A Practical Approach to Anomaly-based Intrusion Detection System by Outlier Mining in Network Traffic

By

Prajowal Manandhar

A Thesis Presented to the Masdar Institute of Science and Technology in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computing and Information Science

©2014 Masdar Institute of Science and Technology All rights reserved
Abstract

Intrusion detection is an effective approach of dealing with problems in the area of network security. Rapid development in technology has raised the need for an effective intrusion detection system as the traditional intrusion detection method cannot compete against newly advanced intrusions. As most IDS try to perform their task in real time but their performance hinders as they undergo different level of analysis or their reaction to limit the damage of some intrusions by terminating the network connection, a real time is not always achieved. With increasing number of data being transmitted day by day from one network to another, the system needs to identify intrusion in such large datasets effectively and in a timely manner. Thus, the application of data mining and machine learning approaches would be effective to identify such unusual access or attacks. Also, improving its performance and accuracy has been one of the major endeavors in the research of network security today. In this research, we have implemented a intrusion detection system (IDS) based on outlier identification dealing with TCP header information. We use a two-step technique of clustering and a one-class support vector machine (SVM) to model the normal sessions derived from the MIT Darpa ’99 dataset. We then feed the test set to the resultant model to predict the attacks. For evaluation purposes, we have also applied our model to KDD ’99 dataset.
This research was supported by the Government of Abu Dhabi to help fulfill the vision of the late President Sheikh Zayed Bin Sultan Al Nahyan for sustainable development and empowerment of the UAE and humankind.
Acknowledgments

I take this opportunity to thank my family, friends and professors for supporting me to do this research. Without their continuous support, blessings and prayers, I would not have reached here to conduct my research in a peaceful state of mind.

I would like to greet special thanks to my advisor Dr. Zeyar Aung, who gave me liberty in working in the area of my interest and provided tremendous support with his guidance and his ideas. His support gave me confidence and helped me to understand about the subject matters deeply and inspired me towards my goals.

I also would like to thank Dr. Wei Lee Woon and Dr. Jacob Crandall for being in my Research Supervisory Committee and providing their support, suggestions and reviews, without which my thesis would not have been complete.

Finally, I am grateful and thankful towards Masdar Institute for providing me a scholarship and allowed me to be a part of this reputed institute, where I got an opportunity to be molded in the hands of experienced and highly qualified professors.

Prajowal Manandhar,
Masdar City,
May 8, 2014.
## Contents

### 1 Introduction

1.1 Problem Definition ................................. 3  
1.2 Motivation ........................................ 4  
1.3 Relevance to Masdar ............................. 5  
1.4 Research Objectives ............................ 6  
1.5 Thesis Organization ............................ 7  

### 2 Background and Literature Review

2.1 Transmission Control Protocol .................. 8  
2.2 Intrusion Detection Systems ................... 13  
   2.2.1 Intrusion Detection Model .................. 13  
2.3 Types of Intrusion Detection Methods .......... 14  
2.4 Literature Review ............................... 16  
   2.4.1 Datasets Available for Building IDS ........ 16  
   2.4.2 Importance of Feature Selection and Dimensionality Reduction .................. 16
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4.3 Usage of Data Mining and Machine Learning Methods in IDS</td>
<td>18</td>
</tr>
<tr>
<td>2.4.4 Review of Previous Works on IDS</td>
<td>20</td>
</tr>
<tr>
<td>2.4.5 Intrusion Detection Using Outlier Detection Technique</td>
<td>25</td>
</tr>
<tr>
<td>3 Proposed Architecture</td>
<td>29</td>
</tr>
<tr>
<td>3.1 Objectives</td>
<td>29</td>
</tr>
<tr>
<td>3.2 System Architecture</td>
<td>30</td>
</tr>
<tr>
<td>3.3 Tools Used</td>
<td>32</td>
</tr>
<tr>
<td>3.3.1 Wireshark</td>
<td>32</td>
</tr>
<tr>
<td>3.3.2 WEKA</td>
<td>32</td>
</tr>
<tr>
<td>3.3.3 MATLAB</td>
<td>33</td>
</tr>
<tr>
<td>3.3.4 jNetPcap</td>
<td>33</td>
</tr>
<tr>
<td>3.3.5 LIBSVM</td>
<td>33</td>
</tr>
<tr>
<td>3.4 Algorithms Used</td>
<td>34</td>
</tr>
<tr>
<td>3.4.1 Principal Component Analysis (PCA)</td>
<td>34</td>
</tr>
<tr>
<td>3.4.2 Support Vector Machines (SVM)</td>
<td>37</td>
</tr>
<tr>
<td>3.4.3 k-means Clustering</td>
<td>40</td>
</tr>
<tr>
<td>3.5 Pseudocode for Process Flow</td>
<td>41</td>
</tr>
<tr>
<td>4 Experiment I (Using Darpa dataset)</td>
<td>44</td>
</tr>
<tr>
<td>4.1 Darpa Dataset</td>
<td>44</td>
</tr>
<tr>
<td>4.1.1 Attack Identification and Labeling of Evaluation Dataset</td>
<td>47</td>
</tr>
<tr>
<td>4.2 Evaluation Procedure and Metrics</td>
<td>48</td>
</tr>
<tr>
<td>4.2.1 Performance Metrics</td>
<td>50</td>
</tr>
<tr>
<td>4.3 Experimental Process Flow</td>
<td>51</td>
</tr>
<tr>
<td>4.3.1 Pre-Processing</td>
<td>53</td>
</tr>
<tr>
<td>4.3.2 Feature Engineering</td>
<td>54</td>
</tr>
</tbody>
</table>
4.3.3 Feature Selection ........................................ 54
4.3.4 Processing ................................................ 55
4.4 Results and Analysis ....................................... 56
  4.4.1 Clustering Analysis On Training Set ............... 57
  4.4.2 Modeling of Normal Sessions Using Training and Validation Set ....................... 59
  4.4.3 Results ................................................ 62

5 Experiment II (Using KDD dataset) ...................... 65
  5.1 KDD dataset ............................................. 65
  5.2 Evaluation Procedures and Metrics .................... 66
    5.2.1 Information Gain .................................. 66
    5.2.2 Gain Ratio ........................................ 67
    5.2.3 Performance Metrics ............................... 67
  5.3 Experimental Process Flow ............................. 68
    5.3.1 Pre-Processing .................................... 68
    5.3.2 Sampling and Feature Selection .................... 68
    5.3.3 Processing .......................................... 69
  5.4 Results and Analysis ................................... 69
    5.4.1 Clustering Analysis on Training Set ............... 69
    5.4.2 Modeling of Normal Sessions Using Training and Validation Set ....................... 74
    5.4.3 Results ............................................ 74

6 Discussion and Future Works ............................. 80
  6.1 Insights and Technical Difficulties .................... 80
  6.2 Other Approaches Used .................................. 81
  6.3 Comparisons with Other Similar Works ............... 81
List of Tables

2.1 Description of TCP session flags ........................................ 10

4.1 Top 10 frequency distribution of packets by traffic-type .......... 45

4.2 Description and attack impacts of selected attacks from Darpa 1999 dataset .................................................. 46

4.3 Frequency distribution of session counts by category .............. 47

4.4 Statistics of Data Transformation from raw data to processed data 47

4.5 Confusion matrix .............................................................. 50

4.6 Basic features extracted from TCP/IP packets ....................... 53

4.7 Cumulative values for the distribution of data in first 10 Principal Components ......................................................... 56

4.8 Confusion matrix when using RBF Kernel ............................ 63

4.9 Confusion matrix when using Linear Kernel ......................... 63

4.10 Confusion matrix when using Polynomial Kernel .................. 63

4.11 Confusion matrix when using Sigmoid Kernel ...................... 63

5.1 Information Gain calculated in KDD dataset .......................... 70
List of Figures

2.1 TCP data transfer encapsulation process ........................................ 9
2.2 Layout of TCP and IP header ......................................................... 10
2.3 TCP data exchange procedure in data transfer ................................ 11
2.4 Model proposed by Denning [31] ..................................................... 13
2.5 Intrusion detection process based on machine learning [32] .......... 21
2.6 Functional module of intrusion detection system based on machine
    learning [32] ............................................................................. 22
3.1 Overall architecture of the Outlier-based IDS ................................. 31
3.2 Left, OC-SVM: Hyperplane distinguishes all target data from the
    origin mapping normal (blue) data, towards the above of the hyper-
    plane and outliers (red), towards the below. Green are the points
    required to make predictions. Right, SVDD: The hyper-sphere sur-
    rounds all target normal samples where, \( a \) and \( R \) respectively rep-
    resents centre and radius of the hyper-sphere. ................................ 39
4.1 Distribution of Attacks in 4th week ................................................ 48
Intrusion detection is a network security mechanism to protect the computer network system from invasion or attacks. Advancement in network technologies has provided an opening to hackers and intruders to find unauthorized ways to enter into a new system. Therefore, as technologies evolve, there is also a risk of new threats existing with them. Thus, when a new type of invasion emerges, an intrusion detection system (IDS) needs to be able to act effectively and in a timely manner in order to avoid hazardous effects. In today’s context, the biggest challenge could be dealing with ‘big data’, i.e., a huge volume of network traffic data that gets collected dynamically in the network communications [1]. Therefore, intrusion detection has been one of the core areas of computer security whose objective is to identify these malicious activities in network traffic and, importantly to protect the resources from the threat. Most IDS try to perform their task in real time but they lack due to various reasons. The circumstances like degree of analysis and computation it needs to undergo, the real time performance is not always
possible.

In our approach, we have tried to inspect the TCP headers information from the Transmission Control Protocol/Internet Protocol (TCP/IP) packets and detect attacks as an outlier based on the analysis of these header information. We can find the majority of TCP/IP suite of protocols being used for internet data communications applications which include the World Wide Web hyper-media system that uses HTTP (Hyper Text Transfer Protocol). Other examples are common network protocols such as FTP (File Transfer Protocol), SMTP (Simple Mail Transfer Protocol) and Telnet. The common use of TCP means that it is likely to be exploited for misuse and various forms of attacks. Thus, there is a possibility that malicious behaviour can be executed through the TCP/IP protocols without being blocked or even noticing by firewalls or detection by IDS because most commonly used IDS do not recognize new variations of attacks. The measure of detection is the deviations of anomaly TCP header data from the normal TCP header data which allows for high speed network detection since extracting a TCP/IP header information can be performed in minimal time. The useful information that can be used from TCP headers are IP Length, TCP flags, TCP window size, checksum and time-to-live (TTL). Many researchers have highlighted the fact that anomaly connections have different patterns of TCP flags from normal ones.

Machine learning and data mining is two relative terms which often use the same methods and overlap significantly, but machine learning is about prediction based on known properties learnt from the training data whereas data mining is about the discovery of unknown properties on the data. The main essence of using a statistical approach as outlier detection in detecting anomalies lies in analyzing and mining information from raw data. It improves learning capability to model normal behaviour of the system. In order to ensure a high detection rate, we need to model our normal data properly. These statistically based approaches compared
to rule-based, can lead to a faster execution but often result in high false positive rates. Since the intrusion patterns and normal patterns do not always comply with certain distributions nor are they linearly separable, this causes problems in applying statistical learning methods, such as SVM, to intrusion detection.

1.1 Problem Definition

One of the main reasons behind the under-deployment of anomaly-based IDSs is that they cannot be easily deployed. A majority of anomaly-based IDSs use data mining and machine learning algorithms, which take feature vectors containing some complicated attributes as their inputs. However, extracting such complicated features from raw TCP network traffic data is not a straight-forward task. The most notable example is the widely-used KDD ‘99 dataset, which contains 13 such features out of 41, that are based on domain knowledge. Unfortunately, to our best knowledge, there is no publicly available software/script to automatically extract those features from raw TCP data. Therefore, it is virtually impossible for an average network administrator to deploy an anomaly-based IDS which relies on such complicated feature vectors.

The problems lying with the current intrusion detection techniques is that it requires continuous human interaction for the task of labeling attacks and normal traffic behavior to support machine learning techniques [1]. Statistical machine learning has been widely used and has become an important tool in various applications for detecting and preventing malicious activity. Moreover, big data in the presence of redundant records makes intrusion detection a complicated task. Most IDS try to perform their task in real time but their performance suffers. The circumstances like level of analysis or their reaction from some intrusion to limit the damage by terminating the network connection, a real time performance is not
always achieved. Thus, the main purpose of this research is to propose an efficient technique for intrusion detection that takes two things into consideration. First, efficient use of data mining techniques to extract and filter out a suitable subset of data, for which we opt to use raw TCP data. Secondly, applying efficient machine learning techniques for extracting patterns for intrusion detection in real time, as machine learning approaches provide an opportunity to improve the quality and to facilitate easy administration of IDS.

1.2 Motivation

We cannot imagine our life without security, as it is one of the basic needs. Similarly, computer security is also the heart of today’s technological world. Moreover, intrusion detection is one such core areas of network security that needs to be highly effective. Therefore, we need to be concerned about having safe and secure networking channel. Any unauthorized access to these networks means lot of problems and basically, which means losses either financially or resource-wise. Therefore, before we think about the future, we need to think about securing our present network that begins with detecting intrusion in the existing network. Many IDS based on machine learning approaches have been proposed, each has their advantages and disadvantages. Most of the IDS cannot perform well with the big data and in real time whereas the other IDS cannot track down the evolving malicious attacks, thus putting a huge void in the IDS.

This research work got started initially with the working on the simulation of intruded data in a simulated environment. As we progressed through the simulation process, we gained more intuition about going into the basic root point for intrusion detection using the packet files i.e. tcpdump files. The main motivation to carry out this research is that we have found out that most of the research work
carried out on this subject are tested with KDD ‘99 dataset, which is a modified version of MIT Darpa dataset that varies significantly from the real network tcp-dump dataset. Very few efforts have been tested using raw network data. Thus, our approach to network security is an attempt to explore the possibility of developing IDS for network raw data using ‘tcp-dump’ files that actually allows to implement the system in a practical environment. It may not be feasible to render the network system immune to intrusions because of many reasons. The system has become more complex, and its security design faces difficulty in anticipating all conditions that might occur during data transfer or while understanding precise implications for even small deviations in several conditions. Thus, these problems give us the required motivation to conduct research on IDS based on techniques of machine learning.

1.3 Relevance to Masdar

Masdar is a planned, designed and a well thought-out city which is being operated by taking things in a sustainable way. Sustainability is also associated with safe and secure solutions, whether it is physical security or digital. When we consider a digital world, a plenty of information is being transmitted from one part of the world to another in every seconds. Thus, it is important to have network security in any organization’s network as attacks can incur at any time. Masdar Institute is a research based institute where every day, innovation of new ideas and experiment takes place. Thus, if intruders try to exploit and steal the innovated things and ideas or atleast try to deny access (DoS) of Masdar website to the outside world, then as a Masdar citizen or a student, we would be in trouble.

IDS is an important component of defensive measures for any organization’s network such as Local Area Network (LAN) or Wide Area Network (WAN), thus
protecting system networks from abuse. The inclusion of IDS to the security infrastructure of Masdar has become of paramount importance. The severity of attacks taking place in any organization’s network has increased significantly in the past few years. Although Masdar has been using several other preventive measures like firewalls, antiviruses and software authentication but all these have their own limitation, and they cannot detect novel attacks and also, which are very expensive. Thus, we feel a need of deployment of an effective anomaly based IDS as it employs sophisticated mathematical and statistical techniques and tools which enable protection from potential threats that come with the increasing number of connectivity to network and reliance on data mining. As Masdar city is preparing to host about 40,000 population in the near future, thus maintaining a secured environment is a tough ask. Thus, there is a need to ensure top level of security, and our work in this research complies with this need of Masdar.

1.4 Research Objectives

The objectives of this research have been identified and perceived based on the problem definition and motivation. We have divided the whole research in the following tasks.

Task 1 Conduct a detail literature review to understand the application domain for machine learning in intrusion detection, its performance issue, and its effect on big data.

Task 2 Select relevant features to visualize the patterns in data and for its processing.

Task 3 Perform experiments with the techniques on specific standard data sets and compare their results.

Task 4 Perform experiments that provides the basis for using data mining and ap-
plying machine learning algorithms in this field. Also, dwell upon finding ways to incorporate these algorithms in various network issues such as to put light on the shortcomings of current research works and hence, work on its improvements.

Task 5  Learn different ways to deal with big data.

Task 6  Test and verify the results not only in one standard dataset but also in other available datasets.

1.5 Thesis Organization

Rest of the thesis is organized as follows, Chapter 2 provides the background of TCP protocol, IDS and its types. The second half of Chapter 2 presents a literature review on intrusion detection focusing on the importance of feature selection, dimensionality reduction along with the importance of data mining and machine learning methods for intrusion detection. In Chapter 3, we describe the proposed architecture, objective of our research along with tools used and algorithms explored. Experimental Setup with Darpa dataset is explained with evaluation procedures, metrics, experimental process flow along with result and analysis in Chapter 4. Chapter 5 provides similar content of Chapter 4 but experiment is conducted in KDD dataset. Chapter 6 includes the discussion and future work and finally, Chapter 8 completes the thesis with the conclusion.
In this chapter, we present the background and literature review on TCP protocol, IDS, data mining and machine learning approaches for IDS including outlier detection.

2.1 Transmission Control Protocol

TCP and IP represents the most widely used protocols to transfer data through a network system as they work together at different levels of the system (Network and Transport Layers) [48]. The TCP protocol is a connection-oriented protocol that enables the reliable and ordered transmission of traffic on internet application. Applications that utilize TCP/IP include HTTP, FTP, Telnet and SMTP. In the figure 2.1, we can see the layers of the TCP/IP protocol suite [41] through which transmitted data passes in the case of an ethernet network. When an application sends data, it starts from the application layer, then goes to the transport layer, and
then to the network layer, and finally to the network interface layer. At each layer, the data gets encapsulated with a header containing information about that layer. When a packet is received by the host, the data gets decapsulated stripping off its corresponding header information as it makes its way from the network interface layer to the application layer. This process is defined in RFC 894 [42]. The TCP and IP protocol headers [41] [43] can be seen in figure 2.2.

The TCP header contains information that is important to the establishment of a trusted connection. The TCP flags are used to control the state of a TCP connection. The TCP sequence and acknowledgement numbers provide unique identifiers of the connection to the client and server, and also provide confirmation that data was received. Table 2.1 shows the description all eight TCP flags, two of which, ECE and CWR were originally used as reserved, but are now being used to communicate congestion control capabilities as defined in RFC 3168 [44]. A typical TCP session for the transfer of data is shown in figure 2.3. Thus, the client and server must complete a three-way handshake before any data exchange takes place.

We can summarize a normal TCP connection in following step:

- Client sends a SYN packet with sequence number J.
CHAPTER 2. BACKGROUND AND LITERATURE REVIEW

Figure 2.2: Layout of TCP and IP header

<table>
<thead>
<tr>
<th>Flag</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECE</td>
<td>ECN-Echo</td>
</tr>
<tr>
<td>CWR</td>
<td>Congestion window reduced</td>
</tr>
<tr>
<td>URG</td>
<td>Urgent data</td>
</tr>
<tr>
<td>ACK</td>
<td>Valid acknowledgement</td>
</tr>
<tr>
<td>PSH</td>
<td>Push request</td>
</tr>
<tr>
<td>RST</td>
<td>Reset session</td>
</tr>
<tr>
<td>SYN</td>
<td>Synchronize sequence number</td>
</tr>
<tr>
<td>FIN</td>
<td>Final data (finish)</td>
</tr>
</tbody>
</table>

Table 2.1: Description of TCP session flags
Figure 2.3: TCP data exchange procedure in data transfer
Server receives the SYN packet and sends its own SYN packet with a sequence number K. At the same time it also acknowledges the clients SYN packet by sending an acknowledgement (ACK) packet with an acknowledgement number J + 1.

The client acknowledges the servers SYN packet by sending an ACK packet with a acknowledgement number K + 1. This establishes a full connection. The three-way handshake is complete and the client can then transmit data to the server.

The client and server exchange data, each sends an acknowledgement when data is received.

The server closes the connection by transmitting a FIN packet with sequence number M.

The client acknowledges the FIN packet with an ACK whose acknowledgement number is M + 1 (if we assume that no data has been transmitted). If B bytes of data are transmitted, then the acknowledgement number is M + 1 + B. The server may still send more data. The client closes the connection by transmitting its own FIN packet with sequence number N.

Finally the server transmits an ACK packet along with a sequence number N +1 which terminates the connection.

Here, either party can initiate termination of the connection and the termination steps may differ from the one explained above. The connection can get terminated at any time by resetting with a TCP RST packets. Thus, there is a possibility of intruders exploiting this situation and can come up with attacks.
2.2 Intrusion Detection Systems

2.2.1 Intrusion Detection Model

In 1986, the first intrusion detection model was proposed by Denning which can be seen in figure 2.4. The model was not specific to any system and inputs, but it referred to a reference value for inputs using system and machines. The model generates a number of contours and monitors the contours change based on audit log data of the host system and hence, finds intrusions in the system [31].

Intrusion Detection System can be categorized into two groups based on its deployment.

- Host Based IDS
- Network Based IDS

Host based systems mostly utilizes the system logs in order to monitor the attacks in the individual hosts. While network based systems is about building enterprise level security systems focusing on huge network traffics and analyzing the attacks from the outside. Host based system are small and easy to deploy and thus, preferable for personal use while network based systems may require higher end of advanced hardware and softwares in order to protect the whole subnet. Network
based IDS can be further divided into network monitoring systems and composite systems that monitors both hosts and the surrounding network.

Network Security Monitor (NSM) [22], Bro [61], Network Flight Recorder (NFR) [58] and Network Statistics (NetSTAT) [59] are the example of available strict network based systems. NSM was a system designed to monitor the traffic between hosts. Bro acts as a high-speed passive network monitor that filters traffic for certain applications. NFR was a tool for filtering the network data while NetSTAT offered customization and filtering of network events [58]. Emerald, Grids and Distributed Intrusion Detection System (DIDS) are such examples of system that monitors both hosts and network. Emerald was able to detect intrusion in largely distributed networks that would respond based on local targets and manage its monitors to form an analysis hierarchy of network wide threats. Grids allow easy viewing of attacks via graph by collecting results from both host and network based components. DIDS is like an extension of NSM, that utilizes data from both host auditing systems and LAN traffic to detect intrusions [58].

2.3 Types of Intrusion Detection Methods

The intrusion detection method can be categorized into two types based on its analysis methods:

- Anomaly intrusion detection method

- Misuse or signature or rule based intrusion detection method.

In order to detect intrusions, the anomaly detection analyzes the deviation from the normal behaviors at user or system level whereas the misuse detection matches sample data to known intrusive rules or patterns. Machine Learning frequently forms the basis for anomaly method and it can detect novel attacks by comparing suspicious ones with normal traffics but has high false alarm rate due to difficulty
of modeling normal behaviors for protected system [3]. With misuse method, pattern matching on known signatures leads to high accuracy for detecting threats but it cannot detect novel attacks as novel attack signatures are not available for pattern matching. The most of current IDS follows a signature-based or misuse approach which is similar to virus scanners, where events are detected after matching with specific pre-defined patterns known as signatures. The limitation identified with signature-based IDS is their downfall to detect new attacks and also neglect minor variations of known patterns. Besides that it is found to have significant administrative overhead cost attached with it in order to maintain signature databases.

Anomaly based detection was started using statistics. Haystack [21] used statistics to analyze changes in user activities. NSM also used statistics along with rules to analyze and monitor LAN traffic. The Nides [23] statistical component set the standard for statistical based intrusion detection as it computes a historical distribution of continuous and categorical attributes that gets updated over time and deviation are found using Chi-Square tests. Emerald statistical component was found to be inherited from Nides system.

Most of the network IDSs like Bro, NSM, NFR, NetStat use rule based methods for detection. Snort [62] is a rule based network IDS whose features include simple rule format based on packet payload inspection, content pattern matching and streamlined architecture based upon source and destination. Network Analysis of Anomalous Traffic Events (NATE) [18] and Light-weight Intrusion Detection System (LISYS) [20] are two light weight anomaly based IDS, both shares similar attributes. In early 1990s, Time-based Inductive Machine (TIM) was another anomaly based IDS, that used inductive learning of sequential user patterns in Common Lisp on a Vax 3500 computer [58].
2.4 Literature Review

2.4.1 Datasets Available for Building IDS

In order to build effective intrusion detection, there are different datasets available for research work and previously many competitions were and are still being held. The MIT Lincoln Labs developed Darpa ‘98 and ‘99 dataset [50] and its modified version, KDD ‘99 dataset [54] which is freely available for the research community to work on intrusion detection problems. Darpa dataset and KDD dataset both are very large dataset consisting of US air force local network data with a variety of simulated attacks. KDD dataset consists of 41 features, defined for each connection samples which are further divided into four categories namely features of TCP protocol, content features, time-based and host-based traffic features. It was noticed that NSL-KDD dataset [55] was developed to overcome the problems related to KDD 99 dataset. It was done by reducing the redundant problem in the dataset. It significantly reduced the size of the dataset that made the data evaluation and validation of the learning algorithm much easier and convenient. The removal of redundant records helped researchers to use dimension reduction techniques like Rough Set Theory based classifiers which make their work unbiased towards both the frequent and infrequent records, and helped them to get a more consistent result as claimed by Suthaharan et al. [1]. The other datasets that are known for testing IDS are ECML-PKDD HTTP 2007 [56] and CSIC HTTP 2010 [57].

2.4.2 Importance of Feature Selection and Dimensionality Reduction

One of the important thing that defines the quality of output of any system is an input that goes into it. Therefore, proper features need to be selected from various available attributes. Feature extraction transforms the data from the high-dimensional space to lower dimensional space [60]. The data transformation ap-
CHAPTER 2. BACKGROUND AND LITERATURE REVIEW

proaches can be linear like Principal Component Analysis (PCA), Independent Component Analysis (ICA) and Singular Value Decomposition (SVD) as well as non-linear approaches.

Suthaharan et al. [1] presented Rough Set Theory (RST) based relevance feature selection approach for capturing the scenario of network-based IDS with NSL-KDD dataset [55]. They focused on monitoring changes in the traffic patterns and detected intrusion based on anomaly detection models. They have integrated data cleaning with RST-based approach to enhance their model. In order to select relevant features, they have used rigorous steps of procedures. At first, the attack was analyzed along with normal traffic data. Secondly, dataset was cleaned to eliminate the duplicate records. After that, features were selected based on the suspicion of relevant importance and from the previous observation. Then, the new duplicates that got introduced after reducing the number of features was further cleaned, and the dataset was standardized to represent a record with respect to all other records. They visualized the obtained dataset on a 3-D space to see if additional processing was required for feature selection. If outliers were found to be present then, a data cleaning method was adopted. They claimed that the observed outliers could lead to abnormal results in RST-based intrusion detection; hence, they used data approximation technique to detect anomalies. Thus, they have used Singular Value Decomposition (SVD) and Gaussian distribution properties to eliminate the outliers. Then random sampling of attacks was performed to select unbiased equal number of attacks and normal records. After thoroughly cleaning the dataset, the dataset was discretized over range of intervals as needed that provided decision tables with the output of attacked and normal records. Later, RST was applied to this discrete dataset to classify the objects and make an appropriate decision regarding the attack records. Then, they have used lower approximation to classify the objects and calculate the level of dependency to decide which feature contributed
to attack and then, the decision was made based on weighted average degree of dependency.

Zhang et al. [2] introduced a Network Intrusion Detection System (NIDS) which used SVM method to identify the scan based on RST to reduce the number of features from 41 to 29. They trained SVM using varied frequencies of normal and attack packets which were collected from the network in every 4 seconds. Thus, SVM was able to classify whether the incoming unknown packet was normal or attack. With this approach, they claimed to correctly detect 95% of the attack packets. Their proposed system used two packages, WINPCAP and JPCAP; to capture all the packets in the network for analysis. 14 features were extracted for pre-processing in their implementation, and they have used packet sniffer to store network packet information. Feature Selection was done using RST method which helped in reducing the dimensionality of the patterns. Their model identified the intrusion by profiling normal network behavior with various attack behavior by using PCA detection.

2.4.3 Usage of Data Mining and Machine Learning Methods in IDS

Data mining is an approach used to identify useful informative patterns and relationships within the large amount of data by a variety of data analysis tools and techniques, such as classifications, associations, clustering and visualization [4]. It has obtained popularity in many research fields such as marketing, bio-medical and telecommunication along with others [5]. It has also been a preferred candidate for assisting in the analysis of network data. Data mining can provide answers to the end-users about big data problems by filtering out big data into useful subsets of information. Essentially applying data mining tools to network data provides the ability to identify the various underlying contexts associated with the network being monitored. There are two important learning strategies in data mining tech-
CHAPTER 2. BACKGROUND AND LITERATURE REVIEW

nique: supervised learning and unsupervised learning. In supervised learning, each instance of data is mapped to align with its associated label in order to find the interpretation of these pattern labels. Decision tree and estimation techniques are examples of supervised learning. Unsupervised learning is to discover a number of pattern labels, subsets, or segments within the data, with no prior knowledge of the target classes or concepts. Examples of unsupervised learning are clustering, blind signal separation and Self Organizing Maps (SOM).

Different data analysis techniques have been applied for IDS. Top-down Learning method is one such statistical technique which is used when there are general ideas about specific relations and hence is used along with mathematical computations to orient researches and studies. Machine Learning techniques are used when there is no first hand knowledge about the pattern of the data and is also known as bottom-up methods. Machine Learning and Data Mining is two relative terms which often employ the same methods and overlap significantly, but Machine Learning is about prediction based on known properties learnt from the training data whereas Data mining is about discovery of unknown properties on the data. Machine Learning includes a number of advanced statistical methods for handling regression and classification task with multiple dependent and independent variables.

Many data mining algorithms are considered good for attack detection; likewise decision trees is considered one of the most powerful and effective ways of detecting attacks in anomaly detection [6]. We have found other different machine learning approaches like Naive Bayes, Neural Network (NN), Support Vector Machines (SVM) and Random Forest (RF) that have been used to analyze and learn the traffic system. Attack signatures are required for misuse detection which can be found using approaches like decision tree (DT), different types of neural network and SVM. It was also perceived that Ensemble Learning which is a combination of
multiple machine learning approaches, being used to improve the result of instruction detection as compared to the result of a single classifier. However it was learnt that each individual base classifiers are needed to be independent of others in order to get effective and correct classification otherwise there would be no significant improvement from ensemble result. The hybrid classifier is a combinatory method based on algorithms like DT, KNN, fuzzy logic and neural networks which detects both anomaly and misuse attacks in a system [6].

2.4.4 Review of Previous Works on IDS

Intrusion Detection Model based on Machine Learning

Jun et al. [32] describes the intrusion detection system based on machine learning as the process of generating the detection rules, then collecting the appropriate data and pre-processing it before being used by the gene method to generate the model library. After which it classifies the data to predict attacks or produces further analysis results using the classifier and model [32] as shown in figure 2.5. Machine learning methods is used by the system to detect the data packets retrieved from the network. Its main components are network packet capture module, data pre-processing module, misuse rule processing module and machine learning modules; which are used to detect intrusions. Network packet capture model monitors and verifies real-time network status, workflow in order to achieve the interception and analysis of data packets in each protocol layer of the network. Data pre-processing module processes the original data packets received from the network packet module which includes packet decoding, sub-sheet and sub-flow restructuring, filtering out erroneous data and duplication of data. Rule processing module compares the collected information with the known network intrusion and system models database to find the behavior in violation of security policy. When misuse rule processing module detects an attack, the resulting treatment is carried out. Otherwise,
it would send the packet back to machine learning model for further processing to determine whether it is attack data or not. The functional module of intrusion detection system based on machine learning system is shown as in figure 2.6.

Jun et al. presented the design of the machine learning module which consisted of the generation of rule learning module and test data packet module. Learning generation rule module regularly collects a large number of network packets, and then it removes the attack data. Then, it generates the un-attacked training data along with a normal data packet rules which it gets stored in a computer system, in the form of a document for the detection of data packets. The detection data packet module would then work by opening the rules file, and calculating the anomaly score for the packet for each rule in the file. If the anomaly score is found to exceed the threshold, the alarm gets turned on.

**IDS combinatory of Machine Learning methods**

Sarvari et al. [6] presented an IDS combinatory of Machine Learning Methods (IDSCML) system in which different methods like 1NN and DT in conjunction were applied to KDD Cup dataset [54]. This ensemble approach provided the better
results than individual methods. Therefore, depending upon the improved result obtained, the other methods like 2NN, 3NN, SVM were added one at a time. It kept on showing improved attack detection result but once Learning Vector Quantization (LVQ) and Back propagation neural network [8] were added, the result got worse or remained constant. Thus, they inferred that the accuracy of the system increases by adding the number of classifiers but the improvement of the result stops at one point and remains constant from that point onwards. Their best model contained the combination of techniques of SVM, DT along with 1NN, 2NN and 3NN.

**IDS from Supervised Machine Learning methods**

Gharibian et al. [3] performed a comparative study of supervised machine learning techniques for intrusion detection in KDD dataset [54]. They also analyzed the sensitivity of supervised techniques in intrusion detection and discussed three different approaches applied for classification. First of which, included consideration of all attacks without categorization. Secondly, the attacks were classified into four
categories and last approach was about classifying attacks into normal and abnormal classes. The initial results of NB and DT showed that neither the usage of five classes nor two classes improved the classification quality [3]. Thus, Random Forest technique was then applied in NIDS to improve the detection rate, which helped to build patterns over balanced data. In order to improve the detection rate of the minority intrusions, they built balanced dataset by oversampling the majority classes and the result showed that the time to build patterns got reduced and outperformed the best results from [54] contest. The hybrid neural network and DT model for misuse detection also showed the performance improvement by selecting sample attacks from each category instead of selecting only attack categories [3].

The simulation work presented by Gharibian et al. [3] showed that the maximum detection rates in each attack category for the three population categories remained same. Their predictive techniques (RF and DT) showed good results in detecting ‘DoS’ attack while probabilistic techniques like Gaussian and NB showed better results in other attack categories for KDD dataset. It was observed that the Gaussian technique exhibited the least sensitivity to different training dataset, as Gaussian also showed least sensitivities to Probe and U2R attacks compared to DT and RF. Thus, it indicated that the probabilistic techniques showed more robustness than predictive techniques when trained using different training datasets. Also, with the dataset with lesser samples such as R2L, U2R and Probe, the probabilistic techniques showed better detection rate. However, with datasets containing more sample of DoS attacks, the predictive techniques were found to perform better.

Secure Machine Learning

Liao et al. [9] introduced various methodology in constructing a secure machine learning system against various attacks. They claimed that the most of machine learning algorithms assume the training data and evaluation data were drawn from
the same distribution, but the data were found manipulated by adversary through modeling the adversary condition to make the learner produce false negatives. Thus, they provided the insight for making secure machine learning by developing a framework for representing the interaction between classifier and adversary, and presenting a classifier that is optimal for the presence of given adversary condition. They also highlighted methodology of two-person Stackelberg game to model the interaction between an adversary and the learner, where the adversary changes its strategy to avoid being detected by the current learner, while the learner updates itself based on new possible threats. Thus, adversary tries to maximize its return from false negative items while learner tries to minimize the error cost. They reported that solving linear Stackelberg game is NP-hard and claimed that a simulated annealing algorithm is used for the equilibrium condition when two players knows each other’s payoff and genetic algorithm is used when the players are unaware about each other’s payoffs. They have reported that about 55 studies have been performed between 2000 and 2007 focusing on developing either single, hybrid or ensemble classifiers. They have also highlighted different potential attacks like Polymorphic Blending Attack (PBA), whose main objective is to create each polymorphic instance in such a way that the statistics of attack packets match the normal traffic profile [9]. Thus, they claimed that generating a PBA that optimally matched the normal profile was NP-complete hard problem. They have further stated that this problem could be reduced to Integer Linear Programming problem to find a near-optimal solution. The other way the intruders could avoid intrusion detection was by modeling an offensive mechanism to exploit the weakness of the anomaly based intrusion detection by making the IDS blind in the presence of common attacks.
2.4.5 Intrusion Detection Using Outlier Detection Technique

Outlier can be understood as an observation that deviates from the other normal observations as to cause suspicion that it was generated by a different mechanism. In intrusion detection, the outliers are of two types. The first type is the one that deviate significantly from others within their own network peripherals while the second type is the one whose patterns belongs to other network services other than their own service. It was also observed that researchers have tried two approaches either to model only normal data or both normal and abnormal data for finding intruders. Modelling both the normal and abnormal data allows the system to be tight on false positive and false negative as both the normal as well as abnormal data needs to be modelled but it puts the limitation in modeling the abnormal patterns. Similarly, using only normal patterns allows the system to model the boundaries of normal data but due to the unknown nature of the boundaries of abnormal data, it gives a possibility of alarming false positive, as well. Thus, a proper tuning needs to be done for defining the threshold of the system to balance the number of true positives and false positives. Outlier detection has been a popular technique for finding intrusions, which can be observed with the amounts of work done in papers [11, 12, 13, 14]. Most of the research works have pointed out that it is extremely difficult to find out outliers directly from high dimensional datasets so, most of the researches has been seen proceeding with the reduction of the dimensionality of the dataset[30].

There are various approaches for Outlier Detection, which includes Model-based approach, Proximity-based approach and Angle-based approach [24]. In model based approach, we apply the model to represent normal data points, and we assume those points that do not fit the model are outliers. Here, probabilistic tests based on statistical models, depth-based approaches and deviation-based approaches can be used for this outlier detection model. In Proximity-based ap-
proaches, we examine the spatial proximity of each object in the data space and then we consider the object as an outlier if the proximity of an object deviates considerably from the proximity of other objects. Here, distance-based approaches and density-based approaches can be used for this outlier detection model. The rationale behind angle-based approach is that we measure the spectrum of pairwise angles between a given point and all other points. So, the outliers would be the points that have a spectrum featuring high fluctuation problems.

Stream Projected Outlier deTector (SPOT)

Kershaw et al. [11] presented an anomaly based network intrusion detection using outlier subspace analysis. They have implemented a SPOT, capable of quickly processing high dimensional streaming data analyzing its stream. It employs a window based time model that summarizes a decaying data which allows data to be processed timely and effectively. The normal behavior of a process is modeled using the collection of specific set of system call combinations. In the first step of offline learning, a Sparse Subspace Template (SST) is constructed using a group of subspaces to detect outliers. A multi-objective genetic algorithm is used to produce these subspaces. And any arriving data mapped to a cell with low density achieves a high abnormality value and are placed in an ‘Outlier Repository’. Thus, this repository is used to determine outlier threshold which helps to distinguish normal and intrusive traces.

Unsupervised Anomaly-based Outlier Detection

Zhang et al. [14] presented an anomaly based outlier detection by using proximities of network service type. In their work, they have performed pattern mining of network services using random forest algorithm and have used concept of proximities of network service type to built an IDS. Da et al. [12] have used k-means
clustering to model their normal patterns and predicted attacks based on outlier factors determined by using statistical test. Pareek et al. [15] have used SOM approach to model the behavior learning of internet using randomized parameters with weighted factors, and have used Euclidean distance to obtain Best Matching Unit (BMU) for each parameter which gets updated over satisfying their model.

**Reduction of False Positives using Outlier Detection**

Xiao et al. [16] came up with two-phase framework for an outlier detection. The first learning phase consists of building a feature set of false positives which gets updated over a period of time and a threshold of true alert is calculated. And in the second phase of online filtering, the outlier score of each new alert is compared with this threshold to determine where it is false positive or not.

**Outlier Ensemble Detection**

Huang et al. [17] used PCA to reduce the dimensionality of the KDD dataset. They have used two outlier mining algorithms, one is based on similar coefficient sum, and the second is based on kernel density to determine the anomaly set. And then, the two outlier mining algorithms are repeated M no. of times. An ensemble approach using voting mechanism provides the final anomaly set.

**Network Analysis of Anomalous Traffic Events (NATE) System**

Taylor et al. [18] [19] presented a method based on statistical approach, that addresses the problems of monitoring high speed network traffic and the time constraints using only packet headers. NATE [18] [19] is a network intrusion detection system that analyzes TCP packets with low cost in real time. It performs clustering of normal network traffic behaviors. Deviations of a TCP header data from the existing clusters was used as the measure of detection. Thus, it allows monitor-
ing of high speed network traffic because once the normal pattern is constructed, it significantly reduces analysis complexity and uses little system resources. They have experimented using a sampling method in the selected files of Darpa ‘98 and ‘99 dataset. They performed a cluster analysis and found 7 such clusters using Euclidean distance that represents their normal samples while they have used both Mahalanobis distance[18] as well as Euclidean distance[19] as a measure to detect intrusions, to perform the boundary analysis using Chebyshev’s inequality. They have identified that false positives could be a potential problem with this tool and hence, the significance level could be used as a tuning parameter in response to decrease the number of false positives. Our approach to this research is based on the ideas of this system using cluster analysis in TCP header information.
3.1 Objectives

It is always easier to know the behaviour of normal patterns as that of abnormal. When we completely model the normal data then anomaly can be suspected measured by its deviation from a normal data. Thus, our objective here is to model the normal network data and then predict the abnormality based on the deviation from the normal patterns. The normal data are also of various types and shows various kinds of behaviors so; it is important to understand and analyze the behavior for each of these normal patterns and try to distinguish one another from abnormal data. Thus, we try to explore the TCP protocol in the network data and traverse through the TCP headers to understand the normal patterns. Exploring the TCP headers allows monitoring of high speed network traffic as it can be performed in real time, and once the normal pattern is constructed, it greatly reduces analysis complexity. Therefore, the purpose of this research is to identify suitable algo-
rchms and methods to model the normal network data and then, implement the same to materialize the improved output.

3.2 System Architecture

Our approach to use basic TCP headers helps to detect the anomaly intrusions in real time. It was important that the model that we have used, could be generalized to work with training, validation and test datasets and hence, subsequently even with any other dataset containing similar features. We tried modelling using different models like clustering and one class classifiers and tried to vary over a range of different model parameters in order to achieve a stable and precise model. Hence, finally we came up with a model that has highest silhouette value, accuracy and recall along with a high degree of consistency.

The proposed architecture of the system is depicted in figure 3.1, where we take the tcpdump data files, analyze the TCP headers and then predict the anomaly based intrusions. Our model can be used to classify any network data using the similar set of features, but the clustering analysis needs to be done in order to measure the correct number of clusters required to model the normal data.

As shown in figure 3.1, our proposed model follows a sequential architecture where it uses the tcpdump network data files and then the TCP headers information is extracted from it to generate the features that describe the behavior of the TCP protocol network data. Then, we have applied Principal Component Analysis (PCA) technique to reduce the dimensionality of our feature set. It allows to reduce the number of centers for clustering purpose that also allows to simultaneously reduce the processing time. After that, the training set of normal network data is divided into clusters to represent the normal model by means of k-means clustering. Then, the Euclidean distance is used to measure the distance from k-centres to
Figure 3.1: Overall architecture of the Outlier-based IDS
each training and testing points. Thus, distance values from each k-centres to training and validation data are sent to One Class SVM method. The results from these two techniques are used to create our model before performing testing on the test set to give the final output of our system. We have implemented an outlier based anomaly detection method that differs from other approaches in its speed and ease of use.

3.3 Tools Used

3.3.1 Wireshark

Wireshark [65], previously known as Ethereal, is a free and open-source packet analyzer tool used for network troubleshooting and analysis for Unix and Windows systems. It has been also used as a packet sniffing application and uses the pcap library to capture packets. It traces network data using a libpcap file format supported by libpcap and WinPcap, so it can exchange captured network traces with other applications that use the same format, including tcpdump. Wireshark is a tool that understands the structure of different networking protocols and hence can parse the fields, along with their meaning as specified by different networking protocols. Apart from capturing live traffic, it can also read many different capture file formats. This tool was very useful for us in analyzing the raw network data i.e. tcpdump file and understanding the network data, which also allows different filtering options to visualize the network data.

3.3.2 WEKA

WEKA [33] is a tool which consists of numbers of machine learning algorithms for data mining tasks such as data preprocessing, classification, clustering, regression, association rule mining along with data visualization functions. It is freely
available for downloading and can be used for academic purpose. We have used this for calculating ‘Information Gain’ and ‘Gain Ratio’.

3.3.3 MATLAB

Matlab [34] is a tool used for numerical computation, visualization and programming. It provides an easy platform to code and analyze data along with in-built functions to carry-out classification, clustering and visualization tasks. It can be also used with other programming languages such as C/C++, Java and Python. It has a rich set of toolkits which are easy to use and helps to reduce the time and increase the efficiency of the work. We have used Matlab for the purpose of our main processing which includes performing clustering and classifying the normal patterns using one-class SVM.

3.3.4 jNetPcap

jNetPcap [66] is an open-source java library which contains java wrapper for nearly all libpcap library native calls. jNetPcap uses a mixture of native and java implementation for optimum packet decoding performance. It helps to decode captured packets in real-time and provides a large library of network protocols. It provides a framework for the users to easily add their own protocol definitions using java SDK. We have used this library for the extraction of TCP headers information.

3.3.5 LIBSVM

LIBSVM [36] is a library for Support Vector Machines. It supports vector classification, (C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR) and distribution estimation (one-class SVM). It also offers multi-class classification. LIBSVM provides a simple interface which users can easily use to link it with their own programs. We have used this tool for the purpose of using one-class SVM in Matlab.
The main features of LIBSVM are:

- Deals with different SVM formulations
- Allows efficient multi-class classification
- Can perform cross validation for model selection
- Allows probability estimation
- Easy use of various kernels (including precomputed kernel matrix)
- Allows to use weighted SVM for unbalanced data
- Provides both C++ and Java sources
- Provides a simple GUI demonstrating SVM classification and regression
- Provides extension to tools like Python, R, MATLAB, Perl, Ruby, Weka, Common LISP, CLISP, Haskell, OCaml, LabVIEW, and PHP interfaces along with C#.NET code and CUDA.
- Allows automatic model selection which can generate contour of cross validation accuracy.

3.4 Algorithms Used

3.4.1 Principal Component Analysis (PCA)

Principal Component Analysis is used to compute a linear transformation by mapping data from a high dimensional space to a lower dimension. The first principal components shows the highest contribution to the variance in the original dataset and so on. Therefore, in the dimension reduction process, the last few components can be discarded as it only results in minimal loss of the information value. The transformation works in the following way [25]:

\[
X' = XW
\]
CHAPTER 3. PROPOSED ARCHITECTURE

Considering a set of observations \(x_1, x_2, \ldots, x_M\) are \(N \times 1\) vectors where each observation is represented by a vector of length \(N\) and can be represented as below matrix.

\[
X_{M \times N} = \begin{bmatrix}
x_{11} & x_{12} & \cdots & x_{1N} \\
x_{21} & x_{22} & \cdots & x_{2N} \\
\vdots & \vdots & \ddots & \vdots \\
x_{M1} & x_{M2} & \cdots & x_{MN}
\end{bmatrix} = [x_1, x_2, \ldots, x_M]
\]

The mean value for each column is defined by the expected value as shown in Equation 3.1. Once the mean value is subtracted from the data yields expression Equation 3.2.

\[
\bar{x} = \frac{1}{M} \sum_{i=1}^{M} x_i \tag{3.1}
\]

\[
\phi_i = x_i - \bar{x} \tag{3.2}
\]

\(C\) is a correlation computed from matrix \(A=[\phi_1 \phi_2 \ldots \phi_M] (N \times M)\) matrix as shown in Equation 3.3. Sampled \(N \times N\) covariance matrix characterizes how data is scattered [26].

\[
A = [\phi_1 \phi_2 \ldots \phi_M], C = \frac{1}{M} \sum_{n=1}^{M} \phi_n \phi_n^T = AA^T. \tag{3.3}
\]

The eigenvalues of \(C\): \(\lambda_1 > \lambda_2 > \ldots > \lambda_N\) and the eigen vectors of \(C\): \(u_1, u_2, \ldots, u_N\) have to be calculated. Because \(C\) is symmetric, \(u_1, u_2, \ldots, u_N\) form a basis (i.e. any vector \(x\) or actually \((x - \bar{x})\) can be written as a linear combination of the eigen vectors as shown in Equation 3.4.

\[
x - \bar{x} = b_1.u_1 + b_2.u_2 + \ldots + b_N.u_N = \sum_{i=1}^{N} b_i.u_i \tag{3.4}
\]

During the dimensionality reduction, only the terms corresponding to the \(K\)
largest eigen-values are taken into consideration as shown in Equation 3.5 [27]

\[ \hat{x} - \bar{x} = \sum_{i=1}^{K} b_i u_i, \text{where } K << N \] (3.5)

The representation of \( \hat{x} - \bar{x} \) into the basis \( u_1, u_2, \ldots, u_K \) is thus, \([b_1 b_2 \ldots b_K]^T\) The linear transformation \( R^N \rightarrow R^K \) by PCA that performs the dimensionality reduction is shown in Equation 3.6.

\[
\begin{bmatrix}
    b_1 \\
    b_2 \\
    \vdots \\
    b_K
\end{bmatrix}
\begin{bmatrix}
    u_{1}^T \\
    u_{2}^T \\
    \vdots \\
    u_{K}^T
\end{bmatrix}
(x - \bar{x}) = U^T (x - \bar{x})
\] (3.6)

If we consider the new variables (i.e. \( b_i \)'s) to be uncorrelated. The co-variance matrix for the \( b_i \)'s can be represented in Equation 3.7. The covariance matrix represents only second order statistics among the vector values.

\[
U^T. CU =
\begin{bmatrix}
    \lambda_1 & 0 & 0 & 0 \\
    0 & \lambda_2 & 0 & 0 \\
    0 & 0 & \ddots & 0 \\
    0 & 0 & 0 & \lambda_n
\end{bmatrix}
\] (3.7)

Suppose \( n \) be the dimensionality of the data. The covariance matrix is used to calculate a diagonal matrix, \( U^T CU \). \( U^T CU \) is sorted and rearranged in the form of \( \lambda_1 > \lambda_2 > \ldots > \lambda_N \) so that the data exhibits greatest variance in \( y_1 \), the next largest variance in \( y_2 \) and so on, with minimum variance in \( y_n \) [28][29].
3.4.2 Support Vector Machines (SVM)

Support Vector Machine is a supervised learning model with associated learning algorithms that perform regression and classification tasks by constructing nonlinear decision boundaries [39]. SVM implements the idea of mapping feature vectors to higher dimensions to use for classification purpose. SVM exhibits a large degree of flexibility in handling classification and regression tasks of varied complexities due to the nature of the feature space in which boundaries are defined. The basic SVM treats two class label problems in which the data are separated by a number of support vectors. And then the boundary is defined between the two classes when SVM cannot separate two classes. Firstly, it maps the input data into high-dimensional feature space using the kernel function [40]. It uses kernel functions to convert the feature into higher dimensions. During the process of training, only few instances among the large training dataset are identified as support vectors. Then, a simple linear decision boundaries are used to classify based on those support vectors. The inherent problem in SVM lies in defining the kernel mapping function. Trial and error method is usually used in order to determine better kernel function which are of following types; Linear, Polynomial, Radial basis and Sigmoid.

These kernels can be represented mathematically as:

\[
\text{Linear} : K(x_i, x_j) = x_i^T x_j; \quad (3.8)
\]

\[
\text{Polynomial} : K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0; \quad (3.9)
\]

\[
\text{Radial Basis Function} : K(x_i, x_j) = exp(-\gamma \|x_i - x_j\|^2), \gamma > 0; \quad (3.10)
\]
\[ Sigmoid : K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r); \] (3.11)

Here, \(\gamma\), \(r\) and \(d\) are kernel parameters.

**One Class-Support Vector Machines (OC-SVM)**

The OC-SVM was used as a support vector method for outlier detection [45]. Let us suppose a dataset \(\{x_i\}_{i=1}^{l} \in \mathbb{R}^N\) which belongs to a given class of interest known as target class. The objective of OC-SVM is to describe the target class by a function that maps it to a region where the function can be non-zero. Thus, for an outlier, origin can be treated as the only available member of the non-target class then, the problem can be treated by exploring a hyperplane with a high-dimensional Hilbert feature space \(H\) such that samples are mapped through a non-linear transformation \(\phi(.)\) as shown in figure (Left) 3.2 [47].

To separate the dataset from the origin, we need to minimize

\[
\min_{\omega, \rho, \xi} \left\{ \frac{1}{2} \|\omega\|^2 - \rho + \frac{1}{vl} \sum_{i} \xi_i \right\}, \quad \forall i = 1, ..., l \tag{3.12}
\]

constrained to

\[
(\omega, \phi(x_i)) \geq \rho - \xi_i \tag{3.13}
\]

In this context, \(\omega\) is a vector which is perpendicular to the hyperplane \(H\) and \(\rho\) is the distance to the origin. A set of slack variables \(\xi_i \geq 0\) is used to deal with outliers, similarly to the SVM framework. The parameter \(v \in (0,1]\) manages the tradeoff between the number of data of the training set mapped as positive by the decision function \(f(x) = \text{sgn}(\langle \omega, \phi(x) \rangle) - \rho\) and having a small value of \(\|\omega\|\) in order to control a model complexity.

After applying limitations from equation 3.13 into equation 3.12 through the
Figure 3.2: Left, OC-SVM: Hyperplane distinguishes all target data from the origin mapping normal (blue) data, towards the above of the hyperplane and outliers (red), towards the below. Green are the points required to make predictions. Right, SVDD: The hyper-sphere surrounds all target normal samples where, \( a \) and \( R \) respectively represents centre and radius of the hyper-sphere.

Use of Lagrange multiplier \( \alpha_i \neq 0 \), i.e., \( \omega = \sum \alpha_i \phi(x_i) \). Then, we can introduce a kernel function which elements defined as \( K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \) avoids the direct computation of the nonlinear mapping \( \phi(\cdot) \) and the dual problem can be represented as in Equation 3.14.

\[
\min_{\alpha} \left\{ \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) \right\} \tag{3.14}
\]

constrained to

\[
0 \leq \alpha_i \leq \frac{1}{vl} \sum \alpha_i = 1 \tag{3.15}
\]

After solving for the dual problem, a set of model weights \( \alpha_i \) can be obtained and the decision function for any test vector \( x_* \) can be represented 3.16.

\[
f(x_*) = \text{sgn}(\sum \alpha_i K(x_i, x_*) - \rho) \tag{3.16}
\]

There exists an alternative and equivalent formulation of OC-SVM, which is
known as Support Vector Domain Description (SVDD) [46]. SVDD attempts to find a hypersphere with minimum volume surrounding only the target class samples instead of looking for the hyperplane which separates the samples of the target class from origin with maximum separation as shown in figure (Right) 3.2. This problem can be defined as:

$$\min_{R,a} \{R^2 + \frac{1}{\nu n} \sum_i \} \forall i = 1,\ldots,l$$  \hspace{1cm} (3.17)

subjected to

$$\|\phi(x_i) - a\|^2 \leq R^2 + \xi_i \forall i = 1,\ldots,l$$ \hspace{1cm} (3.18)

$$\xi_i \geq 0 \forall i = 1,\ldots,l$$ \hspace{1cm} (3.19)

### 3.4.3 k-means Clustering

k-means clustering is a cluster analysis technique whose objective is to partition n observations into k-clusters such that each observation belongs to the cluster with the nearest mean. k-means clustering divides n observations into k sets (k \leq n) S = S_1, S_2, \ldots, S_k so as to minimize the sum of squares within a cluster which is represented by Equation 3.20.

$$\arg\min_S \sum_{i=1}^{k} \sum_{j \in S_i} \|x_j - \mu_i\|^2,$$ \hspace{1cm} (3.20)

where $\mu_i$ is the mean of points in $S_j$. 
Silhouette Value

The quality of a clustering result can be evaluated using 'silhouette analysis' [53]. The silhouette value $s(i)$ of a data point $i$ is defined as in equation 3.21

$$s(i) = \frac{(b(i) - a(i))}{\max(a(i), b(i))}$$  \hspace{1cm} (3.21)

where $a(i)$ is the average distance of $i$ to all other points in its own cluster, and $b(i)$ is the average distance of $i$ to all points in its nearest neighbor cluster. The silhouette value provides a reference to how well each data point lies within its cluster. Its value range from -1 (the worst case) to +1 (the ideal case).

3.5 Pseudocode for Process Flow

The pseudocode for the processes can be seen summarized with the three algorithms which is depicted in following Algorithms 1, 2 and 3.
Algorithm 1 Algorithm-cluster-analysis\((trainSet,validSet,testSet)\)

1: \texttt{procedure} CLUSTER-ANALYSIS\((trainSet, validSet, testSet)\)

2: \hspace{1em} \( A \leftarrow trainSet \) \quad \triangleright \text{Read attributes after extracting TCP headers}

3: \hspace{1em} \( B \leftarrow validSet \)

4: \hspace{1em} \( C \leftarrow testSet \)

5: \hspace{1em} \( A \leftarrow \text{normalized}(A) \) \quad \triangleright \text{Normalizing dataset}

6: \hspace{1em} \( B \leftarrow \text{normalized}(B) \)

7: \hspace{1em} \( C \leftarrow \text{normalized}(C) \)

8: \hspace{1em} \( trainFeature \leftarrow \text{PCA}(A, 8) \) \quad \triangleright \text{Extract 8 principal components from overall attributes}

9: \hspace{1em} \( validFeature \leftarrow \text{PCA}(B, 8) \)

10: \hspace{1em} \( testFeature \leftarrow \text{PCA}(C, 8) \)

11: \quad [\text{index, centre}] \leftarrow kMeans(trainFeature, k, 'distance', 'sqEuclidean'); \quad \triangleright \text{Perform k-means clustering in the training dataset, which return k centre values and index for each instance in the dataset}

12: \quad i \leftarrow 0

13: \quad \textbf{while} \ i < |trainFeature| \ \textbf{do}

14: \quad \hspace{1em} trainDist_k(i) \leftarrow \text{euclideanDistance}(\text{centre}(k), trainFeature(i)) \quad \triangleright \text{Calculate Euclidean distance between each k centre and training instance}

15: \quad \hspace{1em} i \leftarrow i + +

16: \quad \textbf{end while}

17: \quad j \leftarrow 0

18: \quad \textbf{while} \ j < |validFeature| \ \textbf{do}

19: \quad \hspace{1em} validationDist_k(j) \leftarrow \text{euclideanDistance}(\text{centre}(k), validFeature(j)) \quad \triangleright \text{Calculate Euclidean distance between each k centre and validation instance}

20: \quad \hspace{1em} j \leftarrow j + +

21: \quad \textbf{end while}

22: \quad \textbf{return} \ validationDist_k \quad \triangleright \text{Returns Euclidean distance of validation dataset}

23: \texttt{end procedure}


**Algorithm 2** Algorithm-modelingNormalInstances\((trainDist_k, validDist_k)\)

1. **procedure** MODELINGNORMALINSTANCES\((trainDist, validDist)\)
2. modelOneTrainFeature ← \([\text{trainDist}_1, \text{trainDist}_2, \ldots, \text{trainDist}_k]\)
3. modelOneValidFeature ← \([\text{validDist}_1, \text{validDist}_2, \ldots, \text{validDist}_k]\)
4. modelOne ← svmTrain\((\text{labelOfTrainingSet}, \text{modelOneTrainFeature}, 'OC-SVM'); \) ▷ Perform One-Class SVM Classification
5. predictLabel ← svmPredict\((\text{modelOneValidFeature}, \text{modelOne})\); ▷ Predict labels of validation dataset
6. \(m ← 0\)
7. \(n ← 0\)
8. **while** \(m < |\text{validFeature}|\) **do**
9. **if** \((\text{predictLabel}[m] = \text{labelOfValidSet}[m] = 0)\) **then** ▷ Model only normal instances detected by the first model
10. newModelFeature\((n,:)\) = validFeature\((m,:)\); ▷ Create new Feature consisting only normal instances from validation dataset detected by the first model
11. \(n ← n + +;\)
12. **end if**
13. \(m ← m + +;\)
14. **end while**
15. **return** newModelFeature ▷ Returns normal instances in the validation dataset detected by the first Model
16. **end procedure**

**Algorithm 3** Algorithm-predictOutput\((\text{newModelFeature}(i,j), \text{testSet}(i,j))\)

1. **procedure** PREDICTOUTPUT\((\text{newModelFeature, testFeature})\)
2. modelFinal ← svmTrain\((\text{labelOfNewModelFeature, newModelFeature}, 'OC-SVM'); ▷ Perform One-Class SVM Classification on new model feature
3. predictFinalLabel ← svmPredict\((\text{testFeature, modelFinal})\); ▷ Predict labels of testing dataset
4. **return** predictFinalLabel ▷ Returns final output of the outlier based anomaly IDS
5. **end procedure**
Experiment I (Using Darpa dataset)

Once the intrusion problem was analyzed, and the objectives of the research were visualized, the experiment was carried out in this standard Darpa dataset using publicly available tools. The description about the dataset, evaluation procedure and metrics, processing along with result analysis are presented in this chapter.

4.1 Darpa Dataset

We used a publicly available MIT Darpa ‘99 dataset [50] for our analysis that was explicitly created for testing IDS. The Darpa dataset was created by MIT Lincoln Labs for the evaluation study for which they published two dataset in ‘98 and ‘99. It is a very large dataset consisting of US air force local network data with a variety of simulated attacks. They have been a sort of benchmark for IDS for a long time now. One drawback with Darpa dataset concerns with its simulated nature as pointed out by McHugh [10]. The network architecture of Darpa 1999 can be
CHAPTER 4. EXPERIMENT I (USING DARPA DATASET)

Table 4.1: Top 10 frequency distribution of packets by traffic-type seen in [Appendix A.1]. However, we chose to proceed with this dataset because our results would be comparable with others when we use standard and recognized dataset. We have used the three weeks of data starting from third, fourth and fifth. The third week of dataset comprises of only normal packets and hence used for training purpose. While fourth and fifth week’s dataset containing attacks have been used for validation and evaluation purpose of our system. Each daily tcpdump file consists up to 1 million TCP records. We have found in total, 4 corrupted tcpdump files, which is also reported by [35]. However, we were able to recover only partial content from those corrupted files and proceed with it. The third week dataset consists of 16 tcpdump files which comprise of 6.46 GB, captured by inside and outside packet sniffer. The dataset of the fourth week consist of 9 tcpdump files which are of 2.80 GB while the fifth week of dataset is 4.85 GB consisting of 10 tcpdump files. The simulated attack fall in one of the following four categories. The underlying different attacks under these categories can be found in [Appendix A.2].

1. Denial of Service Attack (DoS): In this category, the attacker makes full usages of computing or memory resources so that it cannot handle other legitimate requests or deny legitimate user access to the machine.
CHAPTER 4. EXPERIMENT I (USING DARPA DATASET)

2. User to Root Attack (U2R): In U2R, the attacker attacks with normal user account and then exploits the vulnerability of the system to obtain root access to it.

3. Remote to Local Attack: The attacker in this category sends packets to the machine over a network but does not possess an account on that machine and hence, exploits the vulnerability of the system to gain local access as a user of that machine.

4. Probing Attack: Here, the attacker attempts to gather information about network of computers for the apparent purpose of circumventing its security.

We observed using Wireshark [65] that most of the tcpdump packets are IP packets, among which TCP, Telnet and HTTP packets takes more than 95%, which can be seen in frequency distribution table 4.1. The different types of TCP attacks that we have selected from the Darpa 1999 dataset are Portsweep, Satan, Neptune, Mailbomb and nmap with its details as shown in table 4.2.

<table>
<thead>
<tr>
<th>Attacks</th>
<th>Description</th>
<th>Attack impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portsweep</td>
<td>Scans multiple ports for services</td>
<td>Probe Services</td>
</tr>
<tr>
<td>Neptune</td>
<td>Syn flood denial of service</td>
<td>Temporary Denial of Service</td>
</tr>
<tr>
<td>Satan</td>
<td>Networking probing tool</td>
<td>Probe Services</td>
</tr>
<tr>
<td>nmap</td>
<td>Network Mapping using nmap</td>
<td>Probe Machines</td>
</tr>
<tr>
<td>Mailbomb</td>
<td>DOS for the mail server</td>
<td>Deny permission temporarily</td>
</tr>
</tbody>
</table>

Table 4.2: Description and attack impacts of selected attacks from Darpa 1999 dataset

In our experiment, we look at the distribution of session numbers over the three weeks data. Based on the assumption presented by [18] system, we define our ‘session’ as the aggregated network traffic consisting of all the traffic packets between a unique combination of source IP and port number to its destination IP and port number. This definition varies from the standard one practised in network traffic, which is one complete TCP connection beginning with SYN flags from source, re-
CHAPTER 4. EXPERIMENT I (USING DARPA DATASET)

4.1 Attack Identification and Labeling of Evaluation Dataset

Table 4.3: Frequency distribution of session counts by category

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Types</th>
<th>Session Count</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Training Set</strong></td>
<td>Normal</td>
<td>1318656</td>
</tr>
<tr>
<td></td>
<td>Attacks</td>
<td>0</td>
</tr>
<tr>
<td><strong>Validation Set</strong></td>
<td>Normal</td>
<td>1072427</td>
</tr>
<tr>
<td></td>
<td>Attacks</td>
<td>159929</td>
</tr>
<tr>
<td><strong>Testing Set</strong></td>
<td>Normal</td>
<td>639326</td>
</tr>
<tr>
<td></td>
<td>Attacks</td>
<td>2304</td>
</tr>
</tbody>
</table>

Table 4.4: Statistics of Data Transformation from raw data to processed data

<table>
<thead>
<tr>
<th>Data</th>
<th>Total No. Of packets</th>
<th>No. Of TCP Packets</th>
<th>No. Of TCP Sessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 3</td>
<td>28014164</td>
<td>25289485</td>
<td>1318656</td>
</tr>
<tr>
<td>Week 4</td>
<td>13773651</td>
<td>11723806</td>
<td>643934</td>
</tr>
<tr>
<td>Week 5</td>
<td>21336421</td>
<td>18240162</td>
<td>1392279</td>
</tr>
</tbody>
</table>

The raw network data i.e. tcpdump file does not consist of any attack information. It only comprises of the information about the packets and its header information. Thus, the work was required to be done to create the evaluation dataset, identifying
attacks. MIT Lincoln Lab provides the summarized attack information consisting of start time, duration, source and destination [67, 68]. Therefore, traffic is labelled attack for all the packets based on the starting time of the recorded attack till the duration time from that particular source to its destination.

The distribution of attacks and its composition in Week 4 and Week 5 is depicted in figure 4.1 and 4.2.

4.2 Evaluation Procedure and Metrics

For the evaluation of algorithms, we have used two step evaluation on the provided dataset. The provided dataset consist of three weeks of data, the third week’s dataset, we have used for Training purpose. The fourth week’s dataset is for vali-
CHAPTER 4. EXPERIMENT I (USING DARPA DATASET)

Figure 4.2: Distribution of Attacks in 5th week
CHAPTER 4. EXPERIMENT I (USING DARPA DATASET)

Table 4.5: Confusion matrix

dation while fifth is used for testing purpose. We form our model using a Training dataset and then update the model based on the validation set and finally, use the testing set for evaluation purpose to achieve higher accuracy.

4.2.1 Performance Metrics

It is necessary to evaluate our model by certain evaluation criteria. The commonly used criteria for performance evaluation are False Positive Rate, Accuracy, Precision and Recall. In our experiment, we have used above mentioned three measures along with F- Measure. The below specified confusion matrix in table 4.5 allows visualization of the performance of an algorithm. The False Positive Rate, Precision, Recall and Accuracy can be calculated as:

\[
\begin{align*}
\text{True Positive Rate (TPR)} & = \frac{True \text{ Positive}}{True \text{ Positive} + False \text{ Negative}} \quad (4.1) \\
\text{False Positive Rate (FPR)} & = \frac{False \text{ Positive}}{True \text{ Negative} + False \text{ Positive}} \quad (4.2) \\
\text{Precision} & = \frac{True \text{ Positive}}{True \text{ Positive} + False \text{ Positive}} \quad (4.3) \\
\text{Recall} & = \frac{True \text{ Positive}}{True \text{ Positive} + False \text{ Negative}} \quad (4.4) \\
\text{Accuracy} & = \frac{True \text{ Positive} + True \text{ Negative}}{True \text{ Positive} + True \text{ Negative} + False \text{ Positive} + False \text{ Negative}} \quad (4.5)
\end{align*}
\]
Receiver Operating Characteristic (ROC) Curve

ROC curve which is also known as relative operating characteristic curve, is a graphical plot which shows the performance of a two classifier system. It compares two operating characteristics of TPR and FPR as the criterion or threshold changes. It is plot of the fraction of true positives out of the total actual positives (TPR) against the fraction of false positives out of the total actual negatives (FPR) at various threshold settings. ROC analysis provides ways to select possibly optimal models and to reject suboptimal ones which are independent from either cost context or from the class distribution [38].

Area Under Curve (AUC)

The area under the ROC curve gives the probability that a classifier will rank a randomly selected positive instance higher than a randomly selected negative one. It can be calculated as shown in equation 4.6.

\[
AUC = \int_{-\infty}^{\infty} TPR(T).FPR'(T).dT = \int_{-\infty}^{\infty} TPR(T).P_0(T).dT = \langle TPR \rangle, \quad (4.6)
\]

where, \( T \) is threshold parameter and \( P_0 \) is probability of not belonging to class. The angular brackets represent average from the distribution of negative samples.

4.3 Experimental Process Flow

The experimental process flow of our experiment conducted is shown in figure 4.3. Feature creation and selection is the primitive step used in most classification or regression tasks. It is a frequently used process in machine learning which starts from either the creation of new features from the data or from the selection of an optimal subset from the pool of existing features set. The selected subset of features or feature reduction process contributes in increasing the accuracy of the model.
Figure 4.3: Process flow diagram
CHAPTER 4. EXPERIMENT I (USING DARPA DATASET)

Table 4.6: Basic features extracted from TCP/IP packets

The experimental flow from the raw captured network data to the final model is explained in this section. We start extracting basic information from tcpdump files.

4.3.1 Pre-Processing

The tcpdump files are a raw network data from which tcp information can be extracted using pcap libraries. We have used jNetPcap library in Java to extract basic TCP header information [Appendix A.2 ] along with other packet information that can be seen in the table 4.6. The extracted information is feed into the MySql database where we separate the TCP/IP protocol from other protocols. First, we defined the sessions, based on the unique combination of Source IP and Source Port along with Destination IP and Destination Port for that particular file.
4.3.2 Feature Engineering

Intrusion data consists of multidimensional feature. Since not all features contribute to intrusion thus, only relevant feature needs to be selected. Zhang et al. [2] mentioned that the classifier could be designed by feature extraction and feature selection by reducing the dimensionality of the patterns. A feature could be relevant or irrelevant and could hold a different probability for discriminating a class. RST and PCA are commonly used techniques to find the subset of relevant features as it allows in determining the most important attributes from a classificatory point of view since, much information in an information system are redundant and needs to be eliminated without losing their essential information that can classify the data.

4.3.3 Feature Selection

Feature Selection is a crucial step towards building any system such as robust intrusion detection system. Once, session is defined, then the first basic features are created using SQL queries, where we define features like FSR, which can be seen in [Appendix A.1]:

\[
FSR = \frac{FIN + SYN + RST}{\text{Packets count in a session}}
\]

This feature FSR shows the weightage of connection control flags in total packets of the session. It was observed that FSR for abnormal traffic has high values. Since, we are dealing in sessions, and not in packets level. Therefore, we are dividing each feature with the packets count (except itself) in that session to normalize our feature parameters. After the basic features have been normalized, the values were still varying significantly from one feature to another so, we used a natural logarithmic transformation. In order to avoid negative infinite values because of logarithm on zero counts, we have modified and used a transformation function as
f(x)=\log(1+x). After this transformation, we have applied the PCA technique to reduce the dimensionality of our new feature set to 8 which covers above 99.6% of the variance of the prior feature set.

4.3.4 Processing

Once feature selection procedure is completed, the training feature set of normal network data is divided into clusters to represent the normal model by means of k-means clustering. The number of clusters to represent normal data is decided once the cluster analysis is done based on the silhouette value with its highest value obtained. Once ‘k’ is determined, the Euclidean Distance is used to measure the distance from k-centres to each point in training and validation feature set. Thus, distance values from each k-centres to training and testing data are sent to OC-SVM. In OC-SVM, we have used trial and error method using all the available kernel types and varying the degree and gamma values in the kernel function. We obtained the best results using the kernel type of radial basis function with other parameters such as gamma value set to its default value of 1/number of features and ‘nu’ parameter set to 0.1. The results of these two techniques, clustering and one-class SVM are combined and updated to create a final Model. We update our model with the true negatives i.e. ‘normal sessions’ detected in our 1st model. Thus, updating our model helps to allow to detect the anomaly as we have extracted out the probable intersected zone of attack and normal sessions. Once we get our final model using the effective points from the selected validation feature set, we test the testing feature set to predict the final output of our detection system.
CHAPTER 4. EXPERIMENT I (USING DARPA DATASET)

4.4 Results and Analysis

In this chapter, we present the results obtained by using different approaches during our research work. Our approach for detecting attacks sessions based on outlier detection consists of employing PCA for dimensionality reduction of features, clustering and using one-class classifier. We have tuned the various available parameters of our model so that they can work robustly with the training and validation set and hence, generalized it to detect attacks as outliers in the test set. In the process, we have tried with a range of various model parameters that contributed towards achieving the maximum silhouette value, maximum generalization of normal sessions in validation set along with system’s recall and accuracy.

Our first objective is to model the normal sessions, for which we begin with performing PCA on the features of training, validation and testing set. Hence, we have reduced the dimensionality of our features. Here, we have chosen to proceed with first 8 principal components out of total features used, as it covered more than 99.7% of variance of data with these first 8 components which can be seen in table 4.7. We can partially visualize the training, validation and testing datasets using the first two principal components as depicted in respective figure 4.4, figure 4.5 and

<table>
<thead>
<tr>
<th>Components</th>
<th>Training Set</th>
<th>Validation Set</th>
<th>Testing Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Principal Component 1</td>
<td>0.4894</td>
<td>0.5887</td>
<td>0.5565</td>
</tr>
<tr>
<td>Principal Component 2</td>
<td>0.8171</td>
<td>0.8848</td>
<td>0.7456</td>
</tr>
<tr>
<td>Principal Component 3</td>
<td>0.9117</td>
<td>0.9269</td>
<td>0.8537</td>
</tr>
<tr>
<td>Principal Component 4</td>
<td>0.967</td>
<td>0.9548</td>
<td>0.9509</td>
</tr>
<tr>
<td>Principal Component 5</td>
<td>0.9834</td>
<td>0.9702</td>
<td>0.9759</td>
</tr>
<tr>
<td>Principal Component 6</td>
<td>0.9899</td>
<td>0.9818</td>
<td>0.987</td>
</tr>
<tr>
<td>Principal Component 7</td>
<td>0.9959</td>
<td>0.9889</td>
<td>0.9959</td>
</tr>
<tr>
<td>Principal Component 8</td>
<td>0.9986</td>
<td>0.9949</td>
<td>0.9977</td>
</tr>
<tr>
<td>Principal Component 9</td>
<td>0.9994</td>
<td>0.9979</td>
<td>0.9992</td>
</tr>
<tr>
<td>Principal Component 10</td>
<td>0.9997</td>
<td>0.9992</td>
<td>0.9997</td>
</tr>
</tbody>
</table>

Table 4.7: Cumulative values for the distribution of data in first 10 Principal Components
Figure 4.4: Normal session points in Training Dataset

The first two components almost cover 76% variance in all the three datasets. Here, we can observe from the first two principal components values that, all three datasets almost follow the same pattern. Now, in order to model the normal sessions in the training set we have used clustering method.

4.4.1 Clustering Analysis On Training Set

We performed k-means clustering by using different ‘k’ starting from 2. The ‘k’ is chosen based on the highest mean of silhouette value obtained, which is 0.8055 when k=3. We also noticed that as we increase the value for ‘k’, the silhouette value started to decrease, which is depicted in figure 4.8. The silhouette value provides a reference to, how well each object lies within its cluster. The average silhouette value close to 1 means it has the proper clustering of similar objects and close to 0 means that the clustering consists of dissimilar objects. The following
Figure 4.5: All session points in Validation Dataset

Figure 4.6: All session points in Testing Dataset
Figure 4.7: k-Means clustering of normal sessions when k=3 and k=4

Figure 4.8: Silhouette Value distribution for different clusters

figure 4.7 shows the clustering of the training set using values for ‘k’ 3 and 4.

The reasoning for picking 3 clusters is supported by the fact from the distribution of the traffic-type in those three clusters which is depicted in Figure 4.9. It shows majority of the traffic content in the TCP protocol lying in separate clusters while http, tcp, ssh and ftp do tend to share two of those three clusters.

4.4.2 Modeling of Normal Sessions Using Training and Validation Set

Figure 4.10 shows the outline view of the results of modeling normal instances. Normal and attack instances to the certain extend do show similar properties and
Figure 4.9: Clustering Analysis by Traffic-Type in 3 clusters

Figure 4.10: Modeling true negatives/normal instances in validation dataset
Figure 4.11: Modeling true negatives detected in validation dataset

characteristics. Therefore, we need to capture only the properties that truly defines the normal instances. For that, our optimal task is to get as many true negatives as possible i.e. to push the walls of the true negatives in figure 4.10 so that we can precisely define our normal instances, which helps to reduce false negatives and false positives, as well. Once, we have performed clustering of the normal sessions using training dataset. We then compute the distance of the validation dataset from each center obtained and then, send its corresponding distances to the OC-SVM method. Here, our objective is to extract the normal sessions from attacked sessions. Hence, we proceeded with the parameters after tuning that allowed us to achieve maximum true negatives i.e. normal sessions and maximum true positives i.e. attacked sessions. We then modelled using the features of these true negatives instances of the validation dataset that provides the final model. We were able to model best of 83% of the total normal sessions in validation dataset.

Figure 4.11 shows the true negatives detected by our first model in the validation dataset. And figure 4.11 shows the data distribution of validation dataset containing attacked-sessions.
4.4.3 Results

The first two principal components of the testing dataset is depicted in figure 4.12.

To evaluate our system, we focused in two major indicators of performance, which are detection rate and false positive rate. The evaluation of our system is done based on confusion matrix using measures ‘Recall’ and ‘Accuracy’. Using our approach, we have been able to achieve best results with Recall of 1, which means 100% detection rate and accuracy of 91.85% when we have used Radial Basis Kernel as shown in figure 4.8, with ‘nu’ parameter set to 0.1, degree set to 3 and gamma set to 0.125. About 8.17% of false positive rate detected can be suspected because of the underlying similarities between the normal and attack sessions.

The various parameters are required to be tuned while performing this experiment for getting better results. We have an option of using different kernel types in SVM which can be seen in [Appendix A.3]. The results based on the types of kernel used, can be seen in the figures 4.9, 4.10 and 4.11.

The ROC curve obtained for Table 4.8 is shown in figure 4.13, which has Area Under Curve (AUC) value of 0.95. This means that with 0.95 probability, our ap-
CHAPTER 4. EXPERIMENT I (USING DARPA DATASET)

<table>
<thead>
<tr>
<th>Using Radial Basis Kernel</th>
<th>Actual Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confusion Matrix</td>
<td></td>
</tr>
<tr>
<td>Predicted Class</td>
<td></td>
</tr>
<tr>
<td>Attack</td>
<td>2304</td>
</tr>
<tr>
<td>Normal</td>
<td>52293</td>
</tr>
<tr>
<td>Class</td>
<td>0</td>
</tr>
<tr>
<td>Normal</td>
<td>587033</td>
</tr>
</tbody>
</table>

Measures | Values |
---------|--------|
Accuracy | 0.9185 |
Recall   | 1      |

Table 4.8: Confusion matrix when using RBF Kernel

<table>
<thead>
<tr>
<th>Using Linear Kernel</th>
<th>Actual Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>n=0.5</td>
<td></td>
</tr>
<tr>
<td>Confusion Matrix</td>
<td>Attack</td>
</tr>
<tr>
<td>Predicted Class</td>
<td></td>
</tr>
<tr>
<td>Attack</td>
<td>2201</td>
</tr>
<tr>
<td>Normal</td>
<td>103</td>
</tr>
<tr>
<td>Normal</td>
<td>502004</td>
</tr>
</tbody>
</table>

Measures | Values |
---------|--------|
Accuracy | 0.8061 |
Recall   | 0.9401 |

Table 4.9: Confusion matrix when using Linear Kernel

<table>
<thead>
<tr>
<th>Using Polynomial Kernel</th>
<th>Actual Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>degree=3</td>
<td></td>
</tr>
<tr>
<td>Confusion Matrix</td>
<td>Attack</td>
</tr>
<tr>
<td>Predicted Class</td>
<td></td>
</tr>
<tr>
<td>Attack</td>
<td>2166</td>
</tr>
<tr>
<td>Normal</td>
<td>138</td>
</tr>
<tr>
<td>Normal</td>
<td>515047</td>
</tr>
</tbody>
</table>

Measures | Values |
---------|--------|
Accuracy | 0.8061 |
Recall   | 0.9401 |

Table 4.10: Confusion matrix when using Polynomial Kernel

<table>
<thead>
<tr>
<th>Using Sigmoid Kernel</th>
<th>Actual Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confusion Matrix</td>
<td>Attack</td>
</tr>
<tr>
<td>Predicted Class</td>
<td></td>
</tr>
<tr>
<td>Attack</td>
<td>2285</td>
</tr>
<tr>
<td>Normal</td>
<td>19</td>
</tr>
<tr>
<td>Normal</td>
<td>532681</td>
</tr>
</tbody>
</table>

Table 4.11: Confusion matrix when using Sigmoid Kernel
Figure 4.13: ROC Curve in Darpa Dataset using RBF kernel when k=3

This approach can rank a randomly selected attack session higher than a randomly selected normal session.
Experiment II (Using KDD dataset)

The works of Faisal et al. [51, 52] in KDD dataset has given new sight towards IDS. However, Experiment I gave us the basis to carry out our experiment in KDD dataset, which allows to evaluate our model; which is mostly used dataset for IDS in recent times.

5.1 KDD dataset

KDD 1999 dataset, is a modified version of Darpa MIT dataset. It provides the designer of IDS with a standard benchmark which they can use to evaluate their methodologies. After performing experiments on Darpa dataset, we thought about using our model to compare our results with this KDD dataset. It provides a strong basis for the evaluation of our approach towards making an efficient IDS as KDD is more widely used dataset for IDS benchmarking in recent times. In KDD dataset, 41 features [Appendix A.3] were defined for each connection samples, which are
further divided into four categories namely primary and content features of TCP protocol, time-based and host-based traffic features. It also consists of four groups of attacks which includes Denial of Service (DoS) attack, Probing attack, Remote to Local (R2L) attack and User to Root (U2R) attack. And different types of attacking records are presented within each category such as back, Neptune, pod, land, smurf and teardrop attacks in case of DoS. It was observed that ‘back’ attack has received most attention as KDD ‘99 dataset has 2,203 records of ‘back’ attacks and each these records have 41 features displaying different attack level.

5.2 Evaluation Procedures and Metrics

The feature selection can be done based on Information Gain and Gain Ratio, which signifies how much a feature contributes towards the resulting class for classification.

5.2.1 Information Gain

Let us consider \( S \) be a set of training set samples along with their corresponding labels. Considering there are ‘m’ classes and the training set consists of \( s_i \) samples of class I where ‘s’, represents the total number of samples in the training set. Then, expected information required to classify a given sample is given by Equation 5.1:

\[
I(s_1, s_2, ..., s_m) = - \sum_{i=1}^{m} \frac{s_i}{s} \log_2 \left( \frac{s_i}{s} \right)
\]  

(5.1)

Considering a feature \( F \) with values \( \{f_1, f_2, ..., f_v\} \) which can divide the training set into v subsets \( \{S_1, S_2, ..., S_v\} \) where \( S_j \) is the subset that has the value \( F \) for feature \( F \). And if \( S_j \) contains \( s_{ij} \) samples of class i. Then, we can represent the
entropy of the feature F as shown in the equation Equation 5.2

\[ E(F) = \sum_{j=1}^{v} \frac{S_{1j} + \ldots + S_{mj}}{S} * I(s_1, s_2, \ldots, s_m) \] (5.2)

Therefore, we can mathematically represent Information Gain as shown in Equation 5.3.

\[ Information \ Gain(F) = I(s_1, s_2, \ldots, s_m) - E(F) \] (5.3)

5.2.2 Gain Ratio

For calculating Gain Ratio, we require to calculate Split Information which is then used to reduce the bias of information gain. In equation 5.4, C is a candidate, n is the number of partition splits by C. And D denotes a set of instances whereas \( D_i \) denotes the number of instances in \( i^{th} \) partition.

\[ Split \ Information_C(D) = - \sum_{i=1}^{n} \frac{D_i}{D} \log \frac{D_i}{D} \] (5.4)

Thus, Gain Ratio can be calculated as shown in Equation 5.5

\[ Gain \ Ratio(C) = \frac{Information \ Gain(C)}{Split \ Information_C(D)} \] (5.5)

5.2.3 Performance Metrics

The metrics used for performance evaluation is similar to the one used in Experiment I. However, additionally, we have used F-measure as well for evaluation in KDD dataset.
F-Measure:

F-Measure is also known as F₁ score which is used to evaluate the performance of the model in the scenarios of different class distribution. It considers both precision and the recall of the system to compute the score. It can also be interpreted as a harmonic mean of the precision and recall. When the class distribution is balanced, F₁ score is used while for imbalanced class distribution Fβ score is used, where β stands for weight of recall over precision.

\[
F_1 \text{-score} = 2 \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]

\[
F_\beta \text{-score} = (1 + \beta^2) \frac{\text{precision} \times \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}
\]

5.3 Experimental Process Flow

The experimental process flow of our experiment conducted is same as earlier section 4.3 and can be seen in figure 4.3.

5.3.1 Pre-Processing

The KDD dataset consist of numeric and non-numeric data. First, non-numeric data were mapped to numeric data based on their ranking for each attribute. Among numeric data, continuous values were also converted to discontinuous values for the purpose of calculating Information Gain and Gain Ratio.

5.3.2 Sampling and Feature Selection

KDD dataset has been known for consisting of redundant data so, we have used a ‘distinct sql query statement’ [Appendix A.4] to get the unique sample of each data. In KDD dataset, we have a full dataset of size 708 MB and test dataset of size
45 MB. Here, we have used a full dataset for two purposes, one as a training set and other as a validation set. Training set consists of only normal instances where as complete full dataset is used as a validation set. We have used Weka to calculate Information Gain [Table 5.1] and Gain Ratio [Table 5.2], by means of which we have been able to figure out 28 features out of 41, that contributed mostly towards discriminating the given class from the others. The [Appendix A.4] also provides the names of the 28 features used for processing the KDD dataset.

The data distribution of normal and attack data is shown in table 5.3, which also gives the overview of the redundancy of the data used in KDD dataset.

5.3.3 Processing

The processing steps are described in earlier section 4.3.4. After selecting the features from KDD dataset, we fed these three datasets to our model where processing takes place.

5.4 Results and Analysis

5.4.1 Clustering Analysis on Training Set

We performed k-means clustering by using different ‘k’ starting from 2. The ‘k’ is chosen based on the highest mean of silhouette value obtained, which is 0.64 when k=5. As with Experiment I, we observed that as we increase the value for ‘k’, the silhouette value started to increase and then decrease, which is depicted in figure 5.2. The following figures 5.1 shows the clustering of the training set using different values for ‘k’ from 3 to 5.

Here, we have chosen to proceed with first 8 principal components out of total features used, as it covered more than 99.00% of variance of data with these first 8 components which can be seen in table 5.3. We then start it off by performing
### Table 5.1: Information Gain calculated in KDD dataset

<table>
<thead>
<tr>
<th>attribute</th>
<th>no.</th>
<th>rank</th>
<th>merit</th>
<th>rank</th>
<th>infoGain</th>
</tr>
</thead>
<tbody>
<tr>
<td>src_bytes</td>
<td>5</td>
<td>1</td>
<td>0.016</td>
<td>1</td>
<td>0.001</td>
</tr>
<tr>
<td>service</td>
<td>3</td>
<td>2</td>
<td>0.071</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>dst_bytes</td>
<td>6</td>
<td>3</td>
<td>0.032</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>flag</td>
<td>4</td>
<td>4</td>
<td>0.519</td>
<td>4.1</td>
<td>0.001</td>
</tr>
<tr>
<td>diff_srv_rate</td>
<td>30</td>
<td>5</td>
<td>0.519</td>
<td>4.9</td>
<td>0.001</td>
</tr>
<tr>
<td>same_srv_rate</td>
<td>29</td>
<td>6</td>
<td>0.51</td>
<td>6</td>
<td>0.001</td>
</tr>
<tr>
<td>dst_host_srv_count</td>
<td>33</td>
<td>7</td>
<td>0.476</td>
<td>7</td>
<td>0.001</td>
</tr>
<tr>
<td>dst_host_same_srv_rate</td>
<td>34</td>
<td>8</td>
<td>0.438</td>
<td>8</td>
<td>0.001</td>
</tr>
<tr>
<td>dst_host_diff_srv_rate</td>
<td>35</td>
<td>9</td>
<td>0.411</td>
<td>9</td>
<td>0.001</td>
</tr>
<tr>
<td>dst_host_error_rate</td>
<td>36</td>
<td>10</td>
<td>0.406</td>
<td>10.1</td>
<td>0.001</td>
</tr>
<tr>
<td>logged_in</td>
<td>12</td>
<td>12</td>
<td>0.05</td>
<td>10.9</td>
<td>0.001</td>
</tr>
<tr>
<td>dst_host_srv_error_rate</td>
<td>37</td>
<td>13</td>
<td>0.39</td>
<td>18</td>
<td>0.001</td>
</tr>
<tr>
<td>error_rate</td>
<td>25</td>
<td>14</td>
<td>0.39</td>
<td>13</td>
<td>0.001</td>
</tr>
<tr>
<td>count</td>
<td>21</td>
<td>15</td>
<td>0.384</td>
<td>14</td>
<td>0.001</td>
</tr>
<tr>
<td>srv_error_rate</td>
<td>28</td>
<td>16</td>
<td>0.37</td>
<td>15</td>
<td>0.001</td>
</tr>
<tr>
<td>dst_host_srv_diff_host_rate</td>
<td>37</td>
<td>17</td>
<td>0.198</td>
<td>17</td>
<td>0.001</td>
</tr>
<tr>
<td>dst_host_count</td>
<td>82</td>
<td>18</td>
<td>0.189</td>
<td>18</td>
<td>0.001</td>
</tr>
<tr>
<td>dst_host_same_src_port_rate</td>
<td>36</td>
<td>19</td>
<td>0.142</td>
<td>19</td>
<td>0.001</td>
</tr>
<tr>
<td>srv_diff_host_rate</td>
<td>31</td>
<td>20</td>
<td>0.26</td>
<td>19</td>
<td>0.001</td>
</tr>
<tr>
<td>srv_count</td>
<td>26</td>
<td>21</td>
<td>0.194</td>
<td>20</td>
<td>0.001</td>
</tr>
<tr>
<td>dst_host_srv_error_rate</td>
<td>41</td>
<td>22</td>
<td>0.088</td>
<td>21</td>
<td>0.001</td>
</tr>
<tr>
<td>protocol_type</td>
<td>2</td>
<td>23</td>
<td>0.051</td>
<td>22</td>
<td>0.001</td>
</tr>
<tr>
<td>error_rate</td>
<td>27</td>
<td>24</td>
<td>0.052</td>
<td>23</td>
<td>0.001</td>
</tr>
<tr>
<td>dst_host_error_rate</td>
<td>28</td>
<td>25</td>
<td>0.052</td>
<td>24</td>
<td>0.001</td>
</tr>
<tr>
<td>srv_errors_rate</td>
<td>1</td>
<td>26</td>
<td>0.036</td>
<td>25</td>
<td>0.001</td>
</tr>
<tr>
<td>duration</td>
<td>10</td>
<td>27</td>
<td>0.012</td>
<td>26</td>
<td>0.001</td>
</tr>
<tr>
<td>host</td>
<td>16</td>
<td>28</td>
<td>0.01</td>
<td>27</td>
<td>0.001</td>
</tr>
<tr>
<td>wrong_fragment</td>
<td>8</td>
<td>29</td>
<td>0.007</td>
<td>28</td>
<td>0.001</td>
</tr>
<tr>
<td>num_compromised</td>
<td>15</td>
<td>30</td>
<td>0.004</td>
<td>29</td>
<td>0.001</td>
</tr>
<tr>
<td>num_root</td>
<td>2</td>
<td>31</td>
<td>0.002</td>
<td>30</td>
<td>0.001</td>
</tr>
<tr>
<td>num_access_files</td>
<td>19</td>
<td>32</td>
<td>0.001</td>
<td>31</td>
<td>0.001</td>
</tr>
<tr>
<td>num_file_creations</td>
<td>17</td>
<td>33</td>
<td>0.001</td>
<td>32</td>
<td>0.001</td>
</tr>
<tr>
<td>num_su_attempted</td>
<td>15</td>
<td>34</td>
<td>0.001</td>
<td>33</td>
<td>0.001</td>
</tr>
<tr>
<td>root_shell</td>
<td>14</td>
<td>35</td>
<td>0.00</td>
<td>34</td>
<td>0.001</td>
</tr>
<tr>
<td>num_shells</td>
<td>18</td>
<td>36</td>
<td>0.001</td>
<td>35</td>
<td>0.001</td>
</tr>
<tr>
<td>land</td>
<td>7</td>
<td>37.6</td>
<td>0.07</td>
<td>36</td>
<td>0.049</td>
</tr>
<tr>
<td>num_failed_logins</td>
<td>11</td>
<td>38.1</td>
<td>0.038</td>
<td>37</td>
<td>0.87</td>
</tr>
<tr>
<td>host_login</td>
<td>25</td>
<td>38.8</td>
<td>0.089</td>
<td>38</td>
<td>0.87</td>
</tr>
<tr>
<td>num_outbound_cmds</td>
<td>20</td>
<td>39.6</td>
<td>0.049</td>
<td>39</td>
<td>0.49</td>
</tr>
<tr>
<td>urgent</td>
<td>9</td>
<td>40.8</td>
<td>0.00</td>
<td>40</td>
<td>0.06</td>
</tr>
</tbody>
</table>
### Table 5.2: Gain Ratio calculated in KDD dataset

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Gain Ratio</th>
<th>Average Rank</th>
<th>No. Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>logged_in</td>
<td>0.418</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>srv_error_rate</td>
<td>0.375</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>flag</td>
<td>0.34</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>error_rate</td>
<td>0.394</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>dst_host_src_error_rate</td>
<td>0.331</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>dst_bytes</td>
<td>0.277</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>diff_srv_rate</td>
<td>0.268</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>dst_host_srv_error_rate</td>
<td>0.267</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>src_bytes</td>
<td>0.258</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>same_srv_rate</td>
<td>0.234</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>service</td>
<td>0.179</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>dst_host_srv_diff_host_rate</td>
<td>0.154</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>dst_host_count</td>
<td>0.136</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>dst_host_same_srv_rate</td>
<td>0.134</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>dst_host_same_srv_port_rate</td>
<td>0.127</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>dst_host_diff_srv_rate</td>
<td>0.114</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>srv_diff_host_rate</td>
<td>0.097</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>dst_host_srv_count</td>
<td>0.097</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>dst_host_count</td>
<td>0.093</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>srv_error_rate</td>
<td>0.082</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>num_root</td>
<td>0.079</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>num_file_creations</td>
<td>0.077</td>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td>num_access_files</td>
<td>0.074</td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td>protocol_type</td>
<td>0.072</td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.072</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>protocol_type</td>
<td>0.067</td>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.062</td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.056</td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.054</td>
<td>29</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.049</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.044</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.039</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.035</td>
<td>33</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.029</td>
<td>34</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.022</td>
<td>35</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.022</td>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.022</td>
<td>37</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.022</td>
<td>38</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.022</td>
<td>39</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.022</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.022</td>
<td>41</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.022</td>
<td>42</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.022</td>
<td>43</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.022</td>
<td>44</td>
<td>0</td>
</tr>
<tr>
<td>normal</td>
<td>0.022</td>
<td>45</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table 5.3: Data distribution in KDD dataset

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Types</th>
<th>Count</th>
<th>Distinct Count</th>
<th>Percentage of Unique Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Set</td>
<td>Normal</td>
<td>972780</td>
<td>812735</td>
<td>83.55</td>
</tr>
<tr>
<td></td>
<td>Attacks</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>Validation Set</td>
<td>Normal</td>
<td>972780</td>
<td>812735</td>
<td>83.55</td>
</tr>
<tr>
<td></td>
<td>Attacks</td>
<td>3925051</td>
<td>262179</td>
<td>6.68</td>
</tr>
<tr>
<td>Testing Set</td>
<td>Normal</td>
<td>60593</td>
<td>57876</td>
<td>95.52</td>
</tr>
<tr>
<td></td>
<td>Attacks</td>
<td>250436</td>
<td>29349</td>
<td>11.72</td>
</tr>
</tbody>
</table>
CHAPTER 5. EXPERIMENT II (USING KDD DATASET)

Figure 5.1: k-Means clustering of normal sessions when $k=3$ to $k=5$

Figure 5.2: Silhouette Value distribution for different clusters
The data distribution of $1^{st}$ and $2^{nd}$ principal component of training set and testing set can be seen in figure 5.4 and figure 5.6 respectively.
5.4.2 Modeling of Normal Sessions Using Training and Validation Set

We achieved our best results where, we were able to model 90% (731,466 out of 812,735) of the total normal data of validation set as true negatives. We have used training set which is extracted from validation set, containing only normal instances. The modeling of normal instances should have been 100% of the training dataset but, our model only uses these 90% because OC-SVM tries to avoid over-fitting and allows slack variables to come into play. After modeling the true negatives as shown in [Figure 5.5], we then feed the test set to calculate the output from our model.

5.4.3 Results

The result obtained while processing the KDD dataset in our model can be seen in the following tables 5.4 and 5.5. Looking from the overall perspective of accuracy, f1-score, precision and recall; we achieved our best result using 5 clusters along with other parameters such as ‘nu’ set to 0.1, ‘g’ set to 0.035 (i.e. 1/28 where, 28 is no. of features used) and with Radial Basis kernel. Thus, the best achieved result is shown in the table 5.6.
Figure 5.6: Data distribution in testing dataset

<table>
<thead>
<tr>
<th>Predicted Class</th>
<th>Confusion Matrix</th>
<th>Actual Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack</td>
<td>29230 15589</td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>119 31887</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.4: Result obtained using RBF Kernel when k=3
CHAPTER 5. EXPERIMENT II (USING KDD DATASET)

Table 5.5: Result obtained using RBF Kernel when k=4

<table>
<thead>
<tr>
<th>Predicted Class</th>
<th>Actual Class</th>
<th>Confusion Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack</td>
<td>29243</td>
<td>15969</td>
</tr>
<tr>
<td>Normal</td>
<td>106</td>
<td>31907</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measures</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.7918</td>
</tr>
<tr>
<td>F1 Score</td>
<td>0.7844</td>
</tr>
<tr>
<td>Precision</td>
<td>0.6468</td>
</tr>
<tr>
<td>Recall</td>
<td>0.9964</td>
</tr>
</tbody>
</table>

Table 5.6: Result obtained using RBF Kernel when k=5

<table>
<thead>
<tr>
<th>Predicted Class</th>
<th>Actual Class</th>
<th>Confusion Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack</td>
<td>28954</td>
<td>7705</td>
</tr>
<tr>
<td>Normal</td>
<td>395</td>
<td>40113</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measures</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.8943</td>
</tr>
<tr>
<td>F1 Score</td>
<td>0.8765</td>
</tr>
<tr>
<td>Precision</td>
<td>0.7885</td>
</tr>
<tr>
<td>Recall</td>
<td>0.9865</td>
</tr>
</tbody>
</table>
CHAPTER 5. EXPERIMENT II (USING KDD DATASET)

The figure 5.7 provides precision-recall graph. The best precision and recall obtained by our model is 78.85% and 0.98 respectively. But, it can be inferred from the graph that we can improve the ‘precision’ to 98% if we compromise the ‘recall’ by lowering it down to 0.8.

And if we lower the the ‘recall’ to 0.9 the ‘precision’ stays around 95%. The ROC curve obtained for RBF Kernel when k=5 is shown in figure 5.8, which has AUC value of 0.9122. This means that with 0.9122 probability, our approach can rank a randomly chosen attack higher than a randomly chosen normal one.

The attacks that were undetected by our approach using different cluster value is provided in table 5.7. We can see that attacks such as ‘apache2’ and ‘mscan’ were detected when we used cluster 3 and 4, but it resulted in bit higher false positive rate. The attacks such as ‘buffer_overflow’, ‘httptunnel’, ‘mailbomb’, ‘multihop’, ‘ps’ and ‘rootkit’ were undetected while using 3 and 4 clusters, but they were detected when trying to lower the false positive rate while analyzing with 5 clusters.
Figure 5.8: ROC Curve in KDD Dataset using RBF kernel when k=5

But it gave much better false positive rate which got reduced by almost half of what was detected with 3 and 4 clusters.
<table>
<thead>
<tr>
<th>No. Of Attacks</th>
<th>Class ID</th>
<th>Class Name</th>
<th>No. of attacks undetected when k=5</th>
<th>No. of attacks undetected when k=4</th>
<th>No. of attacks undetected when k=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>apache2.</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>buffer_overflow</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>guess_passwd.</td>
<td>233</td>
<td>57</td>
<td>63</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>http_tunnel.</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>mailbomb.</td>
<td>61</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>msdos</td>
<td>0</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td>multihop.</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>25</td>
<td>ps.</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>26</td>
<td>rootkit.</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>31</td>
<td>snmpgetattack.</td>
<td>89</td>
<td>42</td>
<td>47</td>
</tr>
<tr>
<td>11</td>
<td>38</td>
<td>warezmaster.</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Total no. of attacks undetected</strong></td>
<td><strong>395</strong></td>
<td><strong>106</strong></td>
<td><strong>119</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Total no. of distinct attacks undetected</strong></td>
<td><strong>9</strong></td>
<td><strong>5</strong></td>
<td><strong>5</strong></td>
</tr>
</tbody>
</table>

Table 5.7: Attacks undetected by attack class name
Discussion and Future Works

In this research, we have used various approaches before reaching to the final approach, which are presented briefly in this chapter. Along with this, various insights, difficulties together with comparison and future work are also discussed.

6.1 Insights and Technical Difficulties

The motivation to go for intrusion detection at network level started off with exploring various tools such as Nessi2 [63] and Sim-Grid [64]. However, due to various technical problems and software issues, our attempt to create a simulated attacked network was hindered by the lack of technical support from their development teams. Apart from software issues, we also had to deal with limited resource problems. We started off trying with network flow detection in TCP packets but because of huge data, network flow was not able to be traced properly mainly because of constrained resources, which made us divert towards analyzing the TCP
headers for processing the network data. And then towards using KDD dataset for the evaluation of our model. In our approach, we have used k-means algorithm, which is affected by first initialized centre value, therefore, prior to deciding with ‘k’, we made sure we got the reliable value of ‘k’, by executing the clustering process rigorously multiple times for each value of ‘k’.

The other problems we had during the experiment is the corrupted 4 tcpdump files after downloading these data files. However, we were able to recover the partial information from those files and use it to the certain extend.

6.2 Other Approaches Used

Apart from doing a clustering analysis by using k-means method with Euclidean distance, we have also tried using with ‘Mahalanobis’ and ‘correlation’ distance. But the other two did not give the satisfactory results. Additionally, we also used K-Medoid clustering, which also did not help our cause. We have also played with various tuning parameters for getting the best results in OC-SVM, which includes using different kernel types as well as ‘nu’ parameters. We even played with the sampling techniques, from taking the full set to the distinct set of datas for modeling purpose, that allowed us to achieve better and consistent results. We have started our approach by trying with only single stage consisting of either clustering or OC-SVM and then, we came up with this updated model with two stages.

6.3 Comparisons with Other Similar Works

In the work presented by Taylor et al. [18], they have detected the selected ‘dos’ and ‘probe’ attacks but failed to detect ‘Mailbomb’. That means their recall is less than 100%. In our case, we are able to detect all attacks presented in the 4th week of DARPA ’99 dataset with a perfect recall of 100%. Our model overcomes the
Table 6.1: Comparison of our results in KDD dataset with others’ work

<table>
<thead>
<tr>
<th>Methods</th>
<th>Recall</th>
<th>False Positive Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clustering</td>
<td>93%</td>
<td>10%</td>
</tr>
<tr>
<td>K-NN</td>
<td>92%</td>
<td>8%</td>
</tr>
<tr>
<td>SVM</td>
<td>98%</td>
<td>10%</td>
</tr>
<tr>
<td>SOM Hierarchy</td>
<td>89%</td>
<td>7.60%</td>
</tr>
<tr>
<td>Our Approach GC-SVM*</td>
<td>95%</td>
<td>8%</td>
</tr>
</tbody>
</table>

problem of NATE, which requires absolutely normal packets for normal session modeling. Since we have used the validation set to cross-check our normal session modeling, our final model reflects both the normal session and the attack session (by means of the negative space). In terms of the false positive result (ours is around 8%), we are not able to make a direct comparison against NATE because this is not explicitly reported there, and is pretty much dependent on their tunable parameter called sensitivity level.

Also in the work of Carrascal et al. [48], their best results in analysis of TCP/IP traffic, in 1998 Darpa dataset was 73% detection rate with 2% false positive rate with an approach using SOM with LVQ technique along with SVM. Our approach seems to be comparable using one class modeling while they have used both classes for classification.

Our work and results of processing KDD datasets also showed promising signs as results were pretty much comparable if not better than the previous results [48][49], which is depicted in table 6.1.

6.4 Future Work

For future work, apart from using basic TCP header information; we look to use derived information from the connection analysis and try other approaches that might help us to lower down the false positive rate detected. The current work only performs the analysis in one way traffic from the source to destination. How-
ever, it would be interesting to observe a network flow in both ways (i.e. source to destination and vice-versa) and process the results which offer the prospect of dealing at packets level rather than at session level, that we believe would prove to be more real time application. In this research we have focused only on TCP protocols whereas in the future, we also look to explore other protocols apart from TCP, for making a complete intrusion detection system. The program for deployment of our work is available at http://www.aungz.com/IDS/ and any additional future work regarding this would be updated in this link accordingly.
Conclusion

Intrusion detection has improved with age, but this improvement seems to be a continuous process as advancement in the technology opens the door with a loop-hole for intruders every time. During the literature review, it was observed that recently, many researchers were and are still performing their experiment to increase the effectiveness of intrusion detection in standard datasets like KDD ‘99 and NSL-KDD rather than the original MIT Darpa dataset. These datasets are all easily available for conducting the research, and known information about these datasets, as well as the published results helps the researchers to perform experiments and compare their results with the previous works which help to evaluate their effectiveness.

When the amount of data in the network started to grow, this led to be a significant challenge in intrusion detection. Therefore, there was a need of dealing with these huge datasets. First attempt was to clean the redundant data in these datasets and then to select and extract the subset of datasets for gaining efficiency. The feature selection for any intrusion detection is the key to decide its effectiveness hence,
relevant features need to be extracted and selected by visualizing their patterns and reducing their dimensionality. Machine Learning approach was introduced in implementation of IDS to decrease the level of human interaction and increase the effective of IDS. Many IDS still lacks the ability to detect all kinds of new attacks in the network, so researchers are inclined towards modeling the normal instances to increase their system effectiveness.

Anomaly detection based on outlier has always been a challenging task for real-time detection. In our research work, we have been able to model the normal data and detect the attacked data as outlier from both ‘99 Darpa and KDD datasets. The main objective of processing the Darpa dataset is to built the real-time IDS. In summarizing the whole research work, we have mainly focused on feature reduction, clustering analysis and modeling the normal instances in presence of attack information. Our approach overcomes the drawbacks of one associated with the rule based approaches and is efficient. We have discussed about the effectiveness of our work on basis on clustering analysis, performance metrics, ROC curve and precision-recall test. The output of the processed results in KDD dataset was also a promising sign for our model to be effective in any other situations. Thus, our work provides a practical solution for construction of better IDS based on ‘Outlier Detection method’. Thus, with our results and analysis, we can say that, we have analyzed the TCP header information that allows to deal with anomaly detection in fast incoming traffic in real time with lower false positive detection rate.


systems by using the combination of machine learning approaches,” in 2010 International Conference of Soft Computing and Pattern Recognition (SoCPaR), 2010, pp. 334-337.


[63] website 2014., http://www.nessi2.de/

[64] website 2014., http://simgrid.gforge.inria.fr/


Supporting Materials

The supporting materials for performing this research are presented in this section.
### Table A.1: Features extracted from TCP headers used in our experiment

<table>
<thead>
<tr>
<th>No.</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Packet Count * (Non-average)</td>
</tr>
<tr>
<td>2</td>
<td>Packet Length</td>
</tr>
<tr>
<td>3</td>
<td>FSR Flags</td>
</tr>
<tr>
<td>4</td>
<td>DeFragment Flag</td>
</tr>
<tr>
<td>5</td>
<td>More Fragment Flag</td>
</tr>
<tr>
<td>6</td>
<td>Reserved Flag</td>
</tr>
<tr>
<td>7</td>
<td>TTL</td>
</tr>
<tr>
<td>8</td>
<td>TOS</td>
</tr>
<tr>
<td>9</td>
<td>CheckSum</td>
</tr>
<tr>
<td>10</td>
<td>PSH Flag</td>
</tr>
<tr>
<td>11</td>
<td>ECE Flag</td>
</tr>
<tr>
<td>12</td>
<td>CWR Flag</td>
</tr>
<tr>
<td>13</td>
<td>URG Flag</td>
</tr>
<tr>
<td>14</td>
<td>ACK Flag</td>
</tr>
<tr>
<td>15</td>
<td>RST Flag</td>
</tr>
<tr>
<td>16</td>
<td>SYN Flag</td>
</tr>
<tr>
<td>17</td>
<td>FIN Flag</td>
</tr>
<tr>
<td>18</td>
<td>Duration</td>
</tr>
</tbody>
</table>

Table A.2: List of attacks in different categories used in the MIT Darpa data set

<table>
<thead>
<tr>
<th>No.</th>
<th>Attack Categories</th>
<th>List of Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DOS</td>
<td>Neptune, back, smurf, pod, land, teardrop</td>
</tr>
<tr>
<td>2</td>
<td>U2R</td>
<td>buffer_overflow, load module, rootkit, perl</td>
</tr>
<tr>
<td>3</td>
<td>R2L</td>
<td>warenclient, multihop, ftp_write, spy, imap, guess_passwd, warezmaster, phf</td>
</tr>
<tr>
<td>4</td>
<td>Probe</td>
<td>portsweep, satan, nmap, ipsweep</td>
</tr>
</tbody>
</table>
Figure A.1: Simulated Darpa 1999 Network Diagram
public static void main(String[] args) {
    final String FILENAME = "D://DarnaDataSet//OriginalDump//week4//inside.tcpdump//ln41.tcpdump";
    final String erbuf = new String();
    final Integer packetId = new Integer();
    final Pcap pcap = Pcap.openOffLine(FILENAME, erbuf);
    pcap.loop(Pcap.LOOP_INFINITE, new IPacketHandler(StringBuilder) {
        final Tcp tcp = new Tcp();
        final byte ip = new Byte();
        public void nextPacket(IPacket packet, StringBuilder erbuf) {
            int i = (int) packet.getFrameNumber();
            if (Arrays.asList(packet.id).contains(i)) {
                byte[] dIP;
                byte[] sIP;
                if (packet.header.getIp().destination()) {
                    dIP = packet.getHeader.ip().destination();
                    sIP = packet.getHeader.ip().source();
                    String protocol = ip.typeDown().name();
                    String sourceIP = org.jnetpcap.packet.format.FormatUtils.ip(sIP);
                    String destinationIP = org.jnetpcap.packet.format.FormatUtils.ip(dIP);
                    byte[] data = packet.getByteArray(0, packet.size());
                    System.out.println("PacketID Frame --- #\%d\%n", packet.getFrameNumber());
                    System.out.println("ip.src=%s\n", sourceIP);
                    System.out.println("ip.dst=%s\n", destinationIP);
                    System.out.println("protocol-%s\n", protocol);
                    System.out.println("byte-%s\n", Hex.encodeHexString(data));
                    System.out.println("flag DF-%s, ip.flags_DF()\n", dIP);
                    System.out.println("flag RF-%s, ip.flags_RF()\n", rIP);
                    System.out.println("flag RC%d\n", ip.flags_Reserved());
                } else {
                    System.out.println("src_port=%s", tcp.source());
                    System.out.println("dst_port=%s", tcp.destination());
                    System.out.println("TCPSource%ko\n", Integer.toString(tcp.source()));
                    System.out.println("TCDest%ko\n", Integer.toString(tcp.destination()));
                    System.out.println("TP_\(\%s\)\n", String.valueOf(tcp.flags_UDP()));
                    System.out.println("TP_\(\%s\)\n", String.valueOf(tcp.flags_TCP()));
                    System.out.println("TP_\(\%s\)\n", String.valueOf(tcp.flags_SIM()));
                    System.out.println("TP_\(\%s\)\n", String.valueOf(tcp.flags_FIM()));
                    System.out.println("TP_\(\%s\)\n", String.valueOf(tcp.flags_PSM()));
                }
            }
        }
    };
    pcap.close();
}

Figure A.2: TCP extraction code snippets
Figure A.3: Options available in SVM

```
options:
  -s svm_type : set type of SVM (default 0)
    0 -- C-SVC
    1 -- nu-SVC
    2 -- one-class SVM
    3 -- epsilon-SVR
    4 -- nu-SVR
  -t kernel_type : set type of kernel function (default 2)
    0 -- linear: u'*v
    1 -- polynomial: (gamma*u'*v + coef0)^degree
    2 -- radial basis function: exp(-gamma|u-v|^2)
    3 -- sigmoid: tanh(gamma*u'*v + coef0)
  -d degree : set degree in kernel function (default 3)
  -g gamma : set gamma in kernel function (default 1/num_features)
  -r coef0 : set coef0 in kernel function (default 0)
  -c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
  -n nu : set the parameter nu of nu-SVC, one-class SVM, and epsilon-SVR (default 0.5)
  -e epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)
  -c cache_size : set cache memory size in MB (default 100)
  -v epsilon : set tolerance of termination criterion (default 0.001)
  -h shrinking: whether to use the shrinking heuristics. 0 or 1 (default 1)
  -b probability_estimates: whether to train a SVC or SRM model for probability estimates. 0 or 1 (default 0)
  -w1 weights : set the parameter C of class 1 to weight*C, for C-SVC (default 1)

The k in the -g option means the number of attributes in the input data.
```

Figure A.4: Sampling for unique instance in test dataset

```
SELECT distinct
  'duration',
  'protocol_type',
  'service_rank',
  'flag',
  'src_bytes',
  'dst_bytes',
  'logged_in',
  'num_compromised',
  'num_access_files',
  'count',
  'srv_count',
  'srv_rate',
  'srv_serror_rate',
  'serror_rate',
  'srv_serror_rate',
  'same_srv_rate',
  'diff_srv_rate',
  'srv_diff_host_rate',
  'dst_host_count',
  'dst_host_serror_rate',
  'dst_host_same_srv_rate',
  'dst_host_diff_srv_rate',
  'dst_host_same_src_port_rate',
  'dst_host_srv_diff_host_rate',
  'dst_host_same_srv_rate',
  'dst_host_srv_serror_rate',
  'dst_host_srv_rerror_rate',
  'class',
  'category'
FROM
  `kdd\testset`
```
### Table A.3: Attributes of KDD ’99 dataset

<table>
<thead>
<tr>
<th>No.</th>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>duration</td>
<td>Duration of the connection.</td>
</tr>
<tr>
<td>2</td>
<td>protocol type</td>
<td>Connection protocol (e.g., tcp, udp)</td>
</tr>
<tr>
<td>3</td>
<td>service</td>
<td>Destination service (e.g., telnet, ftp)</td>
</tr>
<tr>
<td>4</td>
<td>flag Status</td>
<td>Flag of the connection</td>
</tr>
<tr>
<td>5</td>
<td>source bytes</td>
<td>Bytes sent from source to destination</td>
</tr>
<tr>
<td>6</td>
<td>destination bytes</td>
<td>Bytes sent from destination to source</td>
</tr>
<tr>
<td>7</td>
<td>land</td>
<td>1 if connection is from/to the same host/port; 0 otherwise</td>
</tr>
<tr>
<td>8</td>
<td>wrong fragment</td>
<td>Number of wrong fragments</td>
</tr>
<tr>
<td>9</td>
<td>urgent</td>
<td>Number of urgent packets</td>
</tr>
<tr>
<td>10</td>
<td>hot</td>
<td>Number of “hot” indicators</td>
</tr>
<tr>
<td>11</td>
<td>failed logins</td>
<td>Number of failed logins</td>
</tr>
<tr>
<td>12</td>
<td>logged in</td>
<td>1 if successfully logged in; 0 otherwise</td>
</tr>
<tr>
<td>13</td>
<td>compromised</td>
<td>Number of “compromised” conditions</td>
</tr>
<tr>
<td>14</td>
<td>root shell</td>
<td>1 if root shell is obtained; 0 otherwise</td>
</tr>
<tr>
<td>15</td>
<td>su attempted</td>
<td>1 if “su root” command attempted; 0 otherwise</td>
</tr>
<tr>
<td>16</td>
<td>if root</td>
<td>Number of “root” accesses</td>
</tr>
<tr>
<td>17</td>
<td>file creations</td>
<td>Number of file creation operations</td>
</tr>
<tr>
<td>18</td>
<td>shells</td>
<td>Number of shell prompts</td>
</tr>
<tr>
<td>19</td>
<td>access files</td>
<td>Number of operations on access control files</td>
</tr>
<tr>
<td>20</td>
<td>outbound cmds</td>
<td>Number of outbound commands in an ftp session</td>
</tr>
<tr>
<td>21</td>
<td>is hot login</td>
<td>1 if the login belongs to the “hot” list; 0 otherwise</td>
</tr>
<tr>
<td>22</td>
<td>is guest login</td>
<td>1 if the login is a “guest” login; 0 otherwise</td>
</tr>
<tr>
<td>23</td>
<td>Count</td>
<td>Number of connections to the same host as the current connection in the past two seconds</td>
</tr>
<tr>
<td>24</td>
<td>srv count</td>
<td>Number of connections to the same service as the current connection in the past two seconds</td>
</tr>
<tr>
<td>25</td>
<td>serror rate</td>
<td>% of connections that have “SYN” errors</td>
</tr>
<tr>
<td>26</td>
<td>srv serror rate</td>
<td>% of connections that have “SYN” errors</td>
</tr>
<tr>
<td>27</td>
<td>error rate</td>
<td>% of connections that have “REJ” errors</td>
</tr>
<tr>
<td>28</td>
<td>srv error rate</td>
<td>% of connections that have “REJ” errors</td>
</tr>
<tr>
<td>29</td>
<td>same srv rate</td>
<td>% of connections to the same service</td>
</tr>
<tr>
<td>30</td>
<td>diff srv rate</td>
<td>% of connections to different services</td>
</tr>
<tr>
<td>31</td>
<td>srv diff host rate</td>
<td>% of connections to different hosts</td>
</tr>
<tr>
<td>32</td>
<td>dst host count</td>
<td>Count of connections having the same destination host</td>
</tr>
<tr>
<td>33</td>
<td>dst host srv count</td>
<td>Count of connections having the same destination host and using the same service</td>
</tr>
<tr>
<td>34</td>
<td>dst host same srv rate</td>
<td>% of connections having the same destination host and using the same service</td>
</tr>
<tr>
<td>35</td>
<td>dst host diff srv rate</td>
<td>% of different services on the current host</td>
</tr>
<tr>
<td>36</td>
<td>dst host same src port rate</td>
<td>% of connections to the current host having the same src port</td>
</tr>
<tr>
<td>37</td>
<td>dst host srv diff host rate</td>
<td>% of connections to the same service coming from different hosts</td>
</tr>
<tr>
<td>38</td>
<td>dst host serror rate</td>
<td>% of connections to the current host that have an SO error</td>
</tr>
<tr>
<td>39</td>
<td>dst host srv serror rate</td>
<td>% of connections to the current host and specified service that have an SO error</td>
</tr>
<tr>
<td>40</td>
<td>dst host error rate</td>
<td>% of connections to the current host that have an RST error</td>
</tr>
<tr>
<td>41</td>
<td>dst host srv error rate</td>
<td>% of connections to the current host and specified service that have an RST error</td>
</tr>
</tbody>
</table>
APPENDIX B

Abbreviations

ACK  Valid Acknowledgement Flag
AUC  Area Under Curve
CWR  Congestion Window Reduced
DIDS  Distributed Intrusion Detection System
DOS  Denial Of Service Attack
DT  Decision Trees
ECE  ECN -Echo Flag
FPR  False Positive Rate
FSR  FIN+SYN+RST flags
FIN  Final Data Flag
APPENDIX B.  ABBREVIATIONS

FTP  File Transfer Protocol  
ICA  Independent Component Analysis  
IDS  Intrusion Detection System  
IP  Internet Protocol  
KDD  Knowledge Discovery and Data Mining  
LISYS  Light-weight Intrusion Detection System  
LVQ  Learning Vector Quantization  
NATE  Network Analysis of Anomalous Traffic Events  
NetSTAT  Network Statistics  
NFR  Network Flight Recorder  
NIDS  Network Intrusion Detection System  
NN  Neural Network  
NSM  Network Security Monitor  
PBA  Polymorphic Blending Attack  
PCA  Principal Component Analysis  
PSH  Push Request Flag  
ROC  Receiver Operating Characteristic  
RST  Rough Set Theory  
RST flag  Reset session flag  
SMTP  Simple Mail Transfer Protocol
APPENDIX B. ABBREVIATIONS

SOM  Self Organizing Maps

SPOT  Stream Projected Outlier deTector

SST  Sparse Subspace Template

SVD  Singular Value Decomposition

SVM  Support Vector Machines

SYN  Synchronize Sequence Number

TCP  Transmission Control Protocol

TIM  Time-based Inductive Machine

TPR  True Positive Rate

URG  Urgent Data Flag