

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELGAUM**



Project Report on

**“IMPLEMENTATION AND EVALUATION OF NETWORK
ANOMALY DETECTION ALGORITHMS”**

Submitted in the partial fulfillment for the requirements of the degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

By

**Mr. MAHIDHAR C
Mr. NITESH A JAIN
Ms. PADMAVATHI K
Mr. PARTHASARATHY M ALWAR**

**USN: 1BY10CS037
USN: 1BY10CS047
USN: 1BY10CS048
USN: 1BY10CS049**

Under the guidance of

Mrs. A MARI KIRTHIMA

Asst. Professor,
Department of CSE, BMSIT.



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

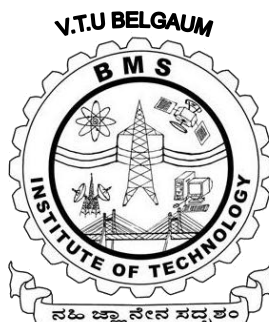
**B.M.S. INSTITUTE OF TECHNOLOGY
Yelahanka, BANGALORE-560064**

2013-2014

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELGAUM**

B.M.S INSTITUTE OF TECHNOLOGY
Yelahanka, BANGALORE-560064

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



CERTIFICATE

This is to certify that the Project work entitled “**IMPLEMENTATION AND EVALUATION OF NETWORK ANOMALY DETECTION ALGORITHMS**” is a bonafide work carried out by **Mr. MAHIDHAR C (1BY10CS037), Mr. NITESH A JAIN (1BY10CS047), Ms. PADMAVATHI K (1BY10CS048), Mr. PARTHASARATHY M ALWAR (1BY10CS049)** in partial fulfillment for the award of **Bachelor of Engineering Degree in Computer-Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2013-14. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in this report. The project report has been approved as it satisfies the academic requirements in respect of project work for the B.E Degree.

Signature of the Guide

Mrs. A MARI KIRTHIMA

Signature of the HOD

Dr. THIPPESWAMY G.

Signature of the Principal

Dr. R.V RANGANATH.

External VIVA

Name of the Examiners

Signature With Date

1.

2.

ABSTRACT

Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behavior. These non-conforming patterns are often referred to as anomalies or outliers. Intrusion detection in computer networks is the process of monitoring for and identifying attempted unauthorized system access or manipulation. This project examines the application of Statistical Modeling for detecting intrusions in computer networks.

Currently, a rule based Network Intrusion Detection System (NIDS) relies on the details of previously known attacks. By knowing the attack vector, rules can be implemented in the firewall and/or other defense tools to recognize the attack and appropriately counter it. However the main drawbacks of this system are - New unknown attacks cannot be countered effectively till encountered and Damage to the system rendering any future counters to the attack useless.

We propose a small NIDS, which is a method to detect and identify attacks. It will concentrate on a limited category of attacks from standard datasets containing well established and tested attack vectors. The network will be modeled by the NIDS using machine learning algorithms based on Naive Bayes Models.

ACKNOWLEDGEMENT

We are happy to present this project after completing it successfully. This project would not have been possible without the guidance, assistance and suggestions of many individuals. We would like to express our deep sense of gratitude and indebtedness to each person who has helped us make this project a success.

We heartily thank our **Principal, Dr. R V Ranganath, BMS Institute of Technology** for his constant encouragement and inspiration in taking up this project.

We are grateful to our **Head of Department, Dr. Thippeswamy G, Dept. of Computer Science and Engineering, BMS Institute of Technology** for his constant encouragement and inspiration in taking up this project.

We sincerely thank our Project guide, **Mrs. A Mari Kirthima, Asst. Professor, Dept. of Computer Science and Engineering**, for her encouragement and advice throughout the course of the Project work.

We gracefully thank our Project Coordinator, **Mr. Rajesh N V, Asst. Professor, Dept. of Computer Science and Engineering**, for his encouragement and advice throughout the course of the Project work.

Our sincere thanks to **Dr. Dipti Deodhare, Scientist 'G', CAIR (DRDO)** and **Mr. Shailesh Sonone, Scientist 'E', CAIR (DRDO)** for providing us the opportunity to do this project under **Defense Research & Development Organization**.

Special thanks to all the staff members of Computer Science Department for their help and kind co-operation.

Lastly we thank our parents and friends for their encouragement and support given to us in order to finish this precious work.

Mahidhar C
Nitesh A Jain
Padmavathi K
Parthasarathy M Alwar

CONTENTS

<i>Abstract</i>	<i>i</i>
<i>Acknowledgment</i>	<i>ii</i>
<i>Contents</i>	<i>iii</i>
<i>List of Figures</i>	<i>vi</i>
<i>List of Tables</i>	<i>vii</i>

<u>Chapters</u>	<u>Page No.</u>
Chapter 1: Introduction	01
1.1 Overview	01
1.2 Network Security	01
1.2.1 Intrusion Detection System	01
1.2.2 Attack Classification	02
1.3 Machine Learning and its Applications	02
Chapter 2: Literature Survey	05
2.1 Intrusion Detection Systems	05
2.1.1 Signature Based Detection	05
2.1.2 Statistical Anomaly Based Detection	05
2.2 Machine Learning	05
2.2.1 Supervised Learning	06
2.2.2 Unsupervised Learning	07
2.2.3 Reinforcement Learning	07
2.3 Types of Anomaly Based Detection	08
2.3.1 Statistical Anomaly Detection	08
2.3.2 Data Mining Based Approach	10
2.3.3 Knowledge Based Detection Techniques	11
2.4 Statistical Modeling	12
2.4.1 Markov Models	12
2.4.2 Markov Chains	13
2.4.3 Hidden Markov Models	13

2.4.4 HMM Based Related Work	15
2.5 Naive Bayes Classifier	16
2.5.1 Bayes Theorem	16
2.5.2 Formal Definition and Background	17
2.5.3 Properties of Naive Bayes	18
2.5.4 Advantages and Limitations	19
2.5.5 Recent Research in Network Anomaly Detection using NBC	19
2.6 Proposed System	21
Chapter 3: System Analysis	22
3.1 Requirements	22
3.1.1 Hardware Requirements	22
3.1.2 Software Requirements	22
3.1.3 Functional Requirements	22
3.2 Datasets	23
Chapter 4: System Design	26
4.1 Design Considerations	26
4.1.1 Modules	26
4.1.2 Use Case Diagram	28
4.1.3 Sequence Diagram	29
4.1.4 Data Flow Diagram	30
4.1.5 State Diagram	31
Chapter 5: Implementation	32
5.1 Language Used for Implementation	32
5.1.1 Python	32
5.2 Libraries Used for Implementation	33
5.2.1 NumPy	33
5.2.2 Flask	34
5.2.3 Matplotlib	34
5.3 Procedure Description	35
5.3.1 Determine Optimal Bands Procedure	35
5.3.2 Update Probabilities Procedure	35
5.3.3 Train Procedure	35

5.3.4 Train Phase Procedure	36
5.3.5 Determine Probability Procedure	36
5.3.6 Determine Threshold Probability Procedure	36
5.3.7 Deploy Procedure	36
5.3.8 Parse Procedure	36
Chapter 6: Results	37
6.1 Screenshots	37
6.2 Results Obtained	41
Chapter 7: Conclusion	42
Chapter 8: Challenges and Future Work	43
Appendix	44
Bibliography	49

LIST OF FIGURES

<u>Figure No.</u>	<u>Name of the Figure</u>	<u>Page No.</u>
Figure 1.1	Intrusion Detection Systems	2
Figure 1.2	Example of Anomalies	3
Figure 2.1	Machine Learning System	6
Figure 2.2	Supervised Learning Model	6
Figure 2.3	Unsupervised Learning Model	7
Figure 2.4	Reinforcement Learning Model	8
Figure 2.5	Taxonomy of Anomaly Based IDS	8
Figure 2.6	Markov Chain Illustration	13
Figure 2.7	Trellis Graph Representation of a HMM	13
Figure 2.8	Experimental Results for TCP	20
Figure 2.9	Comparison of Classification Results of various BN's on full dataset and Wenke Lee	20
Figure 2.10	Comparison of results using 41 attributes	21
Figure 3.1	TCP Packet Format	23
Figure 4.1	Overall System Design	26
Figure 4.2	Use Case Diagram	28
Figure 4.3	Sequence Diagram	29
Figure 4.4	Data Flow Diagram	30
Figure 4.5	State Diagram	31
Figure 6.1	Home Page	37
Figure 6.2	Testing Page	37
Figure 6.3	DARPA No Attack Results	38
Figure 6.4	DARPA Attack Results	38
Figure 6.5	KDD No Attack Results	39
Figure 6.6	KDD Attack Results	39
Figure 6.7	Auckland No Attack Results	40
Figure 6.8	Auckland Attack Results	40
Figure A.1	Performance Matrix for AUCK-II	46
Figure A.2	Performance Matrix for DARPA	46
Figure A.3	Experiments to Determine t (DARPA)	48
Figure A.4	Experiments to Determine t (AUCK-II)	48

LIST OF TABLES

<u>Table No.</u>	<u>Name of the Table</u>	<u>Page No.</u>
Table 1.1	Attacks present in DARPA 1999 Dataset	2
Table 2.1	Markov Models and their relationships	12
Table 3.1	TCP Flag Groups	24
Table 3.2	Window Statistics of the dataset files	25
Table 6.1	Results for various datasets	41

CHAPTER 1
INTRODUCTION

Chapter 1

INTRODUCTION

Computer networks are telecommunication networks which allow computers to exchange data. Network security is implemented on a variety of computer networks to secure daily transactions and communications among businesses, governments and individuals.

1.1 Overview

The current network security technologies (such as Firewall, IDS/ IPS) do provide security to networks but have limitations in terms of detecting only the known attack patterns. The system becomes prone to unknown vulnerabilities until the signatures of such attacks are found and configured in the signature databases of network security solutions. The moment attack pattern changes, the signature based solutions fail.

1.2 Network Security

Network security^[1] consists of the provisions and policies adopted by a network administrator to prevent and monitor unauthorized access, misuse, modification, or denial of a computer network and network-accessible resources. Network security involves the authorization of access to data in a network, which is controlled by the network administrator.

1.2.1 Intrusion Detection Systems

An Intrusion Detection System (IDS)^[2] is a device or software application that monitors network or system activities for malicious activities or policy violations and produces reports to a management station. Intrusion detection and prevention systems (IDPS) are primarily focused on identifying possible incidents, logging information about them, and reporting attempts.

In a passive system shown in figure 1.1, the intrusion detection system (IDS) sensor detects a potential security breach, logs the information and signals an alert on the console and/or owner. In a reactive system, also known as an intrusion prevention system (IPS), the IPS auto-responds to the suspicious activity by resetting the connection or by reprogramming the firewall to block network traffic from the suspected malicious source. The term IDPS is

commonly used where this can happen automatically or at the command of an operator; systems that both “detect (alert)” and “prevent”.

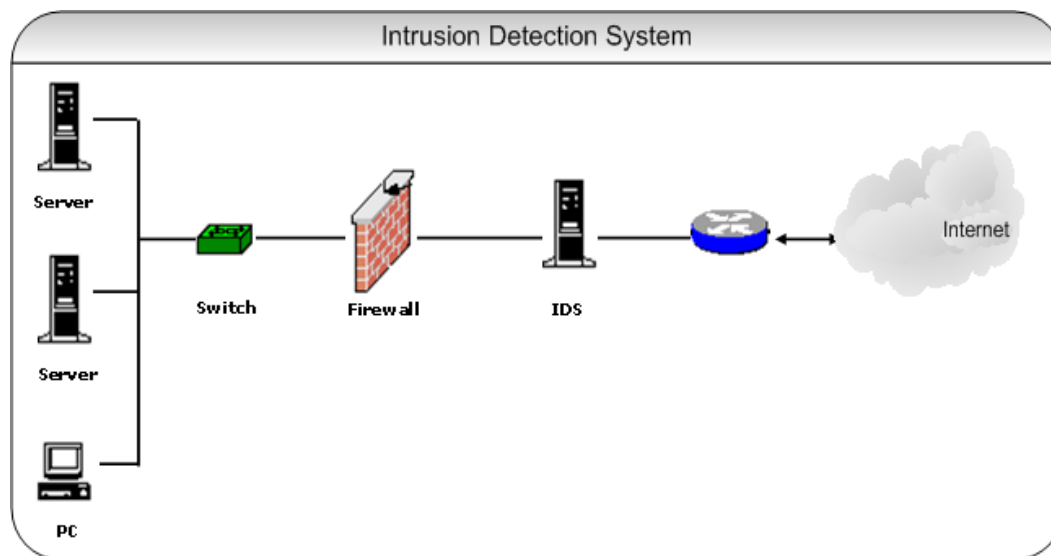


Figure 1.1 Intrusion Detection Systems

1.2.2 Attack Classification

In computer networks, an attack is any attempt to destroy, expose, alter, disable, steal or gain unauthorized access to or make unauthorized use of an asset.[3]

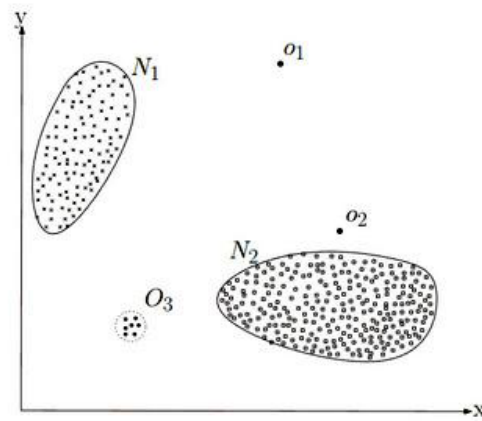
Attack Class	Attack Type
Probe	portsweep, ipsweep, quesoatan, msscan, ntinfoScan, lsdomain, illegal-sniffer.
DoS	apache2, smurf, neptune, tosnuke, land, pod, back, teardrop, tcpreset, syslogd, crashis, arppoison, mailbomb, selfping, processtable, udpstorm, warezclient.
R2L	dict, netcat, sendmail, imap, ncftp, xlock, xsnoop, ssttrojan, framespooF, pppmacro, guest, netbus, snmpget, ftpwrite, hhtptunnel, phf, named.
U2R	sechole, xterm, eject, ps, nukewp, secret, perl, yaga, fdformat, ffbconfig, casesen, ntfsdos, pppmacro, loadmodule, sqlattack.

Table 1.1 Attacks present in DARPA 1999 Dataset

1.3 Machine Learning and its Applications

Machine learning, a branch of artificial intelligence, concerns the construction and study of systems that can learn from data. For example, a machine learning system (shown in

figure 1.2) could be trained on network traffic to learn to distinguish between anomalous and normal traffic. After learning, it can be used to classify the incoming traffic into anomalous and normal traffic.



A simple example of anomalies in a 2-dimensional data set.

Figure 1.2 Example of Anomalies

Machine Learning is used in a variety of fields, a few of which are mentioned below.

- i. **Network Anomaly Detection:** Statistical machine learning is one class of statistical approaches used in the field of Network Anomaly Detection. A choice of a learning scenario depends on the information available in measurements. For example, a frequent scenario is that there are only raw network measurements available and thus unsupervised learning methods are used. If additional information is available, e.g. from network operators, known anomalies or normal network behaviors, learning can be done with supervision. A choice of a mapping depends on the availability of a model, the amount and the type of measurements, and complexity of learning algorithms^[4].
- ii. **Evaluating Learned Knowledge:** Rules induced from training data are not necessarily of high quality. The performance of knowledge acquired in this way is an empirical question that must be answered before that knowledge can be used on a regular basis. One standard approach to evaluation involves dividing the data into two sets, training on the first set, and testing the induced

knowledge on the second. One can repeat this process a number of times with different splits, and then average the results to estimate the rules' performance on completely new problems. Kibler and Langley (1988) experimental methods of this sort for a broad class of learning algorithms. An important part of the evaluation process is experts' examination of the learned knowledge. Evans and Fisher (1994) encourage an iterative process in developing a fielded application.^[5]

CHAPTER 2

LITERATURE SURVEY

LITERATURE SURVEY

The existing literature regarding intrusion detection systems as well as machine learning algorithms are discussed below. Additionally the theoretical details of Hidden Markov Models and Naive Bayes Classifier are explained.

2.1 Intrusion Detection Systems

An Intrusion can be termed as any activity which is aimed at disrupting or denying a service by gaining unauthorized access. Various attacks like DDoS, Identity spoofing, Ping of Death, Buffer Overflow etc. are used to intrude and disrupt a network. Intrusion Detection Systems are used to prevent such intrusions by detecting them at an earlier stage.

All Intrusion Detection Systems use one of two detection techniques:

- i. Signature Based Detection
- ii. Statistical Anomaly Based Detection

2.1.1 Signature Based Detection

Signature-based IDS monitors packets in the network, and compares them with pre-configured and pre-determined attack patterns, known as signatures.^[6]

2.1.2 Statistical Anomaly Based Detection

Anomaly based detection techniques model a “normal” scenario and any deviation from that is considered an anomaly. Anomaly based detection is preferred over Signature based detection due to their ability to detect novelty attacks.^[6]

2.2 Machine Learning

Machine learning is a branch of artificial intelligence, concerned with the construction and study of systems that can learn from data. Machine learning focuses on the development of computer programs that can teach themselves to grow and change when exposed to new data.^[7] A Machine learning system is shown in figure 2.1.

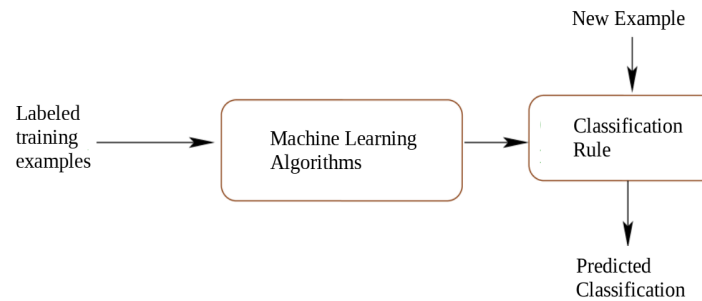


Figure 2.1 Machine Learning System

Machine learning algorithms are classified into 3 types-

- i. Supervised Learning
- ii. Unsupervised Learning
- iii. Reinforcement Learning

2.2.1 Supervised Learning

In supervised learning, the model defines the effect one set of observations, called inputs, has on another set of observations, called outputs.^[7] As shown in Figure 2.2, the inputs are assumed to be at the beginning and outputs at the end of the causal chain. The models can include mediating variables between the inputs and outputs. A supervised learning algorithm makes it possible, given a large data set of input/output pairs, to predict the output value for some new input value that you may not have seen before. In supervised learning, one is trying to find the connection between two sets of observations.

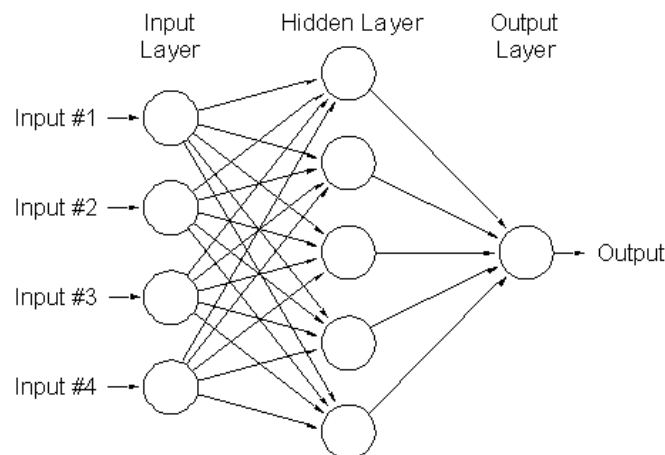


Figure 2.2 Supervised Learning Model

2.2.2 Unsupervised Learning

In unsupervised learning, all the observations are assumed to be caused by latent variables, that is, the observations are assumed to be at the end of the causal chain.^[7] With unsupervised learning it is possible to learn larger and more complex models than with supervised learning. The learning can proceed hierarchically from the observations into ever more abstract levels of representation. Each additional hierarchy needs to learn only one step and therefore the learning time increases (approximately) linearly in the number of levels in the model hierarchy unlike supervised learning where the learning task increases exponentially. Unsupervised learning can be used for bridging the causal gap between input and output observations. The latent variables in the higher levels of abstraction are the causes for both sets of observations and mediate the dependence between inputs and outputs. Figure 2.3 describes an Unsupervised Learning model.

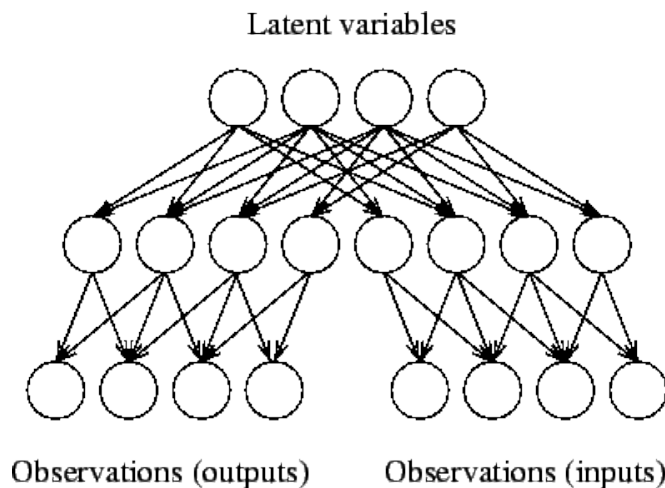


Figure 2.3 Unsupervised Learning Model

2.2.3 Reinforcement Learning

Reinforcement learning is learning what to do - how to map situations to actions - as to maximize a numerical reward signal. A Reinforcement learning agent learns from the consequences of its actions, rather than from being explicitly taught and it selects its actions on basis of its experiences (exploitation) and also by new choices (exploration), which is essentially trial and error learning. Figure 2.4 describes a Reinforcement Learning model.

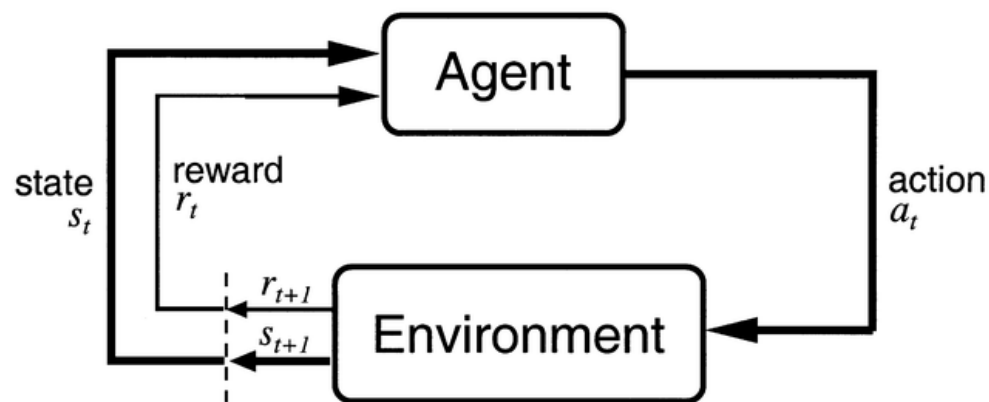


Figure 2.4 Reinforcement Learning Model

2.3 Types of Anomaly Based Detection

The taxonomy of anomaly based IDS is shown in figure 2.5. We now review the different techniques of Anomaly based IDS. The most important ones are-

- i. Statistical Anomaly Detection
- ii. Data Mining Based Detection
- iii. Knowledge Based Detection

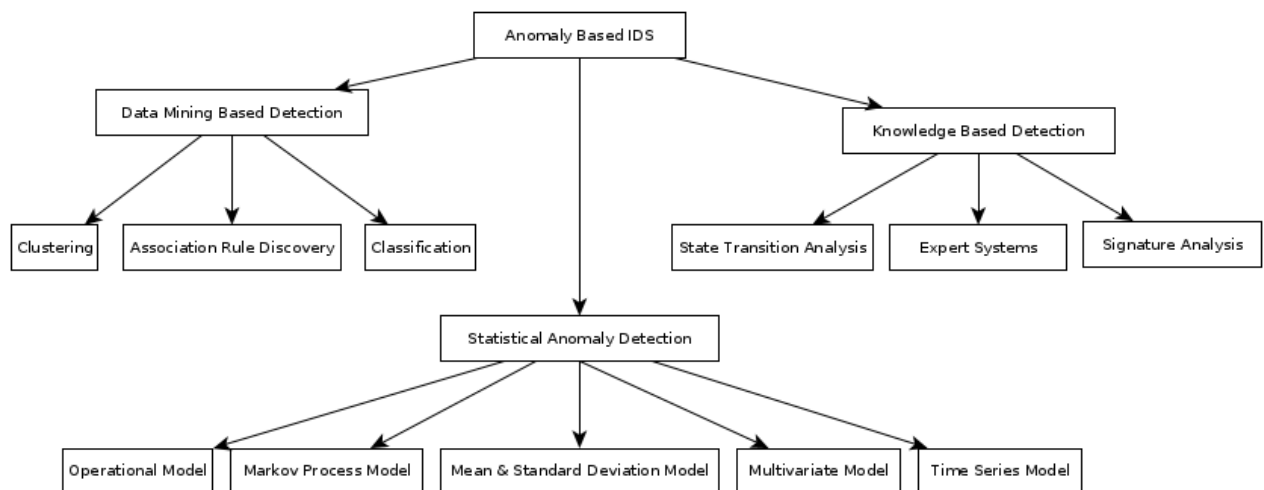


Figure 2.5 Taxonomy of Anomaly Based IDS

2.3.1 Statistical Anomaly Detection

Statistical based anomaly detection techniques use statistical properties and statistical tests to determine whether “Observed behavior” deviate significantly from the “expected

behavior”. Statistical based anomaly detection techniques use statistical properties (e.g., mean and variance) of normal activities to build a statistical based normal profile and employ statistical tests to determine whether observed activities deviate significantly from the normal profile. The IDS goes on assigning a score to an anomalous activity. As soon as this score becomes greater certain threshold, it will generate an alarm.^[8] Statistical Anomaly Based IDS can further be classified into the following categories-

- **Operational Model:** This model is based on the operational assumption that an anomaly can be identified through a comparison of an observation with a predefined limit. Based on the cardinality of observation that happens over a time an alarm is raised. The operational model is most applicable to metrics where experience has shown that certain values are frequently linked with intrusions. For example, an event counter for the number of password failures during a brief period, where more than say 10, suggest a failed log-in.
- **Markov Process Model:** The Markovian Model is used with the event counter metric to determine the normalcy of a particular event, based on the events which preceded it.^[8] The model characterizes each observation as a specific state and utilizes a state transition matrix to determine if the probability of the event is high (normal) based on the preceding events. This model is basically used in two main approaches: Markov chains and Hidden Markov models. A Markov chain keeps track of an intrusion by examining the system at fixed intervals and maintains the record of its state. If the state change takes place it computes the probability for that state at a given time interval. If this probability is low at that time interval then that event is considered as an anomalous.
- **Mean and Standard Deviation Model:** The Mean and Standard Deviation Model is based on the traditional statistical determination of the normalcy of an observation based on its position relative to a specified confidence range.^[8] In this sub-model, event that falls outside the set interval will be declared as an anomalous.
- **Multivariate Model:** This model can be applied to intrusion detection for monitoring and detecting anomalies of a process in an information system.^[8] This model is

similar to the mean and standard deviation model except that it is based on correlations among two or more features. This model would be useful in the situation where two or more features are related. This model permits the identification of potential anomalies where the complexity of the situation requires the comparison of multiple parameters.

- **Time Series Model:** The Time Series Model, attempts to identify anomalies by reviewing the order and time interval of activities on the network.^[8] If the probability of the occurrence of an observation is low, then the event is labeled as abnormal. This model provides the ability to evolve over time based on the activities of the users. Anomalies in time series data are data points that significantly deviate from the normal pattern of the data sequence.

2.3.2 Data Mining Based Approach

As IDS can only detect known attacks, but it cannot detect insider attacks, the better solution for an IDS can be Data Mining at its core and is defined as “the process of extracting useful and previously unnoticed models or patterns from large data stores”.^[8] The data-mining process tends to reduce the amount of data that must be retained for historical comparisons of network activity, creating data that is more meaningful to anomaly detection.

Data Mining based approach is classified into following techniques:

- **Clustering:** Clustering^[8] is an unsupervised technique for finding patterns in unlabeled data with many dimensions (number of attributes). Mostly k-means clustering is used to find natural groupings of similar instances. Records that are far from any of these clusters indicate unusual activity that may be part of a new attack.
- **Association Rule Discovery:** Association rules mining finds correlation between the attributes. This technique was initially applied to the so-called market basket analysis, which aims at finding regularities in shopping behavior of customers of supermarkets. This method is usually very slow and its being replaced by other powerful techniques like clustering and classification.^[8]

- **Classification:** Classification^[8] is one of the main techniques used in data mining. Its main goal is to learn from class-labeled training instances for predicting classes of new or previously unseen data. Instances in the training set have labels. New data is classified based on the training set. Basically a classification tree (also called as decision tree) is built to predict the category to which a particular instance belongs.

2.3.3 Knowledge Based Detection Techniques

Knowledge based detection Technique^[8] can be used for both signature based IDS as well as anomaly based IDS. It accumulates the knowledge about specific attacks and system vulnerabilities. It uses this knowledge to exploit the attacks and vulnerabilities to generate the alarm. Any other event that is not recognized as an attack is accepted. Therefore the accuracy of knowledge based intrusion detection systems is considered good. Knowledge based detection Technique can further be classified as:

- i. **State Transition Analysis:** This technique describes the attacks with a set of goals and transitions, and represents them as state transition diagrams.^[8] State transition diagram is a graphical representation of the actions performed by an intruder to archive a system compromise. In state transition analysis, an intrusion is viewed as a sequence of actions performed by an intruder that leads from some initial state on a computer system to a target compromised state. State transition analysis diagrams identify the requirements and the compromise of the penetration. They also list the key actions that have to occur for the successful completion of an intrusion.
- ii. **Expert Systems:** The expert system^[8] contains a set of rules that describe attacks. Audit events are then translated into facts carrying their semantic significance in the expert system, and the inference engine draws conclusions using these rules and facts. This method increases the abstraction level of the audit data by attaching semantic to it. It also encodes knowledge about past intrusions, known system vulnerabilities and the security policy. As information is gathered, the expert system determines whether any rules have been satisfied.

- iii. **Signature Analysis:** Signature analysis follows exactly the same knowledge-acquisition approach as expert systems, but the knowledge acquired is exploited in a different way.^[8] The semantic description of the attacks is transformed into information that can be found in the audit trail in a straightforward way. For example, attack scenarios might be translated into the sequences of audit events they generate, or into patterns of data that can be sought in the audit trail generated by the system.

2.4 Statistical Modeling

The formalization of relationships between variables in the form of mathematical equations results in a **Statistical Model**. It describes how random variables are related to each other. The model is statistical as the variables are not deterministically but rather **stochastically** related.

2.4.1 Markov Models

In order to understand the definition of a Markov Model, a clear definition of the mathematical principle known as Markov Property must be established. A stochastic process has the Markov property if the conditional probability distribution of future states of the process depends only upon the present state, rather than on the sequence of events that preceded it.

In other words, a Markov Model is used mainly to analyze temporal data, and it assumes that the future is independent of the past states, and is dependent only on the data contained in the present state. Given the following set $D = \{X_1 \dots X_n\}$, we can say X_t depends on $X_{t-1} \dots X_{t-m}$ with the simplest case being $m=1$. We further simplify our model by assuming discrete time and space of the sets.^[9]

The most common Markov models and their relationships are summarized in the following table:

	Fully Observable System States	Partially Observable System States
Autonomous System	Markov Chain	Hidden Markov Model
Controlled System	Markov Decision Process	Partially Observable MDP

Table 2.1 Markov Models and their relationships

2.4.2 Markov Chains

Markov chains are used to model autonomous systems whose states are fully observable. Discrete random variables $X_1 \dots X_n$ form a Markov Chain, if they respect

$$P(X_t | X_1 \dots X_{t-1}) = P(X_t | X_{t-1})$$

The above equation is illustrated as:

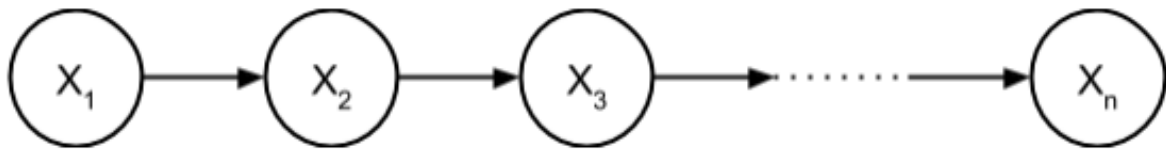


Figure 2.6 Markov Chain Illustration

The biggest drawback of Markov Chains is that it is assumed that the whole system is visible. This assumption is invalid in several real-life scenarios as more often there are two different types of states/variable namely observable variables and hidden variable. Hidden variables are also known as latent variables.

2.4.3 Hidden Markov Model

To overcome the disadvantage of Markov Chains, the concept of Hidden Markov Models (HMM) is used. Hidden Markov Models are useful to define an autonomous stochastic model whose states are partially observable. The concept of HMM is as illustrated in the **Trellis Graph** shown below:

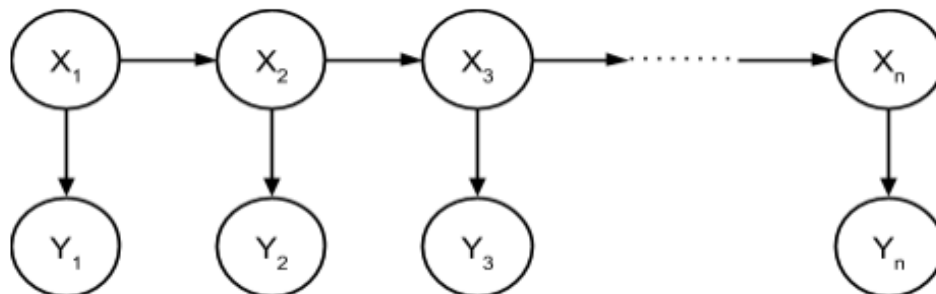


Figure 2.7 Trellis Graph Representation of a HMM

In the above graph, there are two types of states. The hidden/latent variables are represented as,

$$X_1 \dots X_n \in \{1 \dots m\}$$

The visible states, which are observable, are represented by

$$Y_1 \dots Y_n \in \chi(\text{discrete/finite variables})$$

There are three parameters under consideration under Hidden Markov Models.

- i. **State Transition Probability Matrix:** The probability of the HMM transitioning from an existing state X_k to a new state X_{k+1} . It is formally defined as

$$A_{ij} = P(X_{k+1} = j | X_k = i) \forall (i, j \in \{1 \dots m\})$$
- ii. **Emission Probability:** The probability of a given observable variable Y_k occurring given the probability of a latent variable X_k . It is a **probability mass function**, as we are considering only discrete random variables. It is formally defined as

$$B_i(y) = P(Y_k = y | X_k = i) \forall (i \in \{1 \dots m\}, y \in \chi)$$
- iii. **Initial Distribution Probability:** The probability that a given HMM is in an initial state X_k . It is formally defined as

$$\pi_i = P(X_k = i) \forall (i \in \{1 \dots m\})$$

Given the above three parameters, it is possible to formally define HMM's as follows: A Hidden Markov Model (λ) is a five tuple as given: $(\lambda) = [X, Y, A, B, \pi]$, with the joint probability distribution of X and Y defined as,

$$P(Y_1 \dots Y_n, X_1 \dots X_n) = P(X_1)P(Y_1 | X_1) \prod_{k=2}^n P(X_k | X_{k-1})P(Y_k | X_k)$$

There are three problems associated with Hidden Markov Models. These are commonly used to analyse and obtain useful information from the HMM. The three problems are as described.

- i. **Evaluation** of the probability that a given model generate a given sequence of observations. This is solved by the **Forward-Backward Algorithm**.
- ii. **Decoding** the sequence of the hidden states that most likely generated a given sequence of observations. The **Viterbi Algorithm** solves this problem.
- iii. **Learning** the model which most probably underlies the given sample of observations. In other words, learning the parameters of the HMM. This problem is solved using the **Baum-Welch Algorithm**.

2.4.4 HMM Based Related Work

In this section, we discuss the various Hidden Markov Model based related work carried out by various researchers.

- i. **Adaptive Network Intrusion Detection System Using a Hybrid Approach:** In this approach, an adaptive network intrusion detection system, that uses a two stage architecture is described. In the first stage a probabilistic classifier is used to detect potential anomalies in the traffic. In the second stage a HMM based traffic model is used to narrow down the potential attack IP addresses. Due to the difficulties that arise when implementing HMM in real time, HMM model along with NB model was incorporated into a hybrid model for intrusion detection^[10]. The proposed hybrid model performed well in detecting intrusions. HMM with more than five states had 100% accuracy classifying all IPs from CAIDA^[11] as attack and IPs from DARPA^[12] as clean. When tried with HMM with less than five states, few of the attacking streams were classified as clean traffic. This is due to the inability of HMM with very few states to model the data accurately. Using HMM with larger states gave exact results and the number of states to be chosen for a server can be computed empirically.
- ii. **Sensing attacks in Computers Networks with Hidden Markov Models:** In this approach,^[13] an Intrusion Detection model for computer networks based on Hidden Markov Models is proposed. The proposed model aims at detecting intrusions by analysing the sequences of commands that flow between hosts in a network for a particular service (e.g., an ftp session). For each command sequence, a probability

value is assigned and a decision is taken according to some predefined decision threshold. First the system must be trained in order to learn the typical sequences of commands related to innocuous connections. Then, intrusion detection is performed by identifying anomalous sequences. Different HMM models were investigated in terms of the dictionary of symbols, number of hidden states, and different training sets. It was found that good performances can be attained by using dictionary of symbols made up of all symbols in the training set, and adding a NaS (not-a-symbol) symbol in account of symbols in the test set that are not represented in the training set. Performances can be further improved by combining different HMM. As the size of training sets in an intrusion detection application is typically large, it was proposed to split the training set in a number of parts, training different HMM and then combining the output probabilities by three well known combination.

2.5 Naive Bayes Classifier

A Naïve Bayes Classifier is a simple probabilistic classifier which uses the Bayes' theorem with an assumption that the occurrence of an event is totally unrelated to the occurrence of another.

Despite their naive design and apparently oversimplified assumptions, Naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, an analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes classifiers.^[14]

2.5.1 Bayes Theorem

Bayes theorem is used by the Naïve Bayes Classifier. It is particularly suited when the dimensionality of the inputs is high. The formula for the Bayes theorem is represented by the figure below.

$$P(W|L) = \frac{P(L|W)P(W)}{P(L)} = \frac{P(L|W)P(W)}{P(L|W)P(W) + P(L|M)P(M)}$$

where,

$P(W|L) \rightarrow$ Probability of outcome W given L or “posterior probability”

$P(L|W)$ → Probability of outcome L given W or “likelihood”

$P(W)$ → Independent probability of W

$P(L)$ → Independent probability of L or “prior probability”

To explain with an example, let W represent the event that the conversation was held with a Woman and L denote the event that the conversation was held with a long-haired person. Assuming that women constitute half the population, the probability that W occurs is $P(W) = 0.5$.

Suppose it is also known that 75% of women have long hair, which we denote as $P(L|W) = 0.75$. Likewise, suppose it is known that 15% of men have long hair, or $P(L|M) = 0.15$, where M is the complementary event of W.

Calculation of the probability that the conversation was held with a woman, given the fact that the person had long hair, or, in our notation, $P(W|L)$ can be calculated using Bayes formula.

2.5.2 Formal Definition and Background

The naive Bayes classifier applies to learning tasks where each instance x is described by a conjunction of attribute values and where the target function $f(x)$ can take on any value from some finite set V . A set of training examples of the target function is provided, and a new instance is presented, described by the tuple of attribute values $\langle a_1, a_2 \dots a_n \rangle$. The learner is asked to predict the target value, or classification, for this new instance.^[15]

Assumption:

- Training set consists of instances described as conjunctions of attribute values
- Target classification is based on a finite set of classes ‘V’
- Independence assumption: features are independent in a given class.

Task: Predict the correct class for a new instance $\langle a_1, a_2 \dots a_n \rangle$

Key Idea: The Bayesian approach to classifying the new instance is to assign the most probable target value, VMAP, given the attribute values $\langle a_1, a_2 \dots a_n \rangle$ that describe the instance.

$$V_{MAP} = \operatorname{argmax} P(v_j | a_1, a_2, a_3 \dots a_n) \quad \forall v_j \in V$$

Rewriting the above equation using Bayes Theorem we get,

$$V_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} \frac{P(a_1, a_2, a_3 \dots a_n | v_j)}{P(a_1, a_2, a_3 \dots a_n)}$$

$$V_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} P(a_1, a_2, a_3 \dots a_n | v_j) P(v_j)$$

The naive Bayes classifier is based on the simplifying assumption that the attribute values are conditionally independent given the target value. The probability of observing the conjunction $\langle a_1, a_2 \dots a_n \rangle$ is just the product of the probabilities for the individual attributes. That is,

$$V_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} P(a_1, a_2, a_3 \dots a_n | v_j) = \prod_i^n P(a_i | v_j)$$

Hence we get the following classifier:

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \sum_i^n P(a_i | v_j)$$

where,

$v_{NB} \rightarrow$ target value output by the NBC.

2.5.3 Properties of Naive Bayes

The properties of Naive Bayes Classifier are:

- **Incremental approach:** With each training example, the prior and the likelihood can be updated dynamically. This results in a flexible and a robust model.
- **Combines prior knowledge and observed data:** The prior probability of a hypothesis multiplied with a probability of the hypothesis given the training data leads to more accuracy in correct classification.
- **Probabilistic hypothesis:** The output obtained is not just the classification but also a probability distribution over all classes.
- **Meta-Classification:** The outputs of several classifiers can be combined by multiplying the probabilities that all classifiers predict for a given class.

2.5.4 Advantages and Limitations

In this section we discuss the various advantages and disadvantages of the Naïve Bayes classifier.

Advantages are:

- Naïve Bayes Classification is computationally fast and space efficient
- It is faster to train the classifier as it requires a single scan
- It is not sensitive to irrelevant data
- It handles streaming data well

Disadvantages are:

- Requires several records to obtain good results
- When a predictor category is not present in the training data, naive Bayes assumes that a record with that category of the predictor has zero probability. This can be a problem if this rare predictor value is important.

2.5.5 Recent Research in Network Anomaly Detection in NBC

In this section, we discuss the various research work carried out in the field of Network Anomaly Detection using the Naive Bayes Classifier.

- **A System Approach to Network Modelling for DDoS Detection using a Naive Bayesian Classifier by R Vijayasathy, S V Raghavan and Balaraman Ravindran:** In this paper, the authors discuss a practically realized system to detect DoS attacks using a Naive Bayesian Classifier. They have modeled the network for two protocols – TCP and UDP. The concept of windowing is used to have a reasonable estimate and control over the reaction time of the system for attacks and better modeling from larger datasets.^[16] Three datasets were used in the experiment, namely, DARPA dataset, SETS (TCP) dataset and SETS (UDP) dataset. To determine four parameters i.e. false positives, false negatives, true positives, true negatives, they performed two tests: False Positive Test (FPT) and False Negative Test (FNT). They have calculated critical system parameters namely, Accuracy, False Alarm Rate and Miss Rate. The results of these tests are represented in figure 2.8 They report

improvements in accuracy with increase in window size.

WS	DARPA at $t = 1\%$			SETS at $t = 5\%$		
	Acc	FAR	MR	Acc	FAR	MR
50	98.3%	2.3%	0	97%	7%	0
100	98.6%	2%	0	97.4%	6.2%	0.06%
200	98.7%	1.8%	0	97.1%	5%	1.1%

Figure 2.8 Experimental Results for TCP

- From Feature Selection to Building of Bayesian Classifiers: A Network Intrusion Detection Perspective by Kok-Chin Khor, Choo-Yee Ting and Somnuk-Phon Amnuaisuk:** In this paper, the authors proposed a novel approach to select important features by utilizing two selected feature selection algorithms utilizing filter approach. Extra features were added into the final proposed feature set. They then constructed three types of BN namely, Naive Bayes Classifiers (NBC), Learned BN and Expert-elicited BN by utilizing a standard network intrusion dataset. The performance of each classifier was recorded. They found that there was no difference in overall performance of the BNs and therefore, concluded that the BNs performed equivalently well in network attacks. They conclude that Nave Bayes Classifier should be considered if the dataset involved is huge and time for building network and testing is the main factor to consider. Experimental results are represented in the figure 2.9. [17]

Category	NBC	BN	EE	Wenke lee
Normal	96.7	99.8	97.5	N/A
DoS	99.3	99.9	99.2	79.9
Probe	93.5	89.4	93.0	97
R2L	92.8	91.5	86.8	75
U2R	55.8	69.2	69.2	60

Figure 2.9 Comparison of Classification Results of various BN's on full dataset and Wenke Lee

- Combining Naïve Bayes and decision tree for adaptive Intrusion Detection by Dewan Md. Farid, Nouria Harbi, and Mohammad Zahidur Rahman:** In this paper, the authors propose a new learning algorithm for adaptive network intrusion detection using naive Bayesian classifier and decision tree, which performs balance detection and keeps false positives at acceptable level for different types of network attacks, and eliminates redundant attributes as well as contradictory examples from

training data that make the detection model complex. The proposed algorithm also addresses some difficulties of data mining such as handling continuous attribute, dealing with missing attribute values, and reducing noise in training data. The authors tested the performance of their proposed algorithm with existing learning algorithms by employing on the KDD99 benchmark intrusion detection dataset. Their experimental results prove that the proposed algorithm achieved high detection rates (DR) and significant reduce false positives (FP) for different types of network intrusions using limited computational resources. The results of the experiment can be found below. [18]

Method	Normal	Probe	DOS	U2R	R2L
Proposed Algorithm (DR %)	99.72	99.25	99.75	99.20	99.26
Proposed Algorithm (FP %)	0.06	0.39	0.04	0.11	6.81
Naïve Bayesian (DR %)	99.27	99.11	99.69	64.00	99.11
Naïve Bayesian (FP %)	0.08	0.45	0.04	0.14	8.02
ID3 (DR %)	99.63	97.85	99.51	49.21	92.75
ID3 (FP %)	0.10	0.55	0.04	0.14	10.03

Figure 2.10 Comparison of results using 41 attributes

2.6 Proposed System

Researchers are trying to build a fool-proof system which can be used to detect anomalies. The current systems haven't achieved a satisfactory detection rate. Constantly working to counter challenges encountered in the field of network anomaly detection is imperative. Through our proposed system, we plan to increase the existing accuracy and to contribute to the field of Network Security. The proposed system also introduces new challenges in the field of anomaly detection.

CHAPTER 3

SYSTEM ANALYSIS

SYSTEM ANALYSIS

The hardware and software details, functional requirements and the input to the proposed system is outlined in this chapter. The input details consist of the dataset statistics and the parsing techniques used.

3.1 Requirements

The different hardware and software requirements are covered in the next sub-sections. In the subsequent section, we cover the various functional requirements.

3.1.1 Hardware Requirements

- i386 or x64 based Processor
- 50 GB Secondary storage
- 8 GB of RAM

3.1.2 Software Requirements

- GNU/ Linux based OS
- Python 2.6 or greater
- Pycharm community edition
- Numpy 1.7 or greater
- Wireshark
- Git
- Flask
- Matplotlib
- Firefox or Chrome browser

3.1.3 Functional Requirements

Common functional requirements of an Intrusion Detection System are discussed below.

- a) The IDS must continuously monitor and report intrusions.
- b) The IDS should be modular and configurable as each host and network segment will

- require their own tests and these tests will need to be continuously upgraded and eventually replaced with new tests.
- c) The IDS must operate in a hostile computing environment and exhibit a high degree of fault-tolerance and allow for graceful degradation.
 - d) The IDS should be adaptive to network topology and configuration changes as computing elements are dynamically added and removed from the network.
 - e) Anomaly detection systems should have a very low false alarm rate.
 - f) The IDS should be capable of providing an automated response to suspicious activity.
 - g) The ability to detect and react to distributed and coordinated attacks.
 - h) The IDS should be able to work with other Commercial Off-the-Shelf (COTS) security tools, as no vendor toolset is likely to excel in or to provide complete coverage of the detection, diagnosis, and response responsibilities.
 - i) IDS data often requires additional analysis to assess any damage to the network after an intrusion has been detected.

3.2 Dataset

The IDS is trained and tested with standard datasets like DARPA^[19], KDD^[20] and Auckland^[21]. Each dataset is a collection of network packets grouped to form **tcpdump** files. Of the different types of packets in the tcpdump files, we filter only **TCP** packets. The structure of a TCP packet is shown in figure 3.1.

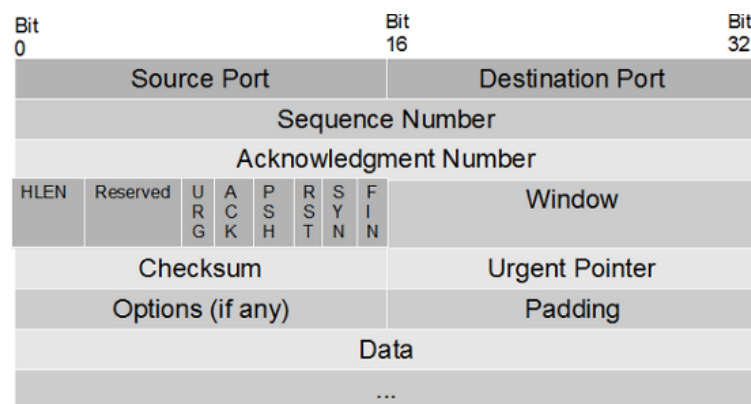


Figure 3.1 TCP Packet Format

Each TCP packet consists of TCP flags like URG, ACK, PSH, RST, SYN and FIN. We filter the flags and group them into six categories as follows:

Type	Flag Set	Flag Name	Description
T1	RST	Reset flag	This flag is set when the remote host rejects the incoming packet from a client on account of trying to establish a connection. This indicates that the remote host has reset the connection
T2	SYN	Synchronization flag	This flag is set when the host is trying to establish a connection by sending SYN (synchronize) packets
T3	ACK	Acknowledgement flag	This flag is set to acknowledge the successful receipt of packets
T4	FIN/ACK	Finish & Acknowledgement flag	This flag is set when the destination host wants to close the connection with the source host after data transfer. The FIN flag is set to indicate termination of connection while the ACK flag is set to acknowledge the last packet received during data transfer
T5	PSH/ACK	Push & Acknowledgement flag	This flag is set when feedback is required immediately from the client and server instead of waiting for the buffer to become full before transmission. The data is immediately pushed when the PSH flag is set and the ACK flag is set to acknowledge the data received
T6	Others	Others	Any other flag/s set other than the above mentioned flags, fall under this category

Table 3.1 TCP Flag Groups

The flags are stored as hexadecimal values which are passed to the program for classifying the incoming traffic as legitimate or malicious. In order to extract only the TCP flags from the datasets we run the following commands -

- ***tcpdump -r inside/w1mon.tcpdump -w test.pcap***

This command is used to read the data present in a tcpdump file (w1mon.tcpdump) and write the contents to a pcap file (test.pcap)

- ***tshark -r test.pcap -Y tcp -T fields -E separator=/s -e ip.dst -e tcp.dstport -e tcp.flags > op_mon***

This command takes test.pcap file as input, filters the tcp packets, extracts destination IP, destination port and TCP flags and stores the output in op_mon

- ***cat op_mon | grep "172.16.112.50 23" | cut -d " " -f 3 > op_flag_mon***

This command selects the particular IP and cuts the third field which contains TCP flag set and stores the output in op_flag_mon

- ***rm op_mon***

This command removes the intermediate file created

Finally, we get the TCP flags in the form of Hexadecimal values which are used to train and test the IDS.

The following table consists of information about the number of attack windows and normal windows present in the dataset files considered to train and test the IDS.

	Normal Windows	Attack Windows
darpaTrain	219600	0
darpaNoAttack	61400	0
darpaAttack	37100	37100
kddTrain	216900	0
kddNoAttack	126400	0
kddAttack	58200	58200
aucklandTrain	219600	0
aucklandNoAttack	126400	0
aucklandAttack	393	2048

Table 3.2 Window statistics of the dataset files

CHAPTER 4

SYSTEM DESIGN

SYSTEM DESIGN

System design is the process of defining the architecture, modules, components, interfaces and the data for the system to satisfy specified requirements. It pertains to the abstract representation of the data flows, inputs and outputs of the system.

4.1 Design Considerations

In this section, we consider the different modules under consideration, use-case diagram, sequence diagram, data flow diagram and state diagram.

4.1.1 Modules

The system is comprised of five modules as shown in figure 4.1. They are -

- Control Unit (CU)
- Traffic Capture Module (TCM)
- Learning Module (LM)
- Detection Module (DM)
- Mitigation Module (MM)

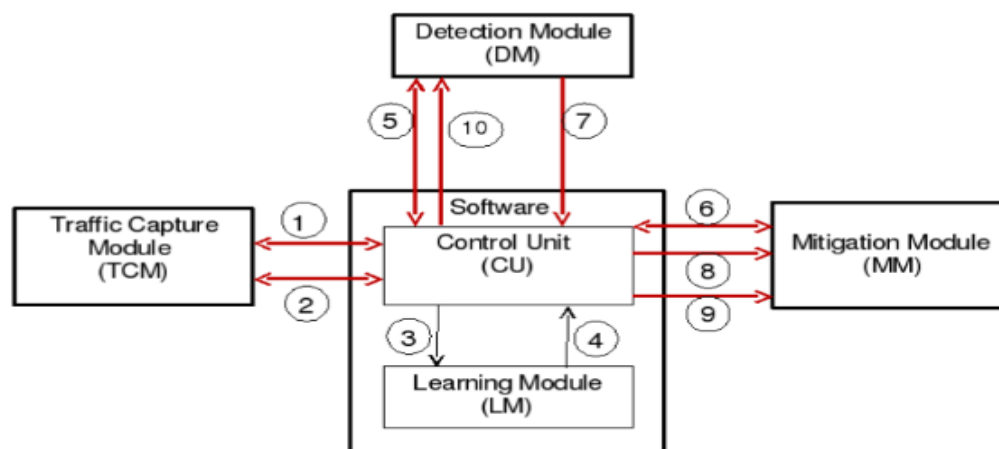


Figure 4.1 Overall System Design

Control Unit (CU):

The Control Unit is a software module, which centrally manages communication between the rest of the modules. CU is resident on the computer, hosting the Learning Module (LM), and interacts with the user (through a Graphical User Interface) to obtain inputs for the above modules. The CU also is responsible for information exchange and dissemination between the modules, and establishes a communication network between each of these modules for this purpose. The CU runs different protocols with the various modules for the task. This is a software module.

Traffic Capture Module (TCM):

The Traffic Capture Module (TCM) is a hardware module, which captures traffic required for learning phase, and computes traffic statistics as required for the Learning Module. The TCM communicates with the Control Unit (CU) to obtain its inputs, namely stream information. Based on triggers, the values of stream configuration are loaded (by CU), traffic captured and statistics supplied back to the Control Unit.

Learning Module (LM):

The Learning Module (LM) is a software module, which resides in the host computer which is connected to the rest of the hardware modules. The LM accepts configuration details of the server targets, and learning traffic statistics (as captured by the TCM) from the CU, and arrives at model probabilities which will later be used by the Detection Module (DM) for classification of normal traffic/anomalous traffic. It is into this module that the ideas developed as a result of this research work are implemented.

Detection Module (DM):

The Detection Module (DM) is a hardware module, which on input the set of model probabilities of normal traffic per stream (output by LM, and supplied by CU), determines the state of the network (normal or abnormal) in real-time. This module is the operations module for the system. The DM talks with the CU to obtain its inputs and interrupts the CU in case an attack flag is asserted. This will cause further action by the CU on the Mitigation Module (MM).

Mitigation Module (MM):

The Mitigation Module is a hardware module, which allows input traffic unaltered when attack flag is not set in any stream, and when attack flag is set in one or more streams, it filters traffic for the stream (s) on the basis of a white list of source IPs corresponding to the stream.

4.1.2 Use Case Diagram

Use case diagram represents a user's interaction with the system. It represents the dynamic aspect of the system. Figure 4.2 represents a Use Case Diagram for our system.

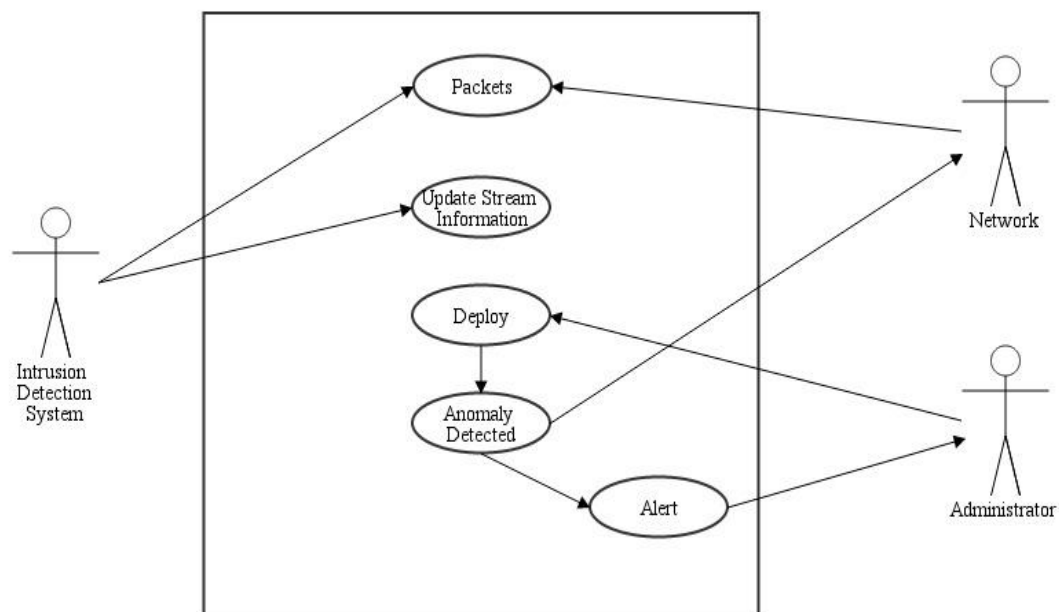


Figure 4.2 Use Case Diagram

The three actors of the system are -

- Intrusion Detection System
- Network
- Administrator

The packets are sent across the network which are accessed by the IDS to update the Stream Information. The updated stream information is used in the testing phase where, based on the incoming traffic, anomalies are detected and are notified to the administrator.

4.1.3 Sequence Diagram

Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It shows object interactions arranged in time sequence. Figure 4.3 represents a Sequence Diagram for our system.

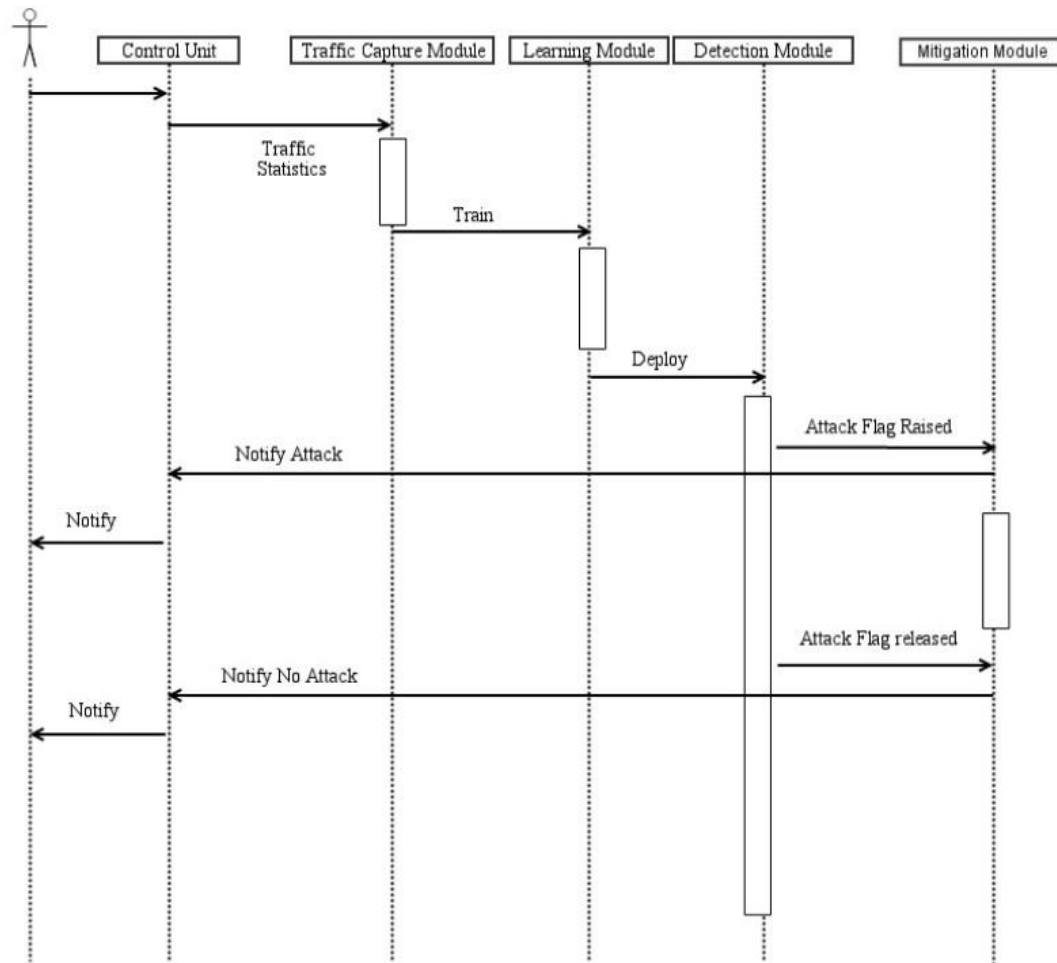


Figure 4.3 Sequence Diagram

The different objects in the system are administrator, CU, TCM, LM, DM and MM. The interaction between each of these objects are denoted by horizontal arrows. The sequence of operation is as follows:

1. The administrator is interacting with the CU to provide input.
2. The input in the form of traffic statistics is given to the TCM.
3. Parameters from the TCM are filtered and provided to the LM to train the model. This

phase is the learning phase.

4. After the learning phase, the model is deployed. The incoming traffic is analyzed by the DM. If any anomaly is detected, an attack flag is raised and sent to the MM.
5. A notification is sent to the CU which then notifies the admin indicating an attack.
6. The MM whitelists the IPs and once the attack flag is released by the DM, the CU is notified that the attack is curbed and in turn informs the system administrator.

4.1.4 Data Flow Diagram

Data Flow Diagram (DFD) is a graphical representation of the flow of data through the system. A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. Figure 4.4 represents a Data Flow Diagram for our system.

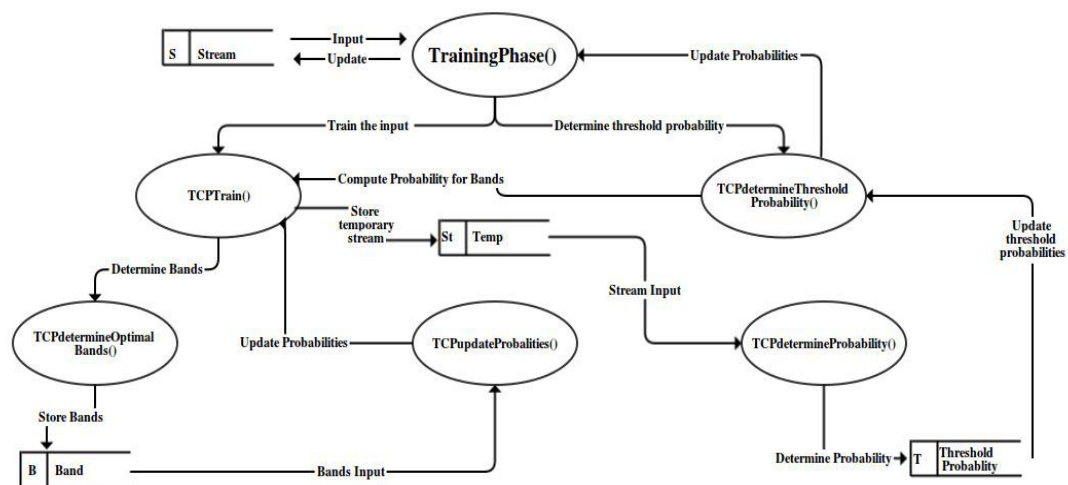


Figure 4.4 Data Flow Diagram

Flow of data in the system is explained as follows. The stream S is passed as an input to the training phase. The input is used to train the model by determining bands. The bands are stored and passed as input to update the probabilities. The updated probabilities are used to determine the threshold probabilities. The threshold probabilities are updated in the stream. Once the training is done, the model is deployed and is used to identify and classify traffic as Normal or Anomalous traffic.

4.1.5 State Diagram

State diagram defines the behavior of the system. It gives an abstract description of the behavior of the system. Figure 4.5 represents a State Diagram for our system.

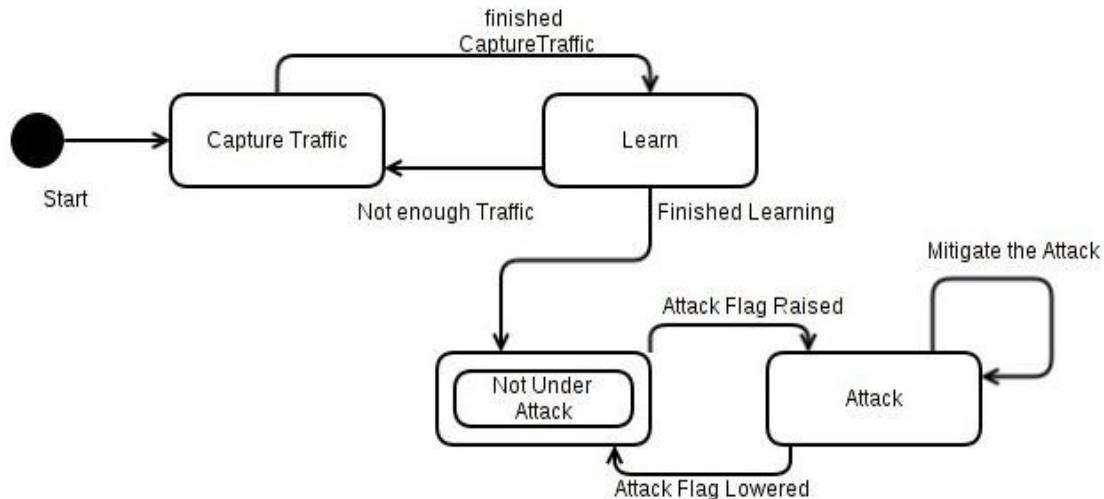


Figure 4.5 State Diagram

The different states are Capture Traffic, Learn, Not Under Attack and Attack. The behavior of the system is explained as follows -

1. Initially, the system is in the **Traffic Capture** state, where it captures the network traffic.
2. Once enough traffic is captured the system moves to the **Learning** state. Here, it learns and determines the normal profile of the network.
3. When the learning process is over, the state is changed to **Not Under Attack**.
4. When an anomaly is detected, the attack flag is raised and the state changes to **Attack** state.
5. While in the Attack state, the system tries to mitigates the attack.
6. After successful mitigation, the attack flag is lowered, and the state is restored to **Not Under Attack**.

CHAPTER 5
IMPLEMENTATION

IMPLEMENTATION

This chapter discusses the various implementation details like information about the programming language used, libraries incorporated and the various procedures involved in the operation of this project.

5.1 Language Used for Implementation

The core system of the project has been written in Python, whereas the front end contains a mix of Python, HTML and CSS.

5.1.1 Python

Python^[22] is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C. The language provides constructs intended to enable clear programs on both a small and large scale.

Python supports object-oriented, imperative and functional programming. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. Like other dynamic languages, Python is often used as a scripting language, but is also used in a wide range of non-scripting contexts. Using third-party tools, such as Py2exe or Pyinstaller, Python code can be packaged into standalone executable programs. Python interpreters are available for many operating systems.

Python is a multi-paradigm programming language: object-oriented programming and structured programming are fully supported, and there are a number of language features which support functional programming and aspect-oriented programming (including by meta-programming and by magic methods). Many other paradigms are supported using extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. An important feature of Python is dynamic name resolution (late binding), which binds method and variable names during program execution. The core philosophy of the language is summarized by the document

"PEP 20 (The Zen of Python)", which includes aphorisms such as:

- i. Beautiful is better than ugly
- ii. Explicit is better than implicit
- iii. Simple is better than complex
- iv. Complex is better than complicated
- v. Readability counts

Libraries like NumPy, SciPy and Matplotlib allow Python to be used effectively in scientific computing, with specialized libraries such as BioPython and Astropy providing domain-specific functionality.

5.2 Libraries Used for Implementation

The project uses the following libraries, which extend the functionality of Python's standard library.

5.2.1 NumPy

Arrays in Python do not have the same definition as that of in traditional programming languages like C. In C arrays are used to store data having the same data type. Python arrays also called lists are similar to C arrays except that they can be used to store mixed type of data. NumPy is an extension to the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.

NumPy^[23] targets the CPython reference implementation of Python, which is a non-optimizing bytecode compiler/interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy seeks to address this problem by providing multidimensional arrays and functions and operators that operate efficiently on arrays. Thus any algorithm that can be expressed primarily as operations on arrays and matrices can run almost as quickly as the equivalent C code.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink whereas NumPy is intrinsically

integrated with Python, a more modern, complete, and open source programming language. Moreover complementary Python packages are available like SciPy which is a library that adds more MATLAB-like functionality and matplotlib is a plotting package that provides MATLAB-like plotting functionality.

5.2.2 Flask

Flask^[24] is a lightweight web application framework written in Python and based on the Werkzeug WSGI toolkit and Jinja2 template engine. It is BSD licensed. Flask takes the flexible Python programming language and provides a simple template for web development. Once imported into Python, Flask can be used to save time building web application.

Flask is called a micro-framework because it keeps the core simple but extensible. There is no database abstraction layer, form validation, or any other components where third-party libraries already exist to provide common functionality. However, Flask supports extensions, which can add such functionality into an application.

Flask has the following features:

- a) Contains development server and debugger
- b) Integrated support for unit testing
- c) RESTful request dispatching
- d) Uses Jinja2 for templating
- e) Support for secure cookies (client side sessions)
- f) 100% WSGI 1.0 compliant
- g) Unicode-based
- h) Extensive documentation
- i) Extensions available to enhance features desired

In the project Flask's server is used to host the front-end. It listens to HTTP requests and responds to them. Based on the request it calls the various modules of the Naive Bayes Classifier and displays the results as a web page via the HTTP response.

5.2.3 Matplotlib

Matplotlib^[25] is a plotting library for the Python programming language and its NumPy numerical mathematics extension. It provides an object-oriented API for embedding

plots into applications. The pylab interface makes matplotlib easy to learn for experienced MATLAB users, making it a viable alternative to MATLAB as a teaching tool for numerical mathematics and signal processing.

Some of the advantages of the combination of Python, NumPy and matplotlib over MATLAB include:

- i. Based on Python, a full-featured modern object-oriented programming language suitable for large-scale software development
- ii. Free and open source
- iii. Native SVG support

5.3 Procedure Description

The procedure description for various algorithms are mentioned in this section.

5.3.1 Determine Optimal Bands Procedure

This procedure groups the events into bands. The band formation is based on the clustering algorithm.

Input: Event array A, window size for stream N, number of bands K.

Output: 2-dimensional array band, containing indices of per group elements in A.

5.3.2 Update Probabilities Procedure

This procedure computes and updates the probability based on the events.

Input: TCP Stream S, 2-D array band containing band indices of groups of A, number of bands K, Index of observable type L

Output: Probabilities of all events updated into corresponding probability array W index in stream S

5.3.3 Train Procedure

This procedure trains the Naive Bayes Classifier using the TCP flags retrieved from the dataset.

Input: Stream S, Flag Array TF, Window size for stream N, number of bands K.

Output: Updated Stream S with learnt probabilities.

5.3.4 Train Phase Procedure

This procedure initiates training. It calls the training module that trains the NBC against the training dataset. It also calls the module that determines the threshold probability.

Input: Window size N , number of bands K , group count, error proportion t and the filename containing the dataset.

Output: Updated Stream S with learnt probabilities and threshold probability for stream.

5.3.5 Determine Probability Procedure

Probability computation involves computing statistics of a window and determining the probability with which the window would have occurred in the training phase.

Input: Stream with NB probabilities S , packet window W

Output: Probability P of window W

5.3.6 Determine Threshold Probability Procedure

This module computes the threshold probability which will be used to classify if a particular window is either an attack window or not

Input: Stream S , Window size for stream N , group count, flag array

Output: Threshold probability P_t for S

5.3.7 Deploy Procedure

This module tests the input traffic and computes if a window is an attack window or not.

Input: Stream S , filename, Abnormal Window Count (AWC)

Output: State of the system and the window

5.3.8 Parse Procedure

This module reads from the input file. It also groups packets into the 6 groups and plots the packet count graph

Input: Name of the file that contains the flag information

Output: Flag array

CHAPTER 6
RESULTS

RESULTS

The screenshots of the Naive Bayes Classifier and the accuracy results of the designed system are outlined below.

6.1 Screenshots

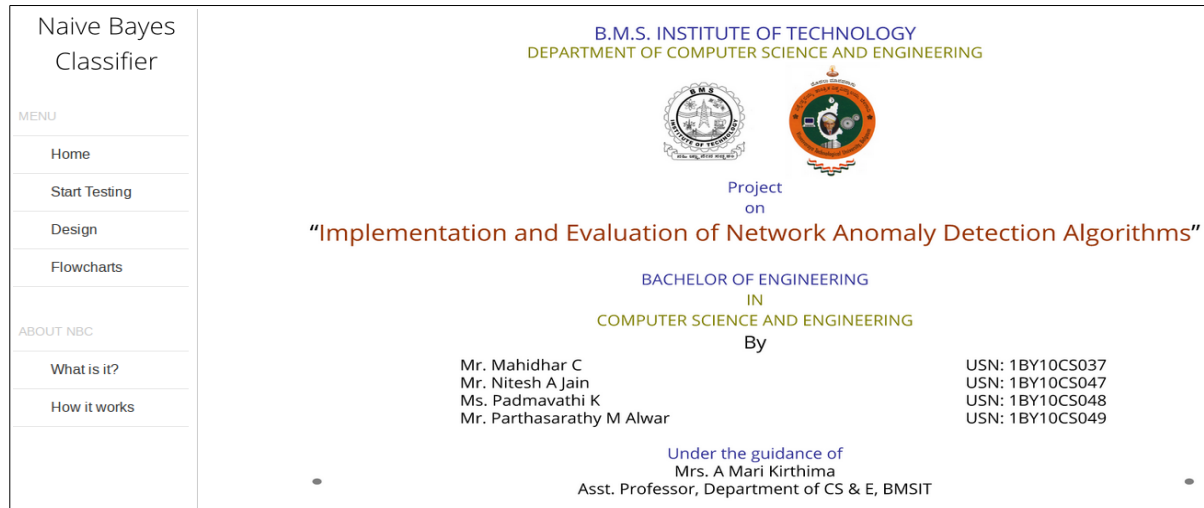


Figure 6.1 Home Page

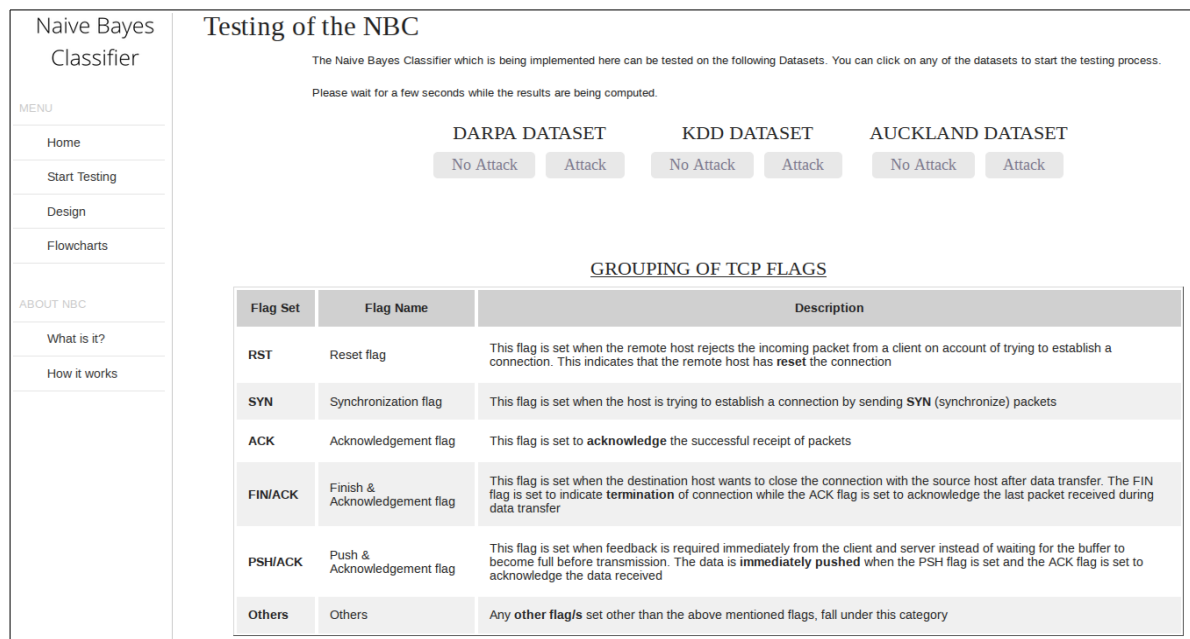


Figure 6.2 Testing Page

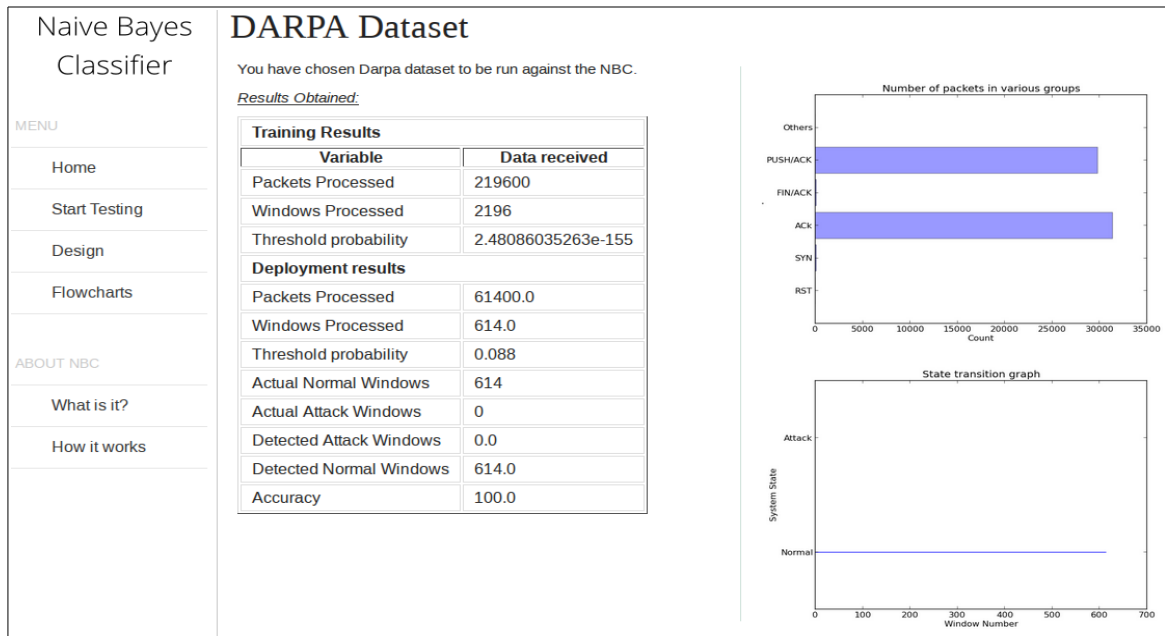


Figure 6.3 DARPA No Attack Results

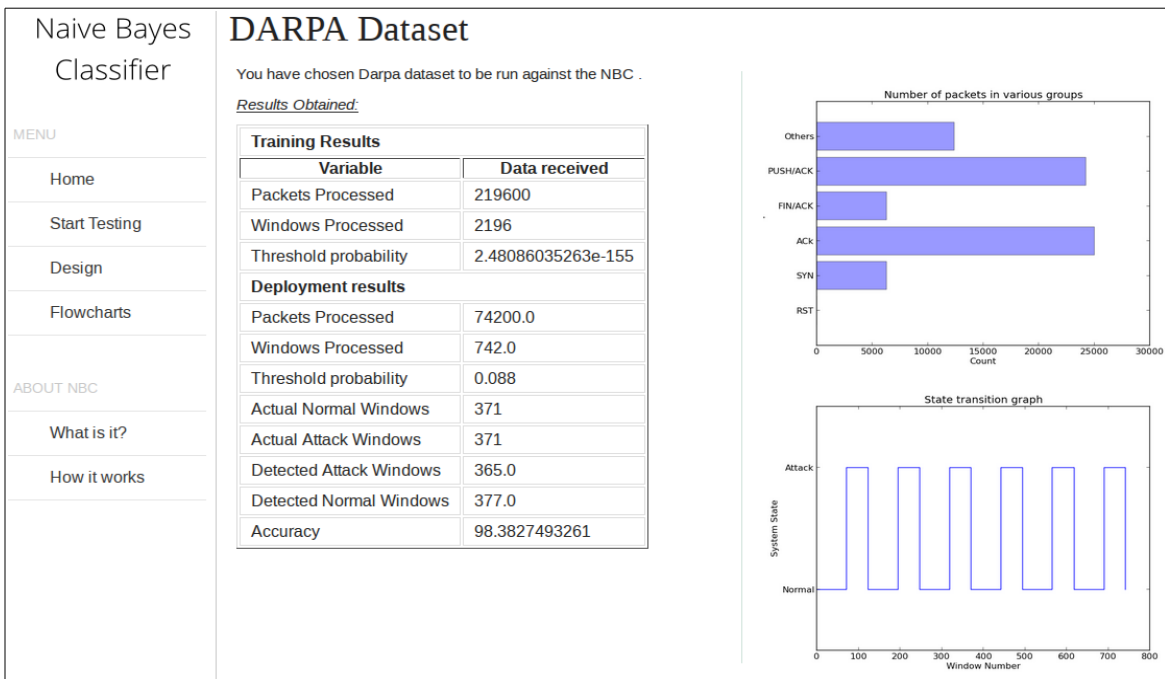


Figure 6.4 DARPA Attack Results

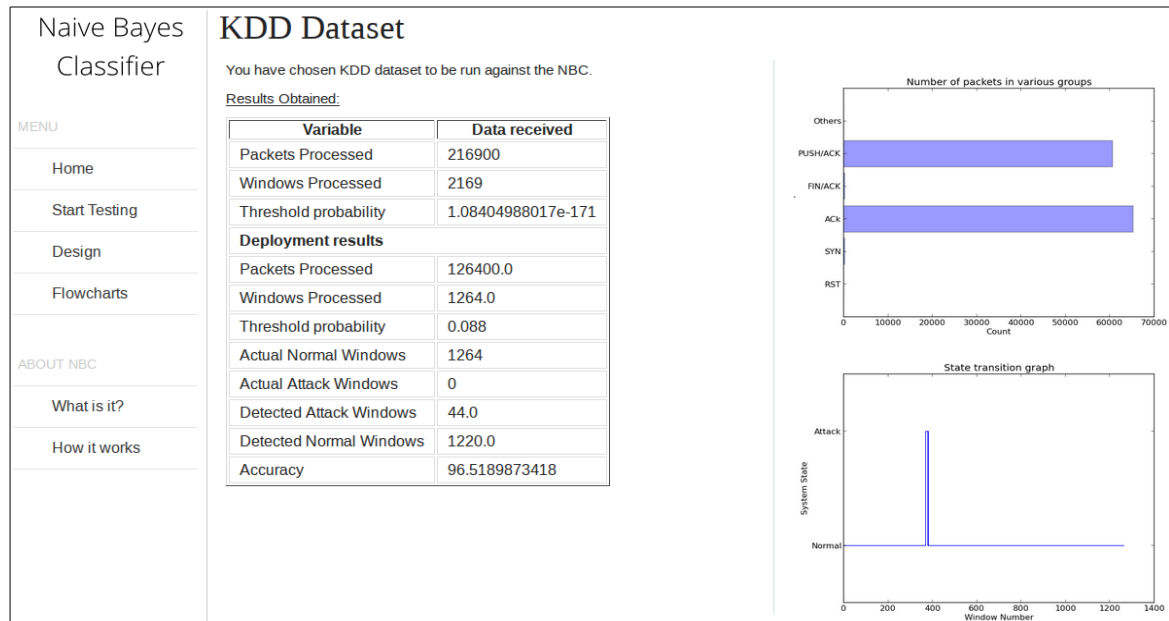


Figure 6.5 KDD No Attack Results

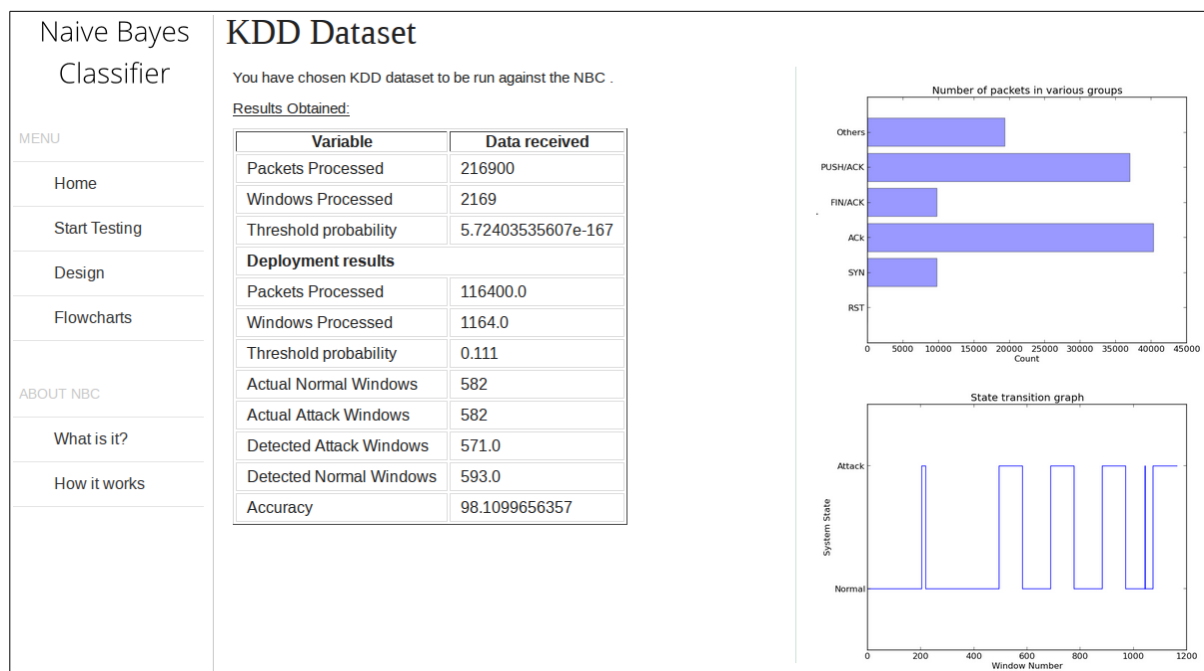


Figure 6.6 KDD Attack Results

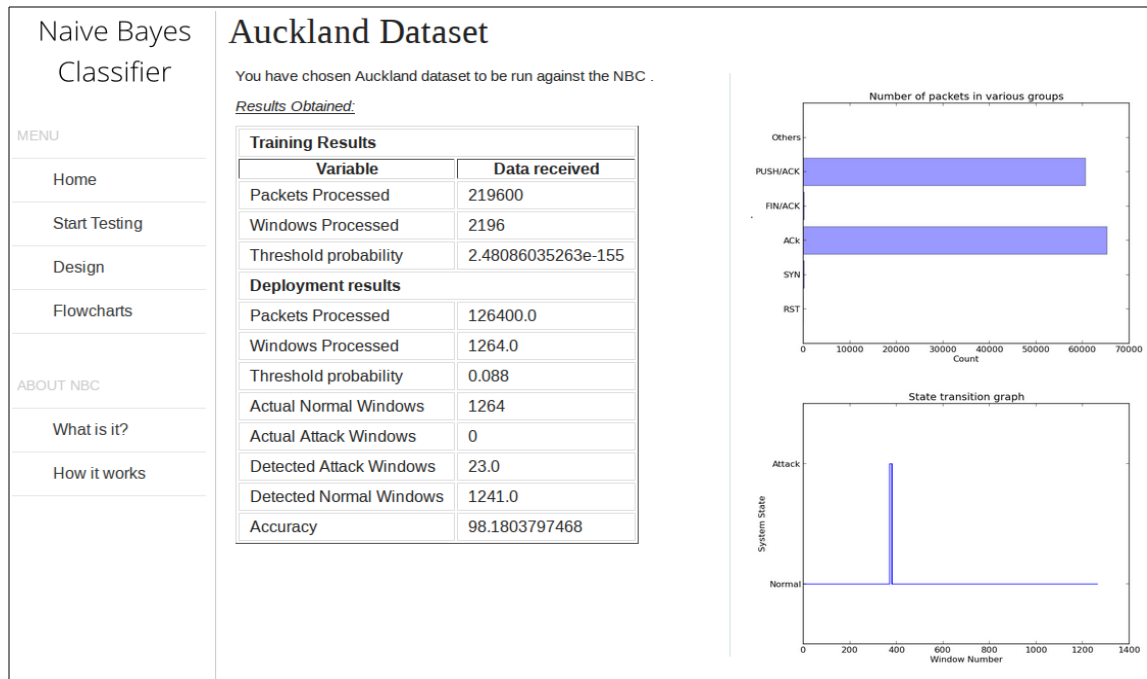


Figure 6.7 Auckland No Attack Results

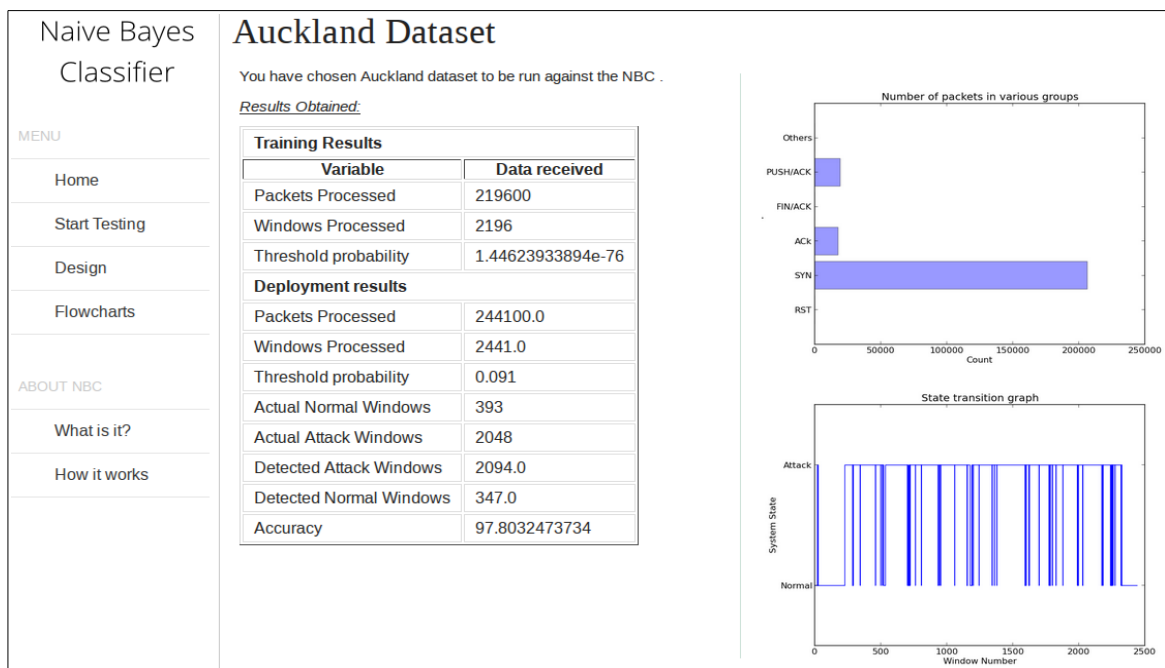


Figure 6.8 Auckland Attack Results

6.2 Results Obtained

The Naive Bayes classifier was tested against the DARPA, KDD and Auckland -II datasets. The formula to calculate the accuracy is as follows

$$Accuracy = \frac{TP + TN}{P + N}$$

where,

TP - True Positives

TN - True Negatives

P - Number of Attack Windows

N - Number of Normal Windows

Each dataset had two types of traffic – **Clean** traffic and **Mixed** traffic. The clean traffic has no attacks whereas the mixed traffic has both clean and attack packets. Using the above formula the accuracies for each dataset are as follows:

Dataset	Accuracy
DARPA	
No Attack	100%
Attack	98.38%
KDD	
No Attack	96.52%
Attack	98.11%
Auckland	
No Attack	98.18%
Attack	97.80%

Table 6.1 Results for various datasets

CHAPTER 7
CONCLUSION

CONCLUSION

The proposed mechanism eliminates the need for a Signature Based System of detection. The advantage of not depending on a Signature Based System is that we can easily detect novelty attacks. The proposed system protects the network by constantly monitoring the TCP flags and checking for outliers. It alerts the administrator whenever in an attack state. The Naive Bayes Classifier which is implemented in this project is a “site-based”, real-time system which is simplistic in its design and deployment. Successful testing on standard datasets such as the DARPA dataset, KDD dataset and the AUCKLAND-II dataset with accuracies over 95% has proved the project to be a success. The proposed mechanism can also be applied for other flag based protocols like the User Datagram Protocol. The Naive Bayes Classifier has successfully detected and alerted the administrator of a Distributed Denial of Service attack.

The accuracy of the system is calculated based on False Positives (FP) and False Negatives when tested against “No Attack” files and “Attack” files. The system has two operational phases; 1. Training Phase and 2. Testing Phase. A normal profile of the network is generated during the Training Phase. The Testing Phase is then started by using an attack file. Based on the threshold probability obtained during the training phase, incoming packets are classified as an attack or a non-attack packet. Finally, we have tested the system with three standard datasets, i.e DARPA, KDD and AUCKLAND-II with satisfactory results.

The proposed system has the following outcomes:

- A systems approach for DoS and DDoS detection at target using a Naive Bayes Classifier for TCP, with a design engineered for real time use and implement-ability.
- Design considerations taking into mind practical user problems with machine learning techniques.
- Light weight detection algorithm, with a note on processing latency and reaction time.

CHAPTER 8

CHALLENGES AND FUTURE

WORK

CHALLENGES AND FUTURE WORK

The model is simple and light weight in terms of detection, but while the design has been made keeping in mind flooding attacks (which form most of attacks carried out to the day), input parameters may have to be further fine tuned in order to be detecting convincingly a broader class of DoS attacks. This may involve including payload features as well. Also, at a user level, making the choice of some inputs to the system is very difficult even for an expert user. For example, it may be extremely difficult for even a seasoned network administrator to determine the right choice for the proportion of abnormal traffic in the training input, unless manual checking is done over the entire dataset.

Further more, the scalability of the model to very high volume servers has to be examined. For example, it is quite possible to see a window full of SYN packets with a high volume server, while the signature for SYN flooding attacks still remain the same. Although it may look like increasing the size of the window may solve this problem, this has to be carefully examined before a conclusion could be given. This essentially calls for testing the system with variety of traffic at various volumes.

Outside of the detection problem itself, a few other interesting areas of research surface when the intention is to build a fool proof DDoS protection system. Some of them are the following:

- Extending to other protocols. While we believe that a similar framework will work for other protocols, the research could be extended to include other protocols in the Internet protocol stack. Examples of such protocols include gateway level protocols, or application level protocols.
- Research on sound mitigation techniques. It is also to be noted that these techniques in many cases may have to be applied in-line, which calls for extremely efficient methods to handle attacks. Possible mitigation techniques include graceful degradation, cryptographic methods to stateless monitoring and making launching further attacks tougher.

APPENDIX

APPENDIX

Tuning System Parameters

After the choice of the grouping algorithm has been made, the next task is to tune system parameters in order to maximize performance. It is seen that some of these parameters could be generalized across user profiles, while the others revolve around the user traffic profile.

The following are the important system parameters in consideration:

1. Window Size (N);
2. Number of bands (K); and
3. Error Percent (t).

The optimal values for each of the above parameters were empirically determined, based on the following factors:

- Maximum Performance – lesser number of false alarms.
- Light-weight detection and quicker response – decrease response time.

Experiments on N and K

The objective of the experiment is to determine the ideal values of $\{N, K, t\}$ at which the performance will be maximal. It is obvious that N and K are related to each other – K refers to the amount of compression allowed on the total number of events in the event space, which in turn, is determined by the value of N , for good performance. A higher value of N will result in more probabilities, which may (or may not) call for an increase/decrease in the value of K in order to accommodate performance requirements. But it is observed that although N and K are related to each other, the value of the two tuple $\{N, K\}$ need not be tied to a particular user traffic profile, and can be generalized across user sites. However, N and K should be chosen in a way that the relation between N and K is still preserved across all user sites. On the other hand, t by definition is specific to the user traffic considered. Hence, t may have to be varied from site to site, and involves trial-and-error.

The performance is mentioned in terms of False Negatives (FN) or True Positives (TP). The parameter considered for measurement is the True Positive Rate (TPR), the ratio of

True Positive windows to the total number of Attack windows. The approach taken to finding the right $\{N, K, t\}$ is as follows:

- The value of t is kept at a constant, and the system performance for different $\{N, K\}$ combinations is observed to derive the relation between N and K .
- The value of t is chosen in a way that it exceeds the maximum value of t for all datasets available for experimentation. This is based on the fact that the system performance saturates at a maximum, beyond a value of t for any dataset. It is seen that $t = 1\%$ is a suitable value for the experiment. Precise values of t will be determined for every dataset separately.
- The varying values of N and K are taken from the following discrete values: $N' = \{50, 100, 150, 200\}$ and $K' = \{5, 10, 15, 20, 30, 40\}$.
- A performance matrix, which describes the performance (TPR) of the system at each possible $\{N, K\}$ combination from N' and K' is arrived at. The resultant matrix is a 6×4 matrix.
- The matrix is analyzed to determine $\{N, K\}$ values for which performance is maximum. Bounds are determined on the values for which the system performance is guaranteed to get maximized. This will also bring to light the relation between N and K , if it is quantifiable.
- Based on the above observation, choose good values of N , K , and determine the exact value of t for every data set.

Experiment

The experiment is the same as the general experiment approach – for a chosen value of $\{N, K\}$, train with normal traffic and pass the training traffic into the testing phase in order to determine the number of False Positives. Train with normal traffic and pass the attack traffic into the testing phase, in order to determine the number of False Negatives. Determine TPR for each $\{N, K\}$ and populate a cell of the performance matrix.

Results and Interpretations

The experiment was conducted (with $t = 1\%$) for the above said values of K for two data sets – DARPA data set and the AUCK-II data set. The performance matrices for the two

data sets are indicated in Figures A1 and A2.

The following are the important observations:

- The performance matrix can be analyzed in two ways – Column wise analysis is done when N is fixed and a suitable K has to be arrived at, and row wise analysis to be done when the right N is chosen for fixed K. The former is more useful.
- The matrices show an important fact discussed above – for a given N , performance saturates after a particular value of K is reached, and addition of a few more bands does not improve the performance any more.
- The well tagged DARPA data set appears to have ideal properties – for even higher values of N , performance starts converging from small values of K. In essence, it means that the data set contains very few event occurrences for all N , which could be captured with fewer bands.
- From the results, it can be seen that when K is upper bound by $N*5$, the system performance peaks for almost all values of N and K in all three data sets.

K	N			
	50	100	150	200
5	0.83%	0.84%	0.9%	0.91%
10	0.84%	0.85%	0.9%	0.85%
15	89.58%	89.57%	0.86%	0.86%
20	89.58%	99.99%	0.87%	0.9%
30	99.99%	100%	100%	89.6%
40	99.99%	100%	100%	100%

K	N			
	50	100	150	200
5	68.14%	68.16%	68.16%	68.18%
10	100%	100%	68.17%	100%
15	100%	100%	100%	100%
20	100%	100%	100%	100%
30	100%	100%	100%	100%
40	100%	100%	100%	100%

Figure A.1: Performance Matrix for AUCK-II

Figure A.2: Performance Matrix for DARPA

Conclusion

Hence, it is hypothesized that K should be at least $N*5$ for the system to achieve good performance levels. The absence of constant values for K for any N is expected to create a few space overheads in hardware implementation (of detection), since N is expected to vary across multiple user sites, depending on their traffic profiles. This can be worked around by

implementing detection assuming a reasonably large value of N , and choosing to keep $N/5$ bands for model probabilities. This space can be dynamically handled for all values of N lesser than the chosen value. This may amount to wastage of reasonably small space, which can be easily handled by modern hardware.

Experiments on Error Percent 't'

The objective of this experiment is to determine precise values of t for different data sets. The previous experiment hypothesized the relation between N and K , and experiments on t preserve this relation. For the purpose of these experiments, N is assumed to be 100, and K , according to our hypothesis is chosen to be 20. With these values fixed, experiment is conducted by varying the value of t until a saturation point in performance, in terms of True Positives is reached. Since it has already been explained that increasing values of t will increase false positives, we are concerned more about the attack traffic and how efficient the system is with respect to classifying attack windows as attack, with a given t .

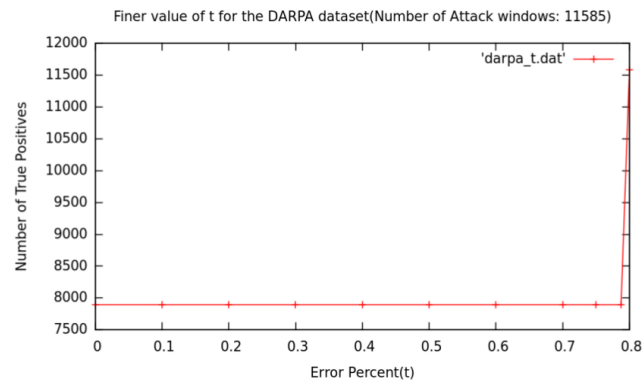
Experiment

The following procedure was used to conduct the experiment:

1. Training phase: Train with normal traffic.
2. Testing phase: Pass as input the attack traffic, and estimate the True Positives for every t .
3. Repeat the experiment for different values of t , and find out the value of t at which the number of True Positives converges to a maximum. This value is the actual value for the error percent t .

Results and Interpretations

The results of the experiment are graphically described in Figure A3 and A4. The value of t differs from site to site. The number of bands K determines the level of accuracy of modeling, and should be carefully chosen during system design. N and K are related to each other.

**Figure A.3 Experiments to determine t (DARPA)****Figure A.4 Experiments to determine t (AUCK-II)**

BIBLIOGRAPHY

BIBLIOGRAPHY

- [1] Simmonds, A; Sandilands, P; van Ekert, L (2004), "An Ontology for Network Security Attacks", Lecture Notes in Computer Science 3285: 317–323
- [2] Scarfone, Karen; Mell, Peter (February 2007), "Guide to Intrusion Detection and Prevention Systems (IDPS)", Computer Security Resource Center (National Institute of Standards and Technology) (800–94), Retrieved 1 January 2010
- [3] ISO/IEC 27000:2009 from ISO, via their ITTF web site
- [4] Marina Thottan, Guanglei Liu, Chuanyi Ji. “Anomaly Detection Approaches for Communication Networks”
- [5] Langely, Pat; Simon, Herbert; “Applications of Machine Learning and Rule Induction” 19950328164 Institute for the Study of Learning and Expertise Technical Report 95-1 February 15, 1995
- [6] Nitin; Mattord, verma (2008), Principles of Information Security Course Technology, pp 290–301, ISBN 978-1-4239-0177-8
- [7] Kevin P Murphy, “Machine Learning, A Probablistic Perspective”, The MIT Press, Cambridge, Massachusetts, London, England
- [8] Mansi Gyanchandani, J.L. Rana, R.N. Yadav, “Taxonomy of Anomaly Based Intrusion Detection System: A Review”, International Journal of Scientific and Research Publications, Volume 2, Issue 12, December 2012
- [9] Feller, W. (1971), “Introduction to Probability Theory and Its Applications”, Vol II (2nd edition), Wiley. ISBN 0-471-25709-5 (pages 9 and 20)
- [10] Rangadurai Karthick R, Hattiwale V.P and Ravindran B, “Adaptive Network Intrusion Detection System Using a Hybrid Approach”, IEEE paper, 2012
- [11] Paul Hick, Emile Aben, kc claffy, Josh Polterock, The CAIDA “DDoS Attack 2007” Dataset, <http://www.caida.org/data/passive/ddos-20070804dataset.xml>
- [12] DARPA intrusion detection evaluation dataset, <http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html>
- [13] Davide Ariu, Giorgio Giacinto and Roberto Perdisci, “Sensing attacks in Computers Networks with Hidden Markov Models”
- [14] Zhang, Harry; “The optimality of Naïve Bayes”, American Association for Artificial Intelligence 2004

- [15] Mitchell, M Tom; “Machine Learning”, McGraw-Hill, ISBN: 0070428077
- [16] Vijayasarathy, R; Ravindran, B ; Raghavan, S.V; “A system approach to network modeling for DDoS detection using a Naive Bayesian classifier”, COMSNETS, 2011
- [17] Kok-Chin Khor; Choo-Yee Ting; Somnuk-Phon Amnuaisuk; “From Feature Selection to Building of Bayesian Classifiers: A Network Intrusion Detection Perspective”
- [18] Dewan Md. Farid; Nouria Harbi; Mohammad Zahidur Rahman, “Combining Naïve Bayes and decision tree for adaptive Intrusion Detection”
- [19] DARPA Dataset, 1999,
<http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/1999data.html>
- [20] KDD Dataset, Cup 1999 Data, <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>
- [21] Auckland Dataset, WITS: Auckland II, http://wand.net.nz/wits/auck/2/auckland_ii.php
- [22] Python Documentation, <https://docs.python.org/2/>
- [23] Numpy Documentation, <http://www.scipy.org/scipylib/download.html>
- [24] Flask Documentation, <http://flask.pocoo.org/docs/installation/>
- [25] Matplotlib Documentation, <http://matplotlib.org/1.3.1/users/installing.html>