# A Hybrid Approach for Intrusion Detection in Streaming Data using Data Mining Techniques

**A DISSERTATION**

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

*Of*

**MASTER OF TECHNOLOGY**

*In*

**COMPUTER SCIENCE & ENGINEERING**

**Submitted by**
***Swati Maheshwari***

**COMPUTER SCIENCE AND ENGINEERING DEPARTMENT**
**INDIAN INSTITUTE OF TECHNOLOGY**
**ROORKEE – 247667**
**MAY 2016**

## CANDIDATE’S DECLARATION

I hereby declare that the work presented in this dissertation “Hybrid Approach for Intrusion Detection in Streaming Data using Data Mining techniques” towards the fulfillment of the requirements for award of the degree of Master of Technology in Computer Science, submitted to the Department of Computer Science and Engineering, Indian Institute of Technology- Roorkee, India is an authentic record of my own work carried out during June 2015 to May 2016 under the guidance of Dr. Durga Toshniwal, Assistant Professor, Department of Computer Science and Engineering, Indian Institute of Technology- Roorkee.

I have not submitted the content presented in this dissertation for the award of any other degree of this or any other institute.

Date: ……………….

Place: Roorkee

**(Swati Maheshwari)**

## CERTIFICATE

This is to certify that the statement made by the candidate in declaration is correct to the best of my knowledge and belief.

Date: ……………….

Place: Roorkee

**(Dr. Durga Toshniwal)**

Associate Professor

Department of Computer Science and Engineering

Indian Institute of Technology, Roorkee

# ACKNOWLEDGEMENTS

First and foremost, I would like to express my deep sense of gratitude and regards to my research supervisor **Dr. Durga Toshniwal,** Associate Professor, Dept. of Computer Science & Engineering, Indian Institute of Technology, Roorkee, for her trust in my work, her invaluable guidance, encouraging attitude and assistance throughout the course of dissertation work.

I would also like to thank my parents, my brother Sharad Maheshwari for providing me constant support, encouragement and care.

Lastly, I thank almighty for the wisdom and perseverance that he bestowed upon me during my research.

**SWATI MAHESHWARI**

# ABSTRACT

With the increase in the dependence on Internet for transactions and communications, ensuring security has become a necessity. Intrusion Detection System (IDS) is one of the important software or hardware devices in security architecture that is used to ensure a safe communication between organizations. Its capability to monitor the complete packet (promiscuously) makes it a good complement tool to other security tools like firewall, anti-virus etc.

Data stream mining is an active research area that has recently emerged to discover knowledge from large amounts of continuously generated data. In this dissertation, we have focused on data streams and data mining techniques to be able to detect attacks on the fly, to learn new attacks for better accuracy in prediction and to generate alarms for the same.

The dissertation has proposed a hybrid approach to efficiently detect intrusions in the network in real time with high accuracy. To improve the accuracy, both the intrusion detection approaches viz. Anomaly and Misuse detection has been used (in sequence). To improve the detection rate, we have used Decision Tree for creating the signatures for the attack data and LERAD's ensemble for anomaly detection in the streaming data. LERAD is a rule-based algorithm for intrusion detection and falls under the category of Anomaly Detection approach of Intrusion Detection. We have implemented the algorithm using Python and its libraries and have tested our results on NSL-KDD dataset, benchmark in intrusion detection field by simulating the datasets as streaming data. All the major challenges with Streaming Data viz. Infinite Length, Concept Drift and Concept Evolution have been addressed. The report has shown incremental improvements in the accuracy and compared the proposed framework with the techniques used.

# CONTENTS

# List of Figures

# List of Tables

# 1 INTRODUCTION

Internet has become a part of our daily life. We do everything online, our computers, laptops and smartphones have become an extension of ourselves so it has become very important to ensure confidentiality, integrity and availability of the resources on Internet.

For ensuring network security, there exist a number of tools, the most famous of them are, firewall, IDS and IPS. All these security tools complement each other to increase the security and protect the integrity.

Firewall allows us to control network traffic by policies, which allows particular protocols, machines only to interact and thereby filtering the network traffic from the plausible malicious intruders. The firewall achieve this by analyzing packet headers and taking actions in accordance with the policies based on the source or destination addresses (which include port number as well) or the services.

IDS (Intrusion Detection System), on the other hand, focus on both packet header and payload of the packets and make a decision based on the nature of the packet received. A log is generated based on the IDS findings. IPS (Intrusion Prevention System) is an extension to IDS by attaching an action to the findings by IDS. The device may either generate an alarm, or deny or pass or block the traffic based on the rules set.

In this dissertation, we have focused on how IDS can be used to improve the detection rate for the streaming data. We shall be discussing more of Intrusion Detection System and Data Streams and their challenges in the following sections.

## 1.1 INTRUSION DETECTION SYSTEM

As explained, IDS is a security device that monitors the network traffic or system logs to enhance the network security of the system or security of an endpoint/ host. It should be able to detect an attack so that an appropriate action can be taken and no serious damage could be done to the system. Thus, IDS is a simple feedback system where it collects information from a system, monitors and inspects it to find intrusions or suspicious threats to the system, and trigger the configured response, as per the policy, to the system back.

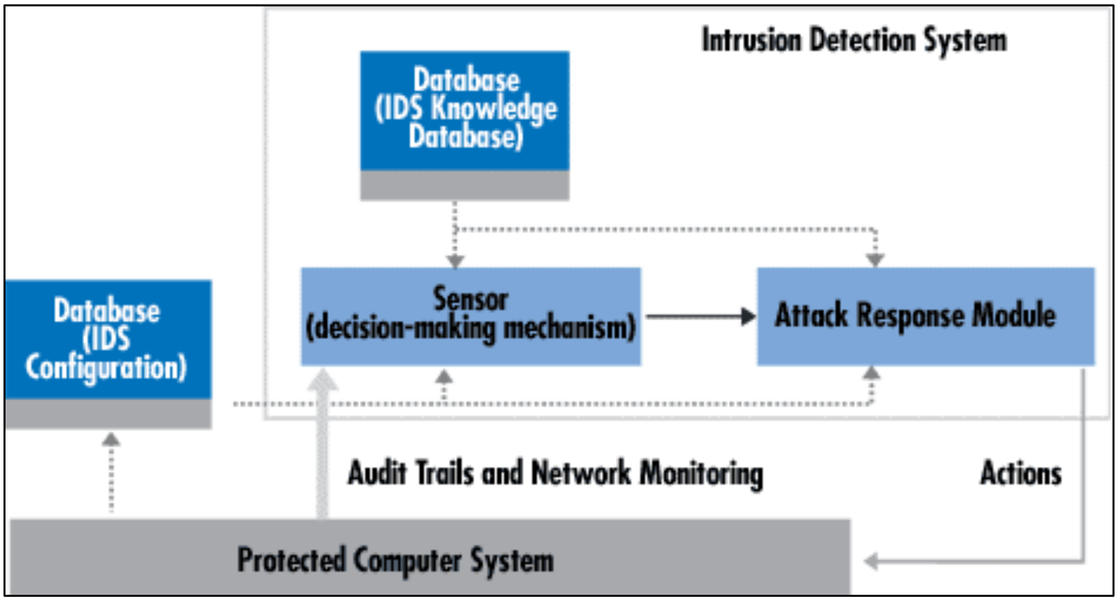

Intrusion Detection System
Database (IDS Knowledge Database)
Database (IDS Configuration)
Sensor (decision-making mechanism)
Attack Response Module
Audit Trails and Network Monitoring
Actions
Protected Computer System

*Figure 1: Simple Intrusion Detection System [4]*

The main component as shown in Figure 1 of a Simple Intrusion Detection System [4] is 'Sensor/ Analyzer' that collects data in the raw format from the system either from the system log for ensuring host based security or from network packets for network security. Sensors, then, analyze information on the basis of the information it gets from the 'IDS Knowledge Database' which contains information about the attacks signatures and normal profile and the value of the important parameters, like threshold or batch size, etc. On the basis of the information, it makes a decision as per the policy in 'IDS Configuration Module', which is then sent to 'Attack Response Module' to take appropriate action on the system. For ex. to block a source, or request on a specific port. Also, information can be either be stored in the sensor in case of static data or be processed on the fly in case of data streams.

IDS can be classified on the basis of multiple parameters. Following are the classifications of IDS.

- On the basis of source of information or where the device is placed. IDS can either be placed along a network segment or be installed in a host.
  - ➢ **NETWORK BASED IDS (NIDS)**

    It is responsible for finding network attacks. It consists of a sensor with a Network Interface Card (NIC) operating in **promiscuous mode** (and thereby serves as a packet sniffer) and is responsible for analyzing the traffic and generating an alarm if an attack is observed. As shown in Fig. 2, NIDS System collects information from the router for all the hosts in the network and take decision which may be for ex. to alter the policies in the firewall.

- **HOST BASED IDS (HIDS)**

  HIDS is entrusted to protect the host i.e. endpoint in the network. The attacks may be directed to a single host that may not even be connected to a network. Capabilities of an HIDS generally include log analysis, monitoring dynamic system behavior (similar to anti-virus packages). For ex, an HIDS may detect an application changing the system database passwords it is not intended to. As shown in Fig 2, HIDS is installed in each host like antivirus software and the information collected may be sent to update the IDS Server.

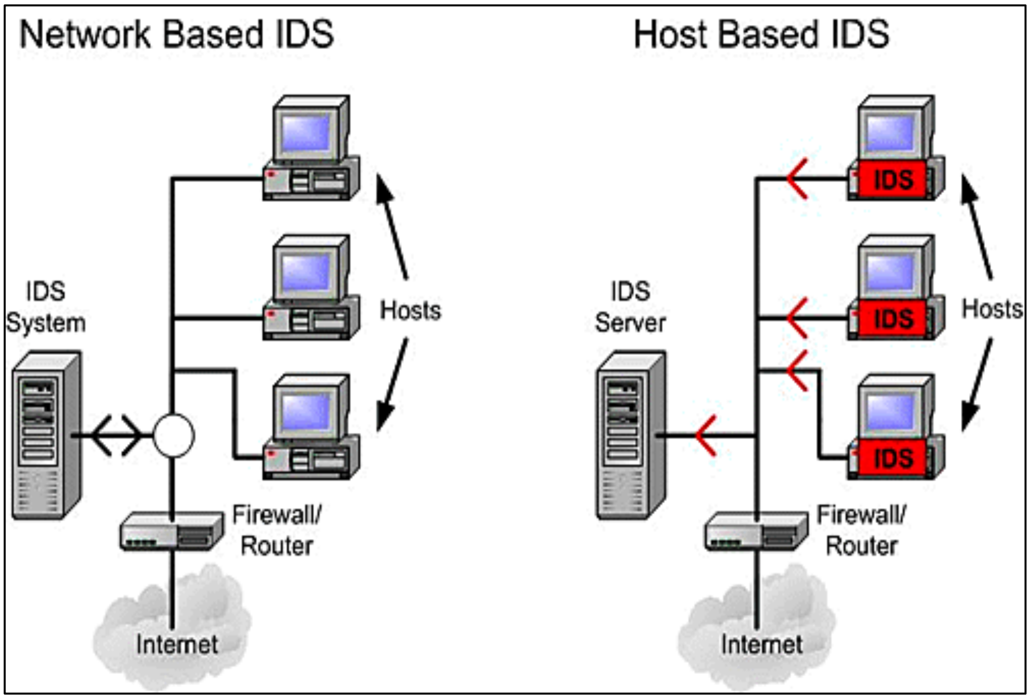


*Figure 2: A comparison of NIDS and HIDS*

We, in the dissertation, have covered algorithms that focus on NIDS and have proposed algorithm for the same.

- On the basis of detection approach. IDS can either create signature for attack or normal data to detect anomalies in the system.

- **MISUSE DETECTION APPROACH:** The Misuse approach detects attacks by creating a signature of the known attacks and checking the packet against the created profiles to detect intrusions. While, the approach is well suited to detect known attacks, it doesn't work well on detection of the unknown attacks or day zero attacks. The new attacks that fall under the signature of the known attacks can be detected though. Since, the definitions of the attacks can be very crisp and clear, this approach sees a lesser number of false positives i.e. attacks being wrongly labeled as normal traffic. Also, with each new attack the model should be updated to reflect them for further traffic.

- **ANOMALY DETECTION APPROACH:** Unlike the previous approach, Anomaly Detection approach creates signatures for the normal data and any deviation from this signature is considered an anomaly. This approach has its own pros and cons as well. It can detect attacks, immaterial if it is new or a known one, assuming that the attacks and normal packets shall have different profiles. But, since the signature of the normal data can be very wide and may not have a very crisp boundary, this approach can generate a number of false alarms.

A difference between the two is summarized in the following table 1

*Table 1: Summary of difference between detection approaches*

| Detection Approach | Pros | Cons |
|---|---|---|
| Misuse/ Signature Detection Approach | 1. High detection rate for previously known attacks.<br>2. Low computational costs | 1. Need to be updated every time with new attacks.<br>2. High false alarm rate for unknown attacks.<br>3. Cannot detect new or variants of the known attacks. |
| Anomaly Detection Approach | 1. Can detect new attacks. | 1. High false positive rates.<br>2. Require larger dataset to create better definition of normal behaviour |

We, in the dissertation, have read and covered algorithms for both the approaches and have proposed a hybrid algorithm that consists of both Anomaly Detection and Misuse Detection Module and thereby brings out the pros of both of them.

- On the basis of action taken by an IDS, it can be classified as Active or Passive IDS
  - **PASSIVE IDS**

    Passive IDS is supposed to only monitor and analyze the traffic, do pattern checking as per the policies set in the database and generate an alarm if an alarm is notices. The decision is left to human administrator to either block, or the let the traffic pass.
  - **ACTIVE IDS**

    Active IDS, also known as IPS (Intrusion Prevention System), apart from monitoring and sensing an attack, is also responsible to take the configured action and thereby provide a real time corrective action. We can have a semi-active system in which some of the alarms generated can be handled by IDS by the configured rules engine and rest attacks detected are sent to human administrator.

### *1.1.1 Types of IDS Alerts*

IDS can generate 4 kinds of alerts viz.

- **FALSE POSITIVE**: IDS generated an alert but there wasn't really an intrusion
- **TRUE POSITIVE**: IDS generated a valid alert i.e. there indeed was an intrusion.
- **FALSE NEGATIVE**: IDS didn't generate an alert when there actually was an intrusion.
- **TRUE NEGATIVE**: IDS labeled the traffic as normal and was correct in labeling.

Thus, to improve accuracy, IDS should focus on improving true positives and true negatives.

## 1.2 DATA ASPECTS

When considering the time aspects of the data, we can distinguish data and thereby IDS in 2 types i.e. if it data collected is either static or streaming. As the name suggests, static data is the one that is stored in disk and can be operated upon as a whole. There is no dynamicity associated with it. On the other hand is streaming data that is a continuous and changing sequence of data and is continuously arriving and needs real time analytics and predictions. The probability distribution of the data may change with time and therefore, the models created using traditional algorithms built using stored static data may soon become irrelevant with time.

Processing of Data Streams has inherent challenges. Following are the main problem in handling data streams:

- Infinite Length: Owing to the continuous stream of data and infinite flow, the data cannot be stored and processed as a single unit. Apart from storage issues, data stream algorithms also need to consider the number of passes on data to provide speedy processing of the coming flow of data.
- Concept Drift: The incoming data is dynamic in nature and therefore, the definition of the models can change with time and should be updates. Thus, data stream algorithms in addition to traditional algorithms should be able to update model to reflect changes in the concept.
- Concept/ Feature Evolution: This refers to the introduction of new classes in the concept. The algorithm should be able to update to address the new concepts created.

Our dissertation has proposed a framework for handling intrusions in data streams using hybrid of the detection approaches, discussed earlier.

## 1.3 MOTIVATION

As stated, Internet has become a part of daily life and with its growth has increased the malware activities and security breaches. It has necessitated the need for device called IDS i.e. Intrusion Detection System for resisting the external attacks and for ensuring confidentiality, integrity and availability of the resources on Internet.

Rules for the attack detection can be entered manually by restricting access to specific hosts and services. But their manual update is very difficult and so is their manual prediction. It can take a lot of human efforts and time. Thus, network security is a great problem of Data Mining algorithms and is an active topic of research as well.

Also, IDS when placed in the network should be able to detect attacks on the fly by generating an alarm for the seen attack signatures and also for the new attacks that is important because of the increase in the number of online threats and zero day attacks.

Also, with the availability of huge network data, it is imperative to be able to handle the size of the data and also updating the models to reflect the most recent definitions of attacks ad normal data.

Despite Intrusion Detection been an old and a popular field, because of the inherent challenges, there is a scope of improvement in terms of accuracy and number of false alarms in prediction, and improvements in time and space complexity.

Some of the challenges intrusion detection suffers from are:

I. Most of the current algorithms are majorly based on anomaly detection approach that result in high false alarms and the ones based on signature detection approach (like SNORT) are unable to detect new attacks and thereby zero day attacks goes missing.

II. The evolving definition of both normal and attack patterns leading to older definitions becoming irrelevant in sometime.

III. In case of data streams, availability of the labeled data is difficult and the system may have to wait for some time before it is able to get the correct labels and be able to learn from the data.

And thus, a need to develop a hybrid based solution to Intrusion Detection in data stream with accuracy comparable to what we can achieve with the same datasets when considered as static data and stored all in one go.

## 1.4 PROBLEM STATEMENT

Most of the data stream algorithms are based on only Anomaly Detection approach to detect Intrusions i.e. they use different data mining techniques to create signatures for the normal data for ex. by clustering [11] or classification with time constraints in [12], etc. Though, with this approach, new attacks can be discovered as well which are more prevalent in real streaming data, anomaly detection approach when used alone tends to have high false positive rate or negative rate resulting into less efficiency in detecting attacks, as more efforts are then put to differentiate between the 2 classes. And therefore, the problem statement of the dissertation is as follows:

*‘To propose a hybrid-based approach (using both detection approaches viz. Anomaly and Misuse modules) to improve the detection rate and to reduce the false predictions for streaming data’.*

The proposed framework shall be tested on NSL-KDD dataset to show its worth and to compare its efficiency parameters with others.

## 1.5 RESEARCH CONTRIBUTION

Following are the contributions of the dissertation:

1. A hybrid model for improving accuracy and false alarm rate in stream data using data mining techniques. The hybrid module contains a Misuse Detection Module and an ensemble of Anomaly Detection modules with majority voting.
2. To avoid pruning of rules in validation phase of LERAD [1], weighted rules have been proposed where weight of a rule is proportional to the importance of the rule.
3. Improvement in accuracy is step-wise shown from the single intrusion detection to their ensemble and their application in data streams.
4. Tested the proposed framework on NSL_KDD Dataset, a standard for Intrusion Detection algorithms.

## 1.6 ORGANISATION OF REPORT

The rest of the report is organized in the following way. Chapter 2 highlights the work does in this field for both static and streaming data. We discuss in Chapter 3, the proposed architecture and the different approached followed. Chapter 4 talks about the implementation details and the results achieved with the Conclusion and Future Work stated in Chapter 5.

# 2 LITERATURE SURVEY

Intrusion Detection has been a field of research for past 2 decades, owing to the sensitivity of the topic. We shall be dividing the research work about NIDS in 2 sections depending on the type of data viz. Static or Streaming Data.

Our literature survey shall follow the classification as shown in Fig.3

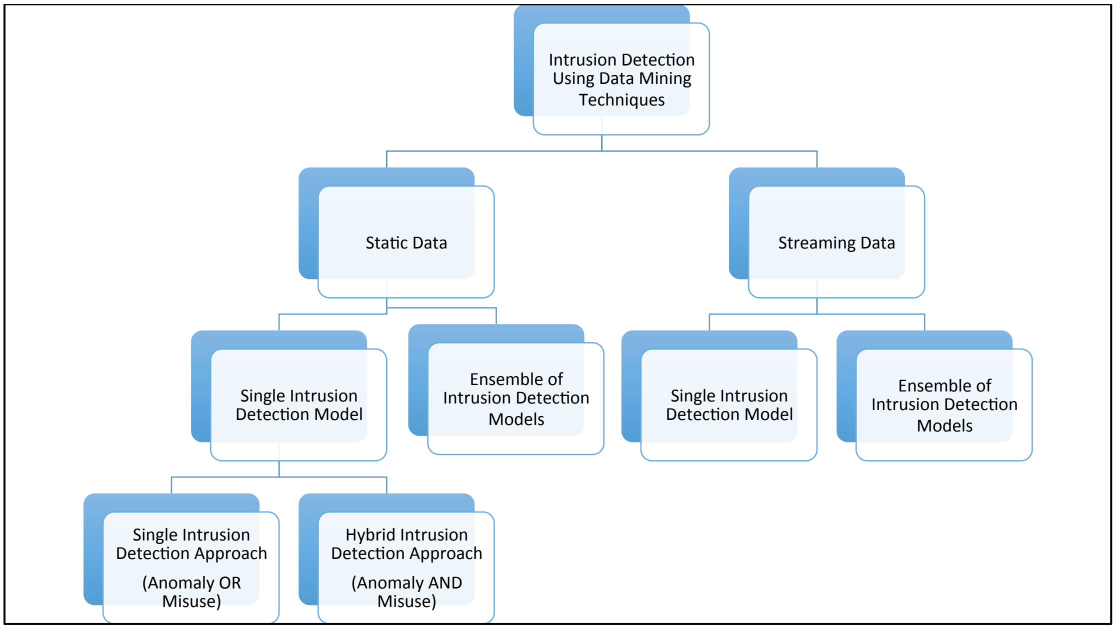

Intrusion Detection Using Data Mining Techniques
Static Data
Streaming Data
Single Intrusion Detection Model
Ensemble of Intrusion Detection Models
Single Intrusion Detection Model
Ensemble of Intrusion Detection Models
Single Intrusion Detection Approach (Anomaly OR Misuse)
Hybrid Intrusion Detection Approach (Anomaly AND Misuse)

*Figure 3: Categorization of Intrusion Detection techniques*

## 2.1 INTRUSION DETECTION METHODS FOR STATIC DATA

In this section, we have discussed three different approaches of intrusion detection for static data.

- *Single intrusion detection methods*. This section contains techniques that use a single data-mining approach to address the problem. We have covered a technique using only Anomaly Detection Module for prediction.
- *Hybrid intrusion detection methods*. It covers methods that leverage both the detection approaches (Misuse and Anomaly Detection approach) to increase the accuracy of prediction of the testing data. We have studied different hybrid algorithm flow models and compared their advantages and disadvantages.

- *Ensemble techniques.* An ensemble of supervised or unsupervised methods is created to enhance the efficiency of the models. The majority of the ensemble predictions can either select the predicted value or some logic can be applied.

### 2.1.1 SINGLE INTRUSION DETECTION METHODS

As discussed, there are 2 approaches to Intrusion Detection, namely Anomaly and Misuse Detection.

Misuse detection modules stores the definition of the attack signatures and the action associated if detected. Many of the popular IDS which uses misuse detection approach to detect intrusion are for ex., SNORT, NetRanger etc. SNORT maintains a huge database of thousands of rules that are a reflective of the plausible attacks.

One of the misuse detection approaches are explained in [5] that proposes an algorithm MADAM ID which is amongst the first research done to incorporate data mining techniques to Intrusion Detection The algorithm computes statistical and temporal frequent patterns from the dataset (DARPA 1998 Dataset) and applies rule learning algorithm Ripper to the learnt patterns to create rules for the intrusions in the data. Statistic patterns may include information about connections from specific source to specific destinations, or the number of packets transferred or data bytes transferred from a source to destination etc. Temporal statistics include features like number of connections per unit time. Such features may be useful in detecting attacks like DoS, probing etc.

An anomaly detection approach is proposed in [1]. LERAD (Learning rules for Anomaly detection) is a method that creates rules to depict the normal packets' behavior. It generates profile, which is in the form of syntactical relationship between attributes, by randomly selecting instances from a smaller section of the dataset and creates rules from its matching attributes. It performs a coverage test to remove rules not covering any new instance. Each rule is assigned a probability which is an indication of it getting violated and on its basis each violating instance is assigned an anomaly score which if exceeds a certain threshold, labels the instance as 'attack'. We have implemented and used this algorithm as a part of the proposed model.

Another anomaly detection method proposed in [7], proposes method to detect anomalies using the pattern matching approach. The technique is an unsupervised method and creates patterns/ frequent patterns of the port sequences used in the connections for a single source. It creates multiple modes as the output where each mode is the representation of the rarity of the event. It updates it library if a new pattern is seen with probability exceeding the threshold.

The techniques presented in this section were focused on a single detection approach. But now a day, most of the proposed techniques follow hybrid approach to take advantage of both the methodologies. The next section shall discuss few hybrid approach techniques.

### 2.1.2 HYBRID INTRUSION DETECTION METHODS

Despite the fact that both the detection approaches are better than the other, they have their inherent disadvantages. And thus, single intrusion detection methods find it hard to match the accuracy with hybrid models.

Explaining in brief the different models used by Hybrid Detection methods:

*I. Anomaly Detection Module followed by Misuse Detection Module.*

In [8], a hybrid detection method is introduced which used different modules for ensuring both anomaly and misuse detection. Fig 4 describes the model diagrammatically. The Anomaly Detection module creates profile for the normal data by using Association Rule Mining and Misuse Module classifies the known attack by doing a supervised learning on the attack data. Also, features are selected to reduce the algorithm complexity.

A part of training attack-free data is first fed to Anomaly Detection module. It computes the frequent itemsets from the data and thereby forms a profile for the normal packets. The rest of the data is then fed to the module. The algorithm works in an online fashion. It maintains a window of fixed size and compute frequent itemsets from the chunk. If the chunk has frequent sets that are deemed normal, no action is taken. Else, a counter is kept to track the support of the itemsets. If the counter exceeds the threshold, it is passed to the misuse detection module which learns from the training data if to classify such a data point as an 'attack' or a 'false alarm'.

For testing data, the online window is used to compute the frequent itemsets. Suspicious data instances are passed through the Misuse Detection module that will label them accordingly with default behaviors kept to 'unknown attack'.

This approach depended on low positives by anomaly and needed the anomaly detection to have a high detection rate. It as well needed misuse module to be able to deal with the false alarms generated by the anomaly detection module.

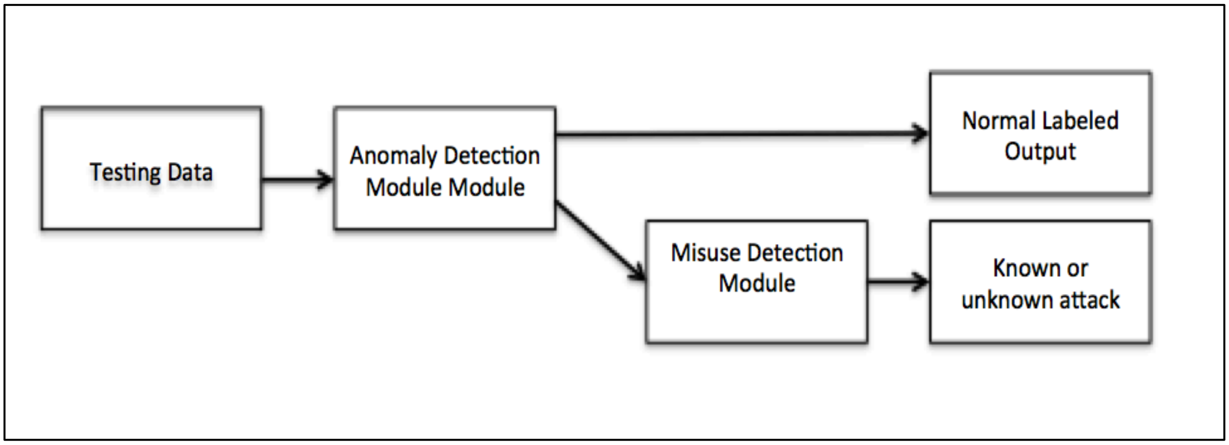


*Figure 4: Anomaly Detection Module followed by Misuse Detection Module*

*II. Anomaly and Misuse Detection model in parallel*

Another method [9] proposes to increase the accuracy in prediction. Fig 5 shows the model of concept. Anomaly and Misuse detection models computed independently their results and merged them. It as well selected few features to improve the efficiency of the algorithms. Since modeling the complete normal set is difficult, the algorithm proposed to create analyzers for different services. For ex, they built analyzers for TCP, UDP and ICMP Anomaly.

The label of the data point is an attack if either of the modules claims it to be an attack. If Misuse Module detects the attack, then the attack can be classified as well on the basis of the supervised learning from the attack data points. If Anomaly detected the attack, the attack will be defined as an ‘Unlabeled Attack’.

It improves the detection rate and eases the modeling of normal behavior but the number of false positives doesn’t improve.

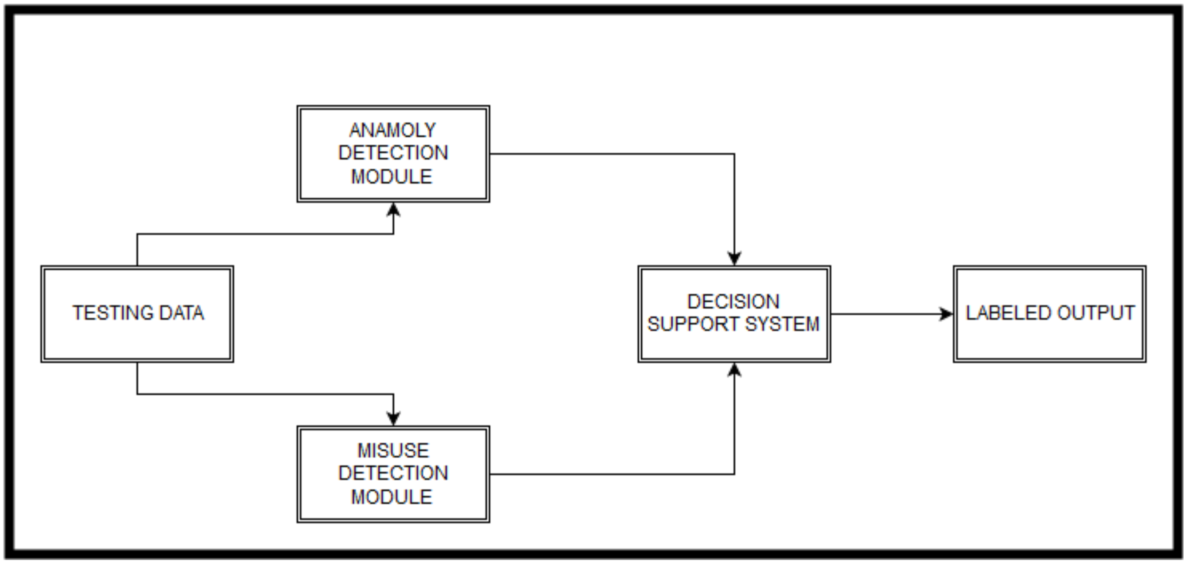


*Figure 5: Anomaly and Misuse Detection Modules in Parallel*

*III. Misuse Detection Module followed by Anomaly Detection Module*

The third technique, which we will describe, shows the best accuracy amongst the hybrid models we studied.

The [10] is on the similar lines as above. It uses the classification approach as well to create the signatures for the known classes.

The algorithm uses Decision Tree and C4.5 for Misuse Detection module in the system. C4.5 decomposes the data into region and assigns each region the class of the majority data points.

The algorithm uses Decision Tree C4.5 Classifier on the complete training set i.e. all normal and attacks data transactions for the Misuse Detection. DT created decomposes the training set for both the classes. For Anomaly Detection, the algorithm uses 1 class SVM to classify the subsets of the training set for normal data nodes created by decision tree. The classifier then creates a decision function for detecting the outliers as possible attacks. This reduces the complexity of building normal profile for the complete training set, as the profile can be very varied.

As the figure 6 suggests, the testing data is fed to first misuse detection module that labels data as either a known attack or an unknown attack label. The unknown attack labeled input is sent to the Anomaly Detection module where it uses the trained classifiers to say if the input is indeed an unknown attack or if its normal.

The above algorithm is tested on NSL-KSS dataset [5], which is a modified version of popular KDD'99 dataset.

Thus, instead of the just integrating the output of both anomaly and misuse models independently, this technique yield better result than them.

The algorithm out-performs earlier techniques presented, because we use anomaly only after misuse module and thereby reducing the number of false positive alarms. Thus, filtering of the data is done before anomaly detection module.

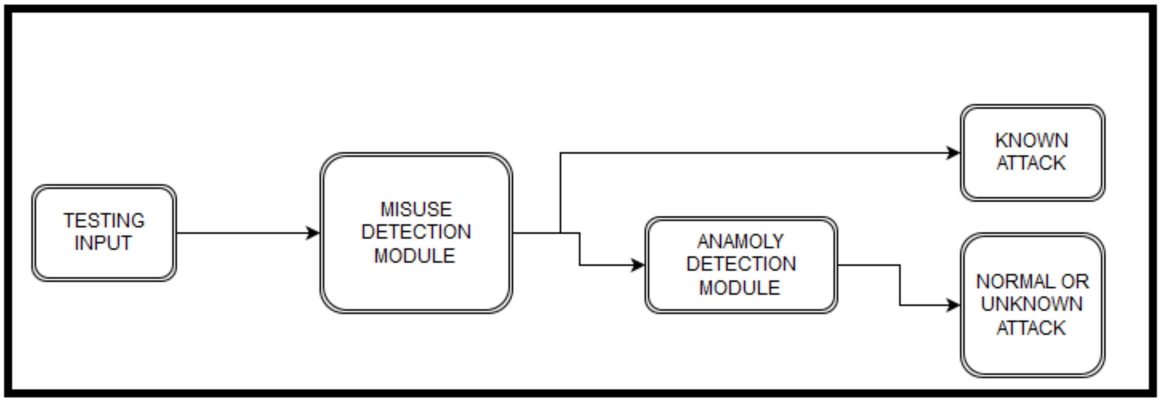


*Figure 6: Misuse Detection Module followed by Anomaly Detection Module*

The above-supervised techniques all need labeled data that may be difficult to obtain for datasets like that for intrusion detection systems as its not feasible every time to know beforehand if a traffic is normal or an attack. Thus, if labels aren't given then classification techniques in the above examples cannot be used.

Thus, next we will discuss a clustering approach to learn signatures of the data points.

### 2.1.3 ENSEMBLE INTRUSION DETECTION METHODS

The next technique uses the ensemble approach to build the intrusion detection system.

The paper [11] uses the public MAWI repository that contains 15 min long raw packets over a span of 10 years for each day. The data thus is inherently a time series data.

The algorithm proposed works on single source single destination packet and generates IP flows for a time slot considered. Each flow contains information like Traffic per time slot, information about the number of source and destination addresses and their ports.

The algorithm proposed by the paper is divided in 2 stages:

***I.*** *Defining the dataset to work on*

The algorithm detects an anomalous time slot of a fixed length and uses some generic change detection algorithm on the slot defined by the above-mentioned parameters. A time slot is flagged as suspicious by algorithm and it generates an alarm for the large deviation in the values of the parameters.

***II.*** *Unsupervised Anomaly Detection using Clustering*

All IP flows in the flagged time slot are considered for the unsupervised learning. IP flows are analyzed at 2 levels or resolutions for an IP source address and for destination address. Thus for each source and destination the information about the following parameters is collected and used for the process of learning. The parameters are: Number of source and destination IP addresses and ports, packet rate, ratio of number of sources to number of destinations, fraction of ICMP and SYN packets from the total packets and average packet size.

The keys are taken to detect 1-N and N-1 anomalies effectively where 1-N refers to the anomalies with a single source and multiple destinations like spreading virus and N-1 help detecting anomalies for multiple source and single destination like DDOS.

In order to be able to automatically detect the parameters required for the clustering algorithms like the number of clusters, the shape etc., the algorithm uses sub space clustering algorithm and then uses Density based clustering on each of the partition created.

The algorithm creates partitions for every 2 attributes and thus creates a total of ${}^{9}C_{2}$ ways. To each partition created DBSCAN (Density Based Clustering of Applications with Noise) is used. The clustering algorithm creates clusters using the notion that clusters are differentiated using low-density regions and the number of points defines a density region or IP flows falling into the subspace. The parameters choses for DBSCAN are: the maximum neighborhood distance is selected from the average distance between flows where 10% flows are randomly selected from the sample. The reason for choosing such small value of the subspace attribute partition size is that DBSCAN works more efficiently on data with low dimensions.

Once clusters are formed for each subspace, each flow is ranked on the degree on anonymity by considering its distance from the largest cluster formed in the time slot, assuming that the normal traffic makes for the majority in the total traffic. Thus, a degree is obtained for each flow from all the subspaces. This matrix thereby stores the dissimilarity of the outliers from the normal points. The algorithm updates the matrix for the outliers detected in each partition.

The matrix obtained then defines the threshold for the anomaly when there is major change in the slope.

Thus, for the testing data their value of dissimilarity is computed and if it is greater than the threshold, it is labeled as an attack. Because of the way the parameters are selected, it is also able to detect various kinds of attack. For ex, port scan usually involve small sized packets in the network from a single, multiple source to a single destination and its multiple ports and is usually characterized by SYN packets.

The algorithm claims to outperform many of the other supervised techniques. It is important to study the unsupervised algorithms because it may not be easy to get signatures every time.

[15] proposes a parallel version of random forest. Random forest creates an ensemble of decision tree with random features and random number of instances and do majority voting to predict the target. The algorithm proposes how selecting features from roulette wheel-based feature selection rather than giving them all equal weights can give better accuracy. Also, it proposed fuzzy rules to combine the results from the decision trees. The algorithm was implemented on Hadoop for better accuracy.

## 2.2 INTRUSION DETECTION METHODS FOR STREAMING DATA

Intrusion Detection for streaming data is challenging owing to the dynamic nature of data streams. There is huge amount of data because of the continuous, and high-speed characteristics of the data streams. Unlike methods for static data, which can do multiple scans on the data, only a single scan is done for data streams because of the space and computational complexity. Also, the data is divided into chunks where each chunk is stored into the memory to be processed. The algorithms also have to consider the 'Concept Drift', notion common to data streams, which mean that the distribution of the input may change with time. In context to intrusion detection, it may for ex, mean that new attacks may be developed with time and the algorithms should be designed to be able to handle the new cases as well. And, since an outlier cannot be readily declared in case of data streams because it may be part of a novel class which is yet to come in next data chunks, the outlier information need to be stored for computation from next chunks. Both the discussed algorithms do take care about the all the concept drift and postponing the outlier decision till fixed number of chunks or time units.

We have divided the algorithms for data streams into single and ensemble methods that we will cover in forthcoming sections.

### 2.2.1 SINGLE INTRUSION DETECTION METHODS

The algorithm in this section discusses an unsupervised outlier detection model that is specifically aimed at using anomaly detection approach. The [12] has proposed 2 techniques and we will be covering the better one in the report which has lesser false positives compared to the other.

In this technique, the network packet instances from streaming data are considered as individual points and are compared with the existing clusters and are then on the basis of their Euclidean distance are labeled as normal or anomalous.

The technique has used a modified version of DBSCAN to process stream data. DBSCAN is a density based clustering algorithm that takes 2 parameters, *epsilon* and *min_points*. While epsilon defines the radius of a well formed cluster, min_points gives the threshold for the data points within a radius of epsilon before a point can be treated as a *core point* of the cluster formed. DenStream, version of DBSCAN for clustering streaming data, use Damping Window to give more importance to the recent data by using a decay function based on time. Using the decay function values for different points in the stream, the updated center and radius of the clusters is computed. It also defines the notion of p-clusters and o-clusters that refer to potential normal and outlier clusters respectively. The definition of the normal signature is also updated with time owing to the dynamic nature of the streaming data, also known as ‘Concept Drift’. The DenStream would for each incoming point check for the nearest micro cluster and label the point accordingly and update the outlier clusters as their weight grows.

The [12] argues that most of the points in the outlier microclusters formed from the training input is in fact anomalous and promoting them to potential clusters would create more false negatives than false positives. Thus, here each data point is treated as anomalous if it comes to an outlier clusters or is forming a new outlier cluster.

Such a techniques is well formed if most of the attacks follow the same pattern. The algorithm doesn’t get intimidated with the attacks containing large number of data points, like DOS attacks. Novel attacks differing from the normal packets’ signatures can be detected easily as well.

But the algorithm doesn’t take into account the concept drift, that is the change in the definitions of the normal packets’ signature over time. We will discuss next an ensemble technique that gives better rates of predictions. Also, it employs the practical use case of incrementing the model by getting knowledge about the datasets seen after some definite interval of time.

### 2.2.2 ENSEMBLE INTRUSION DETECTION METHODS

The paper [13] presents a technique for novel class detection in data streams. Because one of the challenging problems of data streams is the concept evolution, which is apt in case of intrusion detection, this paper rightly detects the new attacks in the network.

The algorithm proposed for the data stream addresses the issue of infinite length by considering the stream in equal sized chunk and doing processing on it for incremental learning. It maintains an ensemble of M classification models to detect the new class and to correctly predict the data values coming using majority voting, the way it happens in ensemble techniques.

The algorithm doesn’t assumes that the data coming from the stream can be unlabeled or rather would be mostly or all labeled and takes the practical approach that a transaction would be labeled after some fixed $T_l$ time units. The assumption looks okay owing to the fact that analysts do label the data after some time. Even for the attack data, the analyst may not classify an attack as it comes, but after some time when it is detected, it is put into the system attack definitions. It also defines $T_c$ that puts a constraint on the classifiers to classify the data with this much time units. And $T_c < T_l$ that makes sense as else there will be no need for the classification model.

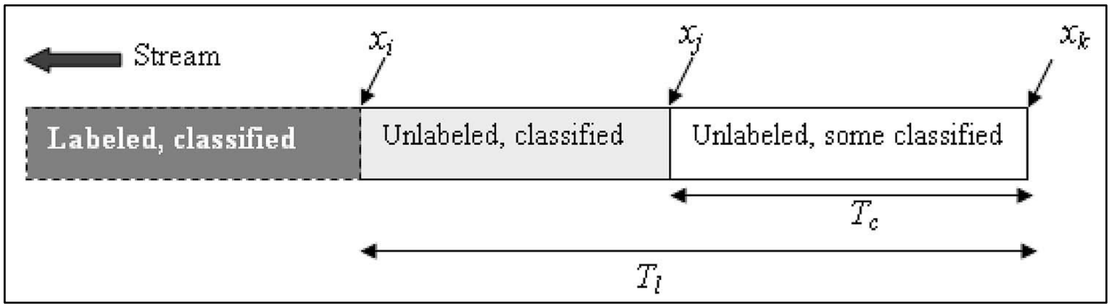


*Figure 7: Description of Time Constraints [13]*

Each data point is checked against the learnt ensemble of classifiers and if it doesn't fall into the region/ signatures defined by the classifiers, the data point is labeled as an outlier. The outlier is information is stored and checked against other outliers to see if a novel class is generated out of them or not. All the labeling should finish by $T_c$ unit time and at $T_l$ when the correct labels are learnt, the ensemble is updated with the new set of information.

The algorithm uses 3 buffers, BUF to store the potential novel class points, U for storing the summary of the classified data points and L for the correctly labeled data points for each chunk.

For each data point in the data stream, all the classifiers do a majority voting. A class is novel only if doesn't fall into any of the classifiers. A data point is stored in BUF if it looks like a possible novel class member. At all times it checks if the data points in U have become older than $T_l$ window. If yes, it is de-queued from U and queued in L. When the size of L reaches the chunk size, a new classifier mode is built to learn the chunk and the replaces the model on the basis of the accuracy in the classification of the learnt chunk.

The algorithm creates K clusters from the training data using kMeans technique and stores only the microclusters and discards the raw points. Each test point is assigned the class label that has the highest label in the microcluster. The classification models either k-NN or Decision Trees uses thereby these clusters to label the data stream objects. The algorithm uses supervision despite clustering to correctly label the clusters, which they couldn't have, in absence of classification models and the definition of the novel class doesn't make sense.

Once learnt, these classifiers represent the region of an ensemble and if a stream data point falls outside this region, it can be a potential member of the novel class.

Thus for the testing data, it is checked if it falls with the ensemble region. If yes, the majority voting is done for the data point, else the point is queued to BUF and is checked at fixed time units if a new cluster is formed from the BUF points. The algorithm uses the concept of q-nearest neighbors to check the nearness with the microclusters. If yes, they are cleared and a novel class is pointed out. The BUF is filtered to remove the points now labeled correctly as they may mislead the results. The algorithm optimizes the process by instead of storing all the data points in BUF, storing just the clusters out of them.

The algorithm manages its ensemble with different classes in the following way:

I. If any of the younger trained classifier labels a stream object with the class not present in the older classifier, the voting process ignores the classification by the older model.

II. If the oldest classifier contains the class signatures not present in any of the younger classifiers, the model is removed from the ensemble as it contains obsolete information.

Thus, while the algorithm uses practical ways to enhance the predictive nature and to update the model and gives high accuracy in predicting the attacks unseen before, the space complexity of the algorithm is high because it has to store a number of models.

## 2.3 RESEARCH GAP

Most of the existing IDS are based on Anomaly Detection approach but they usually have high false positive rate. One of the most popular IDS is SNORT which is based on intrusion detection but it does not have any idea about new attacks and also, they should be frequently updated manually to keep the definitions up-to-date. Also, research work for intrusion detection for real time streaming data is very limited.

# 3 PROPOSED WORK

## 3.1 INTRODUCTION

This chapter discusses the proposed hybrid model, the description of its components and their relevance in increasing accuracy of the overall system.

The hybrid model consists of both Anomaly and Misuse Detection module and feeds on the data stream. The model returns if the incoming data packet is an attack, be it new or a known attack, or if it is a normal packet. The attacks detected can then be used to generate 'alarm/alert' to take appropriate action either automatically or by involving a human administrator.

The following sections shall in detail describe the framework proposed and how it addresses the challenges of managing a data stream.

## 3.2 PROPOSED FRAMEWORK

The following figures 8, 9 and 10 shows all the components in the proposed framework.

The models consist of a Misuse Detection Module, which is responsible for detecting known attacks in the incoming flow. This module contains a signature dictionary that is learnt from a part of training dataset. The profile built is then checked against the incoming data and if pattern is matched, the packet is labeled as an attack. Else, the packet is sent to the Anomaly Detection Module. This module has stored in it the profile of normal packets and any deviation from the signature stored is deemed as a new attack. In this module, we have used an ensemble of 'LERAD' modules, which shall be explained in detail in coming section to improve on the accuracy of the architecture.

To apply the proposed framework on the data streams, the data is broken into chunks of reasonable size and then these chunks are learnt and tested on. The algorithm waits for a chunk to be filled before it started processing. The dissertation aims at solving the challenges posed by a data stream that shall be explained in coming sections. Figure 8, 9 and 10 shows the different phases of learning, testing and updating the model in the proposed framework.

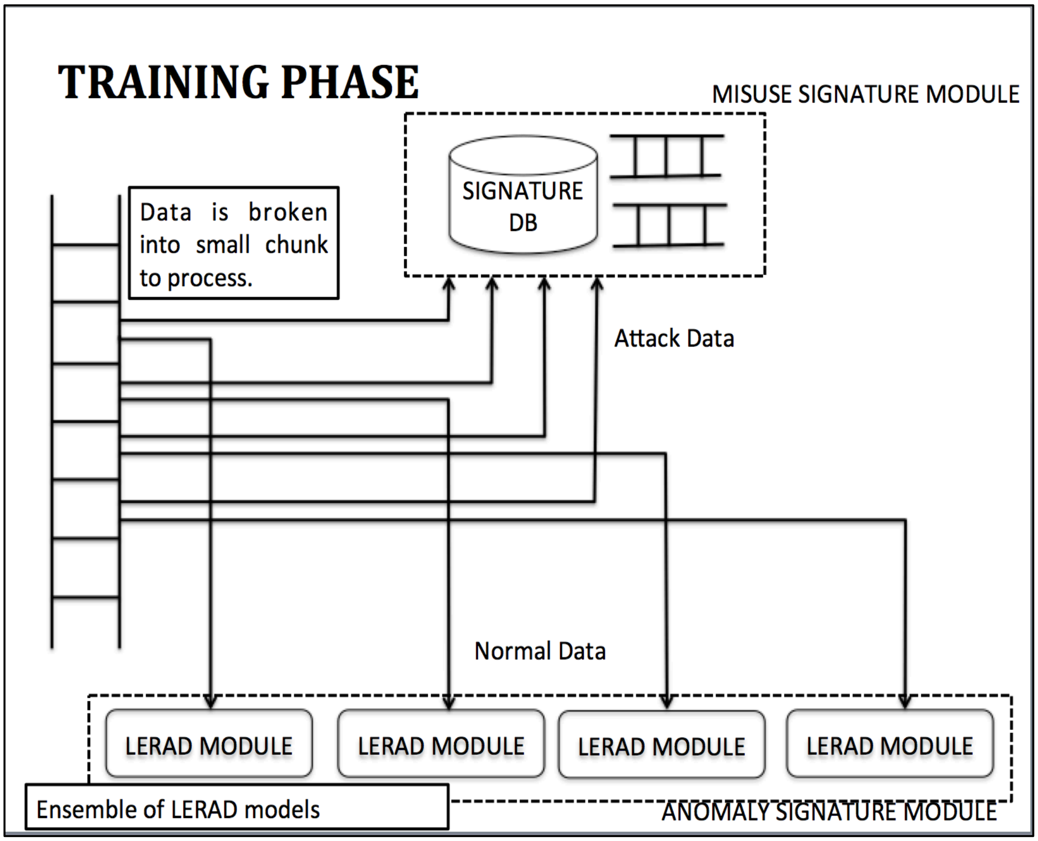


*Figure 8: LEARNING Phase of Proposed Hybrid Architecture for Intrusion Detection*

## 3.3 MODEL ASSUMPTIONS

Following is the listing of the assumptions of the proposed model

1. To address 'Infinite Length', we break data and process it in chunks of fixed length. We therefore, store only the data in latest chunk and the data is discarded once it is processed i.e. predicted and helped back in updating the models.
2. We assume, that we can know the true labels of the incoming data after some unit of chunks of data. The time constraint is shown in Fig 7. Thus, for example, we have taken in our experiment; three chunk time period after which we get their true labels. We store data for 3 chunks and then, with the their true labels, we compute accuracy of the predicted values and update the model. With each chunk, we update the signature database and create a new Lerad model that replaces in ensemble the one with least accuracy.

# 4 EXPERIMENTAL RESULTS

We have implemented the LERAD algorithm for Anomaly Detection Module in Python and imported Python's '**sklearn**' packages to implement Misuse Detection Module containing Decision Tree classifiers. We downloaded the dataset NSL_KDD, from UCI repository and deleted invalid entries (There were some entries containing 85 attributes), if any in the datasets.

We have run and compared the result of the proposed model with the algorithms running individually on the static data and found the results to be almost comparable, which is good considering we were working on data streams and handling each chunk at a time without having the knowledge of the complete dataset.

## 4.1 DATASET USED

Intrusion Detection is an active research field. We have used NSL-KDD, dataset popular in this field. NSL is built on top of a very popular dataset KDD Cup 1999 Dataset and was built to solve the inherent limitation of the KDD Cup Dataset, which shall be explained in coming sections. Also, the size of the training and testing dataset of NSL KDD is reasonable and doesn't require to randomly sample data to conduct an experiment.

We shall be explaining in brief about the KDD Cup followed by the changes NSL introduces to improve the former.

*1. KDD Cup '99 Dataset*

This was the data set used for The Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 The Fifth International Conference on Knowledge Discovery and Data Mining. The competition task was to build a network intrusion detector, a predictive model capable of distinguishing attacks from normal connections. [3]

This data set is prepared by Stolfo et al. and is built based on the data captured in DARPA'98 IDS evaluation program. KDD training dataset consists of approximately 4,900,000 single connection vectors each of which contains 41 features and is labeled as either normal or an attack, with exactly one specific attack type. Following are the categories of the attacks.

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

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

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

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

4) Probing Attack: is an attempt to gather information about a network of computers for the apparent purpose of circumventing its security controls. The attacker scans a machine to find out the weakness or vulnerabilities in the network using mscan etc.

The features of the dataset can be divided into following 3 categories [3]:

1) Basic features: This category list attributes that can be extracted from a TCP/IP network data packet. For ex. protocol type, duration of a connection etc.

2) Traffic features: This includes 'time based' features i.e. corresponding to each current instance, features are extracted on the basis of a window interval of 2sec in this case. It is further divided into 2 sub categories:

2.1) 'Same Host' features: This list features that contains information about the connections to a particular destination (same as the current instance) in the last 2 seconds. It includes statistical features like 'serror_rate' which stands for percentage of connections that have 'SYN error.

2.2) 'Same Service' features: This list features that contain information about the connections to the same service as that of the current connection in the past 2 seconds. An example of such a feature is 'srv_diff_host_rate', percentage of number of connections to different hosts.

3) Content features: DoS and Probing attacks have sequential patterns that are covered by the traffic features. But attacks like U2R and R2L don't contain patterns like them. To detect them, a list of features is embedded in the data portions of the packet to generate suspicious for such attacks. Features like 'num_failed_logins' denoting number of unsuccessful login attempts, etc. can give an insight about such attacks.

To simulate real time traffic details, where the attacks seen are not always known before hand and new attacks are frequently introduced, the dataset has 24 attack types in training dataset and 14 new in test dataset.

Table 3 and 4 give a brief summary of the dataset KDD and NSL-KDD.

*Table 2: Size of the training and Test Dataset in KDD and NSL-KDD*

| Datasets | Training Dataset | Test Dataset |
|---|---|---|
| KDD CUP 1999 Dataset | 4898431 | 300000 |
| NSL-KDD Dataset | 125971 | 22541 |

*Table 3: Normal & attack data size in training and Test Dataset in KDD & NSL-KDD*

| Dataset | Normal Data | Attack Data |
|---|---|---|
| KDD'99 Training Set | 972781 | 3925650 |
| KDD'99 Test Set | 90783 | 209218 |
| NSL-KDD Training data | 67341 | 58630 |
| NSL-KDD Test Data | 9709 | 12832 |

## 2. *NSL_KDD DATASET*

A modification of KDD Cup, NSL KDD dataset is created to resolve the following issues with the above dataset. KDD Cup had following inherent issues that the NSL-KDD resolved. [3]

1) Redundant data: KDD CUP has a number of repeated instances and thereby favors algorithms that may work well for the redundant tuples. NSL KDD is improved by removing all such instances from both training and testing set. Also, invalid instances were removed from the KDD CUP. We also, encountered an invalid in test case of NSL Dataset that we removed. It contained 85 attributes and looked like 2 instance merged into one.

2) Level of difficulty: NSL KDD dataset arranged test and training dataset i.e. the instances whose prediction is easier are least in the new dataset. They achieved this by computing for KDD’s each instance, #successfulPrediction which is an indicator of successful prediction by 21 different trainers. The one with the maximum value are supposedly most easy to guess and their percentage of intake to NSL was proportional to the inverse of their value in #successfulPrediction and thereby avoiding the data to be skewed. The dataset also provides with the set with contains all elements in test data except for the instances that got ‘21’ in the indicator variable.

This filtering reduced the size of the training and test dataset as shown in Table 3.

Thus, NSL-KDD had therefore the following advantages to KDD CUP ’99 Dataset and is therefore a better representative of the accurateness of the proposed framework.

1) This dataset is not biased for frequent records as the redundancy is removed from KDD dataset. The dataset also thereby, doesn’t favor the algorithms that work better on those frequent records that KDD supported.

2) Since, the number of instances of each class is inversely proportional to the percentage of the records in the KDD Dataset, this dataset brings out better comparison of data mining techniques.

3) The size of datasets (training and test) is reasonable to conduct an experiment.

## 4.2 PERFORMANCE PARAMETERS

We have computed for each result, the following parameters [14] which formed the basis of our results and defined our guidelines of comparison with other models:

*Table 4: Performance Parameters*

| ACTUAL VALUE (➔)<br>PREDICTED VALUE (⬇) | ATTACK | NORMAL |
|---|---|---|
| ATTACK | True Positive (TP) | False Positive (FP) |
| NORMAL | False Negative (FN) | True Negative (TN) |

On the basis of the parameters defined above in table 5, we compute following values, which are used to compare different models.

1) Accuracy: Accuracy is defined as the ratio of true predicted values vs. the complete input i.e.

Accuracy = (TP + TN)/ (TP + FP + FN + TN)

Thus, higher the accuracy, better a model is.

2) False Positive Rate or False Alarm Rate: FPR is defined as the ratio of the wrongly classified normal packets. This is important because with each alarm generated, a lot of efforts in terms of human hours are applied and therefore, false alarms should be kept to minimum.

False Alarm Rate = FP/ (FP + TN)

Recall, another popular terminology is 1- FPR i.e.

Recall = TN/ (FP + TN)

3) Precision: Precision is defined as the ratio of the correct predictions for the govern attack data. Higher the precision, better a model is.

Precision = TP/ (TP + FN)

4) F1-score: This parameter is defined as

F1 score = 2 * (R * P)/ (R + P), where

R : Recall; P: Precision

Apart from these parameters, training and testing time should be considered for the completeness of the performance computation of algorithms.

We have compared and tested on the above parameters and shown in the next chapter.

## 4.3 RESULTS AND COMPARISON

We shall in this section cover the steps we took to reach at the final framework and the improvement in the accuracy and other parameters with each step.

We shall be dividing our results in the following steps and shall be explaining them all from the next section

1.1) Implementation of Lerad Alone on Training and Test dataset of NSL-KDD

1.2) Implementation of Lerad with weighted rules.

And to extend the algorithm to handle data stream, we shall be comparing

2.1) A single learnt LERAD module for all the chunks of data stream

2.2) Lerad module (1.2) ensemble working on chunk of data

2.3) Implementation and Comparison of different techniques for Misuse Detection Module

2.4) Misuse Detection Module and Anomaly Detection module together, which is the proposed framework of this dissertation.

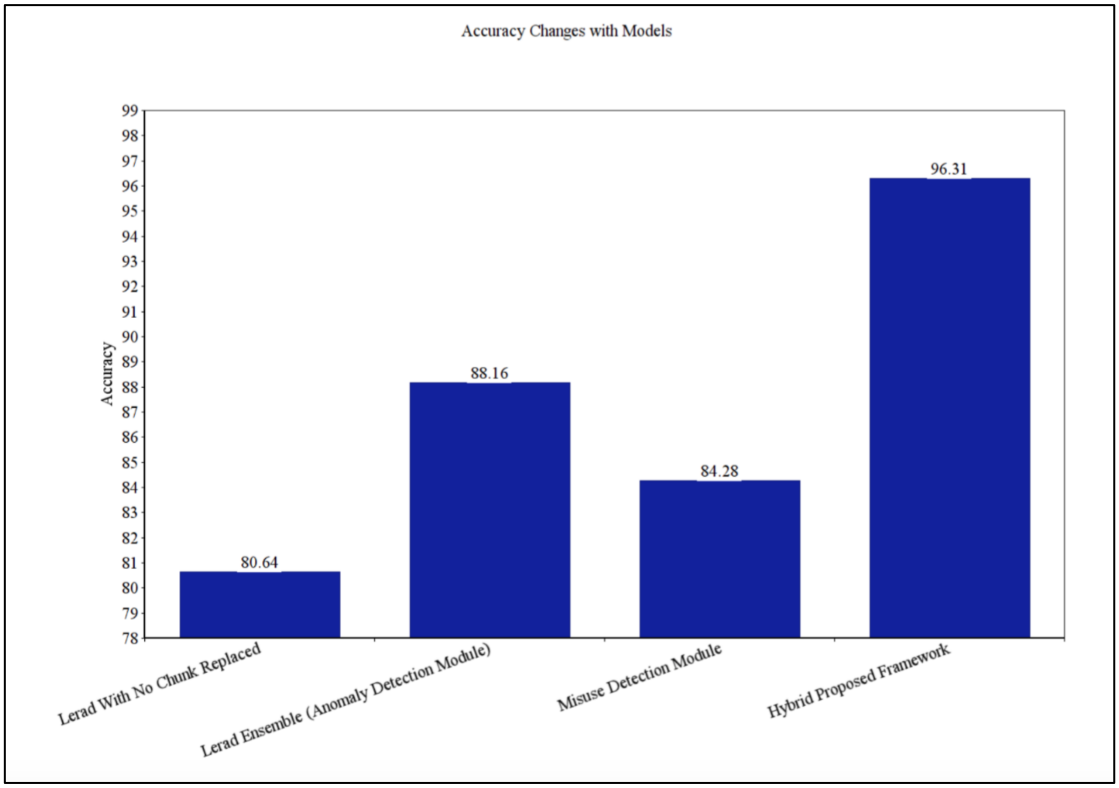


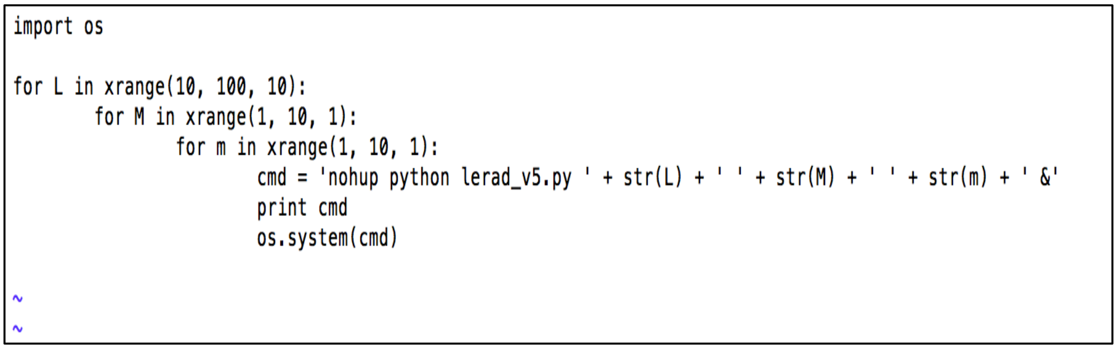
*Figure 9: Accuracy Variations for different Models*

Figure 9 gives a brief idea of the accuracy of the different models. Each model shall be explained in brief with an explanation in the coming sections about how all the parameters had been set for the optimum accuracy.

From the brief figure 15, it is clear that the proposed framework is successful in getting a reasonable improvement over its constituents.

### 4.3.1 Implementation of Lerad Alone on Training and Test dataset of NSL-KDD

We implemented the LERAD algorithm in Python and used it on NSL KDD dataset. Though it faired very well on KDD Cup, its accuracy on NSL-KDD was pretty low. To select the appropriate values of the parameters viz. ‘L’ that is number of random instance pairs to be selected for creating rules, ‘M’ that is number of rules per selected pair and ‘m’ that is upper Bound on the number of matching attributes, following script was run.

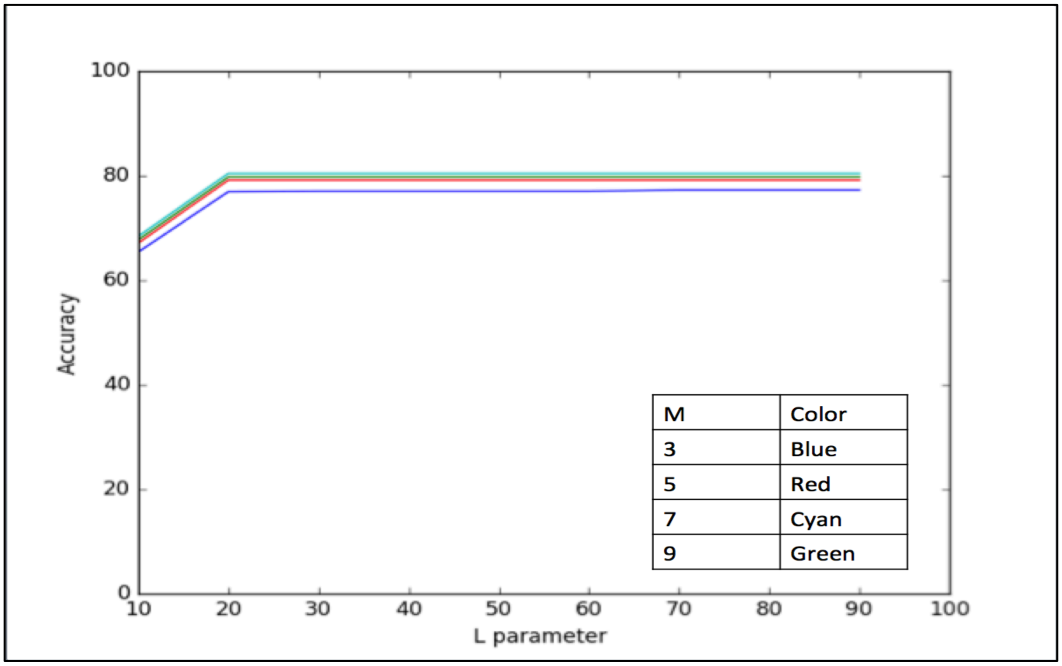


```
import os

for L in xrange(10, 100, 10):
        for M in xrange(1, 10, 1):
                for m in xrange(1, 10, 1):
                        cmd = 'nohup python lerad_v5.py ' + str(L) + ' ' + str(M) + ' ' + str(m) + ' &'
                        print cmd
                        os.system(cmd)

~
~
```

*Figure 10: Script to determine best parameters*

The best accuracy Lerad could achieve was for L = 20, M = 6 and m = 8 was 81.63849299, recall = 0.954189944, precision = 0.61928934 and f1-Score of 0.751099384.

From figure 11, there is a slight improvement in accuracy with more number of rules but after a certain number of rules, the accuracy remains almost constant, and thus, accuracy doesn’t depend a lot on the rules created. It is because the coverage step in general reduces the number to almost same number. But, with the increase in the values of the parameters, increases the time to create and manage rules.



| M | Color |
|---|---|
| 3 | Blue |
| 5 | Red |
| 7 | Cyan |
| 9 | Green |

*Figure 9: ‘L’ VS ‘Accuracy graph for multiple combinations of ‘M’*

### 4.3.2 Implementation of Lerad with modifications

As explained in 3.5.2.2, slight modifications were done to Lerad to make it perform better. A weight was allotted to each rule and the anomaly score is proportional to the weight of the rule. Also, with each violation by a rule, 'alpha' times weight shall be reduces which is added to the rest conforming rules equally.

Slight modification in the efficiency was noted as shown in figure 12. With the change proposed, the modified LERAD was able to achieve a marginal better accuracy with training and testing on the same dataset. It achieved for the same value of parameters and 'alpha' = 0.1, an accuracy of 86.1801525748 with recall 0.919244929924, precision 0.770782875661 and an f1-Score of 0.83849300962.

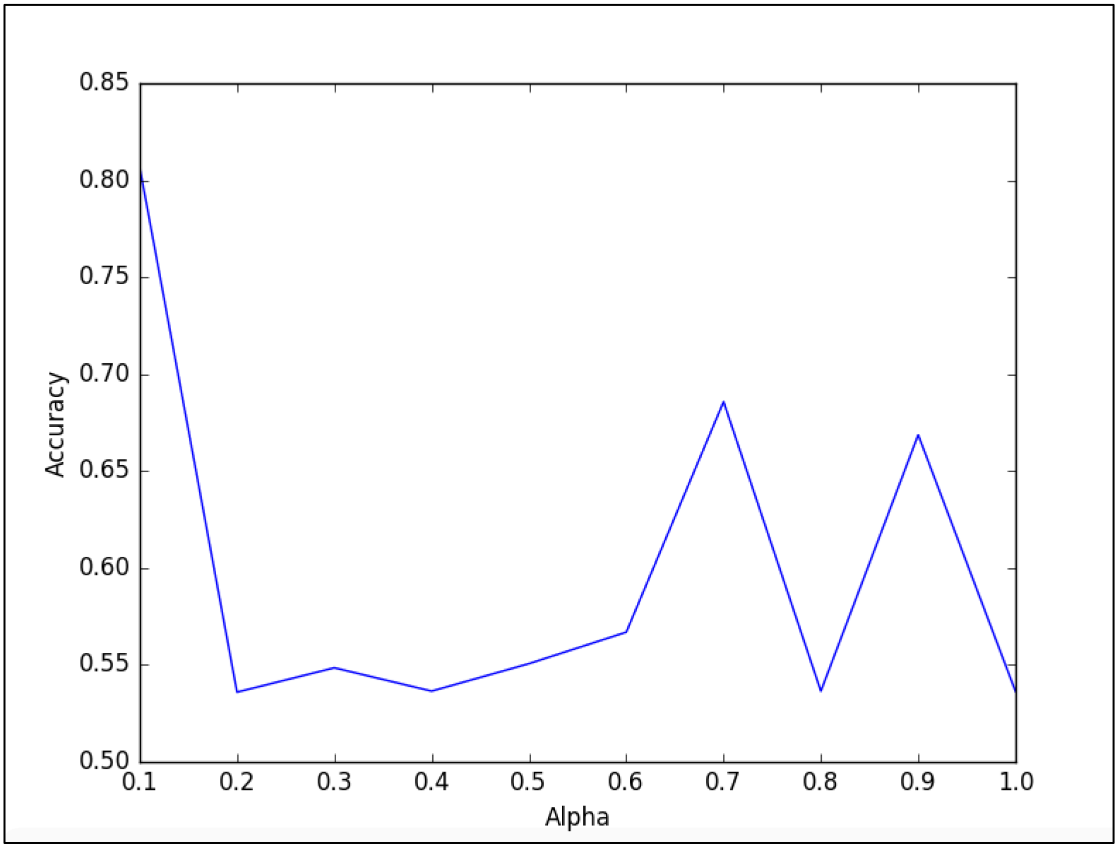


*Figure 2: Alpha vs. Accuracy for weighted rules in Lerad*

For the further examples, it is assumed, that the true label of an instance can be achieved after '3' chunk time. Thus, we have to store our predictions for 3 times the data chunk before we will be able to compute accuracy and be able to update our models.

### 4.3.3 Implementation of Lerad Only in Data Streams' chunks.

To be able to manage data in stream, the framework splits the data in chunks of fixed size and learns, processes and predicts single chunk at a time.

We ran the similar script again this time with an added parameter of chunk size to best select the values of the parameters. A single module of LERAD was learnt from a data chunk and was applied for all the subsequent chunks. This segment was able to handle all the data but couldn't deal with the concept drift or evolution.

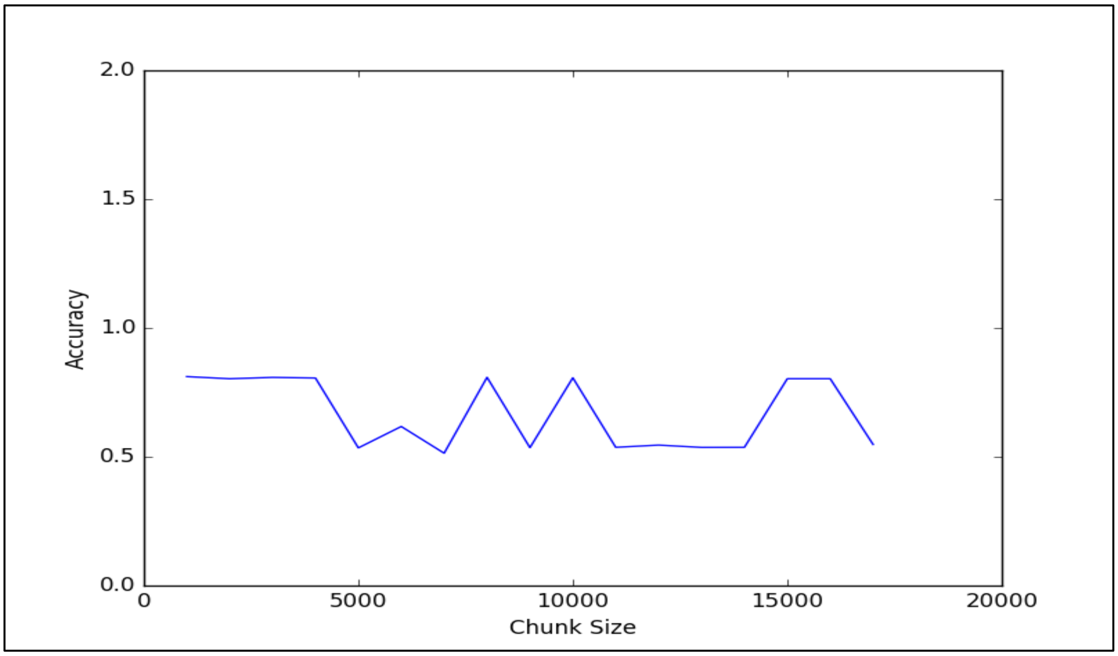


*Figure 10: Accuracy VS Chunk Size for a no replacement LERAD ensemble*

From the figure 13 depicting changes in accuracy with respect to chunk size, we are choosing chunk size to be 10,000 that give an accuracy of 0.806766667 with recall 0.998892171, precision 0.583686974 and f1-Score of 0.73682299

### 4.3.4 Implementation of Anomaly Detection Module

An ensemble of the Lerad modules is used against each chunk of the data stream. Each learnt chunk then contributes to a new Lerad model that replaces the one with the least accuracy. This way, only the most relevant models constitute the ensemble and are thereby able to handle concept drift. Also, since new classes are added to the database as well, the ensemble is able to handle concept evolution as well. We determine using this step, the optimum number of models that should constitute an ensemble. On varying the number of models, it was seen that for 4 models, the maximum accuracy was achieved i.e. 0.881666667 with recall 0.996499045, precision 0.748148148 and an f1-Score of 0.854647195.

Also, the value of precision is considerably low in this case that is due to the fact the anomaly detection logarithms in general fail to reduce the number of false alarms.

### 4.3.5 Implementation of the Misuse Detection Module

Before creating the final proposed framework, various misuse detection modules using different data mining techniques were used. This section contains the accuracy achieved when only misuse detection module was in place. Misuse model can detect the seen attacks, but fails to detect the unseen new attacks. NSL-KDD test dataset contains 14 new attacks not seen in the training dataset

Figure 14 shows variations in accuracy for different classification models. Despite the fact that all the tested techniques work pretty decent, decision Tree seems to be working best for this data and is used in the proposed architecture as well. SVM was found to be comparatively slower with lesser accuracy.

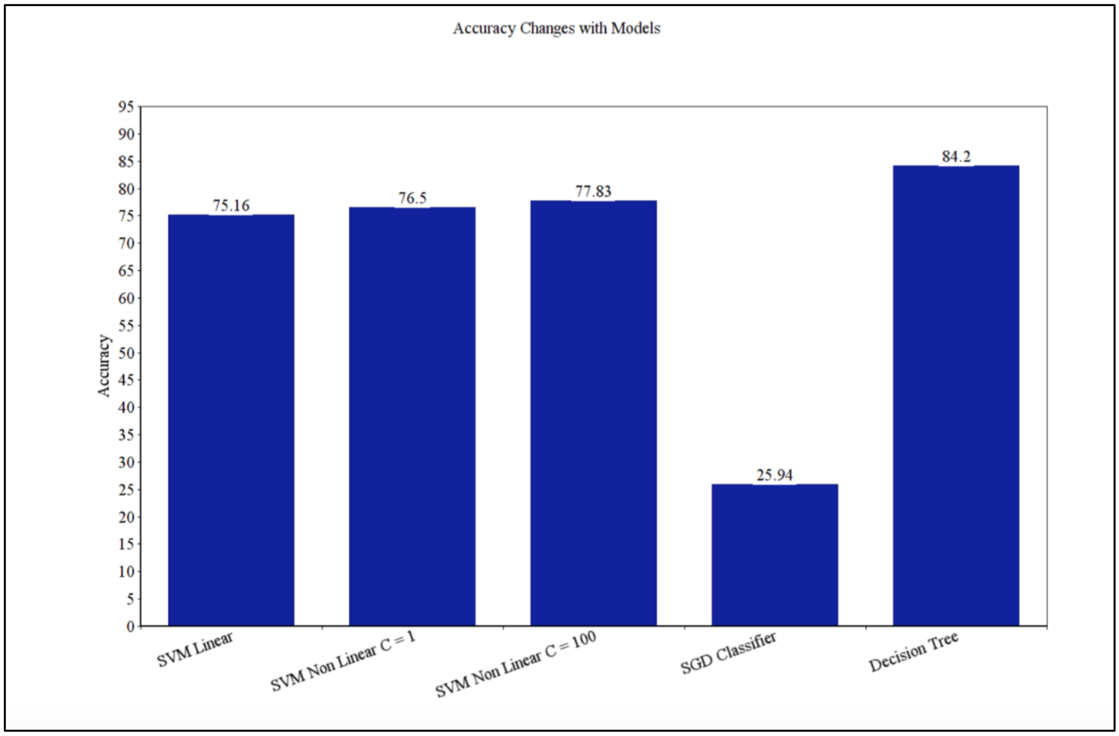


*Figure 14: Accuracy for different classification models*

For decision tree, corresponding to each class, a queue is maintained of fixed maximum length that stores the most recent values of a particular class. Updating classifier on the whole information doesn't take 'Concept Drift' into consideration. Though a marginal increase, a queue size of 10,000 vs 1,000 instances is also shown in the figure 21.

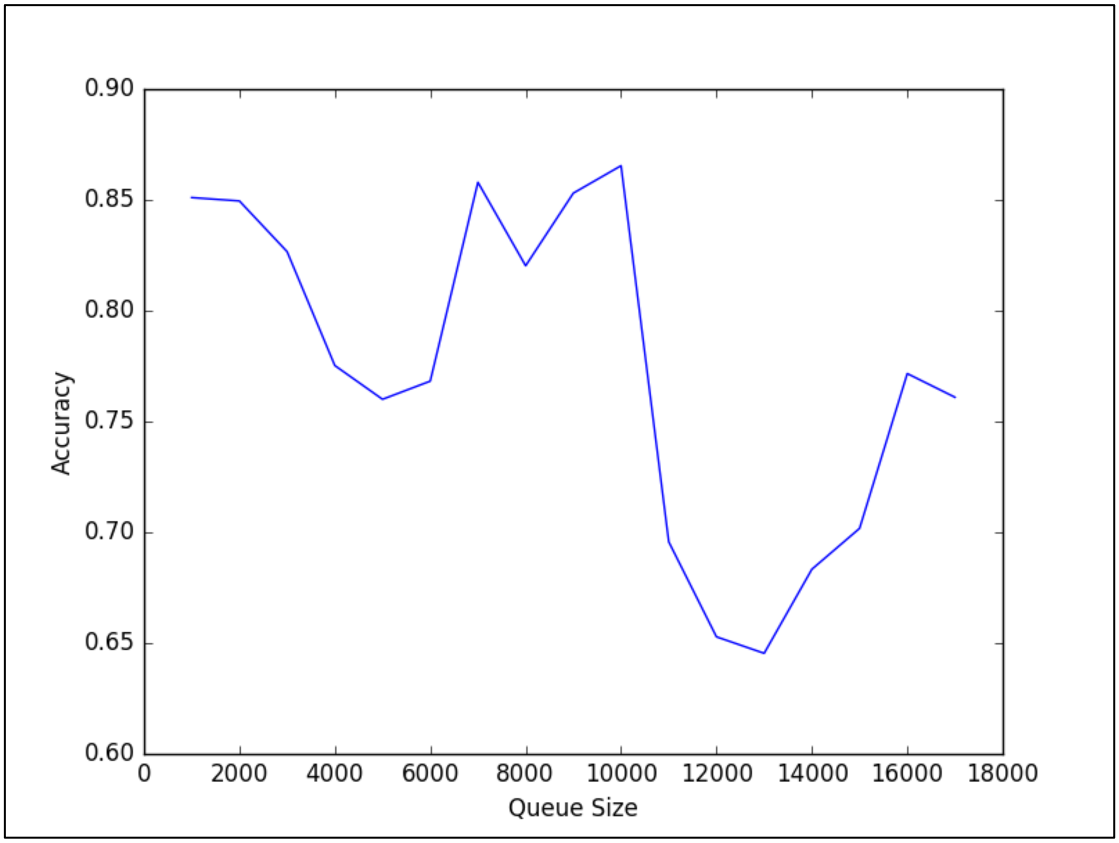


*Figure 15: Queue size VS Accuracy for Misuse Detection Module*

From figure 21, queue size of 1000 shall be fixed for the next section as it gives a reasonable accuracy of 0.851037037 with recall 0.85626572, precision 0.920550342 and f1-score of 0.887245133. Size of 1000 offers lesser space with better accuracy.

### 4.3.6 Implementation of the proposed framework

In this phase, we configured decision tree classifier as well before the anomaly detection module. The chunk size was kept at 10000 instances and fixed length of queue size (1000) was maintained for each of the class labels. The tree was first learnt on the 20% training dataset. And tested on the rest of the training dataset. Also the data chunk is then used to update the models once its true labels are found. The model was able to achieve an accuracy of **0.963157407** with recall of 0.930775398, precision of 0.994700139 and F-Score of 0.961676635 with the selected parameters which is way ahead than what we started with.

Thus, it can be concluded that the proposed architecture fairs way better that the stand-alone algorithm. The main objective of improving accuracy is achieved with better definition of the distinction between normal and anomaly. Also, compared with the initial steps, other parameters viz. precision, recall and f1-score have improved. The improved values of precision and recall are an indicator of lesser false negatives and positives.

# 5 CONCLUSION AND FUTURE WORK

With the growing adoption of Internet and vast dependence on the networked systems, it has become all the way important to secure the networks. New threats are created individually and have become more sophisticated with time. Also, with the increase in the size of data, it is imperative to be able to process large volumes of data and be able to predict on the fly.

This dissertation has proposed a hybrid approach to intrusion detection on streaming data. The architecture consists of 2 main components, viz. Anomaly and Misuse Detection modules. Anomaly Detection Module is based on creating rules for the normal packets and sensing deviation from these rules that are considered as an attack. An ensemble of such models has been created to increase the accuracy and precision in detection. Misuse Detection Module consists of a signature database, which stores the profiles of the attacks. If a packet falls in their signature, it is labeled as a known attack, else, is sent to anomaly detection module that outputs the packet as either normal or an unseen attack. The database also gets updated with the new attacks by storing their corresponding signatures.

The difference between static and streaming data lies in the way the data is handled. For streaming data, we cannot store all the data together and predict. It can only be read in chunks and discarded. And thus, with each iteration, the models should be updated to reflect the data seen. There are 3 major challenges with streaming data, viz. Infinite Length, Concept Drift and Concept Evolution. Storing and processing only a chunk at a time handle infinite Length. Updating the ensemble for Anomaly Detection Module and updating the signature profiles after each chunk solved the problem of Concept Drift. This way the changing definitions of the normal and attack profiles were taken into consideration and reflected in the current models. Regarding, concept evolution, if a new attack is seen, its definition is stored in the dictionary in Misuse Detection Module and is checked for from the next chunks.

While this work has given some promising results, there is always a scope of improvement in systems like Intrusion Detection Systems in terms of accuracy and time factors. Some potential future work can be to introduce weighting of attributes that should change with time to yield better results and realize Concept Drift in the dataset. Also, it may be useful if we can classify the attack in their further categories of DoS, U2R, R2L and Probe.

# REFERENCES

`1. Mahoney, M., & Chan, P. (2003). Learning rules for anomaly detection of hostile network traffic. In IEEE international conference on data mining.

2. KDD CUP 1999 Dataset UCI Repository: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

3. M.Tavallaee, E., W. Lu, and A.A. Ghorbani, ―A detailed analysis of the KDD CUP 99 dataset, ― In Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications, IEEE, 2009.

4. windowsecurity.com, 'Intrusion Detection Systems (IDS) Part I - (network intrusions; attack symptoms; IDS tasks; and IDS architecture), 2003 [Online]. Available: **http://www.windowsecurity.com/articles-tutorials/intrusion_detection/Intrusion_Detection_Systems_IDS_Part_I__network_intrusions_attack_symptoms_IDS_tasks_and_IDS_architecture.html**

5. W. Lee and S.J. Stolfo, A Framework for Constructing Features and Models for Intrusion Detection Systems., *ACM Transactions on Information and System Security,* vol. 3, 4, pp. 227-261, 2000.

6. Aleksandar Lazarevic, Vipin Kumar, Jaideep Srivastava, "INTRUSION DETECTION: A SURVEY", Managing Cyber Threats: Issues, Approaches and Challenges, Vol. 5, 2005, Springer Publisher.

7. A. V aldes, Detecting Novel Scans Through Pattern Anomaly Detection, In *Proceedings of the Third DARPA Information Survivability Conference and Exposition (DISCEX-III2003),* Washington, D.C., April 2003.

8. Barbara, D., Couto, J., Jajodia, S., Popyack, L., & Wu, N. (2001). ADAM: Detecting intrusions by data mining. In Proceedings of the IEEE workshop on information assurance and security (pp. 11–16).

9. Depren, O., Topallar, M., Anarim, E., & Ciliz, M. K. (2005). An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. Expert Systems with

Applications, 29, 713–722.

10. Gisung, K., Seungmin, L., & Sehun, K. (2014). A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. Expert Systems with Applications, 41(4), 1690–1700

11. P. Casas, J. Mazel, P. Owezarski, UNADA: Unsupervised network anomaly detection using sub-space outliers ranking, in: IFIP Networking, 2011.

12. Z. Miller, W. Deitrick, and W. Hu. Anomalous network packet detection using data stream mining. J. Information Security, 2(4):158–168, 2011

13. M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham. 2011. n and Novel Class Detection in Concept-Drifting Data Streams under Time Constraints. IEEE TKDE 23, 6 (2011), 859–874

14. Han,J.and Kamber, J.: Data Mining: Concepts and Techniques, Elsevier 2nd edition, 1011.

15. Saman Masarat, Saeed Sharifian, Hassan Taheri. 2016. Modified parallel random forest for intrusion detection systems. The Journal of Supercomputing. pp 1-24

16. Intrusion Detection Learning:https://kdd.ics.uci.edu/databases/kddcup99/task.html

17. Weka: http://www.cs.waikato.ac.nz/~ml/weka/