Hybrid Intrusion Detection System
Djabeur Mohamed Seifeddine Zekrifa

To cite this version:

HAL Id: tel-01584217
https://tel.archives-ouvertes.fr/tel-01584217
Submitted on 15 Sep 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Hybrid Intrusion Detection System

by

Zekrifa Djabeur Mohamed Seifeddine

M.Sc. in Information Technology Engineering, University of Chicago, 2012

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Ph.D. of Computer Science

In the Graduate Academic Unit of Faculty of Computer Science

Supervisor: Jui yon Lee, Ph.D.,
Faculty of Computer Science, UniSA

Examining Board: John DeDourek, Professor, Computer Science
Rod Cooper, Professor, Computer Science
Brent Petersen, Ph.D.,
Department of Electrical and Computer Engineering

External Examiner: Alex Aiken Stanford professor,
Institute for Information Systems Engineering,

This thesis is accepted by the
Dean of Graduate Studies

THE UNIVERSITY OF SOUTH AUSTRALIA
Zekrifa Djabeur Mohamed Seifeddine

Jun 19, 2014
Dedication

This thesis is dedicated to my parents who have supported me all the way since the beginning of my studies, without whose love, support and inspiration I would never have made it this far.
Abstract

With the enormous growth of computer networks and the huge increase in the number of applications that rely on it, network security is gaining increasing importance. Moreover, almost all computer systems suffer from security vulnerabilities which are both technically difficult and economically costly to be solved by the manufacturers. Therefore, the role of Intrusion Detection Systems (IDSs), as special-purpose devices to detect anomalies and attacks in a network, is becoming more important.

Traditionally, intrusion detection techniques are classified into two categories: misuse (signature-based) detection and anomaly detection. However, some researchers have recently proposed the idea of hybrid detection to reap the advantage of misuse detection by having a high detection rate on known intrusions as well as the ability of anomaly detectors in detecting brand-new attacks. Despite the inherent potential of hybrid detection, there are still two important issues that highly affect the performance of these hybrid systems. First, anomaly-based methods cannot achieve an outstanding performance without a comprehensive labeled and up-to-date training set with all
different attack types, which is very costly and time-consuming to create if not impossible. Second, efficient and effective fusion of several detection technologies becomes a big challenge for building an operational hybrid intrusion detection system.

With respect to the aforementioned shortcomings, in this thesis, we introduce an adaptive hybrid network-based intrusion detection system to recognize malicious network activities and report them to the system administrator. The proposed detection system is based on a multi-layer model which consists of three processing layers: 1) Packet Analysis; 2) Intrusion Detection; and 3) Security Information and Event Management (SIEM).

Packet Analysis layer contains two important modules, namely Flow Analyzer and Traffic Classification which are responsible for grouping the packets and labeling them with the proper application name, respectively. Analyzed packets will be then forwarded to the Intrusion Detection layer for further investigation. Depending on their application labels, each flow will be treated in a specific detection module and will trigger an alert if identified as malicious activity by the hybrid detection system. These alerts are fed into the Security Information and Event Management (SIEM) layer to notify the administrator of potential breaches.
Acknowledgements

I would like to express my deepest and sincere gratitude to my supervisor, Dr. Jui yon Lee, for kindly providing me with a good opportunity to work with him in the area of Network Security. I deeply appreciate his guidance, motivation and support during the preparation of this thesis.

I also wish to express my sincere appreciation to Dr. Wei Lu, Dr. Ebrahim Bagheri, Dr. Natalia Stakhanova, Mr. Ali Shiravi and Mr. Hadi Shiravi for their valuable advice, comments, discussions and guidance during the course of this thesis.

I would like to thank the members of my committee and examining board for taking the time and providing me with constructive comments.
Table of Contents

Dedication ii
Abstract iii
Acknowledgments v
Table of Contents x
List of Tables xii
List of Figures xiv

1 Introduction 1
  1.1 Contributions ................................................................. 3
  1.2 Thesis Organization.......................................................... 5

2 Literature Review 7
  2.1 Anomaly Detection Systems............................................... 7
    2.1.1 Data Labels................................................................. 8
3.5.1 Confusion Matrix................................................................. 47
3.5.2 ROC Curves........................................................................... 49
3.5.3 Precision, Recall and F-Measure........................................... 51
3.5.4 Positive and Negative Predictive Values......................... 52
3.5.5 B-ROC Curves.................................................................. 54
3.5.6 Survey Statistics on the Evaluation Measures.............. 54
3.6 Summary.................................................................................56

4 Proposed Framework 57
4.1 Overall View ......................................................................... 59
  4.1.1 Packet Analysis.............................................................. 59
    4.1.1.1 Flow Analyzer .......................................................... 60
    4.1.1.2 Traffic Classification ............................................... 62
  4.1.2 Intrusion Detection .......................................................... 62
  4.1.3 Security Information and Event Management .............. 63
4.2 Traffic Classification Module ............................................... 63
  4.2.1 Weighted Unigram Model ............................................. 67
  4.2.2 Problem Formulation .................................................... 71
4.3 Intrusion Detection Module .................................................. 78
  4.3.1 Anomaly-based Detector .............................................. 80
  4.3.2 Signature-based Detector .............................................. 80
  4.3.3 Proposed Hybrid Detector ........................................... 81
  4.3.4 Fusing Algorithm .......................................................... 82
4.3.4.1 Dempster's Rule of Combination ....................... 84
4.3.4.2 Dempster's Rule of Combination for Binary
Variables ..................................................... 85
4.3.4.3 Quantification Method .............................. 86
4.4 Performance Analysis ........................................ 89
4.5 Summary .......................................................... 91

5 Data Set Preparation ........................................ 93
5.1 NSL-KDD Data Set .............................................. 93
   5.1.1 Redundant Records ..................................... 94
   5.1.2 Level of Difficulty ..................................... 95
   5.1.3 Our Solution ............................................. 100
5.2 Benchmark Data Sets ......................................... 102
   5.2.1 Network Architecture .................................. 104
   5.2.2 Data Generation ........................................ 106
   5.2.3 Attack Scenarios ....................................... 109
      5.2.3.1 Scenario 1: Infiltrating the network from the
              inside ..................................................... 112
      5.2.3.2 Scenario 2: HTTP Denial of Service ............ 115
      5.2.3.3 Scenario 3: Distributed Denial of Service us-
              ing an IRC Botnet ...................................... 116
      5.2.3.4 Scenario 4: Dictionary Attack against SSH . . 117
   5.2.4 Capturing Traffic ....................................... 118
6 Framework Evaluation

6.1 Phase 1: Evaluation of Traffic Classification Module..............121
6.2 Phase 2: Evaluation of Intrusion Detection Module...............125
6.3 Phase 3: Overall Performance of Hybrid Detection System . . 136
6.4 Summary .............................................................................143

7 Conclusions and Future Work

7.1 Future Work...........................................................................147

Bibliography

A Dempster Rule of Combination

A.1 Dempster Rule for Binary Variables........................................166

Vita
List of Tables

3.1 The details of the surveyed papers. ................................................................. 35
3.2 Publication venues of surveyed papers (top five venues). .................. 36
3.3 The detailed statistics of the applied data sets. ................................. 38
3.4 The detailed statistics of the data processing techniques. .............. 42
3.5 The detailed statistics of the experiments setup............................... 44
3.6 Confusion matrix ......................................................................................... 48

4.1 Payload signatures of some typical network applications ............. 65
4.2 Applied flow-based features ........................................................................ 81

5.1 Statistics of redundant records in the KDD train set .................... 94
5.2 Statistics of redundant records in the KDD test set............................. 95
5.3 Statistics of randomly selected records from KDD train set .... 100
5.4 Statistics of randomly selected records from KDD test set .... 101
5.5 Specification of testbed workstation ........................................................ 104
5.6 Specification of testbed servers ................................................................. 106

6.1 Workload of the ISCX network over an hour ............................... 122
6.2 Workload of the ISP network over an hour ...........................................122
6.3 Accuracy of the proposed method for the ISCX and ISP networks122
6.4 Distribution of normal vs. intrusive flows .............................................126
6.5 Performance of detectors on Day 1 ..................................................... 131
6.6 Performance of detectors on Day 2 ..................................................... 132
6.7 Performance of detectors on Day 3 ..................................................... 133
6.8 Performance of detectors on Day 4 ..................................................... 133
6.9 Performance of detectors on Day 5 ..................................................... 134
6.10 Performance of detectors on Day 6 .................................................. 134
6.11 Performance of detectors on Day 7 ................................................... 135
6.12 Distribution of web-based traffic ....................................................... 137
6.13 Web-based application groups .......................................................... 139
6.14 Performance of detectors in phase 3 on Day 1 ................................ 139
6.15 Performance of detectors in phase 3 on Day 2 ................................ 139
6.16 Performance of detectors in phase 3 on Day 3 ................................ 140
6.17 Performance of detectors in phase 3 on Day 4 ................................ 140
6.18 Performance of detectors in phase 3 on Day 5 ................................ 141
6.19 Performance of detectors in phase 3 on Day 6 ................................ 141
6.20 Performance of detectors in phase 3 on Day 7 ................................ 142
6.21 Comparison of detection rates between phase 2 and 3 ................. 142
6.22 Comparison of false alarm rates between phase 2 and 3 .......... 142
List of Figures

3.1 Data sets usage in the surveyed works.........................................................37
3.2 The whole data space in binary classification. (This figure is adapted from [1]) ................................................................. 49
3.3 Data sets usage in the surveyed works.........................................................55

4.1 The proposed framework .............................................................................60
4.2 Unigram distribution of an HTTP flow.........................................................67
4.3 Average unigram distribution of source payload ...........................................69
4.4 The genetic algorithm-based method for finding the optimal weight vector. ...........................................................................76
4.5 General structure of the hybrid detector .......................................................83
4.6 Evaluated Hybrid Detection System.............................................................90

5.1 The distribution of #successfulPrediction values for the KDD data set records.................................................................97
5.2 The distribution of #successfulPrediction values for the KDD data set records.................................................................98
5.3 The performance of the selected learning machines on KDDTest 98
5.4 The performance of the selected learning machines on KDDTest+ 99
5.5 The performance of the selected learning machines on KDDTest−21 99
5.6 Data set statistics ........................................................................................................... 105
5.7 Data set statistics ........................................................................................................... 107
5.8 Number of flows seen per second during the capturing period 109
5.9 Data set statistics ........................................................................................................... 109
6.1 Fitness value vs. generation number calculated for the ISCX network ........................................................................................................... 124
6.2 Fitness value vs. generation number calculated for the ISP network ........................................................................................................... 124
6.3 Distribution of normal and intrusive flows ......................................................... 127
6.4 Overall performance of the detectors ............................................................... 128
6.5 Overall performance of the detectors ............................................................... 136
6.6 Hybrid Detection System ...................................................................................... 138
Chapter 1

Introduction

With the enormous growth of computer networks usage and the huge increase in the number of applications running on top of it, network security is becoming an important issue. Moreover, almost all computer systems suffer from security vulnerabilities which are both technically difficult and economically costly to be solved by the manufacturers. Therefore, the role of the Intrusion Detection Systems (IDSs), as special-purpose devices to detect anomalies and attacks in the network, is becoming more important.

Generally, IDSs are using two fundamental approaches. The first one is misuse detection, also called signature-based detection. In this type of IDSs, the search for evidence of attacks is based on knowledge accumulated from known attacks. This knowledge is represented by attacks’ signatures which are patterns or sets of rules that can uniquely identify an attack. Being designed based on the knowledge of the past intrusions or known vulnerabilities,
misuse-based IDSs are also called knowledge-base detection. The advantages of knowledge-based approaches are that they have a very good accuracy and very low false alarm rate. Furthermore, the analysis is detailed meaning that there is enough information about the type of detected attacks; thus, it is easier for the system administrator to take preventive and corrective action. On the contrary, drawbacks include the difficulty of gathering the required information on the known attacks and keeping it up-to-date with new vulnerabilities. Moreover, misuse-base IDSs are not complete, i.e., they do not have the ability to detect all types of attacks, especially new ones and those involving an abuse of privileges.

The second type of IDSs is anomaly detection or behavior-based detection. In this approach models of legitimate activities are built based on the normal data, and then the deviation from the normal model will be considered as an attack or anomaly. The main advantage of this approach over misuse detection is that it can detect attempts to exploit new and unforeseen vulnerabilities. It also can help detect “abuse of privileges” types of attacks that do not actually involve exploiting any security vulnerability. However, this approach has its own shortcomings. The main reported problem is high false alarm rate which is caused by two kinds of problems. The first one is the lack of a training data set that covers all the legitimate areas, and the other one is that abnormal behavior is not always an indicator of intrusions. It can happen as a result of factors such as policy changes or offering of new services by a site. Besides the aforementioned problem (high false alarm rate),
behavior-based approaches suffer from some other shortcomings as well:

- They cannot explain why a detected event is an attack
- They cannot provide an explanation for the type of attack
- They cannot provide information to respond to the attack

In order to overcome these challenges, and keep the advantages of misuse detection, some researchers have proposed the idea of hybrid detection. This way, the system will achieve the advantage of misuse detection to have a high detection rate on known attacks as well as the ability of anomaly detectors in detecting unknown attacks. Despite the inherent potential of hybrid detection, there are still two important issues that highly affect the performance of these hybrid systems. First, anomaly-based methods cannot achieve an outstanding performance without a comprehensive labeled and up-to-date training set with all different attack types, which is very costly and time-consuming to create if not impossible. Second, efficient and effective fusion of several detection technologies becomes a big challenge for building an operational hybrid intrusion detection system.

### 1.1 Contributions

In this thesis, we introduce an adaptive hybrid network-based intrusion detection system to recognize malicious network activities and report them to the system administrator. The main contributions of this thesis are as follows:
• We conducted a comprehensive survey on the current state of the experimental practice in the area of anomaly-based intrusion detection and surveyed 276 studies in this area published during the period of 2000-2008 to identify the common pitfalls in the evaluation and comparison of the intrusion detection systems.

• We propose a multi-layer framework for an efficient detection of network intrusion that overcomes the existing shortcomings of intrusion detection systems. This framework consists of three processing layers: 1) Packet Analysis; 2) Intrusion Detection; and 3) Security Information and Event Management (SIEM).

• An online traffic classification method is proposed based on the weighted unigram distribution of the payloads. A genetic algorithm based scheme is then employed to find appropriate weights in order to achieve a higher accuracy.

• We provide a novel hybrid intrusion detection system which can evolve as the network behavior changes. In addition, we have proposed an efficient fusing algorithm that is able to model the uncertainty attached to each detection method.

• Through a statistical analysis on the KDDCUP99 data set, we found two important issues which highly affects the performance of evaluated systems, and results in a very poor evaluation of anomaly detection approaches. To solve these issues, we have proposed a new data set, NSL-KDD, which consists of selected records of the complete KDD
data set and does not suffer from any of identified drawbacks.

• To overcome the shortcomings of available data sets in reflecting real life conditions, we have prepared a data set of full network traces, including packet payloads, which is publicly available to researchers through our website. The ultimate goal is to prepare a public data set for the evaluation of network-based IDSs through imitating our centers network and to conduct several attack scenarios against it.

1.2 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 provides a comprehensive review of three important areas of research that have significant implications for the proposed framework, namely anomaly detection systems, hybrid intrusion detection systems, and network traffic classification. Chapter 3 introduces the drawbacks to the existing evaluation methods of IDSs and provides some solutions for obtaining more realistic results.

In Chapter 4, we propose an adaptive hybrid intrusion detection system to overcome the main shortcomings of the existing IDSs. The proposed detection system is based on a multi-layer model which consists of three processing layers: 1) Packet Analysis; 2) Intrusion Detection; and 3) Security Information and Event Management (SIEM).

Chapter 5 provides two data sets to address some inherent problems in DARPA and KDD data sets which are widely used as the only publicly
available data sets for network-based anomaly detection systems. Chapter 6 presents the experimental results of this thesis, followed by a detailed discussion on the provided results.

The thesis is finally concluded in Chapter 7 which summarizes the current work and presents several suggestions for further research in the area of intrusion detection.
Chapter 2

Literature Review

In this chapter, we review three important areas of research that have significant implications for the proposed framework. First, we explain anomaly detection systems in more detail. In the second section, we review existing hybrid intrusion detection systems. Network traffic classification methods will be discussed in the third section. Finally, we provide some information about the existing evaluation data sets for network-based intrusion detection systems (NIDS).

2.1 Anomaly Detection Systems

As mentioned in the previous chapter, the idea of anomaly detection is to build models of legitimate activities based on the normal data, and then any deviation from the normal model will be considered as an attack or anomaly.
To this end, anomaly detectors basically consist of two phases: a training phase and a testing phase. In the former, the normal model is automatically built based on the training data set applying machine learning techniques; in the latter, the learned model is applied to the new testing instances.

The training data set contains a collection of data instances each of which can be described using a set of attributes (features) and the associated labels. The attributes can be of different types such as categorical or continuous. The nature of attributes determines the applicability of anomaly detection techniques. For example, distance-based methods are initially built to work with continuous features and usually do not provide satisfactory results on categorical attributes.

The labels associated with data instances are usually in the form of binary values, i.e., normal and anomalous. In contrast, some researchers have employed different types of attacks such as DoS, U2R, R2L and Probe rather than the anomalous label. This enables learning techniques to provide more information about the types of anomalies. However, experimental results show that current learning techniques are not precise enough to recognize such type of anomalies.

2.1.1 Data Labels

Since labeling is often done manually by human experts, obtaining an accurate labeled data set which is representative of all types of behaviors is rather costly. As a result, based on the availability of the labels, three operating
modes are defined for anomaly detection techniques:

**Supervised Anomaly Detection:** supervised methods, also known as classification methods, need a labeled training set containing both normal and anomalous samples to build the predictive model. Theoretically, supervised methods provide better detection rate compared to semi-supervised and unsupervised methods since they have access to more information. However, there exist some technical issues which make these methods less accurate than they are assumed to be. The first one is the lack of a training data set that covers all the legitimate areas. Furthermore, obtaining accurate labels is very challenging. Moreover, the training sets usually contain some noise which results in higher false alarm rates. As examples of supervised learning methods we can name Neural Networks, Support Vector Machines (SVM), k-Nearest Neighbors, Bayesian Networks, and Decision Trees.

**Semi-supervised Anomaly Detection:** Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Generally speaking, semi-supervised methods employ unlabeled data in conjunction with a small amount of labeled data. As a result, they highly reduce the labeling processing cost, while maintaining the high performance of supervised methods. In the area of anomaly detection, however, semi-supervised learning methods assume that the training data has labeled instances for only the normal class. This way, they are more practicable than supervised methods to operate in real networks since they do not require any labels for the anomaly
class. One-class SVM is one of the most well-known classifiers of this type which makes a discriminative boundary around the normal instances, and any test instance that does not fall within the learned boundary is declared as anomalous. Although the typical approach in semi-supervised techniques is to model the normal behavior, there exist a limited number of anomaly detection techniques that assume availability of the anomalous instances for training [2, 3]. Such techniques are not widely used since obtaining a training set which covers every possible anomalous behavior is almost impossible.

**Unsupervised Anomaly Detection:** unsupervised techniques do not require training data. Instead, this approach is based on two basic assumptions [4]. First, it assumes that the majority of the network connections represent normal traffic and that only a very small percentage of the traffic is malicious. Second, it is expected that malicious traffic is statistically different from normal traffic. Based on these two assumptions, data instances that build groups of similar instances and appear very frequently are supposed to represent normal traffic, while instances that appear infrequently and are significantly different from the majority of the instances are considered to be suspicious.

While supervised methods are very dependent on the labeled training data which are usually error-prone, time consuming and costly, unsupervised learning techniques avoid such complications by not using the labeled training data and any prior knowledge of attacks or normal instances. Instead, they partition the data into normal operations and anomalies using statistical models.
Applying such techniques to the problem of anomaly detection is one of the possible avenues to build large, reliable anomaly detection systems without the need for extensive and costly manual labeling of instances.

2.1.2 Output Format

In addition to the type of training set, anomaly detection methods can be categorized based on the method they use to report the anomalies. Typically, there are three types of output to report anomalies, namely scores, binary labels, and labels.

**Scores**: in this technique, anomaly detectors will assign a numeric score to each instance which indicates how likely it is that the test instance is an anomaly. The advantage of this technique is that the analyst can rank the malicious activities, set a threshold, and select the most significant ones. Bayesian networks such as Naive Bayes are good examples of this kind of methods in which they provide the administrator with the calculated probabilities.

**Binary Labels**: some of the anomaly detection techniques such as Decision Trees are not able to provide scores for the instances; instead they label the test instances as either anomalous or normal. This approach can be considered as a special case of labeling techniques.

**Labels**: anomaly detection techniques in this category assign a label to each test instance. In this approach, usually there is one label for normal traffic. For the anomalies, however, there are designated labels showing the types of
anomalies. For example, some methods apply the labels normal, DoS, Probe, U2R, and R2L to show the general category of the detected attacks. Most of the non-scoring learners such as Decision Tree methods can be applied for this purpose. The only requirement is that there should be enough samples from each class of labels.

2.1.3 Data Collection

Regardless of the types of data labels and output, anomaly intrusion detection systems (AIDS) can also be classified based on the source of data being acquired and analyzed. Generally speaking, AIDSs are divided into three main categories based on the locus of data which is used to learn normal behavior.

Application-based AIDS: in this approach, sources of data are application log files or network traffic. The research by Kruegel et al. [5] is a good example of application level AIDS. It presents an anomaly detection system that detects Web-based attacks using the Web server log files as input and then produces an anomaly score for each web request.

Host-based AIDS: these techniques analyze activities on a protected host by the acquisition of data from a source that resides on that host. The main data sources of this type of AIDSs are audit-logs and system-calls. In the first method, i.e., audit-logs, the system uses information about a set of events that the operating system (OS) is generating, while in the case of system-calls, the system independently monitors the behavior of each user-critical
application that runs on the operating system.

**Network-based AIDS:** NIADSs are designed to scan and analyze network packets and are not restricted to one host. The applicability of network-based anomaly detectors to large networks, has made them very popular and attracted many researchers to focus their work on network-based anomaly detection.

### 2.1.4 Techniques Used in Anomaly Detection

From the time anomaly detection was formalized by Denning [6], different methods have been proposed for anomaly detection. In the following we briefly explain some of the methods applied for network-base AIDSs:

**Bayesian Networks:** Bayesian Network is a probabilistic graphical model that represents a set of variables and their probabilistic independencies. Bayesian networks, literally, are directed acyclic graphs whose nodes represent variables, and whose edges encode conditional dependencies between the variables [7]. They have been applied in anomaly detection in different ways; for example, Valdes et al. [8] developed an anomaly detection system that employed Naive Bayes, which is a two-layer Bayesian network that assumes complete independency between the nodes.

**Genetic Algorithms:** a genetic algorithm (GA) is a search technique used to find exact or approximate solutions to optimization and search problems. Because of their flexibility and robustness as a global search method, genetic algorithms have been applied in anomaly detection in different ways. Some
approaches [9] have used genetic algorithms directly for the classification of instances, while others [10] have applied this technique for feature selection. **Neural Networks:** a Neural network is a network of computational units that jointly implement complex mapping functions. Initially the network is trained with a labeled data set. Testing instances are then fed into the network to be classified as either normal or anomalous. Support Vector Machines (SVM) is an example of Neural Network technique which is widely used in anomaly detection [11].

**Immune System Approach:** the natural immune system uses a variety of evolutionary and adaptive mechanisms to protect organisms from foreign pathogens and misbehaving cells in the body. Artificial immune systems (AIS) seek to capture some aspects of the natural immune system in a computational framework, either for the purpose of modeling the natural immune system or for solving engineering problems. In either form, a fundamental problem solved by most AIS can be thought of as learning to discriminate between “self” (the normally occurring patterns in the system being protected, e.g., the body) and “nonself” (foreign pathogens, such as bacteria or viruses, or components of self that are no longer functioning normally). Since the role of immune systems in human body is similar to the role of intrusion detection systems, AISs have widely been applied in anomaly detection [12].

**Inductive Rule Generation Algorithms:** the most famous technique in this category is decision tree that is a predictive model mapping observations about an item to conclusions about its target value.
**Clustering:** these techniques are usually based on two important assumptions [4]. First, the majority of the network connections represent normal traffic and that only a very small percentage of the traffic is malicious. Second, malicious traffic is statistically different from normal traffic. If these two assumptions hold, anomalies can be detected based on the cluster size, i.e., large clusters correspond to normal data, and the rest correspond to attacks.

**Outlier Detection:** this approach is based on the idea of semi-supervised learning in which the system is trained based on normal data, and then any instances that do not fit in the normal profile will be considered as anomaly.

### 2.2 Hybrid Intrusion Detection Systems

Despite the great capability of anomaly detection systems in detecting unknown or zero-day vulnerabilities, these methods suffer from a major deficiency namely their high false alarm rate. This is mainly caused for two reasons. The first is the lack of a training data set that covers all the legitimate areas, and the second is that abnormal behavior is not always an indicator of intrusions. The abnormal behavior can also happen as a result of factors such as policy changes or the offering of new services by a site.

In order to overcome these challenges and keep the advantages of misuse detection, some researchers have proposed the idea of hybrid detection. This will help the system take advantage of misuse detection to have a high detection rate on known attacks as well as the ability of anomaly detectors
in detecting unknown attacks. According to this fusion approach, current hybrid IDSs can be divided into two categories: 1) sequence-based, in which either anomaly detection or misuse detection is applied first, and the other one is applied next; 2) parallel-based, in which multiple detectors are applied in parallel, and the final decision is made based on multiple output sources. The most common type of hybrid system is to combine a misuse detection and an anomaly detection together. In such a hybrid system, the signature detection technique detects known attacks, and the anomaly detection technique detects novel or unknown attacks.

In [13], Tombini et al. applied an anomaly detection approach to create a list of suspicious items. Then a signature detection approach is employed to classify these suspicious items into three categories, namely false alarms, attacks, and unknown attacks. The approach is based on an assumption that a high detection rate can be achieved by the anomaly detection component because missed intrusions in the first step will not be found by the follow-up signature detection component.

Zhang et al. proposed a hybrid IDS combining both misuse detection and anomaly detection components, in which a random forest algorithm was applied firstly in the misuse detection module to detect known intrusions [14]. The outlier detection provided by the random forest algorithm was then utilized to detect unknown intrusions. Evaluations with a part of the KD-DCUP’99 data set showed that their misuse detection module generated a high detection rate with a low false positive rate, and at the same time the
anomaly detection component had the potential to find novel intrusions.

In [15], Peng et al. proposed a two-stage hybrid intrusion detection and visualization system that leverages the advantages of signature-based and anomaly detection methods. It was claimed that their hybrid system could identify both known and unknown attacks on system calls. However, evaluation results for their system are missing in the paper. The work is more like an introduction on how to apply a multiple stage intrusion detection mechanism for improving the detection capability of an IDS.

Similar to [14], Depren et al. proposed a novel hybrid IDS system consisting of an anomaly detection module, a misuse detection module, and a decision support system [16]. The decision support system was used to combine the results of the two previous detection modules. In the anomaly detection module, a Self-Organizing Map (SOM) structure was employed to model normal behavior and any deviation from the normal behavior would be classified as an attack. In the misuse detection module, a decision tree algorithm was used to classify various types of attacks. The final system was evaluated with the 1999 KDDCUP intrusion detection data set, and experimental results showed that the proposed hybrid approach gave better performance over individual approaches.

Based on the idea of combining the advantages of low false positive rate of signature based IDS and the ability of anomaly intrusion detection system (AIDS) for detecting new or unknown attacks, Hwang et al. proposed and reported a new experimental hybrid intrusion detection system (HIDS) in [17].
Mining anomalous traffic episodes from Internet connections, the anomaly detector was able to identify some anomalies beyond the capabilities of well-known signature based Snort or Bro IDSs. Moreover, a weighted signature generation scheme was developed to combine AIDS with Snort through modeling signatures from detected anomalies. The HIDS scheme was evaluated with real Internet trace data mixed with 10 days of the 1999 DARPA intrusion detection data set. The obtained results showed that HIDS achieves a 60% detection rate, compared with 30% and 22% detection rate acquired by the SNORT and Bro systems, respectively.

In [18] Hwang et al. proposed a hybrid intrusion detection system combining a signature-based method, Snort, and an anomaly detection system. In contrast to the other hybrid IDSs, their approach only relies on Snort to generate the alerts, and the anomaly detection is only used to automatically generate Snort. To this end, normal traffic is passed to the anomaly system to build a normal profile of frequent episode rules (FER). Having done with the training phase, the real traffic will be fed to the system. FERs generated from the real traffic will be compared to the normal profile and considered as suspicious if it does not match any of the FREs in the normal profile. When the matched rule occurs beyond the threshold, it will be reported as an anomaly and the system will automatically add the rule to the Snort.

Instead of combining signature detection techniques and anomaly detection techniques, some other hybrid systems fuse multiple anomaly detection systems according to some specific criteria considering that the detection capa-
pability for each anomaly detection technique is different. The main goal of such a hybrid system is to reduce the large number of false alerts generated by current anomaly detection approaches and at the same time keep an acceptable detection rate. Some examples of such research work are discussed in [19, 20, 21].

In [19], Xiang et al. proposed a multi-level hybrid classifier utilizing a combination of tree classifiers and clustering algorithms. One of the most interesting ideas of this work is that they fuse both supervised learning (tree classifiers) and unsupervised learning (clustering) techniques. Although the supervised learning technique needs a clear label for training, it was claimed in the paper that unsupervised learning might play an essential role on improving the detection rate. Using the KDDCUP 1999 data set, they evaluated their approach and compared it with other popular approaches (i.e. MADAM ID and 3-level tree classifiers). Evaluation results showed that their hybrid approach was very efficient in detecting intrusions with an extremely low false negative rate of 3.37%, while keeping an acceptable level of false alarm rate of 9.1%.

Sabhnani and Serpen compared and evaluated nine well known pattern recognition and machine learning algorithms with the 1999 KDDCUP intrusion detection data set in [20]. Based on their performance, they selected three of them for obtaining an optimal detection result, including Multilayer Perceptron (MLP) for probing attacks, K-means for DoS attacks as well as U2R, and Gaussian classifier for R2L attacks. Evaluation results with the union of
these three algorithms showed that the hybrid detection system could achieve a better performance than the 1999 KDD Cup’s winner.
In [21], Shon and Moon proposed a new SVM approach (i.e. Enhanced SVM) by combining two existing SVM techniques, namely soft-margin SVM and one-class SVM, in order to provide unsupervised learning capability and to achieve a low false alarm rate. A set of additional techniques had been used to improve the performance of their approach, including creating normal packets profile with Self-Organized Feature Map (SOFM), filtering packets based on Passive TCP/IP Fingerprinting (PTF), selecting features using Genetic Algorithms (GAs) and using the flow of packets based on temporal relationships in data preprocessing. The experimental evaluation with DARPA intrusion detection data sets and a live data set captured from a real network showed that the proposed enhanced SVM approach obtained a low false positive rate similar to that of some real network IDS without requiring pre-labeled data.

2.3 Traffic Classification

Early common techniques for identifying network applications rely on the association of a particular port with a particular protocol [22]. Such a port number based traffic classification approach has been proved to be ineffective due to the emergence of new applications that use dynamic port numbers or hide themselves using encapsulation methods. To overcome these issues, there have recently been significant contributions towards traffic classifica-
tion. Having done a thorough study of research in traffic classification, we have classified the existing approaches into two categories, namely signature-based and statistical. In the remainder of this section, we introduce each category with some typical examples and discuss the limitations for the existing techniques.

2.3.1 Signature-Based Traffic Classifier

An alternative to traditional port number based application classification is to inspect the content of payload and seek the deterministic character strings for modeling the applications. In [23], Gummadi et al. develop a signature model for KaZaA workload characterization through analyzing a 200-day trace of over 20 terabytes of Kazaa P2P traffic collected on a campus network. In [24], Sen et al. analyze the application layer protocols and then generate the signatures of a few P2P applications. Although the protocol semantic analysis improves the accuracy of signatures, it makes the real-time analysis of the backbone traffic impossible since the underlying assumption is that every packet is being inspected. In their consequent work [25], Sen et al. examine available specification and packet-level traffic traces for constructing application layer signatures, and then based on these signatures, P2P traffic are filtered and tracked on high-speed network links. Evaluation results show that their approach obtains less than 5% false positives and false negatives. Obtained from unencrypted traditional applications, signatures can also be extracted from the encrypted traffic. Ehlert et al. examine the hexadecimal
patterns in the Skype packet traces during the initial communication setup phase [26]. Moreover, in [27], Bernaille et al. characterize the application signatures during the early stage of protocol handshakes for mixed network traffic that are carried over an encrypted SSL connection. Although looking for a pattern through inspecting the packet payload content is an effective approach even for encrypted traffic, it usually fails to classify application types for those applications (e.g. Gnutella) with variable-length packets in their protocol handshakes. Other typical examples of using payload content signatures for traffic classification include [28, 29, 30, 31]. Based on the payload signatures, the application classifier can obtain an extremely high accuracy. However, the biggest limitation is that all the above mentioned approaches focus on identifying only one single application (e.g. KaZaA in [23] or Skype in [26]) or one application group (e.g. Chat traffic identification in [29] or P2P traffic identification in [25]).

### 2.3.2 Statistical Traffic Classifier

The usage of statistical properties for network traffic classification or at least traffic behavioral modeling is not new. The early studies on the subject can be traced back to the seminal report by Paxson et al. [32, 33], in which some statistical variables (e.g. packet length, inter-arrival times and flow duration) have been proved to be suitable to express the behavior of a few protocols. With the increase of newly appeared network applications, the problem now has become how to associate a given flow, characterized by a set of statistics,
to a specific application. As a result, machine learning techniques can naturally achieve such a classification task through their training and learning capabilities. In [34], Williams et al. conduct a preliminary performance comparison of 5 machine learning algorithms for practical IP flow classification. Given the same features and flow trace, it was claimed that different machine learning algorithms provide very similar classification accuracy. The basic 5 features proposed in the paper include: protocol, flow duration, flow volume in bytes and packets, packet length, and inter-arrival time between packets. In [35], Crotti et al. present a flow classification mechanism based on three simple properties of the captured IP packets: size of packets, inter-arrival time and arrival order. A new structure, called protocol fingerprints, is defined to express the three trace properties in a compact and efficient way. According to an anomaly score, the protocol fingerprints allow the measurement of “how far” an unknown flow is from the basic characteristics of each protocol. A simple classification algorithm is then applied to classify flows dynamically when packets pass through the classifier, deciding if a flow belongs to a given application layer protocol, or if it was generated by an “unknown” (i.e., non-fingerprinted) protocol. As claimed in their evaluation, the limitation of the approach is that it can identify only 3 protocols, namely SMTP, POP3 and HTTP.

In [36], Bernaille et al. propose a technique that relies on the observation of the first five packets of a TCP connection to identify the application. It was claimed that the size of the first few packets is a good predictor of
the application associated with a flow because it captures the application’s negotiation phase, which is usually a pre-defined sequence of messages and distinct among applications. The result opens a range of new possibilities for online traffic classification since most classification techniques need the statistics of the entire flow to start the traffic classification (e.g. duration and number of packets in a flow), which limits their applicability for online classification. Bernaille et al. then illustrate their idea in [37] for an online traffic classification based on the first few packets of a TCP connection. Specifically their approach consists of two phases: an offline learning phase and an online classification phase. They employ three clustering methods (K-means and Gaussian Mixture Models on an Euclidean space, and Spectral clustering on Hidden Markov Models), and connections not belonging to any cluster are identified as unknown. As a result, it was claimed by the authors that the approach had the potential to detect new applications or new modes of operation of known applications.

In [38], McGregor et al. present a machine learning based methodology to break the traffic trace into different trace clusters in which each cluster has different trace characteristics. Typical clusters include bulk transfer, single and multiple transactions, and interactive trace. For the clustering, they applied Expectation-Maximization (EM) algorithm with a set of statistical features including packet size, inter-arrival statistics, byte counts, connection duration, the number of transitions between transaction mode and bulk transfer mode, and the time spent in bulk transfer and in transaction mode.
The evaluation results basically group traffic into different application types, like bulk transfer, small transactions, etc. However, further work is necessary in order to obtain the more specific applications groups. Although all these techniques show their capability for traffic classification to some extent, the number of applications they can identify is very limited. In addition, the definition of application classes is very rough and is not precise enough to obtain the fine-grained applications. One of the few exceptions is the work conducted by Erman et al. [39], in which they employed a semi-supervised learning technique to classify over 29 applications.

### 2.4 Performance Evaluation

Conducting a thorough analysis of the recent research done in anomaly detection, we encounter some machine learning methods reported to have a very high detection rate of 98% while keeping the false alarm rate at 1% [40]. However, when we look at the state of the art IDS solutions and commercial tools, there is little evidence of using the anomaly detection approach, and people still think that it is an immature technology. Recent studies show that there are some inherent problems in DARPA and KDD data sets which are widely used as the only publicly available data sets for network-based anomaly detection systems.
2.4.1 KDD CUP 99 Data Set Description

Since 1999, KDD’99 [41] has been the most widely used data set for the evaluation of anomaly detection methods. This data set is prepared by Stolfo et al. [42] and is built based on the data captured in DARPA’98 IDS evaluation program [43]. DARPA’98 is about 4 gigabytes of compressed raw data of 7 weeks of network traffic, which can be processed into about 5 million connection records, each with about 100 bytes. The two weeks of test data have around 2 million connection records. The KDD training data set 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 exactly one specific attack type. The simulated attacks fall in one of the following four categories:

1. Denial of Service Attack (DoS) is an attack in which the attacker makes some computing or memory resource too busy or too full to handle legitimate requests, or denies legitimate users access to a machine.

2. User to Root Attack (U2R) is a class of exploit in which the attacker starts out with access to a normal user account on the system (perhaps gained by sniffing passwords, a dictionary attack, or social engineering) and is able to exploit some vulnerability to gain root access to the system.

3. Remote to Local Attack (R2L) occurs when an attacker who has the ability to send packets to a machine over a network, but who does
not have an account on that machine, exploits some vulnerability to gain local access as a user of that machine.

4. **Probing Attack** is an attempt to gather information about a network of computers for the apparent purpose of circumventing its security controls.

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

The data sets contain a total number of 24 training attack types, with an additional 14 types in the test data only. The name and detail description of the training attack types are listed in [44].

KDD'99 features can be classified into three groups:

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

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

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

(b) “same service” features: This type of features examines only the connections in the past 2 seconds that have the same service as the current connection.

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

3. Content features: unlike most of the DoS and Probing attacks, the R2L and U2R attacks don’t have any intrusion frequent sequential patterns. This is because the DoS and Probing attacks involve many connections to the same host(s) in a very short period of time; however the R2L and U2R attacks are embedded in the data portions of the packets and normally involve only a single connection. To detect these kinds of attacks, one needs some features to be able to look for suspicious behavior in the data portion, e.g., number of failed login attempts. These features are called content features.
2.4.2 Inherent Problems of KDD’99 Data Set

As mentioned in the previous section, KDD’99 is built based on the data captured in DARPA‘98 which has been criticized by McHugh [45], mainly because of the characteristics of the synthetic data. As a result, some of the existing problems in DARPA‘98 remain in KDD’99. However, there are some deliberate or unintentional improvements, along with additional problems. In the following, we first review the issues in DARPA‘98 and then discuss the possible existence of those problems in KDD’99. Finally we discuss the author’s observations of the KDD data set.

1. For the sake of privacy, the experiments chosen to synthesize both the background and the attack data, and the data are claimed to be similar to that observed during several months of sampling data from a number of Air Force bases. However, neither analytical nor experimental validation of the data’s false alarm characteristics were undertaken. Furthermore, the workload of the synthesized data does not seem to be similar to the traffic in real networks.

2. Traffic collectors such as TCPdump, which are used in DARPA‘98, are very likely to become overloaded and drop packets in heavy traffic load. However, there was no examination to check the possibility of the dropped packets.

3. There is no exact definition of the attacks. For example, probing is not necessarily an attack type unless the number of iterations passes
an specific threshold. Similarly, a packet that causes a buffer overflow is not always representative of an attack. Under such conditions, there should be an agreement on the definitions between the evaluator and evaluated. In DARPA'98, however, there is no specific definition of the network attacks.

In addition, there exist some other critiques about attack taxonomies and performance measures. However, these issues are not of much interest since most of the anomaly detection systems work with binary labels, i.e., anomalous and normal. While McHugh’s critique was mainly based on the procedure to generate the data set rather than the analysis of data, Mahoney and Chan [46] analyzed DARPA background network traffic and found evidence of simulation artifacts that could result in an overestimation of the performance of some anomaly detection techniques. In their paper, the authors mentioned five types of anomalies leading to attack detection. However, analysis of the attacks in the DARPA data set revealed that many did not fit into any of these categories which are likely caused by simulation artifacts. As an example, the TTL (time to live) values of 126 and 253 appear only in hostile traffic, whereas in most background traffic the value is 127 and 254. Similarly, some attacks can be identified by anomalous source IP addresses or anomalies in the TCP window size field. Fortunately the aforementioned simulation artifacts do not affect the KDD data set since the 41 features used in KDD are not related to any of the weak-
nesses mentioned in [46]. However, KDD suffers from additional problems not existing in the DARPA data set.

In [4], Portnoy et al. partitioned the KDD data set into ten subsets, each containing approximately 490,000 instances or 10% of the data. However, they observed that the distribution of the attacks in the KDD data set is very uneven which made cross-validation very difficult. Many of these subsets contained instances of only a single type. For example, the 4th, 5th, 6th, and 7th, 10% portions of the full data set contained only smurf attacks, and the data instances in the 8th subset were almost entirely neptune intrusions.

Similarly, the same problem with smurf and neptune attacks in the KDD training data set has been reported in [47]. The authors have mentioned two problems caused by including these attacks in the data set. First, these two types of DoS attacks constitute over 71% of the testing data set which completely affects the evaluation. Secondly, since they generate large volumes of traffic, they are easily detectable by other means, and there is no need use anomaly detection systems to find these attacks.

2.5 Summary

In this chapter, we have provided a brief introduction of anomaly intrusion detection systems (AIDS) and their various taxonomies based on the availability of data labels, output format, and sources of data. We have also given an overview of some existing machine learning techniques that have been suc-
cessfully applied for intrusion detection. Since the main focus of this thesis is to propose an application-specific hybrid intrusion detection system, we have devoted two separate sections to review the existing research in the area of hybrid IDSs and also traffic classification methods. Finally, we have discussed the current issues and difficulties that must be overcome to have an accurate performance evaluation of intrusion detection systems.
Chapter 3

Evaluation of Anomaly Detection Systems

During the last decade, anomaly detection has attracted the attention of many researchers to overcome the weakness of signature-based IDSs in detecting novel attacks. Conducting a survey on the recent research done in anomaly detection, we encountered some machine learning methods reported to have a very high detection rate of 98% while keeping the false alarm rate at 1%. However, when we look at the state of the art IDS solutions and commercial tools, there is little evidence of using the anomaly detection approach, and people still think that it is an immature technology.

To find the reason for this contrast, we studied the details of the research done in anomaly detection and considered all the aspects such as learning and detection approaches, applied data sets and evaluation methods. Our
study shows that there are some problems with the employed data sets and evaluation methods. In this chapter, we mention all the drawbacks to the existing evaluation methods and provide some solutions for obtaining more realistic results.

3.1 Evaluation Methodology

In order to conduct a comprehensive survey of anomaly detection systems, we collected all research papers indexed by the Digital Bibliography and Library Project (DBLP) and Google Scholar between the years of 2000 and 2008. From these papers, we removed short papers, extended abstracts, technical reports and papers not written in English. Moreover, we encountered several cases of plagiarism with 90% content overlap. For these cases, we only considered the earlier copy of each work.

Having done all the aforementioned steps, we are left with 276 papers, contained 61 journals and 215 conference/workshop papers. To provide a better analysis of the survey results, we categorized the reviewed papers into four categories. Papers published in ISI journals (40 papers), non-ISI journals (11 papers), frequently cited (FC) conferences/workshops (88 papers) and rarely cited (RC) conferences/workshops (138 papers). Tables 3.1 and 3.2 illustrate the details of the surveyed papers and the top five venues for each category.

In the remainder of this chapter, we review the experimental results of the surveyed papers along with four important aspects of the IDS evaluation: 1)
<table>
<thead>
<tr>
<th>Papers by intrusion detection types</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Host-based studies</td>
<td>93 papers out of 276</td>
</tr>
<tr>
<td>Network-based studies</td>
<td>163 papers out of 276</td>
</tr>
<tr>
<td>Application-based studies</td>
<td>20 papers out of 276</td>
</tr>
<tr>
<td>Applied intrusion detection methods</td>
<td></td>
</tr>
<tr>
<td>Classification-based methods:</td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td>160 papers out of 276</td>
</tr>
<tr>
<td>HMM</td>
<td>25 papers out of 160</td>
</tr>
<tr>
<td>SVM</td>
<td>36 papers out of 160</td>
</tr>
<tr>
<td>Bayesian networks</td>
<td>20 papers out of 160</td>
</tr>
<tr>
<td>Other methods</td>
<td>14 papers out of 160</td>
</tr>
<tr>
<td>Statistics-based methods</td>
<td>65 papers out of 160</td>
</tr>
<tr>
<td>Clustering</td>
<td>62 papers out of 276</td>
</tr>
<tr>
<td>Misc. methods</td>
<td>36 papers out of 276</td>
</tr>
<tr>
<td>(control-flow graph, finite-state automata, etc.)</td>
<td>46 papers out of 276</td>
</tr>
</tbody>
</table>

Table 3.1: The details of the surveyed papers.

employed data sets; 2) data processing; 3) experiments; and 4) performance evaluation.

### 3.2 Employed Data Sets

The applied data sets, both training and testing, play an important role in the evaluation of anomaly detection methods. However, due to the criticism of existing data sets and also privacy issues of employing real traffic, preparing a data set has become one of the biggest challenges in the area of intrusion detection.

As it is shown in Figure 3.1, the most dominant evaluation of anomaly de-
Frequently Cited (FC) category

<table>
<thead>
<tr>
<th>Conference</th>
<th>Cited</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Assurance Workshop (IAW)</td>
<td>7</td>
</tr>
<tr>
<td>International Symposium on Recent Advances in Intrusion Detection (RAID)</td>
<td>5</td>
</tr>
<tr>
<td>ACM Symposium on Applied computing (SAC)</td>
<td>5</td>
</tr>
<tr>
<td>ACM Conference on Computer and Communications Security (CCS)</td>
<td>3</td>
</tr>
<tr>
<td>IEEE International Conference on Data Mining (ICDM)</td>
<td>3</td>
</tr>
</tbody>
</table>

Rarely Cited (RC) category

<table>
<thead>
<tr>
<th>Conference</th>
<th>Cited</th>
</tr>
</thead>
<tbody>
<tr>
<td>International Conference on Machine Learning and Cybernetics (ICMLC)</td>
<td>13</td>
</tr>
<tr>
<td>International Symposium on Neural Networks (ISNN)</td>
<td>6</td>
</tr>
<tr>
<td>International Conference on Availability, Reliability and Security (ARES)</td>
<td>5</td>
</tr>
<tr>
<td>Fuzzy systems and knowledge discovery conference (FSKD)</td>
<td>3</td>
</tr>
<tr>
<td>Security and Management conference (SAM)</td>
<td>1</td>
</tr>
</tbody>
</table>

ISI journals

| Journal                                                        | Cited |
|                                                               |-------|
| Computers & Security, Elsevier                                 | 10    |
| Computer Communications, Elsevier                              | 6     |
| Computer Networks, Elsevier                                   | 3     |
| ACM Transactions on Information and System Security (TISSEC)   | 2     |
| IEEE Transactions on systems, man, and cybernetics             | 2     |

Non-ISI journals

| Journal                                                        | Cited |
|                                                               |-------|
| Journal of software                                           | 2     |
| Journal of Information Assurance and Security                  | 1     |
| International Journal of Non-Standard Computing and Artificial Intelligence | 1     |
| International Journal of Computer Science and Network Security | 1     |
| Information Management & Computer Security                     | 1     |

Table 3.2: Publication venues of surveyed papers (top five venues).

Intrusion detection systems is based on the publicly available data sets which contain 70% of the surveyed papers. 32% of the papers have created their own data sets. 9% have conducted the experiments by means of simulations tools and finally 7% have attempted to test the performance of their proposed methods throughout the deployment in a real network. Note that the surveyed papers are not necessarily classified into one of the above groups, and there exist some papers having done the evaluation on several data sets.
Among the publicly available data sets, the Defence Advanced Research Projects Agency (DARPA) evaluation data set (24%) and the Knowledge Discovery and Data mining (KDD) data set (28%) are the most widely used data sets for anomaly detection. In total these two data sets are used in more than 50% of the studied papers. DARPA [44] refers to a series of data sets generated in 1998, 1999 and 2000 in the MIT Lincoln Laboratories, specifically for the testing of intrusion detection systems. The sets consist of simulated normal traffic and network attacks are manually generated. The KDD set [41], known as KDD’99 is prepared by Stolfo et al. [42] and is built based on the data captured in DARPA98 IDS evaluation program [43]. Although these two data sets have had a significant role in the evaluation of IDSs, their accuracy in simulating real networks conditions have been extensively
Publicly available data sets

<table>
<thead>
<tr>
<th>Percentage</th>
<th>Count (Out of Total)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>24%</td>
<td>67 (out of 276)</td>
<td>employ DARPA data</td>
</tr>
<tr>
<td>28%</td>
<td>77 (out of 276)</td>
<td>employ KDD data set</td>
</tr>
<tr>
<td>36.5%</td>
<td>34 (out of 93)</td>
<td>of host-based studies employ UNM data.</td>
</tr>
<tr>
<td>26%</td>
<td>24 (out of 93)</td>
<td>of host-based studies employ DARPA data set.</td>
</tr>
<tr>
<td>6%</td>
<td>16 (out of 276)</td>
<td>employ other sets</td>
</tr>
<tr>
<td>15%</td>
<td>41 (out of 276)</td>
<td>inserted additional attack traces to the sets.</td>
</tr>
</tbody>
</table>

Self-created data sets for a given study

<table>
<thead>
<tr>
<th>Percentage</th>
<th>Count (Out of Total)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7%</td>
<td>2 (out of 276)</td>
<td>Released data</td>
</tr>
<tr>
<td>31%</td>
<td>86 (out of 276)</td>
<td>Not released data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>34 papers out of 86 are FC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32 papers out of 86 are RC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17 out of 86 are ISI journal papers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 out of 86 are non-ISI journal papers</td>
</tr>
<tr>
<td>28%</td>
<td>24 (out of 86)</td>
<td>Among studies that did not release data sets:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>not provided detailed data description</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 out of 24 are FC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 out of 24 are RC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 out of 24 are ISI journals</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 out of 24 are non-ISI journals</td>
</tr>
<tr>
<td>85%</td>
<td>75 (out of 88)</td>
<td>collected real traffic/host events to generate sets.</td>
</tr>
<tr>
<td>24%</td>
<td>21 (out of 88)</td>
<td>inserted synthetic attack traces into the data.</td>
</tr>
<tr>
<td>23%</td>
<td>20 (out of 88)</td>
<td>inserted real attack traces into the data.</td>
</tr>
</tbody>
</table>

Table 3.3: The detailed statistics of the applied data sets.

criticized in [45, 48, 46].

In the host-based intrusion detection systems, people have mostly used a synthetic data set called the University of New Mexico (UNM) data set [49], which is employed in 36.5% of the host-based research papers, and DARPA has the second place among the host-based research papers with 26%. However, there is no evidence of using the KDD data set since it is specifically designed for network-based IDSs. The detailed statistics of applied data sets in both host-based and network-based intrusion detection systems are listed
3.3 Data Processing

In addition to the current issues with the employed synthetic data sets, data processing is also playing an important role in having an accurate evaluation. In contrast to the papers published in the area of machine learning and data mining, much of the published research in the intrusion detection field do not pay attention to this important stage. In the following we briefly explain some of the common pitfalls identified in the surveyed papers.

3.3.1 Definition of anomaly

One of the biggest issues affecting the performance of anomaly detection systems is the interpretation of data and defining them as either normal or anomaly. As it is defined in the seminal paper of Denning [6], any deviation from normal traffic will be considered as anomaly. In addition, in her paper, she assumes that malicious behavior is anomalous, and therefore detecting anomalous behavior will result in the detection of malicious activities. Although this assumption might have been true in 1987 and specially for the host-based IDSs, it is not applicable to today’s networks for the purpose of intrusion detection. Apparently, based on this assumption people in academia still define anomaly as abnormal behavior, while network administrators think of anomalies as any activities with the potential to threaten
their networks. The necessity of having a clear definition of anomalous behavior has been also highlighted by Gates and Taylor [50] and Ringberg et al. [51]. However, the majority of current research on anomaly detection, 88% of the surveyed papers, do not explicitly define what type of information is considered as anomaly, and the implicit assumption in academia is to refer the abnormal traffic as anomaly while the main target of proposed methods and evaluation systems is to detect attacks and malicious behavior.

3.3.2 Data scaling and normalization

Although scaling is not always necessary, removing this step when the features are not uniform result in a bias toward data with a larger range of values, and consequently impact the final outcome of the algorithm. Equation 3.1 shows one of the most common forms of scaling numeric variables in the range [0, 1]. The drawback of this scaling method is that when there is an outlier in the data set, a “normal” data will be mapped into a very small interval. Normalization, on the hand, transforms data to have a mean value of zero so that outliers can be easily detected. One of the common forms of normalization also called standardization is depicted in Equation 3.2. The main problem of this method is that it assumes the data is generated with a Gaussian law with a certain mean and standard deviation, which might not be true in reality.
While the necessity of scaling cannot be judged in the data sets where little information is available, we can point out with certainty that publicly available data sets such as DARPA and KDD require such normalization. For instance, the KDD set contains 41 features and the majority of features have different scales, e.g., destination host count is in the range of 0 to 255, while source bytes range from 0 to 693375640. Thus, any type of anomaly analysis would require at least scaling of the employed features if does not need the normalization. Given this, only 21% of all studies on these data sets indicated the use of normalization techniques.

3.3.3 Feature selection

Data sets usually contain various features, and a proper feature selection method has a high impact on the performance of detection methods. However, our survey shows that only 24% of network-based approaches have applied feature selection (see Table 3.4). A more important issue here is that 27.5% of the works do not disclose the employed features which makes it almost impossible to reproduce the experiments and have a fair comparison
with future research.
Table 3.4: The detailed statistics of the data processing techniques.

### 3.4 Experiments

Experiment setup is another factor affecting the evaluation results of anomaly detection systems. This factor can be investigated from two different aspects, experimentation procedure and proper documentation. Table 3.5 lists a detailed statistic of the experiments in the reviewed papers.

#### 3.4.1 Experimentation procedure

It is a common practice in machine learning to run the experiments in several rounds and on various data sets to verify the analysis is data set independent and can be generalized for real applications. Cross-validation is one of the techniques that is widely used in machine learning research. Each round of cross-validation involves partitioning the data set into two complementary parts of training and testing. To increase the reliability and validity of the results, researchers perform multiple rounds of cross-validation and average the results over the rounds. As an example, in 10-fold cross-validation the data set will be divided into 10 equal parts, and the experiments will be done
in 10 rounds such that in round \( i \), the \( i \)th section will be used as the testing set and the rest will compose the training set to build the model.

Moreover, since the final goal of anomaly detection systems is to detect unforeseen attacks, researchers have to make sure that the attacks represented in the test set are not fully repeated in the training set. On the other hand an acceptable percentage of anomalies has to be provided in the training set. Such caution seems to be necessary in the area of intrusion detection since the distribution of attack types are generally unequal. For example, the 4th, 5th and 7th, 10% portions of the full KDD data set contains only \textit{smurf} attack, and the data instances in the 8th subset are almost entirely \textit{neptune} intrusions.

Unfortunately, in spite of their significance, the issues of reliability and validity of the experiments were often ignored in the surveyed works. 80% of the papers did not discuss any methods used to ensure the validity and reliability of the experiments. Among the rest, 20% of the studies (6 out of 55) directly stated the number of simulation runs and 34 studies indicated that the produced result was an average of the performed evaluation runs. A total of 12 papers reported the use of cross-validation. There were also a few papers reported that the best obtained result as the overall performance of the system.
<table>
<thead>
<tr>
<th>Experiment Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>37% (72 out of 194)</td>
<td>of studies using public data sets, specified training and testing sets. 18 out of 194 are FC 39 out of 194 are RC 11 out of 194 are ISI journals 4 out of 194 are non-ISI journals</td>
</tr>
<tr>
<td>41% (36 out of 88)</td>
<td>of studies using self-created data sets, specified training and testing sets. 19 out of 88 are FC 9 out of 88 are RC 6 out of 88 are ISI journals 2 out of 88 are non-ISI journals</td>
</tr>
<tr>
<td>21% (34 out of 163)</td>
<td>of network-based studies specified the ratio of abnormal vs. normal activity in the testing set. 23.5% (8 out of 34) used 1-2% abnormal vs. 99-98% normal activity 3% (1 out of 34) used 6-8% abnormal vs. 94-92% normal activity 3% (1 out of 34) used 19-20% abnormal vs. 81-80% normal activity 15% (5 out of 34) used 30-50% abnormal vs. 70-50% normal activity 56% (19 out of 34) used 80-82% abnormal vs. 20-18% normal activity</td>
</tr>
<tr>
<td>19% (52 out of 276)</td>
<td>conducted the performance study. 18 out of 52 are FC 21 out of 52 are RC 11 out of 52 are ISI journals 2 out of 52 are non-ISI journals</td>
</tr>
<tr>
<td>60% (31 out of 52)</td>
<td>of those that conducted the performance study stated the characteristics of the computer. 14 out of 31 are FC 10 out of 31 are RC 5 out of 31 are ISI journals 2 out of 31 are non-ISI journals</td>
</tr>
<tr>
<td>23% (64 out of 276)</td>
<td>did not specify the initial parameters of the algorithms. 15 out of 64 are FC 36 out of 64 are RC 12 out of 64 are ISI journals 1 out of 64 is non-ISI journal</td>
</tr>
<tr>
<td>12% (25 out of 212)</td>
<td>of those that specified the initial parameters justified the selected initial parameters of the algorithm.</td>
</tr>
<tr>
<td>20% (55 out of 276)</td>
<td>indicated methods to ensure validity and reliability of experiments. 6 out of 55 states the number of simulation runs 34 out of 55 indicated that the produced result is an average value 12 out of 55 reported the use of cross-validation 3 out of 55 reported a confidence interval for the obtained results</td>
</tr>
</tbody>
</table>

Table 3.5: The detailed statistics of the experiments setup.
3.4.2 Proper documentation

This phase is very crucial for researchers in order to reproduce the experimental results for evaluation purposes. However, it does not receive proper attention by researchers working on intrusion detection. This is especially important in case of very large data sets such as DARPA and KDD, in which people employ a portion of them for the experiments due to the limited time and available resources. Unfortunately, out of 194 papers using the publicly available data sets, close to two-thirds (63%) did not properly specify which sets were used for training and testing of the approach. Among experimental data sets that were not public, these numbers were slightly higher: out of 88 papers, 59% did not describe the training and testing sets.

Another aspect related to the employed testing set is the ratio of anomalous and normal records in the testing data. An assumption of rareness of anomalies, i.e., the existence of a small portion of anomalous records compared to the volume of normal activity, is common in the intrusion detection domain [50]. Recently there have been several studies showing that this picture is changing and nowadays abnormal traffic on the Internet (including scanning activity) cannot be quantified as rare [52, 53].

Reviewing the network-based studies on the DARPA and KDD data sets, we noticed a great variability in the employed ratio of abnormal to normal activity. Among 34 papers that specified this ratio for the testing set, the majority of the studies (24 out of 34) experimented with a high percentage of abnormal activity (30%-82%) in the data. It should be also noted that 19 of
these studies worked with the KDD data set that has 81% of abnormal activity. As some of these studies employed random sampling, a final percentage of abnormal activity ranged from 80% to 82%. 2 papers experimented with 6%-20% of abnormal activity in the set, and only 8 papers out of 34 (23.5%) assumed a low probability of intrusive activity, using a 1%-2% abnormal to 99%-98% normal activity ratio.

### 3.5 Performance Evaluation

The performance of anomaly detection techniques can be generally evaluated from two perspectives:

1. **Efficiency**: This measure deals with the resources needed to be allocated to the system including CPU cycles and main memory.

2. **Effectiveness**: This measure which is also called classification accuracy represents the ability of the system to distinguish between normal and intrusive activities.

As the current intrusion detection systems are experiencing serious issues to gain acceptable detection and false alarm rates, researchers have mostly concentrated on the effectiveness measures and do not pay much attention to the efficiency of their systems. Our survey shows that only 19% of the papers have conducted a performance study on time and memory complexities.

In contrast, effectiveness of the intrusion detection systems have been extensively studied and there are many approaches proposed to have a better
evaluation of IDSs. In the rest of this section, we will summarize the most popular evaluation metrics which have been employed for the comparison of IDSs.

### 3.5.1 Confusion Matrix

Anomaly detection methods are usually applied to distinguish between anomalous and normal traffic. So, here we are interested in performance measures which are applied in binary classification. To the best of our knowledge, confusion matrix is the best way of presenting the binary classification result (Table 3.6). Due to the two-class nature of the detection, there are four possibilities as follows:

1. **True positive (TP):** Attacks/anomalies that are successfully detected by the IDS.
2. **False positive (FP):** Normal behavior that are incorrectly classified as intrusive by the IDS.
3. **True Negative (TN):** Normal behavior that is successfully labeled as normal by the IDS.
4. **False Negative (FN):** Attacks/anomalies that are missed by the IDS, and classified as normal.

Joshi et al. [1] propose a graphical method to visualize the relation between the four variables in the confusion matrix. Figure 3.2 depicts the normal case of an intrusion detection problem as follows:
• The big circle defines the space of the whole data (i.e., normal and intrusive data)
• The small ellipse defines the space of all predicted intrusions by the classifier. Thus, it will be shared by both TP and FP.
• The ratio between the real normal data and the intrusions is graphically represented by the use of a horizontal line.

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Attack</td>
</tr>
<tr>
<td>Attack</td>
<td>True Positive (TP)</td>
</tr>
<tr>
<td>Normal</td>
<td>False Positive (FP)</td>
</tr>
</tbody>
</table>

Table 3.6: Confusion matrix

Despite the representation power of confusions matrix in binary classification, for the sake of comparison it is not a very useful tool. To solve this problem some performance measures are introduced which extract part of the information from the confusion matrix and produce some numeric values that are easily comparable.

In addition, there are some other metrics derived directly from the confusion matrix values which are briefly explained in the following.

1. **Accuracy:** This metric indicates the total number of connections that are correctly classified including normal and intrusive connections. (Equation 3.3)

2. **Detection Rate (DR):** DR is the ratio between the number of correctly detected attacks and the total number of attacks. (Equation 3.4)
Figure 3.2: The whole data space in binary classification. (This figure is adapted from [1])

3. **False Positive Rate (FPR)**: FPR is the ratio between the number of misclassified normal connections and the total number of normal connections. (Equation 3.5)

\[
Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{3.3}
\]

\[
DR = \frac{TP}{TP + FN} \tag{3.4}
\]

\[
FPR = \frac{FP}{FP + TN} \tag{3.5}
\]

### 3.5.2 ROC Curves

Perhaps the oldest measure used to evaluate machine learning methods is Accuracy. As mentioned this measure shows the ratio of correctly classified samples to the total number of samples in the data set. Since in the data sets used in intrusion detection, normal data usually outnumbers intrusions,
e.g. 95% normal traffic and 5% attack, the Accuracy measure is misleading because a system that always classifies all data as normal would have a high accuracy (95% in our example). So, we need measures being able to evaluate the performance invariant of the bias in the distribution of labels in the data. Receiver Operating Characteristic (ROC) curves were the measure introduced to meet this aim by using a trade-off between false positive and detection rates.

Originating from signal detection theory [54], ROC curves are used on the one hand to visualize the relation between detection and false positive rates of a certain classifier while tuning it, and on the other hand to compare the accuracy of several classifiers.

Although this measure is very effective and widely used in anomaly detection, it has some shortcomings. The first drawback is that it is dependent on the ratio of attacks to normal traffic. In [55] and [14], the authors have used different data sets with various ratios of attack to normal. Their result shows that as we decrease the ratio of attack in the data set, we are getting better performance on the ROC curve. This issue is not very problematic when we are comparing different methods run on the same data set. However, the comparison of anomaly detection methods run on various data sets is completely wrong, unless they both have the same ratio of attack to normal. The second problem with ROC curves is that they might be misleading and simply incomplete for understanding the strengths and weaknesses of a proposed method [48, 56, 57, 58].
3.5.3 Precision, Recall and F-Measure

As previously mentioned, under normal operating conditions there is a big difference between the rate of normal and intrusion data. Thus, the Precision, Recall, and F-Measure metrics ignore the normal data that has been correctly classified by the IDS (TN).

**Precision**: It is a metric defined with respect to the intrusion class. It shows how many examples, predicted by an IDS as being intrusive, are the actual intrusions [59]. The aim of an IDS is to obtain a high Precision, meaning that the number of false alarms is minimized.

\[
\text{precision} = \frac{TP}{TP + FP}, \quad \text{precision} \in [0, 1]
\]  

**Recall**: This metric measures the missing part from the Precision; namely, the percentage from the real intrusions covered by the classifier. Consequently, it is desired for a classifier to have a high recall value [59]. This metric is equivalent to the detection rate (DR).

\[
\text{recall} = \frac{TP}{TP + FN}, \quad \text{recall} \in [0, 1]
\]  

**F-Measure**: Due to the fact that the previously discussed two metrics (i.e., Precision, and Recall) do not completely define the accuracy of an IDS, a combination of them would be more appropriate to use. Being defined
as the harmonic mean of precision and recall, the F-Measure mixes the properties of the previous two metrics [59].

\[
F - \text{Measure} = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}, \quad F - \text{Measure} \in [0, 1] \quad (3.8)
\]

F-Measure is preferred when only one accuracy metric is desired as an evaluation criteria. Note that when Precision and Recall reaches 100%, the F-Measure is maximum (i.e. 1), meaning that the classifier has 0% false alarms and detects 100% of the attacks. Thus, the F-Measure of a classifier is desired to be as high as possible.

### 3.5.4 Positive and Negative Predictive Values

Base-rate fallacy is an error that occurs in conditional probabilities when there is no prior probability of the hypothesis and the evidence [60]. As pointed out by Axelsson [58], this problem occurs in the evaluation of intrusion detection systems since there is a huge difference between the prior probabilities of normal and intrusive connections.

\[
P(A|B) = \frac{\prod_{i=1}^{n} \frac{R(A_i) \cdot R(\hat{A})}{P(A_i) \cdot P(B|A_i)}}{
\prod_{i=1}^{n} \frac{P(A_i) \cdot P(B|A_i)}}
\]

\[
(3.9)
\]

Lets assume that \(I\) denotes intrusion, and \(\neg I\) denotes normal behavior. Similarly, let \(A\) denote an alarm generated by the IDS. Applying general from of
Bayes theorem (Equation 3.9) we can conclude that:

\[
PPV \equiv P(I|A) = \frac{P(I) \cdot P(A|I)}{P(I) \cdot P(A|I) + P(\neg I) \cdot P(A|\neg I)}
\]

\[
= \frac{\text{p} \times \text{DR}}{\text{p} \times \text{DR} + (1 - \text{p}) \times \text{FPR}}
\]

\[
= \frac{\text{p} \times \text{DR}}{\text{p} \times (\text{DR} - \text{FPR}) + \text{FPR}} (3.10)
\]

Where \( P(I|A) \) is called Bayesian detection rate or positive predictive value (PPV) which shows what percentage of alerts are true positive; DR and FPR are detection rate and false positive rate, respectively; and \( p \), also called the base-rate, is the prior probability of having intrusion in the data set and can be estimated using the following formula:

\[
p = P(I) \approx \frac{TP + FN}{TP + FP + FN + TN} (3.11)
\]

To make this problem more clear, let \( p = 10^{-5} \) which means that on average only 1 out of 100,000 connections is an attack. Assuming that the detection rate is 100\% and false positive rate is 1\%, we will have

\[
P(I|A) = \frac{10^{-5} \times 1}{10^{-5} \times (1 - 0.01) + 0.01} = 0.000999 \approx 0.001 (3.12)
\]

This result shows that even for the unrealistically high detection rate of 100\% and false positive rate of 1\%, only one alert out of 1000 alerts is a real intrusion and the rest are false alarms. In other words, to have a Bayesian detection rate of 50\% (half of the alarms will be a true indication of intrusive activity), we need to have a very low false alarm rate of \( 10^{-5} \).
Due to the very low number of attacks in real networks, Axelsson believes that the Bayesian detection rate (PPV) is a more effective metric compared to the detection rate (DR). However, as can be seen in Equation 3.10, PPV is maximized when the false positive rate approaches zero, no matter what the value of the detection rate is. Therefore, he has defined another metric called the negative predictive value (NPV), Equation 3.13, and mentioned that there has to be a trade-off between the values of PPV and NPV.

\[
NPV = P(\neg I|\neg A) = \frac{(1 - p)(1 - FPR)}{p(1 - DR) + (1 - p)(1 - FPR)} \tag{3.13}
\]

### 3.5.5 B-ROC Curves

In order to overcome the deficiencies of ROC curves and the predictive values, Cárdenas and Baras proposed a new metric called B-ROC curves [61]. This new metric uses the same intuition as the ROC curves. However, instead of using false positive rate for the X axis, it uses \(1 - PPV\). This measure is called the Bayesian false alarm rate \(P(\neg I|A)\) and indicates that what percentage of generated alarms are false positive.

### 3.5.6 Survey Statistics on the Evaluation Measures

Generally there are two important points to address the potential inaccuracy of the existing performance measures: 1) a comprehensive set of evaluation metrics should be considered; 2) bursty attacks should be evaluated sepa-
However, the majority of research in the area of anomaly detection still follow the traditional evaluation methods. As Figure 3.3 illustrates, the ROC curve, DR and FPR are still the most commonly used metrics. 42% of the surveyed works employed only these measures, and 29% of the papers complemented these metrics with other evaluation measures. Other traditional metrics commonly employed for the evaluation are accuracy (10% of the papers) and precision and recall (4%).

Unfortunately, the second point, i.e., presentation of separate results for each type of attack, is also far from being a standard practice. Among 163
network-based studies only 26% of studies presented separate results for the types of attacks present in the data set.

3.6 Summary

In this chapter, we summarized our observation about the common pitfalls in the evaluation of intrusion detection systems. The study was based on 276 papers published between 2000 and 2008 in the area of anomaly detection system. Having several round of reviews of the surveyed papers, we classified the existing shortcomings in four categories namely, employed data sets, data processing, experiments, and performance evaluation. Furthermore, each category is extensively studied from different aspects and some solutions are provided. Avoiding the existing pitfalls in the evaluation of IDSs, we hope to have a more accurate evaluation of the proposed hybrid intrusion detection system.
Chapter 4

Proposed Framework

Except for the problems discussed in Chapter 3, Sommer and Paxson [62] point out some other issues which result in rare deployment of anomaly detection systems in real networks. In their paper, they mention that intrusion detection has special characteristics that make it harder to effectively deploy machine learning techniques. In the following we briefly overview these characteristics.

1. **Outlier detection:** Anomaly detection is fundamentally based on the idea of learning normal traffic and then labeling the outliers as anomalies. However, there is a basic rule in machine learning which says that in order to train a system one needs to have an acceptable amount of samples from each class [62]. Any violation from this rule will noticeably affect the performance of the system. Similarly, in their book [63], Witten et al. mention that the idea of learning only positive
examples with the assumption that the rest are negative is called the "closed world assumption". They further state that this assumption is not practical in real-life problems.

2. **High cost of errors:** The cost of misclassification in intrusion detection is extremely high compared to the other applications of machine learning. A false negative, missing an attack, can cause serious damage to an organization. On the other hand, a very small rate of false positive, can make the IDS unusable [58]. Comparing this with spam detection, which is a successful application of machine learning in network security, we observe that false negatives (spam not detected) do not have a significant impact on the system. Even false positives, which have a higher significance, can be easily managed by users.

3. **Semantic gap:** Currently, there is no consensus on the definition of anomaly. Many people believe that the purpose of anomaly detection is to detect what is not seen before, no matter if it is an indicator of intrusive or normal behavior. However, the main goal of all intrusion detection systems, including signature and anomaly based, is to detect intrusive activity. In addition, the definition of intrusive activity can also change from one organization to another depending on the policies. For example, peer-to-peer traffic is considered intrusive in some organizations due to the copyright and bandwidth issues.

4. **Diversity of network traffic:** Network traffic is subject to extreme variability and even the basic characteristics such as number of pack-
ets, duration of connections, type of applications, etc. are quite unpredictable over short time intervals (seconds to hours). This diversity makes the detection of anomalies more difficult and results in an extreme amount of false positives.

5. **Difficulties with evaluation**: As mentioned before, there exist a lot of barriers to have an accurate evaluation of anomaly detection methods including the data sets and performance metrics. Since this issue is fully discussed in Chapter 3, we will not address it in this section.

Having these issues in mind, we have proposed a new framework for network-based anomaly detection which is discussed in the rest of this chapter.

### 4.1 Overall View

To overcome the shortcomings of existing intrusion detection systems, a multi-layer model is provided (Figure 4.1) which consists of three processing layers: 1) Packet Analysis; 2) Intrusion Detection; and 3) Security Information and Event Management (SIEM).

#### 4.1.1 Packet Analysis

Being responsible for all the preprocessing tasks required for the intrusion detection, Packet analysis layer contains two important modules, namely flow analyzer and traffic classification.
4.1.1.1 Flow Analyzer

In order to keep up with the high speed of gigabit links, current network monitoring and management systems use network flow data (e.g., netflow, sflow, ipfix) as their information sources. Network flows are a group of network packets belonging to the same connection. Apparently, these packets have a lot of information in common (e.g., source IP, source port, destination IP, destination port, protocol, application), which can be stored only once
using the flow concept. Furthermore, in order to deal with a huge amount of payload information, usually only the first few bytes of each flow (e.g., 512 bytes), which is more informative for the analysis, will be stored by the monitoring devices.

**Definition 1.** Let \( \pi_i = (t_i, S_i, D_i, s_i, d_i, p_i, f_i) \) be representative of a network packet. A network flow, \( \varphi \), is an ordered set of all possible packets \( \{\pi_1, \pi_2, \ldots, \pi_n\} \) such that

1) \( \forall \pi_i, \pi_j \in \varphi \quad p_i = p_j \)

2) \( \forall \pi_i, \pi_j \in \varphi \quad (S_i = S_j \land D_i = D_j \land s_i = s_j \land d_i = d_j) \lor (S_i = D_i \land D_i = S_j \land s_i = d_j \land d_i = s_j) \)

3) \( \forall \pi_{i=n} \in \varphi \quad (t_i < t_{i+1}) \land (t_{i+1} - t_i \leq \alpha) \)

4) \( [(f_{n-2} = \{F\}) \land (f_{n-1} = \{A,F\}) \land (f_n = \{A\})] \lor (t_{n+1} - t_n > \alpha) \)

where \( t_i, S_i, D_i, s_i, d_i, p_i \), and \( f_i \) represent timestamp, source IP address, destination IP address, source port, destination port, protocol, and TCP flags, respectively.

In order to perform the flow analysis, i.e., capturing network packets, classifying them into flows and extracting the payloads, we used a commercial network security management tool, QRadar [64]. Flows generated by QRadar, qflows, are defined as a group of packets in a single connection sharing all of the following 5 values: 1) source IP address; 2) destination IP address; 3) source port number; 4) destination port number; and 5) IP protocol. Each flow will be terminated by a timeout which is set to 60 seconds (\( \alpha = 60 \)) in our experiments. Except for that, TCP flows can also be terminated upon
proper connection teardown.

4.1.1.2 Traffic Classification

As network applications are getting more diverse and complex, the idea of using special-purpose intrusion detection systems in the application layer becomes more popular. Focusing on a small subset of network applications will have the advantage of designing more specific signatures, which results in a better detection rate and a lower false positive rate. In addition, as illustrated in Figure 4.1, application-based IDSs can be applied in parallel which is of high importance in dealing with large networks with millions of packets per second. Considering the state-of-the-art application-based IDSs, we noticed that these detection systems are mainly deployed at the end points, and there have been few efforts to apply them on the network layer. The reason being that traffic classifiers cannot provide an acceptable level of confidence in terms of classification accuracy. One of the main contributions of this thesis is a powerful method for automatic discovery of network applications, which will be completely discussed in Section 4.2.

4.1.2 Intrusion Detection

As indicated in Figure 4.1, the proposed intrusion detection module consists of several application-based intrusion detection systems components. Sharing a similar architecture and detection mechanism, each component is specifically designed for a special type of application such as Web, FTP, Mail,
etc. There is also a component designed for applications with no specialized intrusion detection system and applications that are not detected with the traffic classifier. In Section 4.3, we provide a detailed description of the hybrid structure of each component, and the mechanism applied to provide automatically labeled training sets for anomaly-based detectors.

4.1.3 Security Information and Event Management

Although a lot of efforts have been done to decrease the number of false alarms generated by intrusion detection systems, we believe that having an IDS with no false alarm is almost impossible due to the dynamic nature of computer networks. However, we can minimize these false alarms by gathering and processing different types of information from various sources such as intrusion detection systems, anti viruses, operating systems logs, application level logs, among others.

Providing a real-time analysis of security alerts, Security Information and Event Management (SIEM) solutions are the most effective approach to increase the detection rate of intrusion while keeping the false alarms at an acceptable level.

4.2 Traffic Classification Module

Accurate classification of network traffic has received a lot of attention due to its important role in many subjects such as network planning, QoS pro-
visioning, class of service mapping, to name a few. Traditionally, traffic classification relied to a large extent on the association of a particular port with a specific protocol [22]. Such a port number based traffic classification approach has been proved to be ineffective due to: 1) the constant emergence of new peer-to-peer networking applications that IANA\(^1\) does not define the corresponding port numbers; 2) the dynamic port number assignment for some applications (e.g. FTP); and 3) the encapsulation of different services into a same application (e.g., chat or steaming can be encapsulated into the same HTTP protocol). To overcome this issue, there have been recently significant contributions towards traffic classification. The most currently successful approach is to inspect the content of payloads and look for the deterministic character strings for modeling the applications. For most applications, their initial protocol handshake steps are usually different and thus can be used for classification. Moreover, the protocol signatures can be modeled through either public documents such as RFCs or empirical analysis for deriving the distinct bit strings on both TCP and UDP traffic.

To have a better understanding of this approach, Table 4.1 illustrates the signatures of 11 typical applications in which to print the signatures, alphanumeric characters are represented in the normal form, while non-alphanumeric ones are shown in the hex form starting with “0x”.

As can be seen in Table 4.1, each application has a unique set of characters

---

\(^1\)Internet Assigned Numbers Authority (http://www.iana.org/assignments/port-numbers)
<table>
<thead>
<tr>
<th>Application</th>
<th>Payload</th>
<th>Offset</th>
<th>Signatures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Torrent</td>
<td>source</td>
<td>1</td>
<td>BitTorrent</td>
</tr>
<tr>
<td>HTTP Image Transfer</td>
<td>destination 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTTP Web</td>
<td>source</td>
<td>0</td>
<td>GET</td>
</tr>
<tr>
<td>Secure Web</td>
<td>destination 0</td>
<td>0x16 0x03</td>
<td></td>
</tr>
<tr>
<td>MSN Messenger</td>
<td>source</td>
<td>0</td>
<td>MSG</td>
</tr>
<tr>
<td>MS-SQL</td>
<td>destination 0</td>
<td>0x04 0x01 0x00 0x25 0x00 0x00 0x01 0x00 0x00 0x00 0x15 0x00 0x06 0x01 0x00 0x1B</td>
<td></td>
</tr>
<tr>
<td>POP</td>
<td>destination 0</td>
<td></td>
<td>+OK</td>
</tr>
<tr>
<td>SMTP</td>
<td>source</td>
<td>0</td>
<td>EHLO</td>
</tr>
<tr>
<td>Windows File Sharing</td>
<td>destination 4</td>
<td></td>
<td>[FF]SMB</td>
</tr>
<tr>
<td>Yahoo! Messenger</td>
<td>destination 0</td>
<td></td>
<td>YMSG</td>
</tr>
</tbody>
</table>

to be identified. *Secure Web* traffic can be identified by searching the hex string “1603” in the beginning of a flow payload. Similarly *Yahoo! Messenger* traffic can be detected by finding the ASCII string of “YMSG” in the payload. However, these signatures do not necessarily start from the beginning of the payload. For example, to identify *HTTP Image Transfer* traffic, we should search for the string “image/” starting from the 5th byte of the payload. This starting point is referred as offset in Table 4.1. Moreover, it is important to know which side of the connection, client or server, produce the signature. For example, the signature to detect *HTTP Web* is in the source payload, i.e., the ASCII string “GET” is sent by the client, initiator of the connection, to the server. This information helps signature-based methods to improve their performance by looking at a fewer number of signatures in either the source or destination payloads.

Although this approach has a high accuracy for identifying network applica-
tions, it fails to detect 20% to 40% of the network flows. Some of the reasons that cause this problem are:

- Whenever a newer version of an application is released, all the signatures should be updated which usually cannot be done immediately.

- There are hundreds of applications being developed by small groups around the world which networking companies are not aware of.

- Peer-to-peer (P2P) applications employ a large number of protocols with different signatures to prevent their traffic being filtered by organizations. Usually these signatures are variants of famous P2P applications (e.g., Bit Torrent, Gnutella, eDonkey) with small changes and kept confidential by crime organizations.

- Encrypted traffic is one the biggest challenges traffic classifiers are facing. Data encryption is becoming more popular with the growth of commercial services on the Internet. Besides, it is widely used to hide malicious activities such as botnet traffic.

Addressing the aforementioned issues, in the rest of this section, we propose a novel approach which is capable of detecting new releases and variants of existing applications. Furthermore, newly created applications and different variant types of P2P software will be classified in a relevant category with similar characteristics. However, detecting encrypted traffic still remains as a challenging issue.
GET / HTTP/1.1
HOST: www.google.com
Connection: keep-alive

HTTP/1.1 200 OK
Date: Fri, 18 Dec 2009 17:31:23 GMT
Expires: -

(a) First 64 bytes of the source payload

(b) First 64 bytes of the destination payload

Figure 4.2: Unigram distribution of an HTTP flow.

4.2.1 Weighted Unigram Model

N-grams are a language-independent means of gauging topical similarity in text documents. Traditionally, the n-grams technique refers to passing a sliding window of \( n \) characters over a text document and counting the occurrence of each n-gram. Being first introduced by Damashek [65], this method is widely employed in many language analysis tasks as well as network security.

Applying the same idea on network packets, one can consider unigram (1-gram) of a network packet as a sequence of ASCII characters ranging from 0 to 255. This way similar packets can be identified using the frequencies of distinct ASCII characters in the payload. Wang et al. were the first who
employed this idea to detect network anomalies [66]. In their paper, they mapped each packet into a 256-dimensional vector such that each dimension represents the frequency of the corresponding ASCII characters in the payload. A normal packet profile is then constructed through calculating the statistical average and deviation value of normal packets. Consequently, anomalies will be alerted once a Mahalanobis distance between testing data and normal profile exceeds a predefined threshold. Although the unigram model can be theoretically extended to n-gram, it has been barely used in the area of networking because of its high complexity. As an example, applying the bigram (2-gram) model on network packets will result in having 65,536 distinct characters.

In order to construct the unigram payload distribution model, we extract the first $M$ bytes of the payload and count the occurrence frequency of each ASCII character. However, since some of the applications bare their signatures in the source payload such as HTTP Web and some of them have their signatures in the destination payload like Secure Web, we consider source and destination payloads as separate pieces of information. In other words, each ASCII character has two frequency values, one for the source payload (data sent by the client, initiator of the connection, toward the server) and one for the destination payload (data received by the client from the server). Figure 4.2 illustrates the payload of a sample HTTP flow and the frequencies of existing characters. In this example, we have applied the unigram distribution on the first 64 bytes of source and destination payloads. Note that
Figure 4.3: Average unigram distribution of source payload
“\r” and “\n” are representative of carriage return and line feed characters, respectively.

By observing and analyzing the known network traffic applications, labeled by a signature-base classifier called MeterFlow [67], over a long period on large-scale WiFi ISP network, we found that the unigram distribution of source and destination payloads can be used as a powerful tool to detect the applications. Figure 4.3 shows the unigram distribution of several applications namely, HTTP Web, Secure Web, SMTP, POP, Bit Torrent, Oracle, HTTP Image Transfer, Web Media Documents. The x axis in the figures is the ASCII characters from 0 to 255, and the y axis shows the frequency of each ASCII character in the first 256 bytes of the source payload. As it can be seen in Figure 4.3, text-based applications (e.g., HTTP Web) can be easily distinguished from binary applications (e.g., Bit Torrent), as well as encrypted applications (e.g., Secure Web) since they use only a portion of ASCII characters, especially alphanumeric ones. Moreover, having an exact investigation of similarly behaving protocols such as HTTP Web and SMTP, we can still separate them based on the most frequent characters.

After applying the unigram distribution model on network flows, we can map each flow to a 512-tuple feature space \((f_0, f_1, \ldots, f_{511})\) such that \(f_0\) to \(f_{255}\) shows the frequencies of corresponding ASCII characters in the source payload. Similarly, \(f_{256}\) to \(f_{511}\) hold the frequencies in the destination payload.

Having specified the applied network features, we focus on the selection of
a high-performance classifier. In order to choose an appropriate classifier, we selected some of the most popular classification methods implemented by Weka [68] and performed some evaluations to compare the accuracy, learning time, and classification time. We finally selected the J48 decision tree, the Weka implementation of C4.5 [69], since it has a high accuracy while maintaining a reasonable learning time. In addition, decision trees are shown to have a reasonable height and need fewer comparisons to reach a leaf which is the final label, and therefore have a very short classification time. Taking advantage of the J48 decision tree as a traffic classifier, we evaluated our proposed method on two real networks.

Although our evaluations on the two networks showed promising results, we still believe that the performance can be improved by assigning different weights to the payload bytes based on the degree of importance. However, finding the appropriate weights is a challenging task. To this end, we employ a genetic algorithm based scheme to find the weights.

In the next section, we formally define our problem and explain how we apply genetic algorithms to improve the accuracy of our proposed traffic classification method.

### 4.2.2 Problem Formulation

In this section, we formally describe how the network application discovery problem can be performed through the combination of genetic algorithms and decision trees. Essentially, we formulate the network application discov-
ery problem as a classification problem, i.e., given the values for a specific set of features extracted from the network flows, we identify the possible application that has generated this payload using a statistical machine learning technique (decision tree). In other words, we assume that each payload uniquely exhibits certain characteristics that very well represents the application that has produced the payload; therefore, these features can be employed to discover that network application.

As was mentioned earlier, Wang et al. [66] were the first to view the payload as a multidimensional vector, where in the context of a unigram analysis of the payload, the frequency of each ASCII character in the payload represents one feature in the vector space. For instance, if the character ‘∞’ is repeated 20 times in the payload, the value for the 236th dimension of the representative vector would be $20^2$. So with this simple yet intuitive formulation of the features of the payload vector space, it is possible to construct a learning classification machine that operates over features that are the frequency of each ASCII character in the payload. We formally describe this as follows:

**Definition 2.** Let $\kappa_s, \ldots, \kappa_s$ be the set of 256 available ASCII characters in the source payload and $\kappa_d, \ldots, \kappa_d$ be those in the destination payload, and $\Psi$ be a given payload generated by an application $\lambda \in \Lambda$, $\Lambda$ being the set of all known network applications. We denote $v_{\kappa_s}(\Psi)$ as the frequency of $\kappa_s$ in $\Psi$. Further, $\mathbf{8}': \Psi \rightarrow \Lambda$ is a learning classification machine that maps a payload $\Psi$ with frequency $\nu$ to a network application such as $\lambda$.

---

236 being the index of $\infty$ in the ASCII character list.
Based on this definition, we need to employ a classifier that infers the correct network application label given the features from the payload. To achieve this we employ the J48 decision tree learning algorithm [69] that builds a decision tree classifier from a set of sample payloads and their corresponding network application labels. The learned decision tree classifier will enable us to find the possible network application label for a payload from an unknown network application. Here, the learned J48 classifier is our required 8’ mapping function.

Now lets consider the HTTP Web application payload. As it can be seen in Figure 4.2, the payload starts with GET and continues with further detailed information about that specific connection such as the protocol version, host and others. So for this specific application, i.e. HTTP Web, the first few characters are representative enough of the application; therefore, by just looking at these 3 characters we are able to identify the signature for this application and easily assert that this payload has been generated by the HTTP Web network application. Similarly, other network applications have specific signatures that are present in some designated positions in the payload. This position is specific to each application, some have chosen the beginning of the payload to place their identifier, e.g., HTTP Web, while others may choose other positions. Hence, the ASCII characters appearing in the payload do not have the same degree of importance in the application discovery process, i.e., the positions of the characters is quite important, e.g., the ASCII characters GET appearing in the first 3 bytes of the payload is an indication of
the generating network application of this payload. So, it can be inferred that some positions in the payload are more discriminative in the process of classifying the payloads for their generating network applications. For this reason, it is important to place more weight on the features that appear in these more important positions. We achieve this through a weighted scheme over the features.

**Definition 3.** Suppose $\Psi$ is a given payload of length $\eta$. We let $\Psi_p$, $p \in [0, \eta]$ denote the $p^{th}$ position in the payload $\Psi$, and $\omega(\Psi_p)$ denote the weight (importance) of position $p$ in the payload. The weighted feature of the vector space is defined as:

$$\omega_{k_{i,j}^{s/d}}(\Psi) = \omega(\Psi_p)$$

where $P$ represents the positions in which $k_{i,j}^{s/d}$ has appeared in $\Psi$.

Simply, this definition defines that each position in the payload has its own significance in the application discovery process; therefore, some of the features may be more discriminative of the applications and hence should receive a higher weight. Based on this weighting scheme, we now revise the dimensions of our vector space such that the frequency of each ASCII character observed in the payload is multiplied by the weight of the position that it was located. For instance, if $\infty$ is observed in positions 5 and 210, and the weight for 5 and 210 are 1 and 8, respectively, the value for the 236$^{th}$ dimension of the representative vector would be 9 rather than simply being 2. Accordingly, we have:
**Definition 4** (Extends Definition 2). \( B^{\omega} : \Psi \rightarrow \Lambda \) is a **learning classification machine** that maps a payload \( \Psi \) with frequency \( \omega \nu \) to a network application such as \( \lambda \in \Lambda \).

It is obvious that \( B' \) is a special case of \( B^{\omega} \) when \( \forall p \in [0, \eta], \quad \omega(\Psi_p) = 1 \). Now, since \( B^{\omega} \) is sensitive to the weights given to the positions in the payload, a method needs to be devised to compute the appropriate weights for the positions. For this purpose, we employ a genetic algorithm based process to find the weights.

Briefly stated, genetic algorithms are non-deterministic and chaotic search methods that use real world models to solve complex and at times intractable problems. In this method the optimal solution is found by searching through a population of different feasible solutions in several iterations. After the population is studied in each iteration, the best solutions are selected and are moved to the next generation through the applications of genetic operators. After adequate number of generations, better solutions dominate the search space therefore the population converges towards the optimal solution. We employ this process over the weights of the positions in the payload to find the optimal weights for each location. The process that we employ is explained in the following:

- The objective is to find the optimal weight vector \( \omega_0 \) from a payload \( \Psi \) with length of \( \eta \). So, initially a pool of random weight vectors of length \( \eta \) are generated: \( \omega_1, \ldots, \omega_n \);
Figure 4.4: The genetic algorithm-based method for finding the optimal weight vector.

- For each $\omega_i$ a learning classification machine ($8^{(\omega_i)}$) is devised given a set of learning instances;

- $8^{(\omega_i)}$ is evaluated based on its performance accuracy over a given set of test instances. The accuracy of $8^{(\omega_i)}$ represents the genetic fitness function; therefore, the accuracy of $8^{(\omega_i)}$ is the fitness of $\omega_i$ among the other weight vectors in the genetic algorithm pool of possible solutions;

- The best set of weights based on their fitness value are selected and are moved onto the next generation and the rest of the weight vectors are discarded;
• The genetic operators, i.e., the mutation and crossover operators, are applied on the weight vectors present in the genetic algorithm pool of possible solutions and new weight vector solution instances are generated;

• The process of the genetic algorithm is repeated for \( \text{Gen} \) generations;

• Once the algorithm reaches a steady state and stops, the weight vector with the best fitness is selected and will be used as the most appropriate weight vector \((\omega_0)\) for the application discovery process using \( \delta^{\omega_0} \).

In the above process shown in Figure 4.4, the weight vectors that are needed for the payload are defined using the genes in the genetic algorithm and the fitness function is defined as the accuracy of the learning classification machine developed based on that specific weight vector (gene). The outcome of the genetic algorithm process provides us with the optimal set of weights for the positions of the ASCII characters in the payloads. This optimal set of weights can be used to learn a classifier that can best find the network application for new payloads whose generating application is not known.

To implement this process, we employed the JGAP (Java Genetic Algorithms Package) software package [70]. In our experiments, reported in the following section, we apply the default crossover technique implemented in JGAP with the population size of 500 evolving for 50 generations. The default crossover rate is \([\text{population size}/2]\), and the value for mutation rate is 1/15. Moreover, we consider the first 256 bytes of source and destination payloads since the
rest of the payload does not contain any signature and decreases the classifier performance by including noise data.

4.3 Intrusion Detection Module

Intrusion detection has been extensively studied since the seminal work by Anderson [71]. Traditionally, intrusion detection techniques are classified into two categories: misuse (signature-based) detection and anomaly detection. Misuse detection is based on the assumption that a large number of cyber attacks leave a set of signatures in the stream of network packets or in audit trails, and thus attacks are detectable if these signatures can be identified by analyzing the audit trails or network traffic behavior. However, misuse detection is strictly limited to the known attacks and detecting new attacks is one of the biggest challenges faced by misuse detection.

To address the weakness of misuse detection, the concept of anomaly detection was formalized in the seminal report of Denning [6]. In this approach models of normal data are build based on normal traffic, and then the deviation from the normal model will be considered as an attack or anomaly. The main advantage of this approach over misuse detection is that it can detect attempts to exploit new and unforeseen vulnerabilities. It can also help detect “abuse of privileges” types of attacks that do not actually involve exploiting any security vulnerabilities. However, this approach has its own shortcomings.
As discussed in the beginning of this chapter, the original idea of anomaly detection, i.e. learning normal behavior and labeling outliers as anomalies, is not practical in real-life problems since it causes a high amount of false alarm rate. This is mainly because of two well-know issues: 1) Lack of a training data set that covers all legitimate areas; and 2) Abnormal behavior is not always an indicator of intrusions. It can happen as a result of factors such as policy changes or offering of new services by a site.

In order to overcome these challenges, and keep the advantages of misuse detection, some researchers have proposed the idea of hybrid detection. This way, the system will achieve the advantage of misuse detection to have a high detection rate on known attacks as well as the ability of anomaly detectors in detecting unknown attacks.

Although with respect to the characteristics of signature-based and anomaly-based methods, the fusion of these two approaches should theoretically provide an effective IDS, there are still two important issues that make this task cumbersome. First, anomaly-based methods cannot achieve an outstanding performance without a comprehensive labeled and up-to-date training set with all different attack types, which is very costly and time-consuming to create if not impossible. Second, efficient and effective fusion of several detection technologies becomes a big challenge for building an operational hybrid intrusion detection system.

In the rest of this section, we provide a novel solution to overcome the aforementioned problems.
4.3.1 Anomaly-based Detector

As the first step to have an effective anomaly detector, we should extract robust network features that have the potential to discriminate anomalous behavior from normal network activities. Since most current network intrusion detection systems use network flow data (e.g. netflow, sflow, ipfix) as their information sources, we focus on features generated based on these flows. The name and description of the applied features are listed in Table 4.2.

Having extracted the features, the next step is to find a very efficient classifier. Evaluating famous classifiers based on detection rate, false alarm rate, classification time, and learning time, we decided to employ C4.5 decision tree algorithm [69].

4.3.2 Signature-based Detector

As our first signature-based detector we chose Snort [5] because of its popularity and availability to researchers. However, our proposed hybrid detection scheme is completely independent from Snort, and any other signature-based detector can be used instead. As mentioned earlier, our anomaly-based detector works on flows. However, Snort is designed to work on packets. To make our detectors consistent, we matched Snort alerts with the existing flows based on the source IP, source port, destination IP, destination port, and time stamp. Since the flows and Snort alerts are generated by differ-
Table 4.2: Applied flow-based features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SrcIP</td>
<td>source IP address</td>
</tr>
<tr>
<td>DstIP</td>
<td>destination IP address</td>
</tr>
<tr>
<td>SrcPort</td>
<td>source port number</td>
</tr>
<tr>
<td>DstPort</td>
<td>destination port number</td>
</tr>
<tr>
<td>ProtocolName</td>
<td>the name of the protocol used in transport layer</td>
</tr>
<tr>
<td>ConnectionDuration</td>
<td>the period of time in milliseconds during which the connection is alive</td>
</tr>
<tr>
<td>SrcBytes</td>
<td>the total number of bytes sent from the source to the destination</td>
</tr>
<tr>
<td>DstBytes</td>
<td>the total number of bytes sent from the destination to the source</td>
</tr>
<tr>
<td>SrcPackets</td>
<td>the total number of packets sent from the source to the destination</td>
</tr>
<tr>
<td>DstPackets</td>
<td>the total number of packets sent from the destination to the source</td>
</tr>
<tr>
<td>SrcBytes/DstBytes</td>
<td>the ratio of “SrcBytes” to “DstBytes”</td>
</tr>
<tr>
<td>SrcPackets/DstPackets</td>
<td>the ratio of “SrcPackets” to “DstPackets”</td>
</tr>
<tr>
<td>SrcBytes/SrcPackets</td>
<td>the ratio of “SrcBytes” to “SrcPackets”</td>
</tr>
<tr>
<td>DstBytes/DstPackets</td>
<td>the ratio of “DstBytes” to “DstPackets”</td>
</tr>
</tbody>
</table>

ent devices, we were not very strict with the time stamps and considered a deviation of up to 5 seconds acceptable.

In addition to Snort, we employed QRadar Rules [64] as our second source of signature-based detection system. Analyzing the extracted information from network packets and flows, QRadar Rules provided a very strong mechanism to discover malicious activities hidden in network communications.

4.3.3 Proposed Hybrid Detector

The most important issues that current anomaly detectors deal with are firstly to prepare a comprehensive labeled data set, and secondly to keep that data set up-to-date. To solve these problems we have proposed to apply
the idea of adaptive learning. To meet this goal we have defined learning time intervals, e.g. 1 day, at the end of which the anomaly-based detector will be trained by the two most recent training sets. These training sets are the flows labeled by the hybrid detector in the previous intervals. Figure 4.5 illustrates the structure of the hybrid detector. In the first interval that there is no training set, the hybrid system only relies on the labels from signature-based detectors. These labels will be used as a training set for the anomaly-based detectors in the next time interval. During the second time interval, final labels of the hybrid system will be used as a training set to feed anomaly-based detectors. In order to optimize the performance of anomaly-based detectors, we have added a filtering module, Train Set Optimizer, to make sure that those records which have a support level of 50% or higher will be forwarded to the learners. Finally, we use a fusing algorithm to combine the results from signature-based and anomaly-based detectors. The final result will be both reported to the administrator and used as a portion of training set in the next time interval.

4.3.4 Fusing Algorithm

Investigating the state-of-the-art fusion methods to combine the detection result of various IDSs, we found out that the existing methods are not able to model the uncertainty attached to each detection method. Since current IDSs are not sufficiently accurate and generate a lot of false positives, there is always an uncertainty attached to them. This uncertainty, which caused
by the lack of knowledge about a system, is called Epistemic Uncertainty. As discussed in [72], classic probability theory is not capable of modeling Epistemic Uncertainty due to the following issues:

1. It requires the probability of all possible events. When the information is not available, the uniform distribution function is often used as justified by Laplace’s Principle of Insufficient Reason. This approach, however, will not produce accurate results as it assumes that all events
with unknown probability distributions are equally likely.

2. It is not capable of assigning probabilities to sets of events. As a result, the summation of all probabilities assigned to possible singleton elements must be equal to one. This assumption is not in line with epistemic uncertainty as it requires to assign probabilities to select sets of events.

4.3.4.1 Dempster’s Rule of Combination

Dempster-Shafer theory of evidence [73, 74] is the most widely used approach to address uncertainty in probability theory. Dempster-Shafer theory is considered as the generalization of probability theory where probabilities are assigned to sets as opposed to singleton elements. This features makes D-S theory capable of quantifying the lack of knowledge with regards to a certain phenomenon.

**Definition 5.** Given a sequence of basic probability assignments \( m_1, \ldots, m_n \), the generalized Dempster’s rule of combination is defined as:

\[
\begin{align*}
  \text{\( n \)-ary:} & \quad (\bigoplus_{i=1}^n m_i)(A) = \frac{1}{1 - \prod_{A_i=A}^{m_n} m_i(A_i)} - \prod_{A_i=A}^{m_n} m_i(A_i) \quad (4.2) \\
  \text{\( 1 \)-ary:} & \quad (\bigoplus_{i=1}^n m_i)(\emptyset) = 0 \quad (4.3) \\
  \mathcal{K} & = \prod_{A_i=\emptyset}^{m_n} m_i(A_i) \quad (4.4)
\end{align*}
\]
4.3.4.2 Dempster’s Rule of Combination for Binary Variables

Although Dempster’s rule of combination is one of the most effective fusion methods, it is computationally very complex, which prevents it being widely used in real-world applications. To overcome the shortcoming of Dempster’s rule, Srivastava proposed an alternative form of Dempster’s rule for combining sources of evidence that pertain to binary variables [75].

**Theorem 1.** Let \( m_1, m_2, \ldots, m_n \) be basic probability assignments of \( n \) independent sources defined on a binary frame of discernments \( \theta = x, \neg x \). The generalized Dempster’s rule of combination can be simplified to:

\[
\begin{align*}
\text{a}^0: & \quad (m_i)(x) = 1 - \frac{n^n}{i=1} (1 - m_i(x))/K \quad (4.5) \\
\text{a}^0: & \quad (m_i)(\neg x) = 1 - \frac{n^n}{i=1} (1 - m_i(\neg x))/K \quad (4.6) \\
\text{a}^0: & \quad (m_i)(\theta) = \frac{n^n}{i=1} m_i(\theta)/K \quad (4.7)
\end{align*}
\]

where \( K \) is defined as:

\[
K = \frac{n^n}{i=1} (1 - m_i(x)) + \frac{n^n}{i=1} (1 - m_i(\neg x)) - \frac{n^n}{i=1} m_i(\theta) \quad (4.8)
\]

In order to provide an efficient mechanism for the fusion of intrusion de-
tectors, we employ the general form of Dempster’s rule of combination for binary variables (Theorem 1). The detailed proof of this Theorem is provided
4.3.4.3 Quantification Method

In order to benefit from Dempster’s rule of combination, one needs to quantify the support (belief) of labels assigned to each set. As most of the intrusion detection systems are not capable of providing support for their final decision, we have proposed a general approach to quantify the decisions made by IDSs. The idea of our approach is based on the fact that intrusive activities are not isolated but related as different stages of attack sequences, with the early stages preparing for the later ones. As a result, intrusions will cause a series of alarms to be generated in detection systems. These series of alarms can be usually related through IP addresses and port numbers.

Taking advantage of this similarity amongst related alarms, we propose a method to cluster network traffic based on four important features: 1) source IP address; 2) destination IP address; 3) source port number; and 4) destination port number. Each cluster will then be analyzed separately to assign a support factor to each network flow labeled as either “normal” or “intrusive”.

Since well-know distance functions such as Euclidean distance are not suitable to measure the similarity between either IP addresses or port numbers, two customized distance function are introduced to provide an effective approach for measuring the similarities.

**Definition 6.** Let $\Omega_1 = (\omega_1, \omega_2, \omega_3, \omega_4)$ and $\Omega_2 = (\omega'_1, \omega'_2, \omega'_3, \omega'_4)$ be either the source or destination IP addresses from two different network flows.
Assuming $\Gamma$ is a set of four binary variables, $\Gamma = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4\}$, the distance between $\Omega_1$ and $\Omega_2$ can be defined as:

$$\Delta(\Omega_1, \Omega_2) = \sum_{k=1}^{4} \gamma_i 2^{4-i} / 2^{4-i}$$

(4.9)

where

1) $\gamma_1 = 0 \iff (\omega_1 = \omega_1^i) \land (\omega_2 = \omega_2^i) \land (\omega_3 = \omega_3^i) \land (\omega_4 = \omega_4^i)$

2) $\gamma_2 = 0 \iff (\omega_1 = \omega_1^i) \land (\omega_2 = \omega_2^i) \land (\omega_3 = \omega_3^i)$

3) $\gamma_3 = 0 \iff (\omega_1 = \omega_1^i) \land (\omega_2 = \omega_2^i)$

4) $\gamma_4 = 0 \iff (\omega_1 = \omega_1^i)$

Definition 7. Let $\rho_1$ and $\rho_2$ be either the source or destination port numbers from two different network flows. Assuming $\Gamma$ is a set of two binary variables, $\Gamma = \{\gamma_1, \gamma_2\}$, the distance between $\rho_1$ and $\rho_2$ can be defined as:

$$\Delta(\rho_1, \rho_2) = \sum_{k=1}^{2} \gamma_i 2^{2-i} / 2^{2-i}$$

(4.10)

where

1) $\gamma_1 = 0 \iff \rho_1 = \rho_2$

2) $\gamma_2 = 0 \iff (\rho_1 \leq 1024) \land (\rho_2 \leq 1024) \land (\rho_1 \neq \rho_2)$

The experimental results on test data sets illustrate a huge improvement over Euclidean distance. It’s also observed that the exponential structure of the defined distance functions is very effective in dividing similar flows into separate clusters.
For the experiments, we employed X-Means clustering algorithm [76] as it achieved the best performance of evaluated clustering algorithms during the preliminary experiments. In addition to the clustering algorithm, choosing the proper parameters is of high importance. After several round of experiments, we decided to configure the X-Means clustering algorithm with the following parameters:

- Minimum number of clusters: 5
- Maximum number of clusters: 20
- Cluster update interval ($\lambda$): 3600000 ms
- Cluster time window ($\omega$): 7200000 ms

Rebuilding the clusters in every hour ($\lambda$) using the flows from last two hours ($\omega$) will result in a strong connection between the new incoming flows and the clusters. In addition, clusters are continuously updated with the new flows in order to maximize the performance.

Once a new flow is labeled by the individual detectors, it will be added to the cluster with the highest degree of similarity. We then calculate the attack/normal ratio for each detector based on the normal vs. intrusive flows in that cluster labeled by the same detector. As an example, let us assume that there is a new flow labeled as an attack by Snort and is classified in Cluster 2. To calculate the attack/normal ratio, we count the number of flows in Cluster 2 labeled as either attack or normal by Snort.
**Definition 8.** Let $\xi$ and $\upsilon$ be attack and normal ratio of a network flow, $\varphi$, respectively. Assuming $\Xi$ and $\Upsilon$ are the average number of attacks and normal flows in the environment. Degree of support for flow $\varphi$, which is labeled by $\Lambda$, is defined as:

\[
\text{if } \Lambda = \text{attack} \Rightarrow \text{Support}_{\varphi}(\Lambda) = \xi^n \tag{4.11}
\]

\[
\text{if } \Lambda = \text{normal} \Rightarrow \text{Support}_{\varphi}(\Lambda) = \upsilon^m \tag{4.12}
\]

where

\[
n = \log(0.50) / \log\left(\frac{\Xi}{\Xi + \Upsilon}\right) \tag{4.13}
\]

\[
m = \log(0.50) / \log\left(\frac{\Upsilon}{\Xi + \Upsilon}\right) \tag{4.14}
\]

### 4.4 Performance Analysis

The proposed framework does not impose any performance constrains on the hybrid detection system, and we can always achieve a better performance by applying more efficient algorithms. However, to have a better understanding of a system’s performance limitation and bottlenecks, we analyze the performance of the implemented detector employed for the evaluation. Figure 4.6 illustrates the products and machine learning techniques that are used for the experimental results.

In our analysis, we do not consider any of the employed commercial products
Figure 4.6: Evaluated Hybrid Detection System

as they are designed to perform in corporate networks and will not affect the performance of our system. With that said, we only focus on the performance of Traffic Classification, Anomaly Detection, Data Fusion modules.

The learning part of the traffic classification module, i.e. extracting application signatures, requires gathering enough samples from each application type and should be done off-line. Considering that we have $m$ attributes and $n$ training instances, the total cost of building a tree is $O(m \cdot n \cdot \log n) + O(n \cdot (\log n)^2)$. As in the traffic classification module we deal a fixed number of features, the time complexity of building a tree would be in the order of
$O(n \ (\log n)^2)$. On the other hand, the time complexity of the application classifier is $O(\log n)$ in the worst case scenario. However, in practice, the depth of the tree is usually not more than 25, which means classification can be done in the order of $O(1)$.

Similarly, the same analysis can be applied to anomaly detection module as they are both based on C4.5 decision tree.

As discussed in the previous section, although the Dempster’s rule of combination for binary variables is very simple and does not impose any complexity to the system, we need to apply a clustering algorithm on the incoming network flows in order to quantify the decisions of detection systems. To this end, we employ X-Means clustering algorithm on the network flows of pervious time interval and then classify the incoming flows based on their similarity with the existing clusters. Considering that we apply the clustering algorithm on 4 attributes and the maximum number of clusters is limited to 50, the total cost of building the clusters is $O(n^{4+(50+1)} \ \log n)$, where $n$ is the total number of network flows in each time interval. This process is off-line and should be done once in each time interval. However, the classification of new flows into the best-fit cluster can be done in the order of $O(1)$.

### 4.5 Summary

In this chapter, we introduced a Multi-layer Intrusion Detection System (MIDS) that overcomes the main shortcomings of the existing IDSs.
As one of our main contributions, we proposed an online traffic classification method, in which the unigram payload distribution model is applied to extract the required set of features. Thereafter the J48 decision tree is employed to classify the network applications based on the unigram features. Having a detail analysis of application signatures, we observed that the signatures are present in some designated positions in the payload, and it is important to place more weight on the features that appear in these more important positions. This is achieved through a weighted scheme over the features. However, finding the appropriate weights is a challenging task. To this end, we employ a genetic algorithm based scheme to find the weights.

We also proposed a new hybrid network intrusion detection framework, combining Snort and QRadar as a signature based detector and a decision tree algorithm (C4.5) as an anomaly detector. Applying Dempster’s rule of combination and with along a clustering-based quantification method, we have introduced an efficient fusing algorithm. Moreover, we presented an adaptive automatic method of training anomaly detectors using the labeled network flows of pervious time intervals.
Chapter 5

Data Set Preparation

As mentioned in Section 2.4.2 of Chapter 2, recent studies show that there are some inherent problems in DARPA and KDD data sets which are widely used as the only publicly available data sets for network-based anomaly detection systems.

5.1 NSL-KDD Data Set

Conducting a thorough analysis of the recent research trend in anomaly detection, one will encounter several machine learning methods reported to have a very high detection rate of 98% while keeping the false alarm rate at 1% [40]. However, when we look at the state of the art IDS solutions and commercial tools, there are a few products using anomaly detection approaches. Practitioners still believe that it is not a mature technology yet. To find the
Table 5.1: Statistics of redundant records in the KDD train set

<table>
<thead>
<tr>
<th></th>
<th>Original Records</th>
<th>Distinct Records</th>
<th>Reduction Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attacks</td>
<td>3,925,650</td>
<td>262,178</td>
<td>93.32%</td>
</tr>
<tr>
<td>Normal</td>
<td>972,781</td>
<td>812,814</td>
<td>16.44%</td>
</tr>
<tr>
<td>Total</td>
<td>4,898,431</td>
<td>1,074,992</td>
<td>78.05%</td>
</tr>
</tbody>
</table>

reason of this contrast, we studied the details of the research done in anomaly detection and considered various aspects such as learning and detection approaches, training data sets, testing data sets, and evaluation methods. Our study shows that there are some inherent problems in the KDDCUP’99 data set [41], which is widely used as one of the few publicly available data sets for network-based anomaly detection systems. In this section we perform a set of experiments to show the existing deficiencies in KDD.

5.1.1 Redundant Records

One of the most important deficiencies in the KDD data set is the huge number of redundant records, which causes the learning algorithms to be biased towards the frequent records, and thus prevent them from learning infrequent records, which are usually more harmful to networks such as U2R and R2L attacks. In addition, the existence of these repeated records in the test set will cause the evaluation results to be biased by the methods which have better detection rates on the frequent records.
Table 5.2: Statistics of redundant records in the KDD test set

<table>
<thead>
<tr>
<th></th>
<th>Original Records</th>
<th>Distinct Records</th>
<th>Reduction Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attacks</td>
<td>250,436</td>
<td>29,378</td>
<td>88.26%</td>
</tr>
<tr>
<td>Normal</td>
<td>60,591</td>
<td>47,911</td>
<td>20.92%</td>
</tr>
<tr>
<td>Total</td>
<td>311,027</td>
<td>77,289</td>
<td>75.15%</td>
</tr>
</tbody>
</table>

To solve this issue, we removed all the repeated records in the entire KDD train and test set, and kept only one copy of each record. Tables 5.1 and 5.2 illustrate the statistics of the reduction of repeated records in the KDD train and test sets, respectively.

While doing this process, we encountered two invalid records in the KDD test set, number 136,489 and 136,497. These two records contain an invalid value, ICMP, as their service feature. Therefore, we removed them from the KDD test set.

5.1.2 Level of Difficulty

The typical approach for performing anomaly detection using the KDD data set is to employ a customized machine learning algorithm to learn the general behavior of the data set in order to be able to differentiate between normal and malicious activities. For this purpose, the data set is divided into test and training segments, where the learner is trained using the training portion of the data set and is then evaluated for its efficiency on the test portion.
Many researchers within the general field of machine learning have attempted to devise complex learners to optimize accuracy and detection rate over the KDD'99 data set. In a similar approach, we have selected seven widely used machine learning techniques, namely J48 decision tree learning [69], Naive Bayes [77], NBTree [78], Random Forest [79], Random Tree [80], Multi-layer Perceptron [81], and Support Vector Machine (SVM) [82] from the Weka [68] collection to learn the overall behavior of the KDD'99 data set. For the experiments, we applied Weka’s default values as the input parameters of these methods.

Investigating the existing papers on the anomaly detection which have used the KDD data set, we found that there are two common approaches to apply KDD. In the first, KDD'99 training portion is employed for sampling both the train and test sets. However, in the second approach, the training samples are randomly collected from the KDD train set, while the samples for testing are arbitrarily selected from the KDD test set.

In order to perform our experiments, we randomly created three smaller subsets of the KDD train set each of which included fifty thousand records of information. Each of the learners were trained over the created train sets. We then employed the 21 learned machines (7 learners, each trained 3 times) to label the records of the entire KDD train and test sets, which provides us with 21 predicated labels for each record. Further, we annotated each record of the data set with a #successfulPrediction value, which was initialized to zero. Now, since the KDD data set provides the correct label for each record,
we compared the predicted label of each record given by a specific learner with the actual label, where we incremented #successfulPrediction by one if a match was found. Through this process, we calculated the number of learners that were able to correctly label that given record. The highest value for #successfulPrediction is 21, which conveys the fact that all learners were able to correctly predict the label of that record. Figure 5.1 and 5.2 illustrate the distribution of #successfulPrediction values for the KDD train and test sets, respectively.

It can be clearly seen from Figure 5.1 and 5.2 that 97.97% and 86.64% of the records in the KDD train and test sets have been correctly labeled by all 21 classifiers. The obvious observation from these figures is that the application of typical learning machines to this data set would result in high accuracy rates. This shows that evaluating methods on the basis of accuracy, detection rate and false positive rate on the KDD data set is not an appropriate option.
Figure 5.2: The distribution of #successfulPrediction values for the KDD data set records

Figure 5.3: The performance of the selected learning machines on KDDTest
Figure 5.4: The performance of the selected learning machines on KDDTest$^+$

Figure 5.5: The performance of the selected learning machines on KDDTest$^{-21}$
Table 5.3: Statistics of randomly selected records from KDD train set

<table>
<thead>
<tr>
<th></th>
<th>Distinct Records</th>
<th>Percentage</th>
<th>Selected Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-5</td>
<td>407</td>
<td>0.04</td>
<td>407</td>
</tr>
<tr>
<td>6-10</td>
<td>768</td>
<td>0.07</td>
<td>767</td>
</tr>
<tr>
<td>11-15</td>
<td>6,525</td>
<td>0.61</td>
<td>6,485</td>
</tr>
<tr>
<td>16-20</td>
<td>58,995</td>
<td>5.49</td>
<td>55,757</td>
</tr>
<tr>
<td>21</td>
<td>1,008,297</td>
<td>93.80</td>
<td>62,557</td>
</tr>
<tr>
<td>Total</td>
<td>1,074,992</td>
<td>100.00</td>
<td>125,973</td>
</tr>
</tbody>
</table>

5.1.3 Our Solution

To solve the issues mentioned in the previous section, we first removed all the redundant records in both train and test sets. Furthermore, to create a more challenging subset of the KDD data set, we randomly sampled records from the #successfulPrediction value groups shown in Figure 5.1 and 5.2 in such a way that the number of records selected from each group is inversely proportional to the percentage of records in the original #successfulPrediction value groups. For instance, the number of records in the 0-5 #successfulPrediction value group of the KDD train set constitutes 0.04% of the original records, therefore, 99.96% of the records in this group are included in the generated sample. Tables 5.3 and 5.4 show the detailed statistics of randomly selected records.

The generated data sets, KDDTrain+ and KDDTest+, included 125,973 and 22,544 records, respectively. Furthermore, one more test set was generated
Table 5.4: Statistics of randomly selected records from KDD test set

<table>
<thead>
<tr>
<th></th>
<th>Distinct Records</th>
<th>Percentage</th>
<th>Selected Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-5</td>
<td>589</td>
<td>0.76</td>
<td>585</td>
</tr>
<tr>
<td>6-10</td>
<td>847</td>
<td>1.10</td>
<td>838</td>
</tr>
<tr>
<td>11-15</td>
<td>3,540</td>
<td>4.58</td>
<td>3,378</td>
</tr>
<tr>
<td>16-20</td>
<td>7,845</td>
<td>10.15</td>
<td>7,049</td>
</tr>
<tr>
<td>21</td>
<td>64,468</td>
<td>83.41</td>
<td>10,694</td>
</tr>
<tr>
<td>Total</td>
<td>77,289</td>
<td>100.00</td>
<td>22,544</td>
</tr>
</tbody>
</table>

that did not include any of the records that had been correctly classified by all 21 learners, KDDTest$^{-21}$, which incorporated 11,850 records. For experimental purposes, we employed the first 20% of the records in KDDTrain$^+$ as the train set, having trained the learning methods, we applied the learned models on three test sets, namely KDDTest (original KDD test set), KDDTest$^+$ and KDDTest$^{-21}$. The result of the evaluation of the learners on these data sets are shown in Figures 5.3, 5.4 and 5.5, respectively.

As can be seen in Figure 5.3, the accuracy rate of the classifiers on KDDTest is relatively high. This shows that the original KDD test set is skewed and unproportionately distributed, which makes it unsuitable for testing network-based anomaly detection classifiers. The results of the accuracy and performance of learning machines on the KDD’99 data set are hence unreliable and cannot be used as good indicators of the ability of the classifier to serve as a discriminative tool in network-based anomaly detection. On the con-
trary, KDDTest\textsuperscript{+} and KDDTest\textsuperscript{−} test set provide more accurate information about the capability of the classifiers. As an example, classification of SVM on KDDTest is 65.01\% which is quite poor compared to other learning approaches. However, SVM is the only learning technique whose performance is improved on KDDTest\textsuperscript{+}. Analyzing both test sets, we found that SVM wrongly detects one of the most frequent records in KDDTest, which highly affects its detection performance. In contrast, in KDDTest\textsuperscript{+} since this record is only occurred once, it does not have any effects on the classification rate of SVM, and provides better evaluation of learning methods.

The new version of KDD data set, NSL-KDD is publicly available for researchers through our website\footnote{http://iscx.cs.unb.ca/NSL-KDD/}. Although, the data set still suffers from some of the problems discussed by McHugh [45] and may not be a perfect representative of existing real networks, because of the lack of public data sets for network-based IDSs, we believe it still can be applied as an effective benchmark data set to help researchers compare different intrusion detection methods.

5.2 Benchmark Data Sets

Due to the criticism of existing data sets and also privacy issues of employing real traffic, preparing a data set has become one of the biggest challenges in the area of intrusion detection. Although applying NSL-KDD data set as an
 interim solution for the preliminary results of this research was quite helpful, the results were not fully reliable due to the inherent issues of KDD data set. As a result, we decided to generate a benchmark data set in a testbed environment. In order to simulate a real network environment, we employ real machines with various operating systems. We then analyzed real traces to create profiles for agents that simulate real traffic for HTTP, SMTP, SSH, IMAP, POP3, and FTP. Having generated normal traffic in our testbed, we carried out various state-of-the-art multi-step attacks to represents current cyber-threads that organizations are dealing with.

The main advantages of our generated benchmark data set are listed in the following:

- **Realistic Traffic:** It is determined that the insertion of post-capture traces and merging them into the real traffic will cause some inconsistencies in network packet parameter such as TTL and sequence numbers. To prevent these issues, both normal and malicious traffic are generated using physical devices.

- **Labeled Data Set:** Having a labeled data set plays an important role in the evaluation of detection systems. As a result, we put a lot of effort to control our testbed environment to distinguish anomalous activities from normal behavior.

- **Total Traffic Capture:** Intrusion detection systems require various types of information to provide an optimal result. Thus, it is very
Table 5.5: Specification of testbed workstation

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Operating System</th>
<th>Service Pack</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>Windows XP</td>
<td>Service Pack 1</td>
</tr>
<tr>
<td>2</td>
<td>Windows XP</td>
<td>Service Pack 2</td>
</tr>
<tr>
<td>1</td>
<td>Windows XP</td>
<td>Service Pack 3</td>
</tr>
<tr>
<td>1</td>
<td>Windows 7</td>
<td>No Service Pack</td>
</tr>
</tbody>
</table>

important to capture both internal and external network interactions to make sure evaluated IDSs can be fed with all required information.

- **Diverse Attack Scenarios:** The data set is enriched with various types of multi-step attacks that are currently threatening cyber environments. Each scenario is carefully handpicked to evaluate different capabilities of detection systems.

## 5.2.1 Network Architecture

As illustrated in Figure 5.6, our testbed network consists of 21 workstations divided into four distinct VLANs. This will help us reduce the broadcast domain for each workstation. All workstations run on select versions of the Windows operating system which is listed in Table 5.5. The reason behind this selection is to be able to apply known exploits to attack the machines. The fifth VLAN contains our servers to provide web, email, DNS, and Network Address Translation (NAT). The NAT server (192.168.5.124) acts as an access provider, for the whole network, to external networks. Being the point of entry of our network, it effectively provides firewall facilities to block unauthorized access while permitting authorized communications. The NAT
server is connected to the Internet through two valid IPs. One IP is multiplexed for the workstations accessing the Internet and the other is designated solely for the Linux Server. Our Linux Server (192.168.5.122) is responsible for the website, email and DNS services. The Windows Server (192.168.5.123) is responsible for internal ASP.NET applications. Both the Linux and NAT Servers are running on Ubuntu 10.04, while the Windows Server is running on Microsoft Windows Server 2003. Table 5.6 specifies more details regarding our servers and the various service providers installed on them.

The sixth LAN, whose traffic is not captured, enabled us to conduct non-disruptive monitoring and maintenance tasks such as loading applications,
tuning certain services among others.

A single layer 3 switch is utilized to provide the required layer 2 and 3 switching and also, the required mirroring of traffic. All connections are explicitly set at 10 Mbps. This was seen as a required setting for the networked devices to operate effectively while keeping the maximum throughput well below the maximum switching capacity. This measure was taken to reduce the probability of packets being dropped by the switch and also capturing devices.

An Ethernet tap (running at 100 Mbps) was efficiently employed to transmit the mirrored traffic to multiple devices without any processing overhead or disruption. These devices provided the means for redundant capturing (e.g. tcpdump), alert generation through various Intrusion Detection Systems (IDS) (e.g. Snort), IDS management systems (e.g. QRadar, OSSIM), and visualization (e.g. ntop).

### 5.2.2 Data Generation

In order to generate realistic background traffic, we captured and analyzed four weeks worth of network activity associated with Information Security

<table>
<thead>
<tr>
<th>Server</th>
<th>Operating System</th>
<th>Services</th>
<th>Providers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux Server</td>
<td>Ubuntu 10.04</td>
<td>Web</td>
<td>Apache 2.2.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>eMail Postfix (MTA) and Dovecot (IMAP/POP3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DNS Bind 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SSH OpenSSH v5.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FTP vsftpd v2.2.2</td>
</tr>
<tr>
<td>NAT Server</td>
<td>Ubuntu 10.04</td>
<td>NAT</td>
<td>IPTables</td>
</tr>
<tr>
<td>Windows Server</td>
<td>Windows Server 2003</td>
<td>Web</td>
<td>IIS v6</td>
</tr>
</tbody>
</table>

Table 5.6: Specification of testbed servers
Center of Excellence (ISCX). This was to initially determine the composition of the traffic in terms of applications used and protocols utilized. Figure 5.7 depicts the arrangement of protocols and applications used throughout the capturing period for our center.

**Centre's Traffic Composition**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Amount (percentage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP</td>
<td>97.737%</td>
</tr>
<tr>
<td>(R) ARP</td>
<td>1.393%</td>
</tr>
<tr>
<td>IPX</td>
<td>0.066%</td>
</tr>
<tr>
<td>IPv6</td>
<td>0.486%</td>
</tr>
<tr>
<td>STP</td>
<td>0.186%</td>
</tr>
<tr>
<td>Other</td>
<td>0.132%</td>
</tr>
</tbody>
</table>

**IP traffic Composition**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Amount (Percentage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>95.112%</td>
</tr>
<tr>
<td>UDP</td>
<td>4.792%</td>
</tr>
<tr>
<td>ICMP</td>
<td>0.067%</td>
</tr>
<tr>
<td>ICPv6</td>
<td>0.010%</td>
</tr>
<tr>
<td>Other</td>
<td>0.019%</td>
</tr>
</tbody>
</table>

**TCP/UDP traffic Composition**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Amount (Percentage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP</td>
<td>6.482%</td>
</tr>
<tr>
<td>HTTP</td>
<td>87.402%</td>
</tr>
<tr>
<td>DNS</td>
<td>3.214%</td>
</tr>
<tr>
<td>Netbios</td>
<td>0.856%</td>
</tr>
<tr>
<td>Mail</td>
<td>0.429%</td>
</tr>
<tr>
<td>SNMP</td>
<td>0.073%</td>
</tr>
<tr>
<td>SSH</td>
<td>1.416%</td>
</tr>
<tr>
<td>Messenger</td>
<td>0.081%</td>
</tr>
<tr>
<td>Other</td>
<td>0.047%</td>
</tr>
</tbody>
</table>

**Figure 5.7: Data set statistics**

The following applications were chosen to be simulated in our testbed; HTTP, SMTP, POP3, IMAP, SSH, and FTP. Other traffic such as NetBios and DNS, among others, are readily generated as the indirect consequence of utilizing real machines and infrastructure. Besides, Yahoo Messenger and
Skype applications were used manually to provide a complete collection of applications.

To fully simulate the behavior of a single users with an agent, we analyzed the behavior of a real user on specific applications namely, HTTP, SMTP, POP3, IMAP, SSH, and FTP. We then extracted the data distribution of different features such as request intervals and application payload length. However, by analyzing the extracted information, we failed to observer a certain statistical distribution over the data considering the following distributions: a) Normal; b) Beta; c) Weibull; d) Erlang; e) Triangular; f) Gamma; g) Exponential; h) Uniform; i) Lognormal.

Moreover, even the behavior of a single user varies during the weekdays. These outcomes are inline with previous research that have suggested more complex distributions are required to model HTTP activity [83].

To prevent these issues, we applied an inverse transformation method [84] which is proven to be more accurate compared to applying well-known distributions.

Having prepared the required infrastructure, services and simulation agents, we started capturing the traffic at 00:01:06 on Friday June 11th, 2010. The capturing lasted for an exact duration of 7 days and ended at 00:01:06 on Friday June 18th, 2010. Attacks were subsequently conducted during this period. The number of flows per seconds is depicted in Figure 5.8, and the overall statistics for our data set is illustrated in Figure 5.9.
Figure 5.8: Number of flows seen per second during the capturing period

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Friday</td>
<td>378667</td>
<td>88427623</td>
<td>16033149459</td>
<td>12921796</td>
<td>8569988</td>
<td>TCP</td>
<td>78796</td>
<td>2478</td>
<td>37.363</td>
<td>3399688</td>
</tr>
<tr>
<td>Saturday</td>
<td>133193</td>
<td>311170654</td>
<td>414820848</td>
<td>3599451</td>
<td>2340671</td>
<td>UDP</td>
<td>90117</td>
<td>37.966</td>
<td>27.244</td>
<td>1032822</td>
</tr>
<tr>
<td>Sunday</td>
<td>275528</td>
<td>433396736</td>
<td>788170674</td>
<td>6734950</td>
<td>4703358</td>
<td>ICMP</td>
<td>31221266</td>
<td>54076</td>
<td>37.4</td>
<td>2157222</td>
</tr>
<tr>
<td>Monday</td>
<td>171380</td>
<td>1051987567</td>
<td>6187034339</td>
<td>87482751160</td>
<td>75685415</td>
<td>L2L</td>
<td>81192</td>
<td>37.309</td>
<td>27.244</td>
<td>1032822</td>
</tr>
<tr>
<td>Tuesday</td>
<td>57198</td>
<td>190513956</td>
<td>2284285536</td>
<td>13254945</td>
<td>434674</td>
<td>L2R</td>
<td>441563</td>
<td>623</td>
<td>37.237</td>
<td>2157222</td>
</tr>
<tr>
<td>Wednesday</td>
<td>522263</td>
<td>789247707</td>
<td>17897978768</td>
<td>14625339</td>
<td>9916103</td>
<td>TCP</td>
<td>87070</td>
<td>513</td>
<td>37.309</td>
<td>2157222</td>
</tr>
<tr>
<td>Thursday</td>
<td>397595</td>
<td>607242240</td>
<td>12510135591</td>
<td>10323340</td>
<td>329378</td>
<td>UDP</td>
<td>67658</td>
<td>547</td>
<td>37.309</td>
<td>2157222</td>
</tr>
</tbody>
</table>

Total: 2450324 5982515383 8748275160 75685415 49667763 1941454 498032 10689 301893 2127168 2381532 68792

Figure 5.9: Data set statistics

5.2.3 Attack Scenarios

Since the generated data set is intended to be used in the area of intrusion detection, we aim to provide a diverse set of attack scenarios representing state-of-the-art multi-step attacks being conducted by malicious hackers. Performed on a separate day, each attack scenario includes the following steps to simulate the real-world attacks;

1. Information gathering and reconnaissance (passive and active)
2. Vulnerability identification and scanning
3. Gaining access and compromising a system
4. Maintaining access and creating backdoors
5. Covering tracks

Reconnaissance or information gathering is the first step in conducting an attack and is perhaps one of the most important ones. Information gathering
can be carried out in both passive and active modes. In passive information gathering, we intend to gather information regarding our target network through secondary Internet sources without directly connecting to the network itself. This includes information gained through WHOIS and DNS databases such as namespaces, server types, server locations, and incorporated software technologies. In active information gathering the boundaries of the target network are identified and are elaborated in greater detail in later stages. Identifying the range of IP addresses or determining if the target network is behind a NAT server is conducted in this stage. Also if our target network possesses an email server, creating a list of users would be a good idea and can be useful in performing social engineering attacks in later stages.

Prior to performing any exploits and after initial reconnaissance, we need to identify the vulnerabilities of the systems residing inside the target network. This requires some information about the type and version of the operating system, along with the running services and open ports. Later this information can be used to query databases containing an up-to-date list of vulnerabilities. Network scanners such as nmap can be used to identify active hosts and services running inside a network while proprietary vulnerability scanners such as Nessus maintain a database of recent security vulnerabilities updated on a daily basis.

Having done with the vulnerability assessment, one can exploit the existing vulnerabilities through exploitation tools such as Metasploit. Once the at-
tacker has acquired access to the network, maintaining that access is often of high priority since the compromised system may simply reboot, leading to the loss of the initial connection or even get patched, forbidding any future attack attempts. Dumping password hash files and installing remote shell connections are typical activities in maintaining access to a computer. Meterpreter is an advanced payload that is integrated in the Metasploit Framework and can be considered as a powerful tool in not only maintaining access to a compromised host but also performing post exploitation tasks such as uploading and downloading files, file system interaction, network pivoting and port forwarding.

It is in the intent of every attacker to move through a network as quietly as possible, raising the least number of alarms and leaving minimum tracks behind. Editing or deleting system or application level log files is often a useful mechanism in avoiding detection by system administrators. In the following we will explain four attack scenarios conducted in our data set based on the aforementioned steps. All four attack scenarios have been carefully designed based on the aforementioned steps but not all attack scenarios necessarily constitute all five. In order to be as realistic as possible, later attack scenarios are based on the results of earlier attacks making them sophisticated, powerful, and harder to detect.
5.2.3.1 Scenario 1: Infiltrating the network from the inside

Many of today’s networks are built on what is called the eggshell principle: hard on the outside and soft on the inside. This means that if an attacker gains access to a host on the inside, she can then use the compromised host as a pivot to attack systems not previously accessible via the Internet such as a local intranet server or a domain controller.

Our attack scenario starts by gathering information about our target including network IP ranges, nameservers, mail servers and user email accounts. This is achieved by querying the DNS for resource records using network administrative tools like nslookup and dig.

Having done the initial reconnaissance, we found out the only system on our target network exposed to the Internet is a NAT server. This makes internal users of our target network inaccessible from outside since the NAT server acts as a firewall, dropping any initial connections from outside the network. This is where conventional attacking techniques are not useful and we are forced into using client side attacks. Based on the nslookup output we can start to enumerate through the mail server guessing potential email addresses that we require to penetrate into the system.

We will be using the Adobe Reader `util.printf()` buffer overflow vulnerability as a starting point for our scenario. An attacker can exploit this vulnerability to execute arbitrary code with the privileges of the user running the application. We create a malicious PDF file using Metasploit and embed a Meterpreter reverse TCP shell on port 5555 inside it. The PDF file is
attached to a system upgrade email and sent on behalf of admin@t3lab.com to all 21 users of the testbed. We have also set up a listener on port 5555 to capture the reverse connection on our attacking machine. Clicking on the file opens Adobe but shows a grey window that never reveals a PDF but instead makes a reverse TCP connection back to our attacking computer listening on port 5555.

The first session is initiated from user5 of the testbed, connecting back to us by exploiting the Adobe Reader vulnerability. Now that we are in, we need to start identifying potential targets such as internal servers or backup databases. We upload nmap to user5 using our Meterpreter session and start to scan potential hosts on two consecutive subnets (192.168.1.0/24 and 192.168.2.0/24). User12 is identified as running Windows XP SP1 with a vulnerable SMB authentication protocol on port 445. This vulnerability is exploited and a scan of the server subnet (192.168.5.0/24) is performed from user12. This scan identifies a Windows Server 2003 running an internal Web application using MSSQL Server as its backend database with only port 80 opened. A remote desktop connection to user12 is created to access the Web application and is tunneled back to the attacking machine using the previously established Meterpreter session.

When a machine has only port 80 opened, even the most advanced vulnerability scanner cannot identify anything useful. This leads to the use of Web application hacking techniques like SQL injection or Cross Site Scripting. SQL injection is the most common type of Web application hacking which
takes advantage of non-validated input vulnerabilities and attempts to pass SQL commands through a web application for execution by the backend database. We start the attack with a single quote trick and input in the username textbox something like this: 'having 1=1--. The Web application responds with an error message signaling a potential vulnerability. We continue by using the information in the error messages to identify the column names and types of the user table. This is achieved by using multiple union and aggregate queries and analyzing their respected errors. We finally manage to create a username tree with password hello and insert it into the database. After logging in and taking a look at different sections of the Web application, we decided to delete the user table to avoid any future logins from the testbed users. This is achieved by injecting a simple query to the database: 'drop table user;--. In order to be able to continue the attacks in later days and avoid undergoing the burdensome process of compromising an internal host, a backdoor is created and configured to connect back to the attacker's machine.

Metasploit's msfencode is used to create a standalone executable of a reverse TCP shell connection and is uploaded to user12 through the already established Meterpreter session. Now we can schedule the backdoor to connect back to us at whatever interval we desire. Scheduling tasks in Windows can be done through the command line using the schtasks command. The backdoor executable is run by the operating system every one hour while making a TCP connection back to the attacker. This marks the final stage
of our first attack scenario as we exit all previously established connections.

5.2.3.2 Scenario 2: HTTP Denial of Service

The second attack scenario is designed towards performing a stealthy, low bandwidth denial of service attack without the need to flood the network. We utilize Slowloris as our main tool in this scenario as it has proven to make Web servers completely inaccessible using a single machine. Slowloris starts by making a full TCP connection to the remote server. The tool holds the connection open by sending valid, incomplete HTTP requests to the server at regular intervals to keep the sockets from closing. Since any Web server has a finite ability to serve connections, after some time all sockets will be used up and no other connection can be made.

We start the attack by running a TCP listener on our attacking machine (different from the previous scenario to have a diverse set of attacking IPs) and waiting for the operating system of user12 to run our backdoor scheduled task as previously configured in the final stage of scenario 1. Upon receiving a connection from user12 a scan of the 192.168.3.0/24 subnet is performed to identify potential vulnerable hosts. User13 is identified as running a vulnerable SMB authentication protocol on port 445. We then exploit this vulnerability and tunnel back a remote desktop connection of user13 to our attacking machine. Slowloris is downloaded from one of our servers and is run against the Apache server hosting the testbed's t3lab.com domain (192.168.5.122). The attack is run for approximately ten minutes leading
to the inaccessibility of the server. The same procedure is taken to exploit users 1, 6, 15 and 17 as Slowloris is downloaded and run on each machine resulting in a Denial of Service attack against the Apache Web server.

5.2.3.3 Scenario 3: Distributed Denial of Service using an IRC Botnet

Botnets are an emerging threat to organizations of all kinds as they are the driving force behind spam, financial information theft and malware distribution. However, botnets pose a much greater risk as their power is in the ability to perform Distributed Denial of Service attacks against corporate networks.

This scenario is designed with the goal of performing a distributed attack using infected hosts on the testbed. An Internet Relay Chat (IRC) bot, written from scratch by our team, is sent as an attachment for an update message to testbed users. The bot has the ability to download malicious programs from remote servers and to execute them with user privileges. A separate Denial of Service program has also been written to perform an HTTP GET attack on a specific target. The program is multithreaded, meaning that each bot can possibly emulate the behavior of hundreds of users making the attack as powerful as possible. The attack is initiated when user12 connects back to our attacking machine using the backdoor configured in the final stage of scenario 1. An IRC server is uploaded to user12 via the Meterpreter session and is configured to start listening on port 6667. An IRC
client is also uploaded to allow for commands to be sent to bots on infected machines. Within a period of 30 minutes, seven users have connected to the IRC server including users 3, 5, 9, 10, 13, 18 and 20. Bots connect to a specific channel on the IRC server with a nickname constructed using the hash of their primary MAC address. We order the bots to download the HTTP GET program using the download httpget IP command. As each user finishes its download, a download complete message is sent over the channel. The Distributed Denial of Service attack is started using the run http IP threads command and is targeted towards the main Apache Web server. The attack is run for a period of 60 minutes leading to inaccessibility or slowing down of the server. The attack is stopped using the stop http command and bots are ordered to quit and to disconnect.

5.2.3.4 Scenario 4: Dictionary Attack against SSH

Brute force attacks are very common against networks as they tend to break into accounts with weak username and password combinations. Our final scenario has also been designed with the goal of acquiring an SSH account by running a dictionary brute force attack against the t3lab.com server. We use brutessh as the main tool for this scenario as it can be easily configured to use our custom made dictionary list. The dictionary is composed of over 5000 alphanumerical entries of varying length. We run the attack for a 30 minute time period resulting in a sudo user account credentials being returned. The user credentials are used to login to the server and is used to download the
/etc/passwd and /etc/shadow files.

5.2.4 Capturing Traffic

In our network configuration, a single layer 3 switch (Omniswitch 6850) was used as the connecting point for all machines. Virtual LANs were configured on the switch to effectively implement the necessary LANs. This enabled us to capture virtually all communication between all of the machines, a priority objective of this experiment.

Directing all of the traffic through a single interface on the switch has its disadvantages, most importantly the dropping of packets due to congestion. To circumvent this matter, we subsequently lowered the speed of all the interfaces on the switch to 10 Mbps, effectively lowering the overall throughput. Our preliminary test runs indicated no substantial effect on the overall quality of the data set. Consequently lower hardware specifications were required to capture, monitor and analyze the traffic, thus enabling the use of commodity and off-the-shelf hardware. As an example, we utilized a simple Ethernet tap to replicate the mirrored traffic, on-the-fly, for various monitoring devices.

5.3 Summary

In this chapter, we have provide two data sets to address some inherent problems in DARPA and KDD data sets which are widely used as the only publicly available data sets for network-based anomaly detection systems.
The first data set, which is called NSL-KDD, consists of selected records of the entire KDD data set. Although, the proposed data set still suffers from some of the problems discussed by McHugh and may not be a perfect representative of existing real networks, because of the lack of public data sets for network-based IDSs, we believe it still can be applied as an effective benchmark data set to help researchers compare different intrusion detection methods.

As our next effort, we have generated a benchmark data set in a testbed environment. In order to simulate a real network environment, we employ real machines with various operating systems. We then analyzed real traces to create profiles for agents that simulate real traffic for HTTP, SMTP, SSH, IMAP, POP3, and FTP. Having generated normal traffic in our testbed, we carried out various state-of-the-art multi-step attacks to represent current cyber-threads that organizations are dealing with.
Chapter 6

Framework Evaluation

In this chapter, we provide the evaluation results of our proposed framework. In order to have a detailed comparison of our hybrid system, we conduct the experiments in three phases. In the first phase, we focus on the Traffic Classification Module and provide a thorough analysis of the classification module using two benchmark data sets. A comprehensive performance analysis of Intrusion Detection Module is conducted in phase 2. This includes a detailed study on the performance of detection module using the prepared benchmark data set which was explained in Chapter 5. Having studied the performance of each module individually, in the last phase, we analyze the overall performance of our proposed hybrid system.
6.1 Phase 1: Evaluation of Traffic Classification Module

To evaluate our proposed method we prepared two data sets from various networks. The first data set is prepared using the traffic we capture in the Information Security Center of Excellence (ISCX) in University of SOUTH AUSTRALIA. For the experiments, we replayed 10-day traffic of ISCX network through QRadar [64]. Having extracted the corresponding flows using QRadar, we fed the flows into MeterFlow [67] to get the application labels. We then filtered out the unknown flows which were about 28% of total flows, and kept the rest for the experiments. Similarly, for our second data set we used 3-hour captured traffic from a large-scale ISP network\(^1\). Tables 6.1 and 6.2 summarize the workload of the IS CX and ISP network over an hour, respectively. Passing this traffic through MeterFlow we ended up with 35% unknown flows since it is an ISP network and contains a lot of peer to peer applications which are really hard to detect by the state of the art signature- based traffic classifiers.

Having prepared the network flows with known applications, we extracted the unigram features of the source and destination payloads to evaluate our decision tree based classifier. However, to have enough samples of each application for training, we decided to only keep those with more than 200 records in the data set. As a result, in our first data set we ended up with

\(^1\)For privacy issues we do not reveal the name of this ISP
Table 6.1: Workload of the ISCX network over an hour

<table>
<thead>
<tr>
<th>Src. IPs</th>
<th>Dst. IPs</th>
<th>Flows</th>
<th>Packets</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>951</td>
<td>756</td>
<td>16,391</td>
<td>9,541,197</td>
</tr>
</tbody>
</table>

Table 6.2: Workload of the ISP network over an hour

<table>
<thead>
<tr>
<th>Src. IPs</th>
<th>Dst. IPs</th>
<th>Flows</th>
<th>Packets</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>17,456</td>
<td>11,585</td>
<td>84,082</td>
<td>2,184,236</td>
<td>1,074,802,236</td>
</tr>
</tbody>
</table>

7 applications namely, FTP, HTTP Image Transfer, HTTP Web, DNS, Secure Web, SSH, Web Media Documents. Applying the same approach for the second data set we are left with 14 applications namely, Bit Torrent, HTTP Image Transfer, HTTP Web, DNS, Secure Web, MSN, MS-SQL, NTP, Oracle, POP, SMTP, Windows File Sharing, Yahoo, Web Media Documents.

For the experiments we applied the J48 decision tree classifier. In the first step we evaluated our classifier using the payload unigram features with equal weights. For the evaluation we employed 10-fold cross-validation to obtain a reliable result. We then applied the genetic algorithm technique to find the appropriate weights to obtain higher accuracy. The experiment took approximately five days to complete since the classifier model should be updated during each run of the fitness function which takes a few seconds.

Table 6.3: Accuracy of the proposed method for the ISCX and ISP networks

<table>
<thead>
<tr>
<th></th>
<th>ISCX Network</th>
<th>ISP Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Accuracy</td>
<td>81.93%</td>
<td>81.72%</td>
</tr>
<tr>
<td>First Generation Accuracy</td>
<td>83.69%</td>
<td>80.74%</td>
</tr>
<tr>
<td>Optimal Accuracy</td>
<td>90.97%</td>
<td>86.55%</td>
</tr>
</tbody>
</table>
As illustrated in Table 6.3, the best fitness value returned is 90.97% and 86.55% for ISCX and ISP networks, respectively. This means that we have achieved about 9% and 5% performance increase in our applied data sets, respectively.

Table 6.3 illustrates the improvement obtained by applying the genetic algorithm. In this Table, base accuracy shows the performance of the classifier using equal weights. First generation accuracy is the average fitness value in the first generation. Finally, optimal accuracy represents the highest accuracy obtained using a genetic algorithm.

Figures 6.1 and 6.2 show the evolution of each generation in the experiments on the ISCX and ISP networks, respectively. In the first data set, Figure 6.1(a), the fitness value of the best individual for each generation has a noticeable increase until generation 15. However, in the next 35 generations there is only about 1% increase in the fitness value. Similarly, as it is illustrated in 6.2(a), for the ISP network there is an approximate steady increase until generation 17, at which point it is apparent that almost the best possible weights have been achieved.

The results presented in this paper show that the genetic algorithm is a promising method to find the appropriate weights for our unigram payload model. We also believe that the weighted unigram model can be applied as an effective intrusion detection method to find the malicious activities on the Internet.
Figure 6.1: Fitness value vs. generation number calculated for the ISCX network

Figure 6.2: Fitness value vs. generation number calculated for the ISP network
6.2 Phase 2: Evaluation of Intrusion Detection Module

In this phase of our evaluation, we focus on the performance of the Intrusion Detection Module independently. This means that incoming traffic will not be separated on the Traffic Classification Module, and they all will be analyzed using the same general purpose intrusion detection systems.

As explained in Chapter 4, for the experiments we decided to employ the C4.5 decision tree algorithm [69] as our anomaly detection system. As the signature based detector, we chose Snort [5] because of its popularity and availability to researchers. Similar to any other intrusion detection solution, we conducted a log of fine tuning to enhance the performance of Snort. As an example, since our FTP traffic was not encrypted, during the first experimental setup, Snort generated a lot of false positives warning that the FTP traffic is sent in clear text format. After several rounds of tuning, we succeeded in gaining a 43% reduction in the number of false positives generated by Snort.

In addition to Snort, we employed QRadar Rules [64] as our second source of signature-based detection system. Analyzing the extracted information from network packets and flows, QRadar Rules provided a very strong mechanism to discover malicious activities hidden in network communications.

As discussed in Chapter 5, due to the inherent issues of publicly available data sets, we decided to prepare our own benchmark data set in a testbed
Table 6.4: Distribution of normal vs. intrusive flows

<table>
<thead>
<tr>
<th></th>
<th>Attack</th>
<th>Normal</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td>0</td>
<td>358,944</td>
<td>358,944</td>
</tr>
<tr>
<td>Day 2</td>
<td>2,076</td>
<td>131,212</td>
<td>133,288</td>
</tr>
<tr>
<td>Day 3</td>
<td>13,044</td>
<td>127,970</td>
<td>141,014</td>
</tr>
<tr>
<td>Day 4</td>
<td>4,446</td>
<td>167,984</td>
<td>172,430</td>
</tr>
<tr>
<td>Day 5</td>
<td>0</td>
<td>564,983</td>
<td>564,983</td>
</tr>
<tr>
<td>Day 6</td>
<td>0</td>
<td>512,764</td>
<td>512,764</td>
</tr>
<tr>
<td>Day 7</td>
<td>4,663</td>
<td>347,575</td>
<td>352,238</td>
</tr>
<tr>
<td>Entire Week</td>
<td>24,229</td>
<td>2,211,432</td>
<td>2,235,661</td>
</tr>
</tbody>
</table>

environment. Although the generated data set is prepared in a testbed environment, we believe it is a good representative of real traffic in our Information Security Center of Excellence (ISCX). In order to simulate a real network environment, we employed real machines with various operating systems. We then analyzed real traces to create profiles for agents that simulate real traffic for HTTP, SMTP, SSH, IMAP, POP3, and FTP. Having generated normal traffic in our testbed, we carried out various state-of-the-art multi-step attacks to represent current cyber-threats that organizations are dealing with. Figure 6.3 and Table 6.4 illustrate the distribution of normal versus intrusive activities during the seven days of experiments.

As indicated in Table 6.4, there is no attack which is conducted during Day 1, Day 5, and Day 6. Apparently, during these 3 days, detectors with lower number of false positive are preferred. On the contrary, during the other 4 days, a combination of higher true positives (TP) and lower false positives (FP) determine which IDS is more efficient. Figure 6.4 illustrates an overall performance of all detectors during the 7 days of evaluation.

In order to reflect the most recent multi-step attacks being conducted by
Figure 6.3: Distribution of normal and intrusive flows
Figure 6.4: Overall performance of the detectors
malicious hackers, we provide a diverse set of attack scenarios representing state-of-the-art multi-step attacks being conducted by malicious hackers. A brief explanation of the attack types conducted in each day is provided in the following:

• **Day 2**

  1. Initial reconnaissance by querying the DNS Server;
  2. Exploiting the Adobe Reader buffer overflow vulnerability;
  3. Scanning the hosts using nmap from an internal vulnerable node;
  4. Conducting SQL injection attack against the identified Web Server.

• **Day 3**

  1. Making a backdoor connection to an internal vulnerable node;
  2. Scanning the hosts using nmap from the exploited node;
  3. Identifying a node vulnerable to SMB authentication protocol on port 445;
  4. Exploiting the SMB vulnerability on an internal machine;
  5. Lunching a stealthy denial of service (DoS) attack against the Apache Web Server using Slowloris.

• **Day 4**

  1. Making a backdoor connection to an internal vulnerable node;
  2. Installing IRC Server on the the exploited node;
3. Scanning the hosts using nmap from the exploited node;
4. Exploiting the SMB vulnerability on 7 internal machines and installing IRC client on them;
5. Launching a distributed denial of service (DDoS) attack against the Main Web Server for about 60 minutes.

- **Day 7**

  1. Running a dictionary brute force attack against the SSH port on the t3lab.com server;
  2. Downloading the “passwd” file which includes the list of all users and groups on the attacked Linux Server.

In order to have a fair comparison of the applied two signature-based detectors, the anomaly detector and our proposed hybrid system, the captured traffic during the 7 days was fed to these detectors and the results were stored in MySQL database. In the remainder of this section, we provide a detailed analysis on the performance of the employed detectors during each day.

**Analysis of Day 1:** Since there has been no attack conducted on Day 1 (see Table 6.5), we only focus on the false positives (FP) generated by each detector. During this day, both Snort and QRadar generated more than 3,000 false alarms. However, because part of the observed suspicious traffic had more in common with normal traffic, the anomaly detector was able to cut the false positives to almost half. Although the anomaly detector had the best performance of all detectors during Day 1, it did not have a high
Table 6.5: Performance of detectors on Day 1

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snort</td>
<td>0</td>
<td>3,186</td>
<td>0</td>
<td>355,758</td>
</tr>
<tr>
<td>QRadar</td>
<td>0</td>
<td>3,774</td>
<td>0</td>
<td>355,170</td>
</tr>
<tr>
<td>Anomaly</td>
<td>0</td>
<td>1,476</td>
<td>0</td>
<td>357,468</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0</td>
<td>2,835</td>
<td>0</td>
<td>356,109</td>
</tr>
</tbody>
</table>

impact on reducing the false positives on the hybrid detector as in many of the cases Snort and QRadar together had more support compared to the anomaly detector.

**Analysis of Day 2:** The attack that is conducted on Day 2 includes four steps. The first step which is querying the DNS Servers for gathering the information is quite normal and cannot be detected by any detectors. Therefore, we did not tag any of the DNS network flows as an attack in our database. However, the next three steps should be definitely identified by detectors as intrusive activities.

Analyzing the behavior of signature-based detectors, we found out that Snort has a very high detection rate with an acceptable low rate of false alarms. Further detailed analysis indicates that the breach of Adobe Reader buffer overflow is successfully detected by Snort as the proper signature has been already added to the known-attack database. Besides, Snort seems to have a very effective mechanism of detecting the traffic from IP/port scanners. However, the SQL injection attack which consists of 3 network flows are not detected by Snort.

On the contrary, as indicated in Table 6.6, QRadar shows a very poor per-
Table 6.6: Performance of detectors on Day 2

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snort</td>
<td>2,073</td>
<td>265</td>
<td>3</td>
<td>130,947</td>
</tr>
<tr>
<td>QRadar</td>
<td>0</td>
<td>3,945</td>
<td>2,076</td>
<td>127,267</td>
</tr>
<tr>
<td>Anomaly</td>
<td>149</td>
<td>2,369</td>
<td>1,927</td>
<td>128,843</td>
</tr>
<tr>
<td>Hybrid</td>
<td>1,461</td>
<td>3,015</td>
<td>615</td>
<td>128,197</td>
</tr>
</tbody>
</table>

performance in detecting any of the attacks during Day 2. In addition, it has generated a very high number of false alarms. This poor performance of QRadar has also affected the training set of the anomaly detector, which leads to having a poor performance on the anomaly detector as well.

Overall, the final results from the hybrid detector is very promising. This is mainly because Snort labels have a higher degree of support compared to QRadar and the anomaly detector.

**Analysis of Day 3:** Similar to Day 2, Snort has shown a high detection rate on the IP/Port scanning activities, the SMB vulnerabilities, and the backdoor connection. However, Snort has not been able to detect the last and the most important step of the attack which is lunching the stealthy DoS against the Apache Web Server.

On the contrary, QRadar has achieved an incredible success in detecting denial of service attack as a result of having powerful rules that keep track of the initiated connection to each host. As illustrated in Table 6.7, the overall detection rate of the hybrid system is very impressive since the combination of Snort and QRadar has covered almost all the attack steps in Day 3. Although the effect of the anomaly detector during this day is not very noticeable, it
Table 6.7: Performance of detectors on Day 3

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snort</td>
<td>3,474</td>
<td>311</td>
<td>9,570</td>
<td>127,659</td>
</tr>
<tr>
<td>QRadar</td>
<td>10,818</td>
<td>3,413</td>
<td>2,226</td>
<td>124,557</td>
</tr>
<tr>
<td>Anomaly</td>
<td>1,804</td>
<td>1,776</td>
<td>11,240</td>
<td>126,194</td>
</tr>
<tr>
<td>Hybrid</td>
<td>11,758</td>
<td>2,601</td>
<td>1,286</td>
<td>125,369</td>
</tr>
</tbody>
</table>

Table 6.8: Performance of detectors on Day 4

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snort</td>
<td>701</td>
<td>1,167</td>
<td>3,745</td>
<td>166,817</td>
</tr>
<tr>
<td>QRadar</td>
<td>2,586</td>
<td>4,528</td>
<td>1,860</td>
<td>163,456</td>
</tr>
<tr>
<td>Anomaly</td>
<td>180</td>
<td>1,872</td>
<td>4,266</td>
<td>166,112</td>
</tr>
<tr>
<td>Hybrid</td>
<td>2,729</td>
<td>4,001</td>
<td>1,717</td>
<td>163,983</td>
</tr>
</tbody>
</table>

has an acceptable impact on reducing the false alarm rates.

**Analysis of Day 4:** Analyzing the results in Table 6.8, we see a very similar pattern with the previous day as most of the steps are conducted in the same way with two exceptions. First, since the communication between the bots is done using IRC protocol, this traffic looks very normal and none of the detectors are able to identify it. Second, since in the distributed denial of service attack (DDoS), the packets are forwarded from a large group of machines in different location, this type of attack is more difficult to identify. As a result, QRadar performance is not as impressive as that of day 3.

**Analysis of Day 5:** Similar to Day 1, we only focus on the false positives (FP) generated by each detector as there has been no attack conducted on Day 5 (see Table 6.9. During this day, Snort has generated 13,630 false alarms which is the highest number of the entire week amongst all detectors. Similarly, QRadar has generated 7,547 false positives which is a very
Table 6.9: Performance of detectors on Day 5

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snort</td>
<td>0</td>
<td>13,630</td>
<td>0</td>
<td>551,353</td>
</tr>
<tr>
<td>QRadar</td>
<td>0</td>
<td>7,547</td>
<td>0</td>
<td>557,436</td>
</tr>
<tr>
<td>Anomaly</td>
<td>0</td>
<td>1,481</td>
<td>0</td>
<td>563,502</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0</td>
<td>10,597</td>
<td>0</td>
<td>554,386</td>
</tr>
</tbody>
</table>

Table 6.10: Performance of detectors on Day 6

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snort</td>
<td>0</td>
<td>901</td>
<td>0</td>
<td>511,863</td>
</tr>
<tr>
<td>QRadar</td>
<td>0</td>
<td>4,404</td>
<td>0</td>
<td>508,360</td>
</tr>
<tr>
<td>Anomaly</td>
<td>0</td>
<td>1,602</td>
<td>0</td>
<td>511,162</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0</td>
<td>3,348</td>
<td>0</td>
<td>509,416</td>
</tr>
</tbody>
</table>

poor performance compared to the other days of the experiment. However, because part of the observed suspicious traffic had more in common with normal traffic, the anomaly detector was able to cut the false positives to 1,481 which is very promising. Although the anomaly detector had the best performance of all detectors during Day 5, it did not have a high impact on reducing the false positives on the hybrid detector as in many of the cases Snort and QRadar together had more support compared to the anomaly detector.

**Analysis of Day 6:** As illustrated in Table 6.10, Snort and Anomaly detectors have an acceptable false alarm rate during Day 6. However, QRadar has relatively generated a higher number of false positives, which is led to a worse performance of the hybrid detector.

**Analysis of Day 7:** During Day 7, a dictionary attack is launched against the SSH port on the t3lab.com server. After trying different username and
Table 6.11: Performance of detectors on Day 7

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snort</td>
<td>934</td>
<td>794</td>
<td>3,729</td>
<td>346,781</td>
</tr>
<tr>
<td>QRadar</td>
<td>805</td>
<td>4,758</td>
<td>3,858</td>
<td>342,817</td>
</tr>
<tr>
<td>Anomaly</td>
<td>1</td>
<td>1,707</td>
<td>4,662</td>
<td>345,868</td>
</tr>
<tr>
<td>Hybrid</td>
<td>1,241</td>
<td>4,584</td>
<td>3,422</td>
<td>342,991</td>
</tr>
</tbody>
</table>

passwords for about an hour, the attacker was successful in gaining access to the server and down the *passwd* file. Apparently, since SSH traffic is encrypted, none of the detectors are able to identify the transfer of the *passwd* file over the link. However, they all should be able to detect the dictionary attack against the Linux Server.

Analyzing the results in Table 6.11, we find that all detectors have shown a poor performance in detecting the intrusive SSH traffic. In addition, QRadar has also generated a high number of false positives and has dramatically affected the performance of the hybrid detector in terms of having an acceptable low number of false positives.

In summary, overall analysis of the detectors shows that the hybrid detector has the best performance of all detectors in terms of identifying the intrusive traffic. This is indicated in Figures 6.5(a) and 6.5(d) where hybrid detector has the highest value of true positives and the lowest share of the false negatives which presents the number of attacks that are not identified by the detectors.

As illustrated in 6.5(b), the anomaly detector has the best performance of all detectors in terms of keeping the false alarms at the very low level. This
strong performance has had a positive impact in decreasing the false positive rate of the hybrid detector. However, the poor performance of QRadar with regards to the false alarms caused the hybrid detector not to achieve the best performance.

### 6.3 Phase 3: Overall Performance of Hybrid Detection System

The purpose of this phase of evaluation is to measure the impact of applying application-based detectors on the overall performance of the system. For the experiments, we employ Ax3soft Sax2 [85] as a signature-based detec-
tor to analyze web traffic. Sax2 is a network-based intrusion detection system which is mainly designed to detect web-based intrusions such as CGI/WWW attacks and SQL injection. In addition, we employ C4.5 decision tree algorithm [69] as our anomaly detection system.

For the rest of the network traffic, we use the general purpose hybrid detector which is explained in the previous section. This detector is composed of Snort, QRadar and C4.5 detectors. Table 6.12 illustrates the distribution of web-based traffic during the 7 days of experiments.

As illustrated in Figure 6.6, network packets will be first sent to Packet Analysis Module in which they will be grouped as network flows, and each flow will be labeled by an application name. The output will be then forwarded to the Intrusion Detection Module for investigation of intrusive activities. Depending on the application type, flows will be either analyzed by web-based or general purpose detectors. Table 6.13 summarizes the application groups that are considered as web traffic for our experiments.

In the remainder of this section, we provide a detailed analysis on the performance of the hybrid detector in general mode (phase 2) vs. application

<table>
<thead>
<tr>
<th>Day</th>
<th>Web Traffic</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td>285,816</td>
<td>358,944</td>
<td>79.63%</td>
</tr>
<tr>
<td>Day 2</td>
<td>82,932</td>
<td>133,288</td>
<td>62.22%</td>
</tr>
<tr>
<td>Day 3</td>
<td>92,545</td>
<td>141,014</td>
<td>65.63%</td>
</tr>
<tr>
<td>Day 4</td>
<td>113,927</td>
<td>172,430</td>
<td>66.07%</td>
</tr>
<tr>
<td>Day 5</td>
<td>408,054</td>
<td>564,983</td>
<td>72.22%</td>
</tr>
<tr>
<td>Day 6</td>
<td>418,377</td>
<td>512,764</td>
<td>81.59%</td>
</tr>
<tr>
<td>Day 7</td>
<td>283,057</td>
<td>352,238</td>
<td>80.36%</td>
</tr>
<tr>
<td>Entire Week</td>
<td>1,684,708</td>
<td>2,235,661</td>
<td>75.36%</td>
</tr>
</tbody>
</table>
specific mode (phase 3).

**Analysis of Day 1:** As indicated in Table 6.14), out of 3,186 false positives generated by Snort only 4 of them was related to web traffic. Similarly, there are only 435 false positives by QRadar that are categorized as web traffic. As a result, applying an application-base IDS on web traffic does not have a huge impact on the overall performance of the detector.

**Analysis of Day 2:** As a large portion of the attacks during this day are not web-based such as Adobe Reader buffer overflow and port scanning, in terms of statistics we do not gain a lot applying the application-based IDS.
Table 6.13: Web-based application groups

<table>
<thead>
<tr>
<th>SecureWeb</th>
<th>WebMediaAudio</th>
<th>WebMediaVideo</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTPImageTransfer</td>
<td>WebFileTransfer</td>
<td>WebMediaDocuments</td>
</tr>
</tbody>
</table>

Table 6.14: Performance of detectors in phase 3 on Day 1

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snort</td>
<td>0</td>
<td>3,182</td>
<td>0</td>
<td>69,946</td>
</tr>
<tr>
<td>QRadar</td>
<td>0</td>
<td>3,339</td>
<td>0</td>
<td>69,789</td>
</tr>
<tr>
<td>Anomaly (General)</td>
<td>0</td>
<td>1,453</td>
<td>0</td>
<td>71,675</td>
</tr>
<tr>
<td>Hybrid (General)</td>
<td>0</td>
<td>2,778</td>
<td>0</td>
<td>70,350</td>
</tr>
<tr>
<td>Sax2</td>
<td>0</td>
<td>51</td>
<td>0</td>
<td>285,765</td>
</tr>
<tr>
<td>Anomaly (Web)</td>
<td>0</td>
<td>35</td>
<td>0</td>
<td>285,781</td>
</tr>
<tr>
<td>Hybrid (Web)</td>
<td>0</td>
<td>47</td>
<td>0</td>
<td>285,769</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0</td>
<td>2,825</td>
<td>0</td>
<td>356,119</td>
</tr>
</tbody>
</table>

However, as indicated in Table 6.15, the web-based IDS is able to detect the SQL injection attack which was missed by Snort in phase 2.

**Analysis of Day 3:** There are about 676 web-based intrusive flows in day 3, which are related to the Slowloris stealthy denial of service attack. Out of 676 flows, only 159 and 299 flows were detected by Snort and QRadar, respectively in phase 2. However, as illustrated in Table 6.16, applying a

Table 6.15: Performance of detectors in phase 3 on Day 2

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snort</td>
<td>2,073</td>
<td>264</td>
<td>0</td>
<td>48,019</td>
</tr>
<tr>
<td>QRadar</td>
<td>0</td>
<td>3,538</td>
<td>2,073</td>
<td>44,745</td>
</tr>
<tr>
<td>Anomaly (General)</td>
<td>149</td>
<td>2,335</td>
<td>1,924</td>
<td>45,948</td>
</tr>
<tr>
<td>Hybrid (General)</td>
<td>1,461</td>
<td>2,987</td>
<td>612</td>
<td>45,296</td>
</tr>
<tr>
<td>Sax2</td>
<td>3</td>
<td>24</td>
<td>0</td>
<td>82,905</td>
</tr>
<tr>
<td>Anomaly (Web)</td>
<td>0</td>
<td>11</td>
<td>3</td>
<td>82,918</td>
</tr>
<tr>
<td>Hybrid (Web)</td>
<td>3</td>
<td>16</td>
<td>0</td>
<td>82,913</td>
</tr>
<tr>
<td>Hybrid</td>
<td>1,464</td>
<td>3,003</td>
<td>612</td>
<td>128,209</td>
</tr>
</tbody>
</table>

139
Table 6.16: Performance of detectors in phase 3 on Day 3

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snort</td>
<td>3,315</td>
<td>309</td>
<td>9,053</td>
<td>35,792</td>
</tr>
<tr>
<td>QRadar</td>
<td>10,519</td>
<td>3,009</td>
<td>1849</td>
<td>33,092</td>
</tr>
<tr>
<td>Anomaly (General)</td>
<td>1,743</td>
<td>1,511</td>
<td>10,625</td>
<td>34,590</td>
</tr>
<tr>
<td>Hybrid (General)</td>
<td>11,321</td>
<td>2,533</td>
<td>1,047</td>
<td>33,568</td>
</tr>
<tr>
<td>Sax2</td>
<td>537</td>
<td>93</td>
<td>139</td>
<td>91,776</td>
</tr>
<tr>
<td>Anomaly (Web)</td>
<td>383</td>
<td>66</td>
<td>293</td>
<td>91,803</td>
</tr>
<tr>
<td>Hybrid (Web)</td>
<td>527</td>
<td>71</td>
<td>149</td>
<td>91,798</td>
</tr>
<tr>
<td>Hybrid</td>
<td>11,848</td>
<td>2,604</td>
<td>1,196</td>
<td>125,366</td>
</tr>
</tbody>
</table>

Table 6.17: Performance of detectors in phase 3 on Day 4

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snort</td>
<td>659</td>
<td>1,166</td>
<td>1,624</td>
<td>55,054</td>
</tr>
<tr>
<td>QRadar</td>
<td>2,035</td>
<td>4,059</td>
<td>248</td>
<td>52,161</td>
</tr>
<tr>
<td>Anomaly (General)</td>
<td>168</td>
<td>1,851</td>
<td>2,115</td>
<td>54,369</td>
</tr>
<tr>
<td>Hybrid (General)</td>
<td>2,117</td>
<td>3,924</td>
<td>166</td>
<td>52,296</td>
</tr>
<tr>
<td>Sax2</td>
<td>1,813</td>
<td>203</td>
<td>350</td>
<td>111,561</td>
</tr>
<tr>
<td>Anomaly (Web)</td>
<td>1,136</td>
<td>88</td>
<td>1,027</td>
<td>111,676</td>
</tr>
<tr>
<td>Hybrid (Web)</td>
<td>1,748</td>
<td>57</td>
<td>415</td>
<td>111,707</td>
</tr>
<tr>
<td>Hybrid</td>
<td>3,865</td>
<td>3,981</td>
<td>581</td>
<td>164,003</td>
</tr>
</tbody>
</table>

web-based IDS solution we were able to successfully detect 527 flows without a major change in the total number of false positives.

Analysis of Day 4: Employing Sax2 to monitor the web traffic has a noticeable improvement in day 4 to detect the distributed denial of service attack against the Apache Web Server. As can be seen in Table 6.17, the new configuration in phase 3 has resulted in having 1,136 more flows successfully detected as intrusive while keeping the false alarm rate at the same level.

Analysis of Day 5: As indicated in Table 6.18, there is not a big difference on the performance of the detector during phase 2 and 3. This is mainly because there is no attack during this day, and web traffic are quite distinctive from attacks signatures.
Table 6.18: Performance of detectors in phase 3 on Day 5

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snort</td>
<td>0</td>
<td>13,572</td>
<td>0</td>
<td>143,357</td>
</tr>
<tr>
<td>QRadar</td>
<td>0</td>
<td>6,916</td>
<td>0</td>
<td>150,013</td>
</tr>
<tr>
<td>Anomaly (General)</td>
<td>0</td>
<td>1,277</td>
<td>0</td>
<td>155,652</td>
</tr>
<tr>
<td>Hybrid (General)</td>
<td>0</td>
<td>10,112</td>
<td>0</td>
<td>146,817</td>
</tr>
<tr>
<td>Sax2</td>
<td>0</td>
<td>83</td>
<td>0</td>
<td>407,971</td>
</tr>
<tr>
<td>Anomaly (Web)</td>
<td>0</td>
<td>29</td>
<td>0</td>
<td>408,025</td>
</tr>
<tr>
<td>Hybrid (Web)</td>
<td>0</td>
<td>37</td>
<td>0</td>
<td>408,017</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0</td>
<td>10,149</td>
<td>0</td>
<td>554,834</td>
</tr>
</tbody>
</table>

Table 6.19: Performance of detectors in phase 3 on Day 6

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snort</td>
<td>0</td>
<td>721</td>
<td>0</td>
<td>93,666</td>
</tr>
<tr>
<td>QRadar</td>
<td>0</td>
<td>3,914</td>
<td>0</td>
<td>90,473</td>
</tr>
<tr>
<td>Anomaly (General)</td>
<td>0</td>
<td>1,391</td>
<td>0</td>
<td>92,996</td>
</tr>
<tr>
<td>Hybrid (General)</td>
<td>0</td>
<td>2,947</td>
<td>0</td>
<td>91,440</td>
</tr>
<tr>
<td>Sax2</td>
<td>0</td>
<td>69</td>
<td>0</td>
<td>418,308</td>
</tr>
<tr>
<td>Anomaly (Web)</td>
<td>0</td>
<td>55</td>
<td>0</td>
<td>418,322</td>
</tr>
<tr>
<td>Hybrid (Web)</td>
<td>0</td>
<td>58</td>
<td>0</td>
<td>418,319</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0</td>
<td>3,005</td>
<td>0</td>
<td>509,759</td>
</tr>
</tbody>
</table>

**Analysis of Day 6:** Similar to Day 5, there is not a noticeable difference on the performance of the detector during phase 2 and 3 (see Table 6.19).

**Analysis of Day 7:** Analyzing the results in Table 6.20, we find that there is no change in the detection rate of hybrid systems during phase 2 and 3. The reason being that the attacks on Day 7 are conducted on SSH service which does not include any web traffic. There is only a slight change in the number of false positives as Sax2 performs better than QRadar on web traffic.

In conclusion, we have gained a slightly better performance employing application-specific detectors. During Day 2, the web-based detector was successful in detecting SQL injection attack. However, since there are only three flows
Table 6.20: Performance of detectors in phase 3 on Day 7

<table>
<thead>
<tr>
<th>Detector</th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snort</td>
<td>934</td>
<td>682</td>
<td>3,729</td>
<td>63,836</td>
</tr>
<tr>
<td>QRadar</td>
<td>805</td>
<td>3,817</td>
<td>3,858</td>
<td>60,701</td>
</tr>
<tr>
<td>Anomaly (General)</td>
<td>1</td>
<td>1,387</td>
<td>4,662</td>
<td>63,131</td>
</tr>
<tr>
<td>Hybrid (General)</td>
<td>1,241</td>
<td>4,203</td>
<td>3,422</td>
<td>60,315</td>
</tr>
<tr>
<td>Sax2</td>
<td>0</td>
<td>27</td>
<td>0</td>
<td>283,030</td>
</tr>
<tr>
<td>Anomaly (Web)</td>
<td>0</td>
<td>21</td>
<td>0</td>
<td>283,036</td>
</tr>
<tr>
<td>Hybrid (Web)</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>283,035</td>
</tr>
<tr>
<td>Hybrid</td>
<td>1,241</td>
<td>4,225</td>
<td>3,422</td>
<td>343,350</td>
</tr>
</tbody>
</table>

Table 6.21: Comparison of detection rates between phase 2 and 3

<table>
<thead>
<tr>
<th></th>
<th>Day 1</th>
<th>Day 2</th>
<th>Day 3</th>
<th>Day 4</th>
<th>Day 5</th>
<th>Day 6</th>
<th>Day 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase II</td>
<td>N/A</td>
<td>70.38%</td>
<td>90.14%</td>
<td>61.38%</td>
<td>N/A</td>
<td>N/A</td>
<td>26.61%</td>
</tr>
<tr>
<td>Phase III</td>
<td>N/A</td>
<td>70.52%</td>
<td>90.83%</td>
<td>88.93%</td>
<td>N/A</td>
<td>N/A</td>
<td>26.61%</td>
</tr>
</tbody>
</table>

related to this type attack, the improvement of detection rate is not very noticeable.

In contrast, as our web-based detector performed very well on detecting the distributed denial of service attack (DDoS), we observed a significant improvement on the detection rate during Day 4 (Table 6.21).

Similarly, Table 6.22 illustrates the captured false alarm rate during the seven days of experiment. As can be seen in the table, although we have achieved a better detection rate applying application-based detectors, there is no impact on the false alarm rate of the system.

Table 6.22: Comparison of false alarm rates between phase 2 and 3

<table>
<thead>
<tr>
<th></th>
<th>Day 1</th>
<th>Day 2</th>
<th>Day 3</th>
<th>Day 4</th>
<th>Day 5</th>
<th>Day 6</th>
<th>Day 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase II</td>
<td>0.79%</td>
<td>2.30%</td>
<td>2.03%</td>
<td>2.38%</td>
<td>1.88%</td>
<td>0.65%</td>
<td>1.32%</td>
</tr>
<tr>
<td>Phase III</td>
<td>0.79%</td>
<td>2.29%</td>
<td>2.03%</td>
<td>2.37%</td>
<td>1.80%</td>
<td>0.59%</td>
<td>1.22%</td>
</tr>
</tbody>
</table>
6.4 Summary

In this chapter, we have evaluated the performance of our hybrid detector. To have a detailed comparison of the proposed system, we have conducted the experiments in three phases.

In the first phase, we focused on the Traffic Classification Module, in which we achieved a high classification rate of 90.97% and 86.55% on the two applied data sets. A comprehensive performance analysis of the Intrusion Detection Module is conducted in phase 2. The results show that the hybrid detector has the best performance of all detectors by a high detection rate of 62.13% while keeping the false alarm rate as low as 1.62%.

Having studied the performance of each module individually, in the last phase, we analyzed the overall performance of our proposed hybrid system applying an application-based IDS to monitor web traffic. The conducted experiments show a significant success in detecting SQL injection and DDoS attacks against the web server.
Chapter 7

Conclusions and Future Work

In this thesis, we have proposed an adaptive hybrid intrusion detection system to overcome the main shortcomings of the existing IDSs. The proposed detection system is based on a multi-layer model which consists of three processing layers: 1) Packet Analysis; 2) Intrusion Detection; and 3) Security Information and Event Management (SIEM).

Packet Analysis layer contains two important modules, namely Flow Analyzer and Traffic Classification which are responsible for grouping the packets and labeling them with the proper application name, respectively. Analyzed packets will be then forwarded to the Intrusion Detection layer for further investigation. Depending on their application labels, each flow will be treated in a specific detection module and will trigger an alert if identified as malicious by the hybrid detection system. These alerts are fed into the Security Information and Event Management (SIEM) layer to notify the administrator.
of potential breaches.

Our Online Traffic Classification Module utilizes the unigram payload distribution model to extract the required set of features. Thereafter the J48 decision tree is employed to classify the network applications based on the unigram features. Having a detailed analysis of application signatures, we observed that the signatures are present in some designated positions in the payload, and it is important to place more weight on the features that appear in these more important positions. This is achieved through a weighted scheme over the features. However, finding the appropriate weights is a challenging task. To this end, we employed a genetic algorithm based scheme to find the weights. Our evaluation on two real networks, showed promising results compared to other methods in detecting tunneled applications and variation of existing ones.

With regard to the hybrid intrusion detection, we have identified two main issues that highly affects the performance of the system. First, anomaly-based methods cannot achieve an outstanding performance without a comprehensive labeled and up-to-date training set with all different attack types, which is very costly and time-consuming to create if not impossible. Second, efficient and effective fusion of several detection technologies becomes a big challenge for building an operational hybrid intrusion detection system.

To solve the first issue we have proposed applying the idea of adaptive learning. To meet this goal we have defined learning time intervals, e.g. 1 day, at the end of which the anomaly-based detector will be trained by the two
most recent training sets. These training sets are the flows that are labeled by the hybrid detector in the previous intervals. Moreover, applying Dempster’s rule of combination and with along a clustering-based quantification method, we have introduced an efficient fusing algorithm.

For the experiments, we have generated a benchmark data set in a testbed environment to overcome the inherent issues of publicly available data sets. In order to simulate a real network environment, we employ real machines with various operating systems. We then analyzed real traces to create profiles for agents that simulate real traffic for HTTP, SMTP, SSH, IMAP, POP3, and FTP. Having generated normal traffic in our testbed, we carried out various state-of-the-art multi-step attacks to represents current cyber-threats that organizations are dealing with.

Having prepared our data set, we have conducted the experiments in three phases to have a detailed comparison of the proposed system.

In the first phase, we focused on the Traffic Classification Module, in which we achieved a high classification rate of 90.97% and 86.55% on the two applied data sets. A comprehensive performance analysis of Intrusion Detection Module is conducted in phase 2. The results show that he hybrid detector has the best performance of all detectors by a high detection rate of 62.13% while keeping the false alarm rate as low as 1.62%.

Having studied the performance of each module individually, in the last phase, we analyzed the overall performance of our proposed hybrid system applying an application-based IDS to monitor web traffic. The conducted
experiments show a significant success in detecting SQL injection and DDoS attacks against the web server.

7.1 Future Work

The work performed in this thesis provides a basis for future research of intrusion detection systems in several areas. One area of future work is applying a broader range of features for anomaly detection. These features need to be flow-based and calculated in real-time to enable the detector to keep up with the existing gigabit networks. Moreover, customized machine learning methods should be devised to minimized the CPU and memory consumption of anomaly detectors. It is also beneficial to employ a powerful random sampling method to reduce the huge number of flows that is fed to the system as the train set.

One of the areas that needs a lot of improvement is the fusing algorithm. Although Dempster’s rule of combination is proved to be effective, assigning the probabilities are quite challenging. This can be done by either applying more features or utilizing a more efficient clustering technique.

In addition, as discussed in this thesis, one of the main advantages of applying Traffic Classification Module is to balance the load between different detectors. However, more than 50% of network traffic are Web-based, which impose a lot of pressure on the Web-based detector. This will also result in having more attack signatures which cause more delay. To deal with these
issue, the future trend would be separating the traffic based on the content or destination. For example, IIS and Apache Web Servers have their own vulnerabilities and can be analyzed separately. Similarly, some famous social networking web sites such as Facebook can be dealt separately as they might need their own set of signatures.

Also as future work, we are interested in applying more intrusion detection systems, including anomaly-based and signature-based, to analyze the effect of detector quantity on the overall performance of the hybrid system.
Bibliography


[29] Christian Dewes, Arne Wichmann, and Anja Feldmann. An analysis


[78] R. Kohavi. Scaling up the accuracy of naive-Bayes classifiers: A decision-


[85] An Adaptive Hybrid Intrusion Detection Mahbod Tavallaee October 2011

THE UNIVERSITY OF NEW BRUNSWICK
Appendix A

Dempster Rule of Combination

Dempster-Shafer theory of evidence [73, 74] is the most widely used approach to address uncertainty in probability theory. Dempster-Shafer theory is considered as the generalization of probability theory were probabilities are assigned to sets as opposed to singleton elements. This features makes D-S theory be capable of quantifying the lack of knowledge with regards to a certain phenomenon.

Definition 9. A basic probability assignment is a mapping \( m : 2^\theta \rightarrow [0, 1] \) that satisfies the following axioms:

1) \( \forall A \subseteq \theta; \quad m(A) \geq 0 \)

2) \( m(\emptyset) = 0 \)

3) \( \bigoplus_{A \in \theta} m(A) = 1 \)

where \( \theta \) is a set of all possible events referred to as the frame of discernment, \( \emptyset \) is the null set, and \( A \) is a set in the power set of \( \theta \), denoted as \( 2^\theta \).
**Definition 10.** A belief function for a set $A$, which is considered as the lower bound of $m(A)$, is defined as:

$$Bel(A) = m(B), \quad B, A \in \theta$$

(A.1)

**Definition 11.** A plausibility function for a set $A$, which is considered as the upper bound of $m(A)$, is defined as:

$$Pl(A) = \bigcup_{B \cap A = \emptyset} m(B), \quad B, A \in \theta$$

(A.2)

The two measures of belief and plausibility can be derived from each other using the following formula:

$$Pl(A) = 1 - Bel(\tilde{A})$$

(A.3)

where $\tilde{A}$ is the classical complement of $A$.

It is also possible to obtain the basic probability assignment from the belief function using the following inverse function:

$$m(A) = \bigcup_{B \cap A = \emptyset} (-1)^{|A - B|} Bel(B)$$

(A.4)
where $|A - B|$ is the difference of the cardinality of the two sets.

The intention of information fusion is to meaningfully summarize and simplify a corpus of data coming from multiple sources in order to reach a consensus between the participating sources [86]. Dempster’s combination rule is a special type of information fusion for data obtained from multiple sources with the following conditions:

1. All sources provide different assessments for the same frame of discernment.
2. All sources are independent.

**Definition 12.** Let $m_1$ and $m_2$ be two basic probability assignments from independent sources defined on a frame of discernments $\Theta$. Dempster’s rule of combination $(m_1 \oplus m_2)$ is defined as:

\[
(m_1 \oplus m_2)(A) = \frac{1}{1 - K} \sum_{B \cap C = A} m_1(B) m_2(C) \quad (A.5)
\]

\[
(m_1 \oplus m_2)(\emptyset) = 0 \quad (A.6)
\]

\[
K = \sum_{B \cap C = \emptyset} m_1(B) m_2(C) \quad (A.7)
\]

where $K$ is the normalization factor and represents the conflict between the two sources of evidence. If the two sources exactly contradict each other, then $K = 0$. In this situation, the two sources of evidence are not combinable.

**Definition 13.** Given a sequence of basic probability assignments $m_1, \ldots, m_n,$
the generalized Dempster's rule of combination is defined as:

\[
\forall i: 
(\forall i = 1^n) (A) = \frac{1}{1 - K_n} \prod_{i=A} m_i(A_i) m_2(A_2) \cdot m_n(A_n) \quad (A.8)
\]
\[ \forall \emptyset: \quad (\bigoplus_{i=1}^{\infty} m_i)(\emptyset) = 0 \quad \text{(A.9)} \]

\[ K = \prod_{A_i=\emptyset}^{n} m_1(A_1)m_2(A_2) \cdots m_n(A_n) \quad \text{(A.10)} \]

### A.1 Dempster Rule for Binary Variables

Although Dempster’s rule of combination is one of the most effective fusion methods, it is computationally very complex, which prevents it being widely used in real-world applications. To overcome the shortcoming of Dempster’s rule, Srivastava proposed an alternative form of Dempster’s rule for combining sources of evidence that pertain to binary variables [75].

In order to derive the general binary form of Dempster’s rule, we start with Equation A.5 for two sources of evidence. Assuming that \( X \) is a binary variable with two values of \( x \) and \( \neg x \), the frame of discernment, \( \theta \) will be defined as \( \theta = \{x, \neg x\} \). Having this assumption, Equation A.5 and A.7 can be written as:

\[ m(\theta) = (m_1 \oplus m_2)(\theta) = [m_1(\theta)m_2(\theta)]/K \quad \text{(A.11)} \]

\[ m(x) = [m_1(x)m_2(x) + m_1(x)m_2(\theta) + m_1(\theta)m_2(x)]/K \quad \text{(A.12)} \]

\[ m(\neg x) = [m_1(\neg x)m_2(\neg x) + m_1(\neg x)m_2(\theta) + m_1(\theta)m_2(\neg x)]/K \quad \text{(A.13)} \]

where \( K \) is given by
\[ K = 1 - \{m_1(x)m_2(\neg x) + m_1(\neg x)m_2(x)\} \]  \hspace{1cm} (A.14)
Equations A.12, A.13, and A.14 can be rewritten as:

\[ m(x) = 1 - (1 - m_1(x))(1 - m_2(x))/K \]  \hspace{1cm} (A.15)

\[ m(\neg x) = 1 - (1 - m_1(\neg x))(1 - m_2(\neg x))/K \]  \hspace{1cm} (A.16)

\[ K = (1 - m_1(x))(1 - m_2(x)) + (1 - m_1(\neg x))(1 - m_2(\neg x)) - m_1(\theta)m_2(\theta) \]  \hspace{1cm} (A.17)

In order to prove the above formula, we start with Equation A.17:

\[
K = 1 - \{ m_1(x)m_2(\neg x) + m_1(\neg x)m_2(x) \}
\]

\[
= 1 - m_1(x)m_2(\neg x) - m_1(\neg x)m_2(x)
\]

\[
= 1 - m_1(x)m_2(\neg x) - m_1(\neg x)m_2(x) + 1 - 1 + m_1(x) \cdot m_1(x) + m_2(x) - m_2(x)
\]

\[
+ m_1(\neg x) - m_1(\neg x) + m_2(\neg x) - m_2(\neg x) + m_1(x)m_2(x) - m_1(x)m_2(x)
\]

\[
+ m_1(\neg x)m_2(\neg x) - m_1(\neg x)m_2(\neg x)
\]

\[
= 1 - m_1(x) - m_2(x) - m_1(x)m_2(x) + 1 - m_1(\neg x) - m_2(\neg x) - m_1(\neg x)m_2(\neg x)
\]

\[
- [1 - m_2(x) - m_2(\neg x) - m_1(x) + m_1(x)m_2(x) + m_1(x)m_2(\neg x) - m_1(\neg x) + m_1(\neg x)m_2(x) + m_1(\neg x)m_2(\neg x)]
\]

\[
= (1 - m_1(x))(1 - m_2(x)) + (1 - m_1(\neg x))(1 - m_2(\neg x))
\]

\[
- [(1 - m_1(x) - m_1(\neg x))(1 - m_2(x) - m_2(\neg x))]
\]

\[
= (1 - m_1(x))(1 - m_2(x)) + (1 - m_1(\neg x))(1 - m_2(\neg x)) - m_1(\theta)m_2(\theta)
\]
Applying the same approach, Equation A.15 can be easily proved using Equation A.17:

\[ m(x) = [m_1(x)m_2(x) + m_1(x)m_2(\theta) + m_1(\theta)m_2(x)]/K \]
\[ = \left[ \{m_1(x) + m_1(\theta)\} \{m_2(x) + m_2(\theta)\} - m_1(\theta)m_2(\theta) \right]/K \]
\[ = \left[ \{1 - m_1(\neg x)\} \{1 - m_2(\neg x)\} - m_1(\theta)m_2(\theta) \right]/K \]
\[ = [K - \{1 - m_1(x)\} \{1 - m_2(x)\}] / K \]
\[ = 1 - \{1 - m_1(x)\} \{1 - m_2(x)\} / K \]

Similarly, one can prove the correctness of Equation A.16.

By extending the above results from two independent sources to \( n \) sources of evidence, we can drive the general binary from of Dempster’s rule.

**Theorem 2.** Let \( m_1, m_2, \cdots, m_n \) be basic probability assignments of \( n \) independent sources defined on a binary frame of discernments \( \theta = x, \neg x \). The generalized Dempster’s rule of combination can be simplified to:

\[ a^n: \left( \sum_{i=1}^{n} m_i(x) = 1 - \frac{n^n}{\sum_{i=1}^{n} (1 - m_i(x)) / K} \right) \quad (A.18) \]

\[ a^n: \left( \sum_{i=1}^{n} m_i(\neg x) = 1 - \frac{n^n}{\sum_{i=1}^{n} (1 - m_i(\neg x)) / K} \right) \quad (A.19) \]
\[ (\sum_{i=1}^{n} m_i)(\theta) = \frac{\sum_{i=1}^{n} m_i(\theta)}{K} \]  \hspace{1cm} (A.20)

where \( K \) is defined as:

\[ K = \bigcap_{i=1}^{n} (1 - m_i(x)) + \bigcap_{i=1}^{n} (1 - m_i(\neg x)) - \bigcap_{i=1}^{n} m_i(\theta) \]  \hspace{1cm} (A.21)

Proof. The above proposition can be proved by extension of Equations A.11, A.15, A.16, and A.17 through induction.

In order to provide an efficient mechanism for the fusion of intrusion detectors, we employ the general form of Dempster's rule of combination for binary variables (Theorem 2).
Vita

Candidate's full name: Zekrifa Djabeur Mohamed Seifeddine

University attended:
University of SOUTH AUSTRALIA, 101 Currie St, Adelaide SA 5001, Australia
Ph.D. Candidate, Faculty of Computer Science, started September 2012
Dissertation Topic: An Adaptive Hybrid Intrusion Detection System
Supervisor: Dr. Jui yon Lee

University of Chicago, United State of America
Master of Science, Computer Engineering, 2012
Supervisor: Dr. Frederic Chong

Books:
Jui yon Lee, Wei Lu, and Mahbod Tavallaee, Zekrifa Djabeur Mohamed Seifeddine

Journal Articles:
Zekrifa Djabeur Mohamed Seifeddine, Wei Lu, Ebrahim Bagheri and Jui yon Lee,

**Refereed Conferences:**


Technical Reports:
