ..., x_{j_n})$ are the M realizations of the $j^{th}$ attribute in the dataset D. Accordingly, the maximum and minimum boundaries of D, after conversion and normalization, can be determined as

$$D_{\text{min}} = \{\min(X_1), \min(X_2), ..., \min(X_n)\},$$
$$D_{\text{max}} = \{\max(X_1), \max(X_2), ..., \max(X_n)\},$$

and the mean will be then $D_{\text{average}} = \frac{1}{2} \cdot (D_{\text{min}} + D_{\text{max}})$. This enhancement exploits these boundaries to initialize the weight vectors meaningfully as

$$w_1 = 0.5(D_{\text{average}} + D_{\text{max}}) + 0.75D_{\text{max}} + 0.25D_{\text{min}},$$
$$w_2 = 0.5(D_{\text{average}} + D_{\text{min}}) + 0.75D_{\text{min}} + 0.25D_{\text{max}}.$$  

These weight vectors are simply cohesion to the dataset. Thus, the input connection will be fast and accurately projected to the BMUs. Figure 5-13
shows two initial weight vectors and the consistency between them and the datasets.

![Figure 5-13: Demonstration example of weight vector initialization](image)

In the previous figure, a small sample dataset with around 100 connections has been selected. Each connection vector consists of 17 features, which have normalized values between 0 and 1. If we assume this dataset is labeled as normal and it will be the input dataset to train a certain map on GHSOM, then the two neurons on the new map can be initialized by the weight vectors $w_1, w_2$ as shown on the figure. These two weight vectors are very cohesion to the input dataset as the figure shows, thus, the 100 connection vectors will be projected precisely on the closest neuron and decrease the quantization error as well.

Generally, there are two known methods to initialize the neurons on the grid, these are the random initialization process and selecting a random weight from the input dataset. The distance error between the input vectors and the initialize weight vectors can be considered the cohesion measure. That means, by calculating the distance error (like quantization error) we can find out how the initial weight vector is affecting the training process, i.e. the smaller the error the faster the training converges. Therefore, a comparison between our proposed initialization method with the previous two initialization methods has been depicted in figure 5-14.
5.3. Proposed Theoretical Modifications on GHSOM

The figure shows clearly the distance error between weight vectors initialized according to the proposed method in this thesis and other methods, which are random initialization weight vector and selecting arbitrary input from the train dataset as a weight vector. The distance error between input connections and the random initialized weight vector is very large, which means that the random initialization process will definitely degrade the training process by increasing the mqe of each map, extending the training time, weakening the topology, and building a weak model. On the other hand, initializing the weight vector from any arbitrary input vector from the train dataset seems more appropriate choice than the random one. In contrast, the proposed method in this thesis, i.e. meaningful initialization process, has achieved a very minimum distance error. That means, accelerating the training process, precise projection for the input vectors, reinforcing the topology, and better classification.

5.3.5. Enhancement 5: Topology Stabilization by Node Splitting

The topology of GHSOM is controlled by the horizontal and vertical growth based on two growth conditions. As long as the map mean quantization error (\( MQE_m \)) is greater than a certain threshold value, i.e. \( MQE_m \geq \tau \cdot mqe_p \), a new column or row must be added between the highest error node \( e \) and its dissimilar node (see algorithm 5-1, section 5.2.2) on the map. Once this condition is not fulfilled then a second condition is examined to check the
vertical growth, which determines the $mge$ of each node on the map, and then the node that fulfills the following condition $mge_i > \tau \cdot mge$, will be further examined on a new map in the next layer (child) and its $RF_i$ will be used to train the new map. However, investigating the heterogeneity of the RF of each node will provide better and higher resolution, stable growth, and robust hierarchical topology. This heterogeneity will be measured as a percentage of normal input vectors to the anomaly ones in the receptive field. If the percentage fulfills a certain threshold then it leads to split the corresponding node into two nodes with two receptive fields on the new layer. During the training, some nodes can have input vectors in the receptive field with different labels such as 20% of the vectors have normal class label and 80% have anomaly class label. Thus, expanding this node on only one map in the new layer will not improve the topology and most probably the new map will have the same problem; that is, the heterogeneity in the RF. Therefore, this enhancement presents a new proposed method that guarantees a fair expansion of the node in the topology, so that the final BMUs are labeled correctly and so the receptive fields. An example of a node $i$ with heterogeneity in its receptive field is illustrated in figure 5-15.

![Figure 5-15: Splitting technique of a heterogeneous node](image)

The figure above shows that layer $n$ has three nodes satisfied the vertical growth condition and hence they should be expanded on a new layer $n+1$. The receptive field of node A has only input vectors with label “anomaly” and node B the same as well but with label “normal”, but the third one has 60% normal input vectors and 40% anomaly. Therefore, nodes A and B can be directly extended to a new map on the new layer $n+1$. Whereas, node $i$ must be expanded on the new layer but onto two maps not only one, i.e. the receptive field of this node must be split according to the class label, that means 60% normal vectors will
be expanded onto one map and 40% anomaly vectors will be expanded onto another. This solution will guarantee a stable expansion and a robust topology. Moreover, this approach will avoid having a BM with mixed input vectors in the receptive field and that, in turn, accelerates the training process and improves the performance of the GHSOM model. Formally, the heterogeneity threshold $\zeta$ that discovers data heterogeneity is a percentage of the normal vectors to the anomaly ones in the RF of a node $i$ such as

$$\zeta_i := \frac{|\{x_{\text{Normal}} \mid x_{\text{Normal}} \in RF_i\}|}{|RF_i|}.$$  

(5-26)

The node is considered heterogeneous if the following condition is satisfied

$$0.45 < \zeta_i < 0.95.$$  

(5-27)

During several training for GHSOM and monitoring the label of input vectors in each RF using various datasets, it comes out that this range is the most plausible one to describe the heterogeneity of a RF. Figure 5-16 illustrates the trade-off between the split threshold and the accuracy of the IDS using a real-time data.

![Figure 5-16: Tradeoff curve between the splitting threshold and the accuracy](image)

The figure shows clearly that the best splitting threshold to achieve a high accuracy is ranging between 45 and 95. Hereby, the first value is the accuracy without any splitting technique, which has achieved the worst accuracy.
5.3.6. Enhancement 6: Normal Network Behavior Model

The detection method in any IDS primarily aims to recognize any deviation from a predefined NNB model. However, researchers are recently exploiting new methodologies to build a significant IDS model without referring to a predefined NNB model to detect anomaly. As mentioned before, the GHSOM model can only classify connections to normal or anomaly. Therefore, a new dynamic classification-confidence threshold has been proposed in section 5.3.1 to enhance the GHSOM by revealing unknown connections. In this enhancement, we define a NNB model based on the training of EGHSOM to further classify the unknown connections. In other words, when the EGHSOM delivers an unknown connection, the NNB model will try to investigate about its normality.

Defining a NNB model using EGHSOM

Due to EGHSOM proficiency in exploring supplemental details in huge amount of data, it has been exploited to build a model that represents the normal behavior of a computer network. The only condition to achieve this goal is to provide the EGHSOM by a clean dataset; that is, anomaly free. In this thesis, clean datasets have been aggregated from the KDD repository, synthetic datasets from our simulated network (see chapter 6) and a real dataset from companies. This mixture of datasets ensures building a reasonable normal model of the corresponding network. Note that, the EGHSOM model principally classifies the incoming connections in real-time to normal, anomaly or unknown connections. Hence, normal connections from EGHSOM are also used to build a NNB which is very accommodated with the concerned network.

If we train the EGHSOM with a clean dataset as mentioned before, then the final model will be \( C, w_c, RF_c, "Normal" \) as described in 5.2. Generally, if a BMU \( c_i \) in the model has attracted a certain number of input vectors, i.e. \( |RF_c| \), together they form a cluster representation such that the BMU is the centroid and the RF is the neighborhood. Accordingly, the maximum radius of each cluster can be determined as

\[ R_i = \max\{d(x, w_i) \mid x \in RF_c\}, i = 1, 2, ..., m. \]  

\[ (5-28) \]

The radiuses for all BMUs can be represented as shown in figure 5-17.
Based on the clustering and the radiuses, the NNB can be determined as

\[(C, w_C, R_C), R_C = \{R_1, ..., R_m\}\]

Intuitively, all clusters are labeled as normal.

**Labeling Unknown Connection**

The intent of defining a NNB model is to support the EGHSOM by further investigating the unknown connections and the ability to classify them. During the online detection of EGHSOM in real-time, there should be few number of connections that are classified as unknown based on the classification-confidence threshold (see 5.3.1.). In this thesis, the defined NNB model will be used to examine if these unknown connections belong to a normal network activity or they are suspicious connections. This target can be achieved by measuring the distance between the unknown connection vector and each cluster in the NNB model. If the unknown connection falls into the vicinity of one cluster then it can be classified as normal connection, otherwise it is unknown-anomaly. Algorithm 5-4 describes clearly the main steps of using the NNB model to classify the unknown connections.
5.3. Proposed Theoretical Modifications on GHSOM

### Algorithm 5-4: Labeling unknown connections using NNB model

```plaintext
// preparing NNB Model (\(G, w_C, R_C\))
Define Normality=Normal.csv // create an empty file
1: For \(i\); // continuous loop
2: Read unknown connection \(\hat{x}(t)\);
3: While \(i \leq m\) // m number of BMUs
4: \(\hat{d} = \|\hat{x}(t) - w_i\|\) the distance to each BMU
5: IF \(\hat{d} \leq R_i\)
6: \(\hat{x}(t) \rightarrow\) Normal _connection
7: Write \(\hat{x}(t)\) to Normality;
8: Return \(\hat{x}(t)\); Break;
9: Else \(i++\);
10: End;
11: End;
12: Return \(\hat{x}(t) \rightarrow Unknown - Anomaly\);
13: End;
```

The previous algorithm is very effective by uncovering unknown connections whether they are normal or not. Once the connection could not be recognized as normal, the NNB labels it as unknown-anomaly. This label describes the expected status of a connection, i.e. unknown means that the connection is still obfuscated and anomaly means that it is most probably suspicious because the EGHSOM and the NNB could not detect its normality. Consequently, all detected normal connections from EGHSOM and NNB will be gathered in a separate file, which will be utilized to update the NNB model to stay adaptive in real-time.

### 5.3.7. Enhancement 7: Updating EGHSOM and NNB Models in Real-time

In the real-time, EGHSOM and NNB models must be adaptive, which imply that they need to be continuously updated. In this enhancement, update procedure of each model is presented individually. EGHSOM update procedure uses all uncovered anomaly connections to update the current online EGHSOM model. On the other hand, NNB update procedure uses all uncovered normal connections to update the current NNB model. It is worth mentioning that, the EGHSOM or the NNB read the connection and then deliver the same connection with extra features, which indicate its status. In other word, the connection vector is getting longer in length.

If we assume \(\hat{x} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n) \in \mathbb{R}^n\) is a new incoming connection, the EGHSOM will classify and deliver it as \(\hat{x} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n, label, BMU) \in \mathbb{R}^{n+2}\), where label is given as numeric value from EGHSOM (0=normal, 1=anomaly, and -1=...
unknown) and BMU is the BMU that has detected this connection. In this thesis, the normal labels have been replaced by the value 0, the anomaly labels by the value 1, and the unknown by the value -1 (i.e. they are real numbers). Accordingly, EGHSOM or NNB writes the detected connections $\tilde{z} \in \mathbb{R}^n$ as normal in a separate file (or database) called “Normal.csv”, the anomaly in “Anomaly.csv”, and the unknown in “unknown.csv”. During the online classification, these files will be used to update the EGHSOM and NNB models at once better than updating these Models after each single connection. Updating the EGHSOM or NNB models after each detected connection directly will harm the network and cause an overhead, especially in large scale and heterogeneous networks. Moreover, these files provide more details about each BMU during the online detection process. Figure 5-18 shows the position of each update mode in the proposed IDS framework during the online process.

As mentioned before, OptiFilter constructs normalized connections and exports them directly via a pipe concept to the EGHSOM, which in turn classifies them and sends the unknown connections to the NNB for further classification. All detected connections will be saved in separate files, which are used by the update model to update the EGHSOM and NNB constantly.

**Update Model of EGHSOM**

The update procedure of EGHSOM considers only the detected anomaly connections to keep the model adaptive. On the other hand, detected unknown connections will not be considered in the update procedure, although they could be anomaly, because they will be forwarded to the NNB for further examination as described before.

Meanwhile, if we update the final EGHSOM model by retraining it directly with each detected anomaly, then the topology will keep growing and tends to be instable with a weak form, which indeed affects critically on the EGHSOM model during the online detection. Therefore, the update procedure should be different than the training.

Basically, online EGHSOM model reveals anomaly connections and then saves the BMU that has classified it in a separate file named *anomaly.csv*. The update
procedure reads this file after certain time (configured manually) and gradually updates the corresponding BMUs in the model using the same cooperative process. Particularly, the update procedure reads the anomaly.csv file (or database) and measures the selection percentage $\eta$ of each BMU, where $\eta$ determines which BMU was active in the online detection process. The use of selection percentage is necessary to expand the topology of the EGHSOM during the online detection. That means the growth of the original EGHSOM model will be kept in a plausible extending and, on the other hand, it fulfills the idea of exploring supplemental details in real-time. Let $Q := \{bmui, bmui, ..., bmui\} \subseteq C$ represent all active BMUs in the online classification process, which are saved in the anomaly.csv file, where $Q \subset C$ (C from table 5-1). Thus, the selection percentage of each BMU can be defined as

$$
\eta_i := \frac{| \{ (x, 1, c) \in \text{anomaly.csv} \mid c = bmui, bmui \in Q \} |}{| \text{anomaly.csv} |}, i = 1, 2, ..., P \tag{5-30}
$$

Where $x$ is the detected connection and $P$ is the total number of individual BMUs in the anomaly.csv file. The value of percentage selection indicates the activity of the BMU in the online detection. This is to imply that, if one BMU has classified more than 95% of the total incoming connections as anomaly then it is considered as a very attracted one and should be divided into two BMUs to improve the topology of the EGHSOM and keep it adaptive. Hence, to split a BMU $i$ into two BMUs, the following condition should firstly be fulfilled

$$
\eta_i \geq 0.95 \tag{5-31}
$$

This condition is manually configured and can be changed upon the classification result. Assume that $bmui$ has satisfied the condition in 5-31, then this BMU should be further analyzed by dividing it into two BMUs $\{bmui, bmui\}$. The weight vectors of each new BMU will be initialized using the enhancement in 5.3.4 as follows. Let the detected connections by $bmui$ represent its RF, then the set $\{bmui, RF\}$ can be considered as the input dataset to initialize the weight vectors of both neurons. After initializing the weight vectors, all corresponding anomaly connections will be reassigned on both new BMUs by the minimal distance measure, which leads to

$$
\{bmui, RF\} \rightarrow \{bmui, RF\}, \{bmui, RF\}. \tag{5-32}
$$
Thus, if the number of individual BMUs in the anomaly.csv file is \(T\) then after performing the splitting in 5-32 the number of BMUs will be \(T+1\), each with its RF.

The online EGHSOM uses the threshold margin \(\delta : [d_{\min}, d_{\max}]\) (see equation 5-21) to classify input connections as described in section 5.3.1, but this margin has been abstracted from the final model after training. Particularly, after performing the splitting of the most attracted BMU, the classification-confidence margin should be also adapted. In equation 5-30, we have defined \(Q\) as the subset of the individual BMUs in the anomaly.csv file. Accordingly, let \(\delta_q\) be the threshold margin for the BMUs in the anomaly file such as \(\delta_q : [d_{\min}^q, d_{\max}^q]\).

Therefore, to adapt the margin we examine the following conditions:

\[
\begin{align*}
d_{\min}' &= \begin{cases} d_{\min}^q, & d_{\min}^q \leq d_{\min} \\ d_{\min}, & \text{else} \end{cases} \quad (5-33) \\
d_{\max}' &= \begin{cases} d_{\max}^q, & d_{\max}^q \geq d_{\max} \\ d_{\max}, & \text{else} \end{cases} \quad (5-34)
\end{align*}
\]

Which results then by a new margin threshold

\[
\delta' := [d_{\min}', d_{\max}'] \quad (5-35)
\]

The last step in the update procedure is updating the BMUs in \(Q\); that is, only the weight vectors of the active BMUs in the online classification process will get adapted and other BMUs will stay unchanged. Each BMU will be updated using its corresponding RF (the detected connections by this BMU) by the following rule

\[
w_i(t + 1) := w_i(t) + \alpha(t) \cdot [\hat{z}(t) - w_i(t)], i = 1, \ldots, T + 1 \quad (5-36)
\]

where \(\alpha(t)\) is a decreasing learning rate function to guarantee that the BMU moves in direction the input vector. Note that the coordinates of each BMU in the final topology are no more valid in the online mode, which explains why the Gaussian function \(h_{ci}(t)\) is not used. The exponential function is the effective function that moves the BMU toward the input vector. The decreasing function that is used in this thesis is...
\[ \alpha(t) = \alpha_0 \cdot \exp \left( -\frac{t}{|RF|} \right) \]  

where, \( t \) is the number of iteration, \( \alpha_0 \) is a constant vale such that \( 0 < \alpha_0 \leq 1 \) and \( |RF| \) is the total number of input connections in the receptive field. For example, if \( bmu_a \) has detected 30 input connections in the online classification then the total number of input connections is 30 and the update rule in 5-36 will be performed for each input connection in the \( RF \).

Finally, the update procedure will replace the current online EGHSOM detection model by the adapted one such as

\[ (C', w_{c'}, Labels, \delta') \leftarrow (C, w_c, Labels, \delta) \]  

In addition, further statistic information is also gathered during the online classification for each BMU such as time of detection or the amount of misclassification. This information is used to handle the inactive BMUs or to remove them from the model.

In summary, the update procedure of the EGHSOM can be summarized in algorithm 5-5. In this algorithm, the update takes place between 30 to 60 minutes after starting the online classification and if within this period the number of detected anomaly has reached 100 connections.

---

**Start Online Classification**;

1. Timer = current(time); // save the current time of starting the classification
2. Define anomaly.csv = detected_anomaly;
3. If \( 30 < \text{minute(time - Timer)} < 60 \) & |anomaly.csv| > 100
4.  \( Q = \text{detected_anomaly.bmus}; \) // active BMUs
5.  \( \text{For } i = 1 : |Q| \)
6.     calculate \( \eta_i \) //equation 5-30
7.  If \( \eta_i \geq 0.95 \)
8.     \( \{bmu_i, RF\} \rightarrow \{bmu_{i'}, RF_{i'\}}, \{bmu_{i''}, RF_{i''}\} \) // divide the BMU
9.  \( \text{End}; \)
10. \( \text{End}; \)
11. Load \( \delta : [d_{\text{min}}, d_{\text{max}}]; \)
12. Calculate \( \delta_{\text{Q}} : [d'_{\text{min}}, d'_{\text{max}}]; \)
13. If \( \hat{d}_{\text{min}} \leq d_{\text{max}} \)
14.     \( d_{\text{min}} \leftarrow d'_{\text{min}} \)
15.  \( \text{End}; \)
16. If \( \hat{d}_{\text{max}} \geq d_{\text{max}} \)
5.3. Proposed Theoretical Modifications on GHSOM

Hence, the EGHSOM topology keeps changing during the classification in real-time by generating new units or removing inactive ones as the following example in figure 5-19 shows.

This figure shows a final EGHSOM model that consists of 9 BMUs at the beginning. The model has been loaded at t=0 to start the online classification in real-time, therefore, all BMUs are still inactive. After 10 min of operating, 5 BMUs were active (4 blue and one red) and 4 were still inactive, the red one has satisfied the split condition in 5-31. Therefore, at t=20 the red neuron has been divided into two neurons and one more neuron becomes active, i.e. the model has now 10 BMUs 8 of them are active and one is satisfying the split condition. At t=30, the two inactive BMU have been removed from the model because they are still inactive; (i.e. the model contains now 9 active BMUs), and so on.

**Update Model of NNB**

The update procedure of the NNB performs different steps than the EGHSOM. It considers only detected normal connections from the online EGHSOM model and the online NNB model. The normality of the network is changing constantly due to the fact that data stream are non-stationary data as well as they are affected by the concept drift [49]. Therefore, the NNB model should be updated continuously. All detected normal connections will be used to update the state...
of each BMU in the NNB model \((C, w_C, R_C)\) using the K-means clustering algorithm. The K-means is examined in the updating of NNB because all BMUs in the model have the same label normal so that any detected normal connection in the real-time can be used to update the entire model and the radiuses as well. That means BMUs that were not active in the online detection could be modified during the update procedure, which in turn enhance the NNB model to be more adaptive and homogeneous with the new normal state of the network.

Let \(\mathcal{C}\) the be the set of all BMUs such that \(m = |\mathcal{C}|\) as shown in figure 5-17 and the detected normal connections be the input dataset \(X_{\text{Normal}}\). Hence, for each cluster the centroid is considered the weight vector of the corresponding BMU. Accordingly, to update the NNB model we perform the K-means algorithm as

\[
K - \text{Means}(m, \text{centroids}, X_{\text{Normal}}, \lambda)
\]

where \(\lambda\) is the number of iterations.

After the algorithm is converged, the radius of each cluster is determined as described in equation 5-28. Finally, the update procedure will replace the current online NNB model by the adapted one such as

\[
(C', w_{C'}, R_{C'}) \leftarrow (C, w_C, R_C)
\]

Hence, the update procedure of the NNB model can be summarized in algorithm 5-6. Note that, almost the same conditions in EGHSOM have been also used by the updating of NNB model.

---

**Algorithm 5-6: Major steps of updating the NNB model**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Timer = current(time); // save the current time of starting the classification</td>
</tr>
<tr>
<td>2.</td>
<td>Define normal.csv = detected_normal;</td>
</tr>
<tr>
<td>3.</td>
<td>IF (30 &lt; \text{minute}(\text{time} - \text{Timer}) &lt; 60) &amp;&amp; (</td>
</tr>
<tr>
<td>4.</td>
<td>(m \equiv</td>
</tr>
<tr>
<td>5.</td>
<td>centroids = (w_C); initialize (\lambda)</td>
</tr>
<tr>
<td>6.</td>
<td>(X_{\text{Normal}} := {\text{normal.csv}});</td>
</tr>
<tr>
<td>7.</td>
<td>(K - \text{Means}(m, \text{centroids}, X_{\text{Normal}}, \lambda));</td>
</tr>
<tr>
<td>8.</td>
<td>For (j = 1 :</td>
</tr>
<tr>
<td>9.</td>
<td>(R_j := \max{d(x, w_j) \mid x \in R_E}, j = 1, 2, \ldots, m)</td>
</tr>
<tr>
<td>10.</td>
<td>End;</td>
</tr>
<tr>
<td>11.</td>
<td>End;</td>
</tr>
<tr>
<td>12.</td>
<td>Upload the new Model ((C', w_{C'}, R_{C'}) \leftarrow (C, w_C, R_C))</td>
</tr>
</tbody>
</table>
In this thesis, the steps of updating the NNB are more efficient than the steps of updating the EGHSOM due to the following reasons:
- The normal behavior of any network changes continuously and within short time, therefore, updating the NNB must be very fast and effective to handle this change.
- The amount of detected anomaly is very small in comparison to the number of detected normal connections. Thus, to process the normal connections professionally the update procedure of NNB must be optimized and efficient to handle the huge amount of data.
- Finally, EGHSOM has several parameters than the NNB, which should be adapted during the classification. Therefore, the time required to update all of these parameter is larger than the time required updating the NNB model.

5.4. PROPOSED PREVENTION METHOD

Actually, this thesis is not handling the intrusion prevention system. However, this section explains a proposed alarm method that shows proper information content of the detected anomaly. Moreover, it presents a suggested alarm categorization method that assigns a status for each detected anomaly.

5.4.2. Alarm Information Content

The standard reaction when detecting an anomaly is raising a message that describes this anomaly. The proposed IDS framework can generate a sufficient alarm content from the IP-header information, other advanced features, and from some host events. Therefore, the alarm information content can be presented as a message that contains:
- Upper section of the message. It assigns an ID and a category to the anomaly. They indicate to the status of the anomaly (critical or warning).
- The next section shows the source and destination IP addresses together with the type of connection such as TCP, UDP, ICMP or other protocols.
- The lower section shows the source and destination ports, as well as the name of the service and the transferred source byte.
- The messages can accommodate more information about the connection vector but this information is omitted for the future use.

Figure 5-20 demonstrates a sample of alarm content.
5.4. Proposed Prevention Method

Figure 5-20: Example of alarm information content

5.4.3. Proposed Alarm Categories

This section proposes a categorization technique that identifies the current status of the detected anomaly. These categories are divided into:

- **Category 1: Severity of the anomaly.** This categorization decides how dangerous the detected anomaly is. The anomaly becomes a new class label based on its severity, such as {policy violation, exploit, error, attack}. Note that some classes such as Exploit needs more details from the packet payload.

- **Category 2: Level of the anomaly.** When the anomaly connection becomes a class label, this category assigns a status to the connection by determining the level of the anomaly class label. Proposed levels are: {Info, Warning, and Critical}. For instance, the anomaly on the figure 5-20 has the level “Info”, which can be recognized from the icon in the identification tab.

- **Category 3: Anomaly groups.** This category shows which part of the network has been often attacked from certain anomaly. This task can be achieved by an intelligent script that filters such information from the detected anomaly. These categories can be automated by an agent or manually by an analyst. As a result, any anomaly connection will be further analyzed upon its severity and distribution.
5.5. **ONLINE IDS FRAMEWORK**

This section explains the proposed IDS framework by combining components together. Moreover, the section will clarify the adaption process of the framework while it operates in the online mode.

5.5.1. **Overview Online Framework**

The online IDS framework consists of two parts. The first part is OptiFilter and the second part is the adaptive classification. OptiFilter has the following tasks:
- Capturing data flow from the corresponding network (data aggregation).
- Preprocessing these flows by generating continuous datasets, converting nominal features to numeric ones and normalizing them to the same scale.
- Exporting normalized connection vectors continuously.

The adaptive classification method performs the following:
- Detecting anomaly and unknown connections using the EGHSOM model.
- Further classifying the unknown connections by a NNB model.
- Updating the framework constantly by adapting the EGHSOM and NNB models.
- Showing a proper alarm content

The general framework architecture can be shown in figure 5-21.

![Figure 5-21: General architecture of the IDS framework](image)

The previous architecture shows the main components of the proposed framework, where the alarm is considered as a component from the adaptive classification. The internal component diagram of the proposed IDS framework is depicted in figure 5-22.
In the framework, there are some components that operate only for one time in the offline operational mode. These components are mainly the training processes, which are colored in gray on the previous figure. Initially, before the framework operates in the online mode, the EGHSOM and NNB models must be trained by a sufficient amount of connection vectors. When the training process finishes, it generates an EGHSOM and an NNB models, which are ready to operate in real-time. These models stay in the standby mode as long as the framework is not operating. Once the framework starts operating in the online mode, both models will be activated and loaded into the online components (online EGHSOM and online NNB).

The online operational mode can be summarized into the following steps:

1. The aggregators (which are part of OptiFilter) gather network packets and hosts’ events continuously in a form of tcpdump data and SNMP traps.
2. OptiFilter processes and analyzes these data in the queue, constructs normalized connection vectors, and sends these vectors sequentially via the pipe to the online EGHSOM model. At the same time, it saves these connections (identified by unique ID) in a database with their original values (i.e. string, numeric and nominal values).
3. The EGHSOM reads the incoming connections from OptiFilter from the same pipe and classifies each connection to anomaly, normal or unknown. It sends the classified connections directly into another pipe to the controller.
4. The controller receives these connections from the same pipe and checks the label. If the label is anomaly, it sends a copy to the alarm component and saves the connection in a separate file (or database) namely “anomaly.csv”. But if the label is normal, the controller sends this connection to the NNB model to get assigned by a new BMU, and then saves it in a separate file (or database) namely “normal.csv”. The last case, if the label is unknown, then the controller sends the connection to the NNB for further classification.
5. The NNB model receives connection vectors from the controller to assign each connection by a new BMU (if the connection is normal, see step 4) or to classify the unknown connection as normal or unknown-anomaly. Then it sends the result back to the controller.

6. Alarm component receives only anomaly or unknown-anomaly connections from the controller. It checks the ID of the connection and fetches the original connection values from the database (see step 2) to raise appropriate alarm content.

7. According to predefined condition, the update models are activated. For example, if the framework updates the EGHSOM and NNB models every 20 minutes, the update models of EGHSOM and NNB will be enabled after 20 minutes, 40 minutes, and so forth.

8. Update EGHSOM component reads the anomyl.csv file and performs the update algorithm (see algorithm 5-5). When the update finishes, a new EGHSOM model is generated and replaced by the current online EGHSOM model. In case of updating the NNB models, the update component reads the normal.csv file, performs the algorithm 5-6, generates a new NNB model, and then replaces it with the current online NNB model.

As a result, the framework adapts its models to the current network traffic to be self-adaptive in real-time. In the following two subsections, the internal structure of each part of the framework will be explained intensively.

5.5.2. Part 1: OptiFilter

Based on figure 5-22, this part includes the aggregator and OptiFilter. This part of the framework handles network traffic to construct normalized connection vectors in the online operational mode. Particularly, the entire architecture of the first part includes the aggregator, connection analysis, conversion, normalization, and exporter. The aggregator consists of a tcpdump sniffer that captures the network traffic from a corresponding network element such as a switch or router, and a SNMP trap that sends traps from hosts and processes them at the server side. All SNMP traps are sent via secure TCP connection to the server. On the other hand, the connections analyzer (dynamic queuing concept) analyzes network packets and hosts’ events and correlates them together to construct a connection vector. The queuing concept is clearly declared in 4.2.2. A conversion method examines the constructed connections on each window to convert nominal values into numeric ones. The Normalizer scales all features in each window to the same rang (i.e. 0 and 1). The last step is to export these normalized connections. Basically, all constructed connections will be exported in a CSV format. The CSV will be gathered accumulatively to generate a single dataset or it will be sent directly via a pipe for further processing. Exporting a single dataset has the benefit of using it for different
purposes such as evaluating IDS models in offline mode. However, to classify the data flow in the online mode, the pipe concept is very appropriate and also preferable from science and industry. Figure 5-23 demonstrates the internal methods or the functionality of OptiFilter during the online operational mode in the real-time.

The novel method in this part is the dynamic queuing concept that is able to handle massive data flow within sufficient time (more details are available in 4.2. and 4.4.).

5.5.3. Part 2: Online Adaptive Classification

The second part of the proposed IDS framework is the adaptive classification. The internal components of this part can be concluded by the online EGHSOM model, the online NNB model, both updates models for NNB and EGHSOM, the controller, and the alarm. These components have been explained clearly in the previous sections. Figure 5-24 shows the internal functionality of the second part architecture.
Online EGHSOM Model

The internal functionality of the online EGHSOM in the proposed IDS framework is illustrated in figure 5-25.

The online EGHSOM model has internally a receiver that reads connection vectors either from a pipe or from a file. The distance between the incoming connection and the BMUs in the model will be calculated. Accordingly, the EGHSOM makes a decision about which label should the connection become, sends this connection to the controller, and writes in the statistic file a new
instance. The EGHSOM can be configured to update the threshold margin after investigating each connection, but in this thesis the update of classification-confidence threshold margin takes place in the update model of EGHSOM.

**Online NNB Model**

Similarly, internal functionality of the online NNB model can be shown in figure 5-26.

![Diagram of NNB model](image)

**Figure 5-26: Internal functionality of NNB model in real-time**

The online NNB model, as mentioned before, is a support component to the EGHSOM. It monitors the normal status of the network and examines the normality of unknown connections. The NNB model receives, constantly, from the controller unknown or normal connections. It assigns a new BMU for each normal connection and sends it back to the controller. It checks the normality status of each unknown connection and similarly sends the result back to the controller.

**Online Controller**

This component performs as a management component. It gathers the classified connections from the online EGHSOM and the online NNB and manages them systematically. Figure 5-27 shows the internal functionality of the controller.
The controller separates the output of EGHSOM from the output of NNB. It takes the EGHSOM output and checks if its anomaly, normal or unknown. In case of anomaly connection, it saves a copy in the anomaly.csv file, reads the original connection vector from the database (generated from OptiFilter) and sends it to the alarm component. But if the connection is normal, it sends it to the NNB model, which assigns a new BMU to this connection (necessary for the update) and returns it back to the controller. The controller saves this connection to the normal.csv file. If the connection is unknown, the controller sends it to the NNB for further classification. If the NNB classifies it as normal, the controller saves it in normal.csv file. But if the NNB could not classify it, the controller sends the connection to the alarm and to an analyst, who examines this unknown connection intensively.

**Update EGHSOM Model**

One of the novel enhancements of the proposed IDS functionality is the Adaptivity during the classification in real-time. Unlike signature-based IDS’s, which update only their databases by new signatures, the proposed IDS framework adapts the EGHSOM and the NNB models and other parameters, such as the classification-confidence threshold to be able to detect new attacks.

The update EGHSOM component performs an algorithm to keep the EGHSOM continuously adapted during real-time classification. The internal functionality of the EGHSOM update model is illustrated in figure 5-28. The functionality in the online mode affects each individual BMU in the model. It monitors the attraction value (see 5.2.1) of each BMU during the online mode and addresses them or removes them from the entire topology.
All detected anomaly from the EGHSOM will be saved in a separated file called anomaly.csv. The update model of EGHSOM reads this file and retrains the current EGHSOM model to adapt it with the new detected anomaly. Hence, any detected attack will be directly recognized once it appears again.

**Update NNB Model**

On the other hand, the NNB model represents the normal behavior of the computer network. Therefore, adapting the current NNB by the new classified normal connections is a very important step. In the update model of NNB, the normal.csv file will be exploited to handle all BMUs in the current model and define new radiuses for the updated BMUs. The update model has the following functionality as shown in figure 5-29.
5.5. Online IDS Framework

Figure 5-29: Internal functionality of the update NNB model in real-time

Online Alarm component

The alarm is the last component in the proposed online IDS framework. It receives the anomaly connections and unknown-anomaly and shows them in a proper alarm message as shown in 5.4.2. In addition, the unknown-anomaly, which has been sent from the controller to an analyst, can be shown in an extra message on the alarm component. Figure 5-30 shows the functionality of the alarm.

Figure 5-30: Internal functionality of the alarm in real-time
Chapter 6. Experimental Study

6.1. Preparation for Evaluation

In the performance evaluation, the proposed IDS framework has been evaluated in the offline and the online operational modes. One of the key aspects in the performance evaluation is the scalability of the first part of the framework. On the other hand, the accuracy and reliability of the second part of the framework are also other important key aspects. Moreover, there are several selected performance metrics to evaluate the framework such as the processing time inside the queue, the overall accuracy, the false alarm, etc.

The result of the proposed hybrid feature selection method (section 4.3) is very essential for generating the datasets in OptiFilter. Therefore, results of the proposed feature selection method will be considered as a preparation for the performance evaluation of the framework.

6.1.1. Resources and Metrics

It is very necessary to explain all sources, which are used in this thesis to evaluate the proposed IDS framework. These sources have been obtained from available offline data flow, synthetic data flow from a simulated network at the university, and data flow from a real-time company. Thus, used data flow in this thesis can be classified as offline, synthetic, and realistic. Moreover, this section will clarify the performance metrics which have been used to evaluate the first and second parts of the proposed IDS framework.

Offline Available Data flow

There are various data flows available for evaluating the proposed IDS framework. But in this thesis, the most used and well-known data flows have been selected. These data flows, mostly, are dump network traffic from certain computer and communication networks. They will be used to evaluate OptiFilter and then the adaptive classifier. The first source is the NSL-KDD, which is
originally derived from the DARPA 1998/1999 dump network traffic (see 4.2.1). The second source that is the Defcon iCTF (international Capture the Flag) dump network traffic.

**Synthetic Data flow**

In addition to the available offline data, the proposed framework has been evaluated by synthetic data flows which are generated in an isolated simulated network. We have installed a test network at the university campus which simulates internal and external traffic on virtual machines. At the internal network, five GNU/Linux server systems and two Microsoft Windows Domain Controllers were installed. They build the basis of an active directory infrastructure for a total of 10 Microsoft Windows client computers running Windows 7 and Windows XP. Additionally, two Microsoft Windows servers running IIS as well as a Microsoft Exchange server have been also installed. At the external network, there are five virtual machines, each providing a dedicated service (i.e. HTTP, DNS, SMTP). The virtual machines got assigned 1024 IP addresses each, ranging over the whole IPv4 address space. This allows simulating the connections to the Internet with a wide range of different IP addresses. In this test network, the applications Metasploit, Nexpose, OpenVAS, and other attack scenarios like ping around or DoS to generate new uncovered attacks (or anomaly traffic) have been used. Network data are captured using tcpdump at the virtual bridge interface of the internal physical server. For the test scenarios, selected 16 most common services which should be present in the network traffic dump have been selected: ftp, ssh, telnet, smtp, smb, nfs, xmpp, http, ntp, dhcp, syslog, snmp, rdp, IMAP, pop3, rsync.

Datasets, which are generated from the synthetic data flow, have given the name “SecMonet”.

**Realistic Data flow**

The last source for evaluating the IDS framework is realistic data flow from a real-time company. In short, the company’s head quarter is in Fulda, where the realistic network traffic has been aggregated. Moreover, all networks in this company are performing either 1 GB or 10 GB Ethernet connections. Datasets, which are generated from the realistic data flow, are called “RealSet”.

**Performance Metrics**

The performance metrics indicate whether the framework is optimal for anomaly detection. They were chosen to evaluate each part of the proposed framework individually.

The performance metrics of the first part OptiFilter:
- Number of windows. This metric indicates the total number of windows inside the dynamic queue. It examines if the total time of aggregation is consistent with the total number of windows inside the queue.
6.1. Preparation for Evaluation

- Number of processed packets. This metric presents the total number of processed packets per window. It indicates on the capacity of a single window especially by a massive data flow.
- Number of connections. Similarly, this metric shows the number of generated connections per window.
- Number of packets drop. Indicator to the total number of packet drops per window.
- Processing time. As mentioned in section 4.2, each window inside the dynamic queue has the length \( t \), i.e. 5 seconds. Therefore, this metric examines if the total time, from reading packets to generating connections, has required more than the length of the window. Indeed, this metric is significant, because it inspects if the processing time stays less than the window length throughout the online operational mode.

In contrast, metrics of the adaptive classification are mainly focusing on the preciseness and accuracy of the classifier. Thus, the performance metrics here are divided into two types: metrics for the offline mode and others for the online mode.

For the offline mode, the Confusion Matrix (also known as contingency table) has been used. It is an N×N matrix that has been firstly defined by Kohavi and Provost 1998 [194]. It shows the information about the actual and predicted classifications. Table 6-1 illustrates a 2×2 confusion matrix.

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Predicted Class</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Anomaly</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>Anomaly</td>
<td>TP</td>
<td>FN</td>
<td>TP+FN (actual positive)</td>
</tr>
<tr>
<td>Normal</td>
<td>FP</td>
<td>TN</td>
<td>FP+TN (actual negative)</td>
</tr>
<tr>
<td>Total</td>
<td>TP+FP</td>
<td>FN+TN</td>
<td>TP+FN+FP+TN (Total)</td>
</tr>
</tbody>
</table>

Hereby, the abbreviations of the confusion matrix:

- **TP** – *True Positive*: also known as number of hits. Number of anomaly connections which have been detected as anomaly from the classifier.
- **FN** – *False Negative*: also known as type II error. Number of anomaly connections which have been detected as normal from the classifier.
- **FP** – *False Positive*: also known as false alarm or type I error. Number of normal connections which have been detected as anomaly from the classifier.
- **TN** – *True Negative*: also known as correct rejections. Number of normal connections which have been detected as normal from the classifier.

Accordingly, the performance metrics which are derived from the confusion matrix can be summarized as follows.
6.1. Preparation for Evaluation

- True Positive Rate (TPR): also known as Detection Rate (DR), Sensitivity or Recall. The TPR can be defined as how the model is precise by detecting anomaly from the total anomaly connections (actual positive). It can be derived as

\[
TPR = \frac{TP}{TP + FN}
\]

- False Positive Rate (FPR): also known as False Alarm Rate (FAR). The FPR can be defined as the flaw percentage while classification connections. It is the detecting of normal connections as anomaly from the total normal connections (actual negative). It can be derived as

\[
FPR = \frac{FP}{TN + FP}
\]

- Accuracy: it is the percentage of correctly classified connections to the total number of connections (total). The accuracy can be computed as

\[
ACC = \frac{TN + TP}{TP + FN + TN + FP}
\]

Note that the confusion matrix can be used to derive other metrics such as the specificity or precision, but the most common performance metrics are the detection rate together with the false positive rate and the accuracy, as well.

In the online operational model, the following performance metrics have been selected to evaluate the framework.

- Accuracy, TPR, and FPR.
- Activity. This metric measures the activity of each BMU during the online classification. It is very important for the update procedure. Number of active BMUs during the real-time indicates also on the model efficiency.
- Attractivity. It is a complementary metric to the activity. However, this metric shows the attraction percentage of each BMU throughout the online classification.
- Distance of the incoming connection to the classification-confidence margin threshold. This metric is considered as a verification parameter, which means it examines if all detected connections are fall into the classified threshold margin.
6.1.2. Feature Selection Results

In this section, the result of the proposed SBS and IG methods from the feature selection will be illustrated. These features will be used in data aggregation and datasets generation in OptiFilter. Therefore, the result is considered as a preparation for the evaluation.

6.1.2.1. Results of SBS

These components have been prepared to examine the SBS algorithm 4-1:
- Features set (appendix B).
- Dataset: NSL-KDD with 130000 instances (105000 normal and 25000 anomalies).
- Classifiers: Neural Network – Multilayer Perceptron (NN), Naïve Bayes (NB), Decision Tree (J48), and Random Tree (RT).
- Training: 75% of the dataset.
- Testing: 25% of the dataset.
- Metrics: TPR and FPR.

The following scenario has been used to evaluate each classifier:
1. Evaluate the first classifier with a full dataset and record the TPR and FPR.
2. Remove one feature from the dataset and the corresponding column and repeat step 1.
3. Perform step 2 for all features.
4. Repeat the above steps for all classifiers.
5. Use TPR and FPR form all classifiers to choose the features that positively affect the performance metrics.

Based on the preparation steps, each classifier will have 42 TPR values and the same for the FPR. To examine which feature has positively affected the metrics, a selection margin has been determined from the TPR and FRP values as follows. Let $\mu_{NN}$ be the mean value of TPR for the NN classifier and $\text{Std}_{NN}$ be the standard deviation of the same classifier, then we can define a selection margin as $[\mu_{NN} - \text{Std}_{NN}, \mu_{NN} + \text{Std}_{NN}]$ that explores if the TPR or FPR after removing certain feature have been positively affected or not. Table 6-2 shows the result of TPR and FPR of the NN classifier and the margin, as well.

<table>
<thead>
<tr>
<th>Removed Features</th>
<th>TPR</th>
<th>FPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full dataset</td>
<td>0.9750</td>
<td>0.0270</td>
</tr>
<tr>
<td>duration</td>
<td>0.9750</td>
<td>0.0270</td>
</tr>
<tr>
<td>protocol_type</td>
<td>0.9720</td>
<td>0.0290</td>
</tr>
<tr>
<td>service</td>
<td>0.9720</td>
<td>0.0300</td>
</tr>
<tr>
<td>flag</td>
<td>0.9730</td>
<td>0.0280</td>
</tr>
<tr>
<td>src_byte</td>
<td>0.9750</td>
<td>0.0260</td>
</tr>
</tbody>
</table>
6.1. Preparation for Evaluation

<table>
<thead>
<tr>
<th>Feature</th>
<th>TPR</th>
<th>STDEV</th>
</tr>
</thead>
<tbody>
<tr>
<td>dst_byte</td>
<td>0.9740</td>
<td>0.0270</td>
</tr>
<tr>
<td>land</td>
<td>0.9750</td>
<td>0.0260</td>
</tr>
<tr>
<td>wrong_fragment</td>
<td>0.9740</td>
<td>0.0270</td>
</tr>
<tr>
<td>urgent</td>
<td>0.9750</td>
<td>0.0260</td>
</tr>
<tr>
<td>count</td>
<td>0.9750</td>
<td>0.0260</td>
</tr>
<tr>
<td>serror_rate</td>
<td>0.9740</td>
<td>0.0270</td>
</tr>
<tr>
<td>rerror_rate</td>
<td>0.9740</td>
<td>0.0270</td>
</tr>
<tr>
<td>same_srv_rate</td>
<td>0.9720</td>
<td>0.0290</td>
</tr>
<tr>
<td>diff_srv_rate</td>
<td>0.9730</td>
<td>0.0280</td>
</tr>
<tr>
<td>srv_count</td>
<td>0.9750</td>
<td>0.0260</td>
</tr>
<tr>
<td>srv_serror_rate</td>
<td>0.9750</td>
<td>0.0270</td>
</tr>
<tr>
<td>srv_rerror_rate</td>
<td>0.9740</td>
<td>0.0270</td>
</tr>
<tr>
<td>srv_diff_host_rate</td>
<td>0.9730</td>
<td>0.0270</td>
</tr>
<tr>
<td>dst_host_count</td>
<td>0.9750</td>
<td>0.0260</td>
</tr>
<tr>
<td>dst_host_serror_rate</td>
<td>0.9740</td>
<td>0.0270</td>
</tr>
<tr>
<td>dst_host_rerror_rate</td>
<td>0.9740</td>
<td>0.0270</td>
</tr>
<tr>
<td>dst_host_same_srv_rate</td>
<td>0.9740</td>
<td>0.0270</td>
</tr>
<tr>
<td>dst_host_diff_srv_rate</td>
<td>0.9750</td>
<td>0.0260</td>
</tr>
<tr>
<td>dst_host_srv_count</td>
<td>0.9750</td>
<td>0.0260</td>
</tr>
<tr>
<td>dst_host_srv_serror_rate</td>
<td>0.9740</td>
<td>0.0270</td>
</tr>
<tr>
<td>dst_host_srv_rerror_rate</td>
<td>0.9730</td>
<td>0.0280</td>
</tr>
<tr>
<td>dst_host_srv_diff_host_rate</td>
<td>0.9750</td>
<td>0.0250</td>
</tr>
<tr>
<td>dst_host_same_src_port_rate</td>
<td>0.9690</td>
<td>0.0310</td>
</tr>
<tr>
<td>hot</td>
<td>0.9750</td>
<td>0.0260</td>
</tr>
<tr>
<td>num_failed_logins</td>
<td>0.9730</td>
<td>0.0280</td>
</tr>
<tr>
<td>logged_in</td>
<td>0.9740</td>
<td>0.0270</td>
</tr>
<tr>
<td>num_compromised</td>
<td>0.9740</td>
<td>0.0270</td>
</tr>
<tr>
<td>root_shell</td>
<td>0.9740</td>
<td>0.0270</td>
</tr>
<tr>
<td>su_attempted</td>
<td>0.9740</td>
<td>0.0270</td>
</tr>
<tr>
<td>num_root</td>
<td>0.9750</td>
<td>0.0270</td>
</tr>
<tr>
<td>num_file_creations</td>
<td>0.9740</td>
<td>0.0270</td>
</tr>
<tr>
<td>num_shells</td>
<td>0.9700</td>
<td>0.0290</td>
</tr>
<tr>
<td>num_access_files</td>
<td>0.9750</td>
<td>0.0260</td>
</tr>
<tr>
<td>num_outbound_cmds</td>
<td>0.9750</td>
<td>0.0260</td>
</tr>
<tr>
<td>is_hot_login</td>
<td>0.9740</td>
<td>0.0270</td>
</tr>
<tr>
<td>is_guest_login</td>
<td>0.9750</td>
<td>0.0260</td>
</tr>
<tr>
<td>Mean value</td>
<td>0.9739</td>
<td>0.0271</td>
</tr>
<tr>
<td>STDEV</td>
<td>0.0014</td>
<td>0.0012</td>
</tr>
</tbody>
</table>

The above TPR values of the NN classifier can be illustrated with the selection margin as shown in figure 6-1.
6.1. Preparation for Evaluation

On the above figure, the selection margin is \([0.9726, 0.9752]\). Hence, if the TPR value after removing a certain feature is still within the selection margin then this feature didn’t affect the detection rate, therefore, it is not valuable. Based on this condition, these features \(\{3, 4, 14, 29, 38\}\) have a positive impact on the NN classifier. Note that positive impact means removing these features reduces the detection rate.

The same procedure has been performed for all classifiers as shown in table 6-3.

![Figure 6-1: TPR results of SBS using the NN classifier](image)

### Table 6-3: Final feature sets of SBS for all classifiers

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Features affect the TPR</th>
<th>Features affect the FPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural Networks (NN)</td>
<td>3, 4, 14, 29, 38</td>
<td>3, 4, 14, 15, 27, 29, 31, 38</td>
</tr>
<tr>
<td>Naïve Bayes (NB)</td>
<td>4, 9, 28</td>
<td>4, 9, 28</td>
</tr>
<tr>
<td>Decision Tree (J48)</td>
<td>11, 22, 30</td>
<td>21, 29</td>
</tr>
<tr>
<td>Random Tree (RT)</td>
<td>3, 6, 8, 9, 10, 20, 21,26,29,34,39</td>
<td>3,6,8,3,10,12,13,20,21,26,29,34,39</td>
</tr>
</tbody>
</table>

The table shows, for each classifier there is a feature set that positively affects the metrics TPR and FPR. Hence, the following Venn diagram for 4 sets has been established to filter out the common features between all classifiers as shown in figure 6-2.
6.1. Preparation for Evaluation

Based on the above Venn diagram, there are common features between the classifiers. There are common features affecting the TPR of one classifier and at the same time the FPR of other classifiers.

Let $Set_{TPR}$ be the universal feature set that positively affects the TPR of all classifiers, $Set_{TPR} = \{3, 4, 6, 8, 9, 10, 11, 14, 20, 21, 22, 26, 28, 30, 38, 39\}$. Then, define a Most Valuable Feature (MVF) set as $Set_{TPR}^{MVF}$, where each feature in this set should be common at least between two classifiers. All other uncommon features are called Valuable ones as $Set_{TPR}^{V}$. Accordingly, the two features sets:

- $Set_{TPR}^{MVF} = \{3, 4, 9, 29\}$
- $Set_{TPR}^{V} = \{6, 8, 10, 11, 14, 20, 21, 26, 28, 30, 38, 39\}$

The same principle is valid for the FPR, $Set_{FPR}^{MVF} = \{3, 4, 9, 21, 29\}$ and $Set_{FPR}^{V} = \{5, 6, 8, 10, 12, 13, 14, 15, 20, 26, 27, 28, 31, 30, 38, 39\}$.

Common features between the MVF set of TPR and FPR are derived as $Set_{TPR}^{MVF} \cap Set_{FPR}^{MVF} = \{3, 4, 9, 29\}$ and common features between the Valuable set of TPR and FPR are $Set_{TPR}^{V} \cap Set_{FPR}^{V} = \{6, 8, 10, 14, 20, 21, 26, 28, 38, 39\}$.

The rest is called Relevant feature set $Set_{R} = \{11, 12, 13, 15, 22, 27, 30, 31, 34\}$.

As a result, the final selected features can be divided into two features sets:

- **MVF and Valuable features sets (will be called as MVF):**
  $Set_{TPR}^{MVF} \cup Set_{FPR}^{MVF} \cup Set_{TPR}^{V} \cup Set_{FPR}^{V} = \{3, 4, 6, 8, 9, 10, 14, 20, 21, 26, 28, 29, 38, 39\}$

- **MVF, Valuable, and Relevant features (MVRF):**
  $Set_{TPR}^{MVF} \cup Set_{FPR}^{MVF} \cup Set_{TPR}^{V} \cup Set_{FPR}^{V} \cup Set_{R} = \{3, 4, 6, 8, 9, 10, 14, 20, 21, 26, 28, 29, 38, 39, 31, 32, 33, 34, 35, 36, 37, 38, 39\}$

These two features sets will be now ranked using the IG method as described in 4.3.1.2.
6.1.2.2. Results of IG method

To achieve accurate results in the online classification, all selected features must deliver sufficient amounts of information about the anomaly class in each connection. Although the previous selected features in SBS have a positive impact on the performance metrics of various classifiers, they will be further analyzed. In the IG method, each feature has been ranked according to the amount of its uncertainty to the class label anomaly. Tables 6-4 and 6-5 depict the final IG result of both features sets.

Table 6-4: Information gain of all features in the MVF set

<table>
<thead>
<tr>
<th>Feature</th>
<th>Information Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>0.6715649</td>
</tr>
<tr>
<td>dst_bytes</td>
<td>0.58435007</td>
</tr>
<tr>
<td>Flag</td>
<td>0.51938822</td>
</tr>
<tr>
<td>diff_srv_rate</td>
<td>0.50089399</td>
</tr>
<tr>
<td>dst_host_serror_rate</td>
<td>0.39539431</td>
</tr>
<tr>
<td>Count</td>
<td>0.34681662</td>
</tr>
<tr>
<td>dst_host_same_srv_port_rate</td>
<td>0.17893258</td>
</tr>
<tr>
<td>dst_host_srv_rerror_rate</td>
<td>0.08504881</td>
</tr>
<tr>
<td>dst_host_rerror_rate</td>
<td>0.05090946</td>
</tr>
<tr>
<td>Hot</td>
<td>0.01093148</td>
</tr>
<tr>
<td>wrong_fragment</td>
<td>0.00960996</td>
</tr>
<tr>
<td>num_access_files</td>
<td>0.00216444</td>
</tr>
<tr>
<td>Urgent</td>
<td>0.0000089</td>
</tr>
<tr>
<td>num_outbound_cmds</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6-5: Information gain of all features in the MVRF set

<table>
<thead>
<tr>
<th>Features</th>
<th>Information Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>0.6715649</td>
</tr>
<tr>
<td>dst_bytes</td>
<td>0.58435007</td>
</tr>
<tr>
<td>Flag</td>
<td>0.51938822</td>
</tr>
<tr>
<td>diff_srv_rate</td>
<td>0.50089399</td>
</tr>
<tr>
<td>same_srv_rate</td>
<td>0.47857645</td>
</tr>
<tr>
<td>dst_host_same_srv_rate</td>
<td>0.41792864</td>
</tr>
<tr>
<td>logged_in</td>
<td>0.40475154</td>
</tr>
</tbody>
</table>
6.1. Preparation for Evaluation

<table>
<thead>
<tr>
<th>Feature</th>
<th>IG Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>dst_host_serror_rate</td>
<td>0.39539431</td>
</tr>
<tr>
<td>serror_rate</td>
<td>0.37787435</td>
</tr>
<tr>
<td>Count</td>
<td>0.34681662</td>
</tr>
<tr>
<td>dst_host_srv_diff_host_rate</td>
<td>0.25915986</td>
</tr>
<tr>
<td>dst_host_same_src_port_rate</td>
<td>0.17893258</td>
</tr>
<tr>
<td>dst_host_srv_rerror_rate</td>
<td>0.08504881</td>
</tr>
<tr>
<td>rerror_rate</td>
<td>0.05451864</td>
</tr>
<tr>
<td>dst_host_rerror_rate</td>
<td>0.05090946</td>
</tr>
<tr>
<td>srv_count</td>
<td>0.04413898</td>
</tr>
<tr>
<td>Hot</td>
<td>0.01093148</td>
</tr>
<tr>
<td>wrong_fragment</td>
<td>0.00960996</td>
</tr>
<tr>
<td>num_access_files</td>
<td>0.00216444</td>
</tr>
<tr>
<td>su_attempted</td>
<td>0.00052957</td>
</tr>
<tr>
<td>num_failed_logins</td>
<td>0.00011326</td>
</tr>
<tr>
<td>Urgent</td>
<td>0.0000089</td>
</tr>
<tr>
<td>num_outbound_cmds</td>
<td>0</td>
</tr>
</tbody>
</table>

Each feature with a zero IG value or close to zero must be removed. Feature is close to zero when its IG value is less than $10^{-4}$ (manually selected). According to these criteria, the final feature sets, based on the IG ranking system, are:

Most valuable Feature set:
\{Service, dst_bytes, flag, diff_srv_rate, dst_host_serror_rate, count, dst_host_same_src_port_rate, dst_host_srv_rerror_rate, dst_host_rerror_rate, hot, wrong_fragment, num_access_files\}.

Most valuable and Relevant Feature set:
\{Service, dst_bytes, flag, diff_srv_rate, dst_host_serror_rate, count, dst_host_same_src_port_rate, dst_host_srv_rerror_rate, dst_host_rerror_rate, hot, wrong_fragment, num_access_files, same_save_rate, dst_host_same_srv_rate, logged_in, serror_rate, srv_count\}.

In the previous two sets, some features can be achieved directly from the IP packet header and other should be calculated. In addition, other features from the IP packet header are necessary, which are Timestamp, Source_IP, Source_port, Destination_IP, and Destination_Port. Note that some features are no longer active in the area of IDS such as the feature “land”. Consequently and based on the hybrid feature selection method, the final feature set that has been used in this thesis is summarized in Table 6-6.
6.2. IDS Performance Evaluation

Table 6-6: Final network and host features

<table>
<thead>
<tr>
<th>Feature group</th>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network basic features</td>
<td>Timestamp</td>
<td>The UNIX time when the first packet of the connection is received.</td>
</tr>
<tr>
<td></td>
<td>src_ip, dst_ip, src_port, dst_port,</td>
<td>Source IP address. Destination IP Address. Source port number. Destination port number</td>
</tr>
<tr>
<td></td>
<td>Protocol_type</td>
<td>The protocol being used at the data-link layer, network layer or transport layer. Considered protocols are ARP, ICMP, TCP and UDP.</td>
</tr>
<tr>
<td></td>
<td>Service</td>
<td>The connections application layer protocol, e.g. http, ssh. Unique numbers in the range 100.001 to 100.002 and 200.000 to 204.999 are used to describe ARP and ICMP messages respectively.</td>
</tr>
<tr>
<td></td>
<td>Src_byte</td>
<td>Total number of bytes that have been transferred from source to destination</td>
</tr>
<tr>
<td></td>
<td>Flag</td>
<td>A flag describing the connection state. Flag calculation is driven from Bro IDS [15].</td>
</tr>
<tr>
<td></td>
<td>Wrong_fragment</td>
<td>The number of fragmentation errors (i.e. overlapping fragments, fragment size is not a multiplier with 8 or reassembling is not possible)</td>
</tr>
<tr>
<td>Traffic features</td>
<td>count, serror_rate, rerror_rate,</td>
<td>All traffic features are statistical and get derived from the basic features. They are divided into two types, time-based traffic features and connection-based traffic features, which are distinguished and treated differently by OptiFilter. The first one is calculated based on a dynamic time window, e.g. the last 5 seconds, while the latter is calculated on a configurable connection window, e.g. the last 1000 connections</td>
</tr>
<tr>
<td></td>
<td>same_srv_rate, diff_srv_rate, srv_serror_rate, srv_rerror_rate, dst_host_srv_serror_rate, dst_host_srv_count</td>
<td></td>
</tr>
<tr>
<td>Content features</td>
<td>num_faidl_login, logged_in, root_shell</td>
<td>These features are obtained directly from the monitored host systems using SNMP</td>
</tr>
</tbody>
</table>

6.2. IDS PERFORMANCE EVALUATION

This section is a brief introduction about the proposed framework. The performance evaluation of the proposed framework has been performed comprehensively to include the offline and the online operational modes. In the offline operational mode, the first part of the framework has been evaluated
6.3. Evaluation in the offline mode

The evaluation in the offline mode includes both two parts of the framework. The first one will use several data flows as described before and generate datasets based on the selected features in table 6-6.

6.3.1. Evaluation Results of Part 1 (OptiFilter)

The evaluation has been performed in two different scenarios (i.e. two experimental runs). Each scenario uses various data flows.

The First Scenario

In this scenario, OptiFilter uses available data flows from DARPA (KDD), iCTF, and synthetic data flows from SecMonet. For each data flow, OptiFilter has been configured twice, and two performance metrics have been also monitored. A summary of preparation for the test scenario is depicted in Table 6-7.

<table>
<thead>
<tr>
<th>Resource of data flow</th>
<th>Available data flow:={DARPA (KDD), iCTF} Synthetic data flow:={SecMonet}</th>
</tr>
</thead>
<tbody>
<tr>
<td>OptiFilter parameters</td>
<td>Queue capacity = 3 windows Time slot window ( w = 2 ) sec, Backlog = 100</td>
</tr>
<tr>
<td></td>
<td>Time slot window ( w = 5 ) sec, Backlog = 1000</td>
</tr>
<tr>
<td>Performance metrics</td>
<td>Number of packets, Processing time in milliseconds</td>
</tr>
</tbody>
</table>

Note that DARPA (KDD) has generated the KDDCup99 dataset using window length of 2 seconds and 100 connections in the backlog. Therefore, OptiFilter has been configured twice. The first configuration uses the same parameters as DARPA (i.e 2 sec, 100 connections) and this will examine if the outdated parameters from DARPA are still valid for the current network traffic. The second configuration parameters are determined empirically. The reason of choosing only two metrics (number of packets and processing time) is to ensure that OptiFilter can handle different offline data flows within sufficient time. The
performance metrics of OptiFilter for this scenario have been demonstrated in figure 6-3.

OptiFilter could process large number of data flows within sufficient time. In the iCTF data flow, OptiFilter could process more than 9000 packets within the 2 seconds window (window Nr. 890) and more than 20000 packets within the 5 seconds window (350). It seems that the maximum processing time by the 2 seconds window is almost 2000 millisecond while in the 5 seconds window is only 4000 milliseconds. It means the configuration of 5 seconds window is more feasible than the 2 seconds window.

The same result can be achieved from the KDD data flows. In contrast, maximum number of packets in the synthetic data is almost the same for each window, because data flows have been automatically and symmetrically generated using a traffic generator, which leads of course to equal processing time. Table 6-8 summarizes the results.
6.3. Evaluation in the offline mode

Table 6-8: Offline evaluation of OptiFilter – First scenario comparison

<table>
<thead>
<tr>
<th>Dataset</th>
<th>max Packet No.</th>
<th>max processing time /ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>KDD</td>
<td>1544</td>
<td>666</td>
</tr>
<tr>
<td>ICTF</td>
<td>8740</td>
<td>1748</td>
</tr>
<tr>
<td>SecMonet</td>
<td>352</td>
<td>165</td>
</tr>
</tbody>
</table>

Note that only one window has oddly needed more than 2 seconds by KDD and SecMonet data flows. This flaw or kink could be referred to several reasons such as corruption in the data flow, executable process in the background, or a programming bug. Moreover, the processing time by the 2 seconds window has needed almost the full window to process the packets. Therefore, choosing 2 seconds window with only 100 connections in the backlog, as the DARPA program did, is not appropriate anymore for the current computer networks. Based on the first scenario, this thesis configures OptiFilter as 5 seconds window and 1000 connections in the backlog. Other approaches such as 3 seconds window have been internally examined and the results were almost the same as the 2 seconds window.

The Second Scenario

In this scenario, OptiFilter uses available data flows from DARPA (KDD), synthetic data flows from SecMonet (includes anomaly so it is called SecMonet+anomaly), and realistic data flows RealSet from company’s traffic. For each data flow, four performance metrics have been monitored accordingly. A summary of preparation for the test scenario is shown in Table 6-9.

Table 6-9: Offline evaluation of OptiFilter – Second scenario

<table>
<thead>
<tr>
<th>Resource of data flow</th>
<th>Available data flow:= {DARPA (KDD)}&lt;br&gt;Synthetic data flow:= {SecMonet+anomaly}&lt;br&gt;Realistic data flow:= {RealSet}</th>
</tr>
</thead>
<tbody>
<tr>
<td>OptiFilter parameters</td>
<td>Queue capacity = 3 windows&lt;br&gt;Time slot window ( w = 5 \text{ sec} ), Backlog = 1000</td>
</tr>
<tr>
<td>Performance metrics</td>
<td>Number of packets&lt;br&gt;Number of connections&lt;br&gt;Number of packet drops&lt;br&gt;Processing time in milliseconds</td>
</tr>
</tbody>
</table>
This scenario is broader than the first one. It includes two more performance metrics, which are number of constructed connections and number of packet drops. The first result is the number of processed packets per window for all data flows as illustrated in figure 6-4.

![Figure 6-4: Offline evaluation of OptiFilter for number of packets– Second scenario](image)

OptiFilter can handle a huge amount of packets as shown on the above figure. Moreover, the visualization of the processed packets (from the data flow SecMonet+anomaly) by OptiFilter provides further information for the administrator (such as determining various types of attacks). Figure 6-5 depicts the result of require processing time of each window.

![Figure 6-5: Offline evaluation of OptiFilter for processing time– Second scenario](image)
6.3. Evaluation in the offline mode

Although the data flows are different in size and characteristics, OptiFilter could process the packets and construct connections within the time slot window. Once the window length, i.e. 5 seconds, is not more enough, OptiFilter can be configured to accommodate any computer network. Figure 6-6 shows the constructed connections per window.

![Figure 6-6: Offline evaluation of OptiFilter for number of connections– Second scenario](image)

The last metric is the packet drops, which is a very important metric in computer management networks especially 1 GB – 10 GB networks. This metric is recently a very vibrant topic in network management; hence, OptiFilter has been also evaluated to examine the number of packet drops per window. Figure 6-7 demonstrates the result.

![Figure 6-7: Offline evaluation of OptiFilter for number packet drops– Second scenario](image)
By the data flow from Secmonet+anomaly, number of packet drops is very small. At the beginning OptiFilter has dropped around 150 packets due to the cold start; that is, OptiFilter starting up phase. Moreover, some IPv6 packets appear at the beginning and OptiFilter dropped them immediately. The data flow RealSet has regularly around 20 packet drops each 100 windows which refers to the IPv6 packets and some packets belong to other dedicated services, which are not configured in OptiFilter. In the data flow from SecMonet, OptiFilter dropped all IPv6 packets and because the data flows are synthetically generated, the number of packet drops is repeated until the last window.

All these metrics have been summarized in one table for better illustration and understanding. The minimum and maximum values of each metric for each data flow are illustrated in Table 6-10.

<table>
<thead>
<tr>
<th></th>
<th>DARPA (KDD)</th>
<th>SecMonet+anomaly</th>
<th>SecMonet</th>
<th>RealSet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of packets</td>
<td>min max</td>
<td>min max</td>
<td>min max</td>
<td>min max</td>
</tr>
<tr>
<td>Packet drops</td>
<td>0 2226</td>
<td>0 24120</td>
<td>0 42838</td>
<td>11592  67445</td>
</tr>
<tr>
<td>Number of Connections</td>
<td>0 83</td>
<td>0 5163</td>
<td>0 493</td>
<td>0 3487</td>
</tr>
<tr>
<td>Processing time (ms)</td>
<td>0.012 16.522</td>
<td>0.004 859.59</td>
<td>0.011 1800</td>
<td>0.017 ≈ 5000</td>
</tr>
</tbody>
</table>

### 6.3.2. Evaluation Results of Part 2 (Adaptive Classifier)

The second part of the offline evaluation focuses on the classifier EGHSOM model. It examines the following performance metrics: The accuracy, the detection rate, and the false positive rate. The most well-known method to evaluate the IDS classifier is the cross-validation technique.

The cross-validation is an accuracy estimation technique that evaluates the preciseness of the classifier model by building a confusion matrix for different datasets. The cross-validation technique divides the dataset to $n$ folds, where the fold can be defined as the total number of subsets of the concerned dataset. For instance, if the classifier is evaluated by 5-fold cross-validation technique then the dataset will be divided into 5 sets each set has the size $n/5$ so that the classifier is trained using 4 sets and evaluated by the last set. This process will be repeated 5 times and finally the mean accuracy is the final accuracy of the classifier. An illustration example for 4-fold cross-validation is shown in figure 6-8.
6.3. Evaluation in the offline mode

According to the example, the first fold trains the classifier and evaluates it using the test set and then records the accuracy, and so forth.

For evaluating the adaptive classifier, the 10-fold cross-validation has been selected because it has shown that 10 is about the right number of folds to get the best estimate of error and it is the widely used technique for evaluating the classifier model [195]. In addition, several labeled datasets have been prepared, which are varying in size and source. The following datasets have been prepared as shown in Table 6-11.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of instances</th>
<th>Normal instances</th>
<th>Anomaly instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSL-KDD</td>
<td>125969</td>
<td>97252</td>
<td>28717</td>
</tr>
<tr>
<td>SecMonet+anomaly</td>
<td>150239</td>
<td>130128</td>
<td>20111</td>
</tr>
<tr>
<td>RealSet</td>
<td>186414</td>
<td>186414</td>
<td>0</td>
</tr>
</tbody>
</table>

The third dataset RealSet has no anomaly connections, which is intentionally constructed from only normal connections to evaluate the classifier even with a clean dataset.

On the other hand, table 6-12 summarizes the EGHSOM parameters, which are used in the evaluation. Note that the proposed classifier model has been tested for several times to find out the optimal parameters. Thus, these parameters have been chosen because they have delivered often the best performance results throughout all tests.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>0.01</td>
<td>Threshold for Horizontal growth</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>0.0001</td>
<td>Threshold for vertical growth</td>
</tr>
<tr>
<td>( \varsigma )</td>
<td>0.65</td>
<td>Threshold for splitting a node</td>
</tr>
<tr>
<td>( r_Y )</td>
<td>0.75</td>
<td>Threshold for Similarity measure between two BMUs</td>
</tr>
<tr>
<td>( \lambda )</td>
<td>1000</td>
<td>Number of Iterations</td>
</tr>
<tr>
<td>( n_{RF} )</td>
<td>250</td>
<td>Minimum number of Attraction value</td>
</tr>
<tr>
<td>$h_c(t)$</td>
<td>Gaussian</td>
<td>Neighborhood function</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>$\sigma(t)$</td>
<td>Exponential</td>
<td>Radius for Gaussian function</td>
</tr>
<tr>
<td>$\alpha(t)$</td>
<td>Exponential</td>
<td>Learning rate function</td>
</tr>
<tr>
<td>$\delta$</td>
<td>Margin threshold</td>
<td>Classification-confidence margin threshold</td>
</tr>
</tbody>
</table>

After performing the 10-fold cross-validation, the following additional metrics have been also monitored after each fold for all above datasets in table 6-11. These metrics are the training time, total number of grids, total number of BMUs before merging, the total number of BMUs after merging, and the number of detected connections as unknown.

The evaluation technique was performed 10 times for each dataset, which means the final result will have 30 confusion matrixes. Thus, table 6-13 shows the confusion matrix for 1-fold of the dataset SecMonet+anomaly.

Table 6-13: Confusion matrix from the 1-fold of the dataset SecMonet+anomaly

<table>
<thead>
<tr>
<th>Predicted Class</th>
<th>Anomaly</th>
<th>Normal</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Class</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anomaly</td>
<td>TP=4068</td>
<td>FN=873</td>
<td>4941</td>
</tr>
<tr>
<td>Normal</td>
<td>FP=54</td>
<td>TN=9030</td>
<td>9084</td>
</tr>
<tr>
<td>Total</td>
<td>4122</td>
<td>9903</td>
<td>14025</td>
</tr>
</tbody>
</table>

The matrix shows that the total classified connections in the test dataset are 14025. Note that there are 9 more confusion matrixes for the same dataset. According to the above confusion matrix and by referring to equations 6-1, 6-2, and 6-3, the TPR= 82.335%, FPR= 0.0059 %, and the ACC=93.39%. Table 6-14 demonstrates the final result (TPR, FPR, and accuracy) of the 10-fold cross-validation for all datasets.

Table 6-14: Performance evaluation result of the adaptive classifier using 10-fold cross-validation

<table>
<thead>
<tr>
<th>Dataset</th>
<th>NSL-KDD</th>
<th>SecMonet+anomaly</th>
<th>ReaSet</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross-Val</td>
<td>TPR</td>
<td>FPR</td>
<td>ACC</td>
<td>TPR</td>
</tr>
<tr>
<td>Fold 1</td>
<td>95.9914</td>
<td>0.0000</td>
<td>95.9914</td>
<td>82.3350</td>
</tr>
<tr>
<td>Fold 2</td>
<td>93.3842</td>
<td>0.0000</td>
<td>93.3842</td>
<td>81.8950</td>
</tr>
<tr>
<td>Fold 3</td>
<td>97.5393</td>
<td>0.0000</td>
<td>97.5393</td>
<td>71.3190</td>
</tr>
<tr>
<td>Fold 4</td>
<td>94.9040</td>
<td>0.0000</td>
<td>94.9040</td>
<td>0.0000</td>
</tr>
<tr>
<td>Fold 5</td>
<td>99.0380</td>
<td>0.0869</td>
<td>98.2620</td>
<td>74.5150</td>
</tr>
<tr>
<td>Fold 6</td>
<td>100.0000</td>
<td>0.0822</td>
<td>99.3770</td>
<td>92.5450</td>
</tr>
<tr>
<td>Fold 7</td>
<td>100.0000</td>
<td>0.0576</td>
<td>99.9803</td>
<td>91.6390</td>
</tr>
<tr>
<td>Fold 8</td>
<td>98.9210</td>
<td>0.0107</td>
<td>96.9830</td>
<td>80.2730</td>
</tr>
<tr>
<td>Fold 9</td>
<td>98.7340</td>
<td>0.0126</td>
<td>95.8770</td>
<td>4.5946</td>
</tr>
<tr>
<td>Fold 10</td>
<td>100.0000</td>
<td>0.0533</td>
<td>96.4270</td>
<td>77.0220</td>
</tr>
<tr>
<td>Average</td>
<td>97.8475</td>
<td>0.0302</td>
<td>96.0052</td>
<td>65.6138</td>
</tr>
</tbody>
</table>

The aim of the cross-validation is to monitor the accuracy of the classifier and to check if the model is superior in classifying the connections. The mean
accuracy of the total 10-fold has reached 97.11 % which is a satisfied result. Moreover, the mean FPR of the model is very low. On the other hand, the mean TPR is only 54 % which is very normal. The reason for such a TPR value refers to, the TPR of the RealSet is 0 % of all folds throughout the cross-validation, because this dataset has no anomaly and of course the classifier will not detect any, so the value of TP in the confusion matrix will stay 0.

The previous results of the performance metrics in table 6-14 are represented on figure 6-9.

Figure 6-9: Performance metrics of the offline 10-fold cross-validation
6.3. Evaluation in the offline mode

The accuracy on the first figure and the FPR in the second figure show promising results for all datasets. The TPR on the third figure shows satisfied result although the TPR values are not large enough.

During the evaluation, the total number of detected connections as unknown has been monitored to confirm that the model is able to detect unknown connections. It is very possible that a test subset (fold) contains anomaly connections which are not trained in the train subset, and then the classifier classifies them as unknown. Figure 6-10 shows the number of detected connections as unknown during the cross-validation for all datasets.

![Figure 6-10: Total number of detected unknown connections during the offline cross-validation](image)

The next metrics is the total number of grids (maps). It has been monitored to examine that the EGHSOM builds almost the same number of maps for each fold as illustrated in figure 6-11.

![Figure 6-11: Total number of maps for each fold in the offline cross-validation](image)
Obviously, the number of maps of each fold for the three datasets is almost the same, and this indicates on a stable topology growth in the EGHSOM.

The next monitored metric during the cross-validation technique is the number of BMUs before merging and after merging. This metric shows the efficiency of the second enhancement on the GHSOM (see section 5.3.2.). Figure 6-12 demonstrates the result of merging BMUs during the cross-validation.

![Figure 6-12: BMUs number before and after merging in the offline cross-validation](image)

The last monitored metric is the training time of each fold as shown in figure 6-13. The training time depends on the number of instances in the train subset; however, the total training time for each fold was very short in comparison to the amount of instances.

![Figure 6-13: Training time after each fold in the offline cross-validation](image)
The RealSet has the maximum number of instances, hence it needs more time than others in training.

The cross-validation technique for the proposed adaptive classifier presents promising performance metrics. The overall result can be summarized as the following:
- The training process is very stable and fast enough.
- The growth topology is stable and conservative.
- The classifier model is able to detect unknown connections effectively.
- The overall performance of the proposed model is optimal for the online classification mode.

6.4. Evaluation in the online mode (Real-time)

Online evaluation is the most important part in this chapter, because it demonstrates the effectiveness and proficiency of the proposed framework in the real-time. Moreover, this part will unveil the capability of the framework in detecting unknown data flow in the real-time. This is to imply that the framework will be installed on a real-time 1 to 10 GB network that generates continuous massive data flow, activated to start processing and classification, and then evaluated intensively throughout the operation.

This evaluation study focuses on the following points:
- The aggregation methods.
- The queue concept.
- Continuous dataset generation.
- Performance metrics of OptiFilter.
- Active BMUs in the EGHSOM model.
- Number of classified connections especially the unknown ones.
- The Adaptivity of the proposed framework in real-time.
- The ability of the NNB to classify unknown connections
- The persistent operation without any flaw or crash throughout the classification.

The above points will be examined by the performance metrics for each component and the results of OptiFilter and the adaptive classifier will be presented individually.

The general scenario for testing the framework includes the following steps:
1. The complete hardware (the server and the proposed framework) is installed in a network segmentation of a real-time computer and communication network from our project partner. More than 400 employees are connected to this network and they perform their daily activities such surfing, mailing, scripting, etc.
6.4. Evaluation in the online mode (Real-time)

2. The first scenario will be at the noon, in which all employees browsing and having access to the external network, i.e. the Internet.
3. The output of OptiFilter, EGHSOM, and NNB will be recorded throughout the entire operation in the first scenario. Accordingly the result will be presented.
4. The second scenario will be in another day and for longer time than the first one. Steps 2 and 3 will be repeated respectively.

6.4.1. First Scenario in the Real-time

The first scenario focuses on the functionality of OptiFilter, EGHSOM and NNB models and the update models are disabled. This step has been taken to ensure that the framework can effectively operate and classify the traffic. Table 6-15 depicts the prepared configuration parameters.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>instances</th>
<th>normal</th>
<th>anomaly</th>
</tr>
</thead>
<tbody>
<tr>
<td>128500</td>
<td>100000</td>
<td>28500</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>0.001</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>0.0001</td>
</tr>
<tr>
<td>$\zeta$</td>
<td>0.7</td>
</tr>
<tr>
<td>$n_{RF}$</td>
<td>200</td>
</tr>
<tr>
<td>$\delta$</td>
<td>[0.4385,2.198]</td>
</tr>
<tr>
<td>anomaly BMUs</td>
<td>34,69,148,179</td>
</tr>
</tbody>
</table>

The NNB model has been built using the same parameter but with only normal instances from the RealSet because the data flow of RealSet is realistic one.

Results of OptiFilter

The proposed framework has started almost at 12:15 and ends at 13:30 and the operational time was 4589 seconds in real time. Hence, OptiFilter has processed around 920 windows (note that each window is 5 seconds long). The most used services during the operational mode are: {$HTTP, SNMP, HTTPS, SMTP, FTP, SSH, Telnet, DHCP, NTP, NetBIOS, others$}. OptiFilter examines only the following protocols: {$TCP, UDP, ICMP, ARP$}.
The nature of the aggregated network traffic from the corresponding network segmentation is shown in figure 6-14.

Figure 6-14: Most used services in real-time – First scenario

The main performance metrics of OptiFilter (number of packets, number of connections, number of packet drops, and processing time) are considered in this scenario. Firstly, the number of processed packets is illustrated in figure 6-15.

Figure 6-15: Number of processed packets by OptiFilter in real-time – First scenario

Some windows could process more than 13500 packets within the 5 seconds in real-time. The average number of packets is 2248 of all 920 windows.
In figure 6-16 and based on the total number of packets pro window, the number of constructed connections is illustrated.

The number of packet drops during the online mode is very small and devolve to be not worth mentioning, because it reaches only 6 packets, which means the framework has processed all captured packets from the network data flow without dropping any. The last window has dropped only 6 packets, which are from the IPv6 packets. Figure 6-17 shows the last metric which is the processing time.
The processing time obviously is very convenient and did not exceed the 5 seconds window length. This successful result confirms on the correctness of choosing the optimal parameters of OptiFilter for the online mode.

**Results of the Adaptive Classifier**

The used classification-confidence margin threshold in the online mode is presented in figure 6-18.

![Figure 6-18: Classification-confidence margin threshold in real-time – First scenario](image)

The previous margin is the average value for min and max distances as illustrated in 5.3.1. The average value will guarantee preciseness in uncovering unknown connections. This margin will be used then to check if the distance of each constructed connection falls inside or outside the average range (dot lines) as shown figure 6-19.

![Figure 6-19: Distances of all connections to the EGHSOM model in real-time – First scenario](image)
As illustrated in the figure, most of constructed connections could be classified by the adaptive classifier because the distance of each individual connection falls in the classification-confidence margin except 3 connections, which then classified as unknown. More details are summarized in the following confusion matrix.

The confusion matrix has been determined according to the following rules:
- All connections between internal IPs are considered normal.
- All connections from internal IPs to external IPs are considered normal.
- All connections from external IPs to internal IPs have been further analyzed and labeled as normal or anomaly based on the source IP. The external IPs have been examined by checking the IP location and the service for the corresponding host. For instance, some IPs like “80.252.91.41” access the internal network and the framework classifies it as unknown due to the distance measure. On the other hand, the framework has classified some connections as anomaly although the connection was between two internal IPs, which then considered as FP in the confusion matrix.

Table 6-16 shows the confusion matrix and the related performance metrics.

<table>
<thead>
<tr>
<th>Predicted Class</th>
<th>Anomaly</th>
<th>Normal</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Class</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anomaly</td>
<td>TP=3</td>
<td>FN=0</td>
<td>3</td>
</tr>
<tr>
<td>Normal</td>
<td>FP=57</td>
<td>TN=303544</td>
<td>303601</td>
</tr>
<tr>
<td>Total</td>
<td>60</td>
<td>303544</td>
<td>303604</td>
</tr>
</tbody>
</table>

Performance metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specificity</td>
<td>TNR</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>TPR, DR, Recall</td>
</tr>
<tr>
<td>1-Specificity</td>
<td>FPR</td>
</tr>
<tr>
<td>Accuracy</td>
<td>ACC</td>
</tr>
</tbody>
</table>

The specificity is the percentage that the adaptive classifier has classified normal connection as normal. In this scenario, the adaptive classifier has classified 53 connections as anomaly. Thus, these connections have been further analyzed to verify the correctness of the detection. But, they were established between internal IPs, therefore, they have considered as FP in the confusion matrix.

It is important to note that, building the confusion matrix in the online operational mode is a very complicated task and without any rules this task tends to be impossible.

The total number of classified connections by EGHSOM and the percentage of normal, anomaly, and unknown connections is summarized in table 6-17.
6.4. Evaluation in the online mode (Real-time)

Table 6-17: Classified connections in real-time – First scenario

<table>
<thead>
<tr>
<th>Classified connection</th>
<th>Total number</th>
<th>percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>303544</td>
<td>99.9 %</td>
</tr>
<tr>
<td>Anomaly</td>
<td>57</td>
<td>0.0188 %</td>
</tr>
<tr>
<td>Unknown</td>
<td>3</td>
<td>0.0009 %</td>
</tr>
</tbody>
</table>

According to the activity of the BMUs in the online mode, some BMUs have classified more than half of the traffic. The reason is that, these BMUs have been built from the realistic dataset RealSet and hence they are very similar to the nature of the constructed connections. Figure 6-20 shows the activity of each BMU.

BMUs 28 and 22 were very active in the classification due to the mentioned reason above. But other BMUs were also active and could classify several connections. Similarly, BMUs attraction value is shown in figure 6-21.
Finally, in the first scenario, OptiFilter has effectively aggregated the traffic without any packet drops and constructed connections within sufficient processing time accordingly. The adaptive classifier is very accurate in classifying these connections. It could reveal the unknown connections using the classification-confidence margin threshold and misclassified some instances as anomaly although they were normal. In general, the evaluation was very promising.

### 6.4.2. Second Scenario in the Real-time

The second scenario has been configured based on the parameters in table 6-18.

<table>
<thead>
<tr>
<th>Dataset instances</th>
<th>normal</th>
<th>anomaly</th>
</tr>
</thead>
<tbody>
<tr>
<td>150000</td>
<td>100000</td>
<td>50000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EGHSOM parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>0.05</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>0.0005</td>
</tr>
<tr>
<td>$\zeta$</td>
<td>0.7</td>
</tr>
<tr>
<td>$n_{RF}$</td>
<td>250</td>
</tr>
<tr>
<td>$\delta$</td>
<td>[0.6253,2.376]</td>
</tr>
<tr>
<td>$\eta$</td>
<td>0.75</td>
</tr>
</tbody>
</table>

| Update EGHSOM    | 25-60 min, 100 anomaly |
| Update NNB       | 30-60 min, 5000 normal |

In this scenario, the classified normal connections from the first scenario have been used to build the NNB model. In addition, after each update of EGHSOM, the classification-confidence margin will be adapted according to the new model as well as the final BMUs as explained in section 5.3.7.

**Results of OptiFilter**

In this scenario the proposed framework has started at 13:30 and ended at 16:00. The total number of processed windows in OptiFilter is 1556 windows. The result of OptiFilter in this scenario will be demonstrated as the same demonstration of the first scenario.
The nature of the aggregated traffic from the network segmentation is illustrated in figure 6-22.

![Pie chart showing traffic distribution](image)

**Figure 6-22: Most used services in real-time – Second scenario**

Note that, the second scenario is performed afternoon while the access to the external network is limited; hence the number of HTTP requests is less than the one in the first scenario.

The number of packets which are processed by OptiFilter for each window individually is shown in figure 6-23.

![Graph showing packets processed](image)

**Figure 6-23: Number of processed packets by OptiFilter in real-time – Second scenario**
It seems that the average number of packets processed inside the window is less than the average one in the first scenario due to the blocked services at the time of evaluation such as the http service. Hence, the number of constructed connections will be smaller than the one in the first scenario as shown in figure 6-24.

![Figure 6-24: Number of constructed connections by OptiFilter in real-time – Second scenario](image1)

However, the packets drops still the same but this time at the beginning of aggregation the framework has dropped 4 packets.

Finally, figure 6-25 illustrates the processing time.

![Figure 6-25: Processing time of OptiFilter in real-time – Second scenario](image2)
As a result, the second scenario in evaluating OptiFilter shows almost the same result as in the first scenario. However, the number of packets and the corresponding constructed connections is decreased here due the less traffic generated by the users. Similarly, the packet drops is almost the same but at the beginning due to the cold start of the framework or the packets were IPv6 packets so they have been dropped. The processing time is still optimal for the online mode.

**Results of the Adaptive Classifier**

In this scenario, other parameters have been examined such as the classification-confidence adaption after the update and the BMUs on the final EGHSOM and NNB models. In addition the same performance metrics as in the first scenario have been also examined here.

The main goal of this scenario is to check whether the EGHSOM and NNB update models perform the update when the update condition is satisfied as shown in table 5-17. The update condition for EGHSOM can be explained as the following. After 25 minutes of starting the framework, if the number of detected connections as anomaly is greater than 100 then perform update for the EGHSOM. If the number is still less than 100 then keep monitoring it for the next 35 minutes. Once the framework has been working since one hour then force an update and use the current anomaly.csv file as the source dataset for the updating process. In the next hour repeat the condition again. This explanation is also valid for the NNB update mode but after 30 minutes not 25. The used classification-confidence margin threshold is illustrated in figure 6-26.

![Figure 6-26: Classification-confidence margin threshold in real-time – Second scenario](image)
6.4. Evaluation in the online mode (Real-time)

Based on the used average margin in figure 6-26, the distances of all connections to the BMUs are depicted in figure 6-27.

![Graph showing distances of all connections to the BMUs](image)

**Figure 6-27:** Distances of all connections to the EGHSOM model in real-time – Second scenario

Note that, all connections have fallen inside the margin, i.e. no unknown vectors. The final confusion matrix with the same rules in the first scenario is illustrated in table 6-19.

**Table 6-19: Confusion matrix of the adaptive classifier in real-time – Second scenario**

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Predicted Class</th>
<th>Anomaly</th>
<th>Normal</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomaly</td>
<td>TP=20</td>
<td>FN=0</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>TP=597</td>
<td>TN=255201</td>
<td>255798</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>617</td>
<td>255201</td>
<td>255818</td>
<td></td>
</tr>
</tbody>
</table>

**Performance metrics**

- Specificity: TNR = 0.99766
- Sensitivity: TPR, DR, Recall = 1
- 1-Specificity: FPR = 0.00233
- Accuracy: ACC = 0.99766

The accuracy is a bit decreased which is not affecting the framework at all. The final number of detected connections from EGHSOM is illustrated in table 6-20.

**Table 6-20: Classified connections in real-time – Second scenario**

<table>
<thead>
<tr>
<th>Classified connection</th>
<th>Total number</th>
<th>percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>255201</td>
<td>99.76 %</td>
</tr>
<tr>
<td>Anomaly</td>
<td>617</td>
<td>0.24 %</td>
</tr>
<tr>
<td>Unknown</td>
<td>0</td>
<td>0 %</td>
</tr>
</tbody>
</table>
The number of active BMUs during the second scenario is also almost the same as the last one. However, there is no certain BMU specifically, all BMUs were active and participate in the classification as shown in figure 6-28.

Consequently, the attraction value of each BMU during the online operational mode can be also depicted from in figure 6-29.

Obviously all BMUs contribute in the classification except the BMU 150 which is more attracted for the input connections. According to the update procedure, the previous condition for updating the NNB and the EGHSOM models has been performed. Accordingly, the radiuses
of the new NNB model have been adapted to the new detected normal connections and the classification-confidence margin has been also adapted to the new detected anomaly.

The adaption between the radiuses in the original NNB model and the last updated NNB model is illustrated in figure 6-30.

![Figure 6-30: Adapted radiuses of the NNB model in real-time – Second scenario](image)

The above figure shows that during the online classification, the radiuses of the NNB model have been adapted to the current network data flow and hence the NNB model successfully continued representing the traffic.

On the other hand, the classification-confidence margin is also updated during the online mode. Figure 6-31 shows the new adapted classification-confidence margin.

![Figure 6-31: Adapted Classification-confidence margin in real-time – Second scenario](image)
The average minimum value of the margin is close to zero therefore it is not obvious on the previous figure. Similarly, the margin is adapted during the online mode in such a way that the classifier is adapted to detect the beforehand detected anomaly connections.

As a result, the framework shows again very promising results in the second scenario. The framework has successfully adapted its models and the classification parameters. Thus, the radiuses of NNB and the classification-confidence margin of EGHSOM are kept updated which means the framework can accommodate the incoming connections from OptiFilter and detect new anomaly significantly.

**6.5. COMPARATIVE STUDY**

In this section, the result of the proposed IDS framework is compared with the best known IDS approaches in this area. It is worth mentioning every approach focuses on certain target. Some approaches consider only the HTTP traffic and ignore other services, and others have evaluate their models only offline or in a simulated network. The comparison study has examined the accuracy, detection rate and false alarm rate. We have obtained these values as follows: For each framework, we have examined whether the framework performed single experimental run or several runs. If the framework has performed several runs then we have calculated the average value for each performance metric. In addition, we have taken notice of other characteristics such as, which operational mode operates the framework and how many data sources have been used in the evaluation study. In spite of that, the comparison can be considered as a general comparison study, because each framework enjoys different properties and prospective. Finally, some frameworks have examined only one performance metric, in this case we left the other metrics for the corresponding framework blank.

The comparison result includes the approach, the source of data, the evaluation mode which is either offline or online, and the performance metrics TPR, FPR, and accuracy.

Table 6-21 summarizes the performance comparison.
The proposed framework in this thesis is considered a competitor for other frameworks. It has achieved a promising result in comparison to others. For instance, the framework from Zolotukhin et al. has considered only HTTP requests and achieved a promising performance result such as 99.8 % accuracy. However, upon their article, they have evaluated their framework only on the offline operational mode. In contrast, our framework, in addition to the offline mode, has been evaluated in the online operational mode but it doesn’t analyze the HTTP requests thoroughly as their framework does. Another discussion aspect is the adaptive property in the IDS. Just few frameworks have investigated this aspect in their work such as Ippoliti et al. However, we consider our framework more adaptive because it examines the Adaptivity task for both classifier models (i.e. EGHSOM and NNB) and several parameters such as the classification-confidence margin, whereas their framework adapts only the normalization method in the offline mode. The comparison shows, there are several IDS approaches that have achieved satisfied accuracy. However, most of them have been evaluated using the outdated KDDCup99 or other offline datasets.

---

1 EGHSOM main target is network and hosts while other approaches have focused only on the network data flows
datasets. In addition, these approaches, in comparison to the proposed one in this thesis, have focused on the network traffic and totally ignored the host events.

Another major point is the ability of our framework to aggregate massive data flow and constructs connections accordingly, while other approaches evaluate their models by offline available datasets. Thus, this is considered as a second superior point than other approaches.
Chapter 7. Conclusion and Future Work

7.1. Conclusion

One of the most key aspects of securing computer and communication networks nowadays is the ability to accurately uncover malicious connections (or the so-called zero-day-attack) and within sufficient detection time. Furthermore, the fundamental problem of current IDSs can be described in two points. The first one is the difficulty of processing the massive data flows and the second one is proposing an adaptive intrusion detection model that operates in the real-time and efficiently reveals anomaly.

This thesis proposes a comprehensive IDS framework that has provided a gradual solution to the problem of securing computer and communication networks. It is comprehensive because it successfully addresses all main steps of developing an IDS framework. These steps can be summarized as data aggregation and preprocessing, feature selection and extraction, and classification.

It presents a hybrid feature selection method that consists of sequential backward search and information gain. The proposed hybrid method has selected the most valuable and relevant features which are very important for the proposed framework, which uses these important features to construct connection vectors from the massive data flow.

Principally, the framework consists of two main parts. The first part is called OptiFilter, which is responsible for aggregating the massive data flows and processing them in a dynamic queuing concept to construct continuous connection vectors based on the valuable and important features. The second part is an adaptive classifier that includes three main components, EGHSOM, NNB, and the update models. The classifier model EGHSOM is an intelligent neural network model which has emerged from the original GHSOM approach.

In this thesis, the following enhancements have been applied on the standard GHSOM algorithm to resolve its shortcomings and weaknesses. The first
enhancement defines a classification-confidence margin to detect unknown data flow. The second enhancement stabilizes the topology growth of the GHSOM by reinforcing the BMUs on the final GHSOM model. In addition, we have stabilized the topology by adding a third threshold value for the growth algorithm, which is a splitting technique that avoids generating a weak teleology. The next enhancement has addressed the initialization process of GHSOM. It presents an expressive weight vectors initialization that robustly controls the topology. EGHSOM has been further utilized to define a NNB model to the entire data flow.

Generally, the proposed framework operates as follows. EGHSOM classifies the constructed connections to normal, anomaly or unknown based on a novel classification-confidence margin threshold. Unknown connections, on the other hand, are presented to the NNB model, which analyzes them thoroughly and classifies each individual unknown connection as belonging to normal or anomaly class. In addition, the proposed update models are used to constantly adapt the actual EGHSOM and NNB models and other parameters such as the classification-confidence margin and the radiiuses of the NNB model. Hence, the update models mitigate the phenomenon of concept drift.

We have evaluated the framework in the offline and the online operational modes. In the offline operational mode, each part of the framework has been separately evaluated. In contrast, in the online operational mode the entire framework has been evaluated in a real 1 to 10 GB computer network (project partner).

For the offline evaluation, the dump data flows from DARPA, synthetic university traffic, iCTF, and realistic company traffic have been used to carry out the evaluation on both parts. The overall result was promising because OptiFilter could process all data flows within sufficient time and with few packet drops. The classifier has also achieved a significant performance by reaching an accuracy of 97% and a FPR of 0.01417.

In contrast, the entire framework has been installed on a real 10GB computer network segmentation of a project partner. OptiFilter has successfully processed the massive data flows without any packet drops and with a sufficient time less than the time slot window provides inside the dynamic queue. The adaptive classifier has achieved a higher accuracy and lower FPR than the offline mode (99% and 0.00233%) and it could adapt the EGHSOM and NNB models every 25 min without any delay.

In addition, the EGHSOM model has classified some connections as unknown while the NNB model has successfully classified most of them as normal. Finally, the proposed framework has been compared to other approaches in IDS. The proposed IDS framework in this thesis is considered rival to all approaches.
Although the proposed framework focuses on achieving the optimal performance, it can show appropriate alarm information to the end user. In summary, the proposed framework is an adaptive real-time anomaly IDS that sufficiently handles voluminous data flow and uncovers malicious connections even if they have been manipulated.

7.2. Future Work and Open Issues

The superiority of the proposed framework over other frameworks does not mean that it should not be subject to other improvements. Quite the contrary, it should be analyzed and additionally improved to be more automated. For instance, the framework can be improved to handle not only the packet header but also the payload and consider as well the IPv6 packets too. In addition, it should be more comprehensive to cover other IT infrastructure such as in the smart grid. Finally, the framework can be further developed to collaborate with other frameworks or to operate as a distributed framework.

The prevention technique is considered as an important open issue in this area. The main target of this dissertation is achieving a comprehensive IDS framework that is able to handle massive data flow and detect new attacks. Therefore, investigating the prevention technique has been partially skipped. To achieve this goal, I suggest investigating the well-known prevention techniques in the area of IDS. The prevention technique should verify each anomaly connection and mark the IP addresses to share and distribute them with other applications or hardware. In addition, it should continuously examine the state of an anomaly and change it to normal, critical connection upon certain conditions or rules.
BIBLIOGRAPHY

Last access of all websites was on February 28th 2014.


[34] OSSEC. [Online]. http://www.ossec.net/


http://www.juniper.net/us/en/

http://www.enterasys.com/


[143] Y. Jin et al., "A modular machine learning system for flow-level traffic classification in large networks," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, no. 1, p. 34, 2012.


APPENDIX A – ATTACK TYPES IN DARPA 98/99 DATASET

<table>
<thead>
<tr>
<th>Attack type</th>
<th>Attack name</th>
<th>1998 present</th>
<th>1999 present</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denial of Service Attacks</td>
<td>Apache2</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>arppoison</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Back</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Crashtis</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>dosnuke</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Land</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Mailbomb</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>SYN Flood (Neptune)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Ping of Death (POD)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Process Table</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>selfping</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Smurf</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>sshprocessstable</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Syslogd</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>tcpreset</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Teardrop</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Udpstorm</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>anypw</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>casesen</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Eject</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### User to Root Attacks

<table>
<thead>
<tr>
<th>Tool</th>
<th>Yes</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fbconfig</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Fdformat</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Loadmodule</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ntfsdos</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Perl</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Ps</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>sechole</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Xterm</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>yaga</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Dictionary</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Ftpwrite</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Guest</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Http_tunnel</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Imap</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Named</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ncftp</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>netbus</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>netcat</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Phf</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ppmacro</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sendmail</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>sshTrojan</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Xlock</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Xsnoop</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>insidesniffer</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>lpsweep</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ls_domain</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Mscan</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NTinfoscan</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Nmap</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>queso</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>resetscan</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Saint, Satan</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### APPENDIX B – FEATURES IN DARPA 98/99 DATASET

<table>
<thead>
<tr>
<th>Feature category</th>
<th>No.</th>
<th>Feature name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic features of individual connections</td>
<td>1</td>
<td>duration</td>
<td>Length (number of seconds) of the connection</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>protocol_type</td>
<td>Type of the protocol, e.g. tcp, udp, etc</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>service</td>
<td>Network service on the destination, e.g., http, telnet, etc.</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>flag</td>
<td>Normal or error status of the connection</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>src_bytes</td>
<td>Number of data bytes from source to destination</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>dst_bytes</td>
<td>Number of data bytes from destination to source</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>land</td>
<td>1 if connection is from/to the same host/port; 0 otherwise</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>wrong_fragment</td>
<td>Number of <code>wrong</code> fragments</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>urgent</td>
<td>Number of urgent packets</td>
</tr>
<tr>
<td>Traffic features computed using a two second time window:</td>
<td></td>
<td>count *</td>
<td>No. of connections to same host as the current connection in past 2 secs</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>serror_rate *</td>
<td>% of connections that have <code>SYN</code> errors</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>rerror_rate *</td>
<td>% of connections that have <code>REJ</code> errors</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>same_srv_rate *</td>
<td>% of connections to the same service</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>diff_srv_rate *</td>
<td>% of connections to different services</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>srv_count **</td>
<td>No. of connections to same service as the current connection in past 2 secs</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>srv_serror_rate **</td>
<td>% of connections that have <code>SYN</code> errors</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>srv_rerror_rate **</td>
<td>% of connections that have <code>REJ</code> errors</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>srv_diff_host_rate **</td>
<td>% of connections to different hosts</td>
</tr>
<tr>
<td>Traffic features Computed using a Hundred-connection window:</td>
<td></td>
<td>dst_host_count *</td>
<td>No. of connections to same host as the current connection in past 2 secs</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>dst_host_serror_rate *</td>
<td>% of connections that have <code>SYN</code> errors</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>dst_host_rerror_rate *</td>
<td>% of connections that have <code>REJ</code> errors</td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>dst_host_same_srv_rate *</td>
<td>% of connections to the same service</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>dst_host_diff_srv_rate *</td>
<td>% of connections to different services</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>dst_host_srv_count **</td>
<td>% of connections to same service as the current connection in past 2 secs</td>
</tr>
</tbody>
</table>
** = same-service cxn

<table>
<thead>
<tr>
<th>Feature ID</th>
<th>Feature Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td><code>dst_host_srv_serror_rate</code> % of connections that have ``SYN'' errors</td>
</tr>
<tr>
<td>26</td>
<td><code>dst_host_srv_rerror_rate</code> % of connections that have ``REJ'' errors</td>
</tr>
<tr>
<td>27</td>
<td><code>dst_host_srv_diff_host_rate</code> % of connections to different hosts</td>
</tr>
<tr>
<td>28</td>
<td><code>dst_host_same_src_port_rate</code> % of connections to same service port</td>
</tr>
</tbody>
</table>

Content features within a connection suggested by domain knowledge

<table>
<thead>
<tr>
<th>Feature ID</th>
<th>Feature Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td><code>hot</code> number of ``hot'' indicators continuous</td>
</tr>
<tr>
<td>30</td>
<td><code>num_failed_logins</code> number of failed login attempts</td>
</tr>
<tr>
<td>31</td>
<td><code>logged_in</code> 1 if successfully logged in; 0 otherwise</td>
</tr>
<tr>
<td>32</td>
<td><code>num_compromised</code> number of ``compromised'' conditions</td>
</tr>
<tr>
<td>33</td>
<td><code>root_shell</code> 1 if root shell is obtained; 0 otherwise</td>
</tr>
<tr>
<td>34</td>
<td><code>su_attempted</code> 1 if ``su root'' command attempted; 0 otherwise</td>
</tr>
<tr>
<td>35</td>
<td><code>num_root</code> number of ``root'' accesses</td>
</tr>
<tr>
<td>36</td>
<td><code>num_file_creations</code> number of file creation operations</td>
</tr>
<tr>
<td>37</td>
<td><code>num_shells</code> number of shell prompts</td>
</tr>
<tr>
<td>38</td>
<td><code>num_access_files</code> number of operations on access control files</td>
</tr>
<tr>
<td>39</td>
<td><code>num_outbound_cmds</code></td>
</tr>
<tr>
<td>40</td>
<td><code>is_hot_login</code> 1 if the login belongs to &quot;hot&quot; list; 0 otherwise</td>
</tr>
<tr>
<td>41</td>
<td><code>is_guest_login</code> 1 if the login is a ``guest'' login; 0 otherwise</td>
</tr>
</tbody>
</table>

Attack type / normal

<table>
<thead>
<tr>
<th>Feature ID</th>
<th>Feature Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td><code>class</code> Normal traffic or specific attack type</td>
</tr>
</tbody>
</table>
List of Publications

Published


Submitted