

分类号 TP393.08

密级                     

UDC                     

编号 10486

武 汉 大 学  
博 士 学 位 论 文

使用聚类-离群值方法和增量 SVM 分类  
的混合入侵检测

研 究 生 姓 名: Chitrakar Roshan

指导教师姓名、职称: 黄传河

学科、专业名称: 信息安全

研 究 方 向: 入侵检测

**2015年5月30日**



Wuhan University  
PhD Thesis

Hybrid Intrusion Detection with  
Clustering-Outlier Technique and  
Incremental SVM Classification

Research Student: Chitrakar Roshan

Supervisor: Huang Chuanhe

Specialization: Information Security

Research Direction: Intrusion Detection

**May 30, 2015**



## 学位论文使用授权书

(一式两份, 此份论文作者保存)

本论文作者完全了解学校关于保存、使用学位论文的管理办法及规定, 即学校有权保留并向国家有关部门或机构送交论文的复印件和电子版, 允许论文被查阅和借阅, 接受社会监督。本人授权武汉大学可以将本学位论文的全部或部分内容编入学校有关数据库和收录到《中国博士学位论文全文数据库》进行信息服务, 也可以采用影印、缩印或扫描等复制手段保存或汇编本学位论文。

本论文提交  当年 /  一年 /  两年 /  三年以后, 同意发布。

若不选填则视为一年以后同意发布。

注: 保密学位论文, 在解密后适用于本授权书。

作者签名:

导师签名:

2015年5月31日

### 武汉大学研究生学位论文作者信息

论文题目	Hybrid Intrusion Detection with Clustering Outlier Technique and Incremental SVM Classification				
姓名	CHITRAKAR RASHAN	学号	201072110001	答辩日期	2015年5月30日
论文级别	博士 <input checked="" type="checkbox"/> 硕士 <input type="checkbox"/>				
院/系/所	计算机	专业	信息安全		
联系电话	13659817175	E_mail	rashanchi@gmail.com		
通信地址 (邮编):	国际教育学院宿舍3-B624 (430072)				
备注:					

### 稿酬领取通知

学位论文出版后, 杂志社向被录用论文作者支付稿酬, 稿酬支付标准为博士论文作者一次性获得价值 400 元人民币的“CNKI 网络数据库通用检索阅读卡”和 100 元人民币的现金稿酬; 硕士论文作者一次性获得价值 300 元人民币的“CNKI 网络数据库通用检索阅读卡”和 60 元人民币现金稿酬。

请作者直接与杂志社联系领取学位论文发表证书和稿酬。联系方式如下:

联系人: 吴老师 010-62791817 (兼传真)、62793176、62701179 (兼传真)


通讯地址: 北京 清华大学邮局 84-48 信箱 采编中心 邮编: 100084



## 论文原创性说明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的研究成果。除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本人的研究做出贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果就本人承担。

学位论文作者(签名):



2015 年 6 月 1 日





## 武汉大学学位论文使用授权协议书

本学位论文作者愿意遵守武汉大学关于保存、使用学位论文的管理办法及规定，即：学校有权保存学位论文的印刷本和电子版，并提供文献检索与阅览服务；学校可以采用影印、缩印、数字化或其它复制手段保存论文；在以教学与科研服务为目的前提下，学校可以在校园网内公布及部分内容。

在本论文提交当年，同意在校园网内提供前十六页全文浏览服务。

1、本论文提交  当年 /  一年 /  两年 /  三年，同意在校园网内允许读者浏览并下载全文。

(保密论文解密后遵守此规定)

论文作者(签名):



学 号： 2010172110001

学 院： 计算机

日期： 2015 年 6 月 1 日



# Contributions of Thesis

1. This thesis discovers that k-Medoids clustering outperforms k-Means clustering when data samples in an IDS are large. The accuracy and detection rate are found to be higher with lower false alarm rate while using k-Medoids than using k-Medoids clustering. k-Medoids shows consistently better performances when they are followed by either NB classification or SVM classification.
2. The research work develops an Incremental SVM classification method for intrusion detection system, which is faster and more efficient. The proposed method named CSV-ISVM adopts iterative approach of SVM classification but reduces its time complexity by implementing a new support vector selection strategy called Half-partition method that works better with almost half the sample than other methods.
3. This work develops an algorithm that combines clustering technique and outlier detection. This algorithm is an extended version of k-Medoids clustering algorithm and can perform k-Medoids clustering and outlier detection simultaneously. By using this algorithm, an IDS can get no. of clusters normally in order to deal with the detection process as well as it obtains outliers data that can be directly analyzed for possible attacks.



## TABLE OF CONTENTS

<b>ABSTRACT</b> .....	I
<b>摘要 (Summary in Chinese)</b> .....	VI
Chapter 1. Introduction.....	1
1.1 Background and Significance of the Research .....	1
1.2 Research Objectives.....	3
1.3 Solutions to Problems and their Contributions .....	4
1.4 Thesis Organization .....	8
Chapter 2. Literature Review.....	10
2.1 Related work in IDS developments .....	10
2.2 Developments in k-Means/Medoids and Outlier Techniques.....	11
2.3 Works related to NB and SVM Classification.....	14
2.4 Recent work in Incremental SVM Classification .....	22
2.5 Supervised / Unsupervised and Incremental Learning .....	26
2.6 Shortcomings of the Current Researches .....	30
Chapter 3. k-Medoids with Naïve Bayes Classification.....	31
3.1 System Model and Problem Description .....	31
3.2 General Description of the Solution .....	32
3.3 Comparision between k-Means and k-Medoids .....	33
3.4 The Proposed Hybrid Approach .....	34
3.5 Experimental Results and Analysis .....	40
3.5.1 Selection of Experimental Data .....	41
3.5.2 Description of Experimental Data .....	41
3.5.3 Data Pre-processing .....	42
3.5.4 The Experimental Procedure.....	42
3.5.5 Performance Evaluation.....	43
3.5.6 Analysis of the Results.....	44

3.6 Chapter Summary .....	47
Chapter 4. k-Medoids-Outlier with SVM Classification.....	48
4.1 System Model and Problem Description .....	48
4.2 General Description of the Solution .....	49
4.3 The Proposed Approach.....	49
4.4 Experimental Results and Analysis .....	55
4.4.1 Selection of Experimental Data .....	56
4.4.2 Description and Samples of Experimental Data .....	57
4.4.3 Data Pre-processing .....	57
4.4.4 The Experimental Procedure.....	58
4.4.5 Performance Evaluation.....	58
4.4.6 Analysis of the Results.....	61
4.5 Chapter Summary .....	63
Chapter 5. Incremental Support Vector Machine with CSV-ISVM.....	64
5.1 System Model and Problem Description .....	64
5.2 General Description of the Solution .....	65
5.3 Candidate Support Vectors based Incremental SVM .....	65
5.3.1 The Improved Concentric Circle Method .....	67
5.3.2 The Half-Partition Strategy.....	70
5.3.3 CSV Selection Algorithm and CSV-ISVM Algorithm.....	74
5.4 Experimental Results and Analysis .....	76
5.4.1 Experimental Data .....	76
5.4.2 Description and Samples of Experimental Data .....	77
5.4.3 Data Pre-processing .....	77
5.4.4 The Experimental Detail .....	77
5.4.5 Performance Evaluation and Analysis .....	78
5.5 Chapter Summary .....	84

Chapter 6. Concluding Remarks.....	86
6.1 Conclusion .....	86
6.2 Future Work.....	87
<b>References .....</b>	<b>88</b>
<b>Published Papers .....</b>	<b>103</b>
<b>Acknowledgement .....</b>	<b>104</b>
<b>Appendix I : Features of Kyoto 2006+ datasets .....</b>	<b>105</b>
<b>Appendix II: Experimental Data Samples.....</b>	<b>109</b>





## ABSTRACT

With the rapid and wide-spread growth of internet technology, security risks and threats are also increasing day by day. Newer versions of attacks and intrusions are evolving continuously by putting extra challenges to the field of intrusion detection. In this present context, this thesis work proposes a hybrid approach of intrusion detection along with a hybrid architecture of intrusion detection system. The proposed architecture is flexible enough to perform intrusion detection tasks either by using a single hybrid module or by using multiple hybrid modules. The “Clustering-Outlier detection followed by SVM classification” is proposed as the first hybrid IDS module to be used in the architecture, whereas the second module proposed is the “Incremental SVM with Half-partition method”.

The Clustering-Outlier detection is an algorithm developed by this research work, which combines k-Medoids clustering and outlier analysis such that both the operations are carried out simultaneously. The selection of k-Medoids for the Clustering-Outlier detection is finalized from a simulation work / experimentation which discovers that k-Medoids clustering outperforms k-Means clustering when used for detecting anomalies in a large databases or network traffic data. The research work shows that k-Medoids consistently yields higher rates of accuracy and detection rate but lower rates of false positives in the mean time – whether it is followed by a Naïve Bayes classification or an SVM classification.

This thesis work also suggests that SVM classification is more suitable than NB classification for an IDS. For this purpose, a simulation / experiment work is carried out, in which a comparative analysis is done between the Clustering-Outlier detection followed by NB classification and the Clustering-Outlier detection followed by SVM classification. It is shown that the combination having SVM is better in terms of accuracy, detection rate and false alarm rates. Hence, Clustering-Outlier detection is then followed by a SVM classification in order to design an IDS.

In the second module to be used in the proposed hybrid IDS architecture, the Half-partition strategy is adopted with the intention to reduce the time and space complexity of incremental SVM classification. In the Half-partition strategy, the

support vectors identified in the current iteration of incremental SVM are selected and retained in a smarter way for the next iteration. By using this method, an algorithm named Candidate Support Vector (CSV) selection algorithm is developed, which works two times faster and storage space is reduced by half compared to other incremental support vector machine (ISVM) algorithms. Thus, CSV-ISVM algorithm is proposed as the final piece of this thesis work.

This and the following few paragraphs explains the problems or motivations behind this thesis work. Intrusion Detection System (IDS) has been established as the most essential and unavoidable component of the whole network security and defense system. In present context, a wide range of attacks and threats are increasing day by day along with rapidly growing network technologies and the Internet. Uncontrolled databases and web servers have been constantly targetted by intruders. Therefore, this thesis choses IDS as its major research work.

Need of applying clustering techniques like k-means and k-medoids into IDS is realized to handle big data and multimedia. Various IDS and IPS have been implemented for quite a long time for protecting and securing information, specially in network environment. Most of them work well with known attacks and work well with small data or network traffics. Due to evolution of big and multimedia databases, an IDS that is able to detect attacks from the huge data samples in an acceptably less amount of time is required.

There is a need of new techniques which are better in detecting anomalies efficiently. In recent years, data mining approaches have been proposed and used as detection techniques for discovering anomalies and unknown attacks. These approaches have resulted in high accuracy and good detection rates but with moderate false alarm on novel attacks. In addition, some attacks and normal connections are not detected correctly. Hence, there is a need to detect and identify such normal instances and attacks accurately in an interconnected network.

Most of IDS related works are focused in increasing accuracy and detection rates. As a consequence in due course, many approaches give rise to false positives and also fail sometimes in detecting new threats like zero-day attacks. Hence, an outlier analysis is felt necessary in order to detect new anomalies very efficiently and also to

help reduce false alarms in classifying the attacks. So, this thesis develops an algorithm that combines clustering with outlier detection.

Time complexity has always been a major concern in IDS. In case of big and multimedia data traffics, online detection generally takes longer time thus compromising the performance of the network speed. Considering this problem, the classification of normal and abnormal data traffic should be done in as less time as possible. And therefore, a faster classification method like Naïve Bayes (NB) classification or SVM becomes really necessary. This thesis addresses this necessity with better SVM approaches.

NB classifiers are based on a very strong independence assumption with fairly simple construction. They work fine with good data distribution. When NB is combined with k-Medoids clustering, time complexity increases as the size of data grows. Therefore, to address the time complexity, Naïve Bayes classification could be replaced with a better unsupervised learning method e.g. Support Vector Machine (SVM) that can produce high detection rate with a small-sized data distribution. Moreover, the time consumed by SVM should also be reduced to give it an extra performance and thus an idea of designing a new algorithm is justified. Therefore, this thesis improves the time consumption of incremental SVM classification by inventing newer methods.

As newer threats and attacks are coming and new security scenarios are developing day by day, it has become a compulsion for an IDS to learn continuously over every new network scenario. Not surprisingly, SVM classification also needs an incremental technique to be incorporated before using it in an IDS. Therefore, a new Half-partition method is suggested in reducing the time taken in incremental SVM classification.

Moreover, implementation aspect of IDS should also be taken into consideration. Since network attacks are quite unpredictable, the security infrastructure in which the IDS is implemented should be flexible enough to incorporate necessary techniques in required ways. Such an IDS architecture, therefore, should also be sought in order to provide a number of options and combinations of detection techniques or components.

With the afore-mentioned motivation, this research is carried out with the aim to provide a flexible IDS infrastructure and propose a hybrid IDS with k-Medoids-Outlier method and incremental SVM classification scheme. Other objectives of this

work are : - (1) To detect intrusion in real time, (2) To guarantee the predictability of the model, (3) To handle infrequent patterns, and (4) To reduce false alarm rates.

In order to meet these objectives, special attention is paid to make sure that the amount of time taken to build the model and detect the anomalies does not create extra overheads to the web servers. Predictability of the model is tested to make sure that it can always produce the desired accuracy in detecting attacks. And also, the proposed model is able to handle infrequent normal patterns or anomalies and learn also from them in order to carry out correct classification. Moreover, iterative detection technique is used in the proposed model to minimize the false alarm rates.

The methodology adopted by the thesis are explained ahead. This thesis first carries out a comparative study of k-Means and k-Medoids clustering technique in order to find out which one is most suitable for an IDS in real time. For this, each clustering is followed by a Naïve Bayes classification method and results are analysed based on intrusion detection parameters.

It also designs an algorithm called “Clustering-Outlier Detection algorithm” that unifies k-Medoids clustering and outlier detection technique by keeping the clustering quality of k-Medoids intact and without increasing the time complexity of the algorithm. Then this algorithm is combined with a classification method to be used in an IDS.

This work also carries out a comparison between NB and SVM classification by applying a simple simulation / experiment method to see whether SVM can perform quickly (using as less data sample as possible) than NB.

This work modifies and improves incremental SVM classification, in which the newly proposed “Half-partition strategy” selects and retains “Candidate Support Vectors (CSV)” sets. A new algorithm named “Candidate Support Vector based Incremental SVM” or CSV-ISVM algorithm implements the proposed strategy.

Separate experiments are carried out for different pieces of approaches and research works. Combined experiments also are done wherever necessary. Types of experiments include and not limited to data pre-processing and extraction, clustering, outlier detection, classification and cross validation etc. The data set used in all the experiments is Kyoto2006+ data sets. The experiments related to clustering and

outlier detection techniques are evaluated on the basis of clustering quality and execution time. They are compared with other similar methods and the methods proposed by this research work have been found better. In case of experiments related to classification methods, the evaluation criteria are performance, accuracy, detection rate and false positive rate of the classification scheme as well as the execution time, in some cases.

Consequences of the research works show that : -

*The k-Medoids clustering technique followed by Naïve Bayes classification method, in case of large data sets, is proven to be more significant than k-Means clustering in terms of accuracy and detection rate. The method also reduces the false alarm rate in the mean time.*

Combination of SVM classification with k-Medoids-Outlier detection method produces better accuracy, detection and false alarm rates. This approach is shown to be better than the combination of k-Medoids with Naïve Bayes classification.

The new algorithm CSV-ISVM method that implements the proposed Half-partition strategy is shown to perform double faster with just half the data samples (support vectors) than other similar incremental SVM methods.

All the proposed approaches and research works have enhanced the detection rate with minimum false positive rates. The proposed algorithms e.g. Clustering-Outlier Detection algorithm and CSV-ISVM are also tested and compared experimentally with other similar methods and are found better to be used by IDS in real-time environment. These proposed methods can be used for network intrusion detection in real-time because of its higher detection rate, improved false alarm rate as well as acceptably less amount of learning time.

**Keywords:** Hybrid Intrusion Detection, Clustering-Outlier Detection, Incremental SVM, Candidate Support Vector, Half-Partition Method.

## 摘要 (Summary in Chinese)

随着互联网技术广泛和迅速发展, 安全隐患和威胁也日益严重。不断出现的新的攻击和入侵对入侵检测领域提出了新的挑战。在这个背景下, 本论文提出了一个入侵检测混合方法和入侵检测系统混合结构。本文所提出的架构十分灵活, 既可以使用单一混合模块, 也可以使用多个混合模块来完成入侵检测任务。该架构主要包括两个混合 IDS 模块, 即聚类离群探测和 SVM 分类模块, 以及具有半隔离方法的增量式 SVM 模块。

本文提出的聚类离群点检测算法结合了 k-Medoids 和离群分析, 使得这两种分析可以同时进行。为离群-聚类探测选择 k-Medoids 是因为通过实验发现在大的数据库或网络流量很大的情况下, k-Medoids 聚类优于 k-Means 聚类。研究工作表明, 不管是使用贝叶斯分类还是 SVM 分类, k-Medoids 总能获得较高的精度和较低的误报率。

本文的工作也表明, SVM 分类比贝叶斯分类更适合比较合适 IDS。对此本文进行了一个模拟实验来对结合贝叶斯分类的聚类-离群和结合 SVM 分类的聚类-离群进行比较分析。实验表明, 结合 SVM 的方法取得了较好的精度、探测速率和误报率。因此, 使用聚类-离群探测结合 SVM 分类来设计一个 IDS。

所提出的混合 IDS 架构中的第二个模块采用半分隔离策略来减少增量式 SVM 分类的时间和空间复杂度。在半分隔离策略中, 增量式 SVM 当前迭代中的支持向量被选择, 并以一种智能的方法保留到下一次迭代。以这种方式开发的候选支持向量机 (CSV) 算法, 其速度是其他增量式支持向量机 (ISVM) 的两倍, 而存储空间减少了一倍。因此本文最后部分提出了 CSV-ISVM 算法。

入侵检测系统 (IDS) 在信息和网络安全领域, 特别是对于构建良好的网络防御设施起着不可替代的作用。签名检测技术和异常检测技术是构建这种设施的两个主要模块。由于安全威胁日益严重, 所以异常检测变得更加重要。近年来的异常检测技术主要基于一些分类学习方法, 如贝叶斯 (Naïve Bayes) 分类和支持向量机 (SVM) 分类等。为了使分类的结果更加准确、有效, 通常也采用与数据挖掘技术相结合的混合方法。此外, 增量式学习方法也用于每一个增量学习阶段以得到更好的检测率, 而支持向量机分类也不例外。在一个增量式支持向量机 (SVM) 分类中, 前一阶段已分类的数据对象被标记为非支持向量, 并与使用卡罗需-库恩-塔克 (KKT) 条件验证的新数据样本一起, 作为下一分类阶段的训练数据。

入侵检测被认为是一个分类问题。因此，分类的重要性是毋庸置疑的。对于一种准确和实时的检测方法，学习方法也必须是快速的，并包含高度准确的知识。显然，这样的学习方法需要足够的训练数据，并且应该进行多个阶段的学习。为了处理用于训练的大量数据，可以利用聚类分析技术。此外，为了在整个检测过程中实现实时检测，还需要创建有效的资源和快速的算法。因此，使用一个实时参数“可预见性”来评价整个检测系统。

混合模型入侵检测系统的应用范围不受限制。该方法可以在任何具有高速出入流量和大量用户的网络中实现。该方法可以用于数据交换量大而数据安全和隐私需要严格保护的网路。但本文提出的方法在要求实时和嵌入式系统的环境中有特殊的适用范围。

入侵检测系统(IDS)作为整个网络安全和防御系统最基本和必需的组件已经建立。目前，随着网络技术和互联网的迅速发展，各种网络攻击和威胁日益增加。数据库和 Web 服务器不可控制地成为入侵者不断攻击的目标。因此，本文将 IDS 作为研究的主要工作。

受上述思路启发，本文的研究目标是要提供一个灵活的 IDS 架构和提出一个基于 k-Medoids-Outlier 方法和增量式 SVM 分类架构的混合 IDS。本文的其它研究目标还包括：（1）实时入侵检测；（2）保证模型的可预测性；（3）处理非频繁模式和（4）减少误报率。

通过研究，这里提出的解决方案解决的问题如下： -

问题 1：如何检测大流量情况下的网络入侵？

目标：（1）找到一个合适的聚类技术进行实时检测；（2）找到一个合适的分类方案用于实时入侵检测系统。

解决方案：k-medoids 聚类技术和支持向量机分类的方案选择

解释：线性分类的可预测性不能得到保证。因此，对离线的可预测性也进行了分析，确保模型在持续的良好水平下进行检测工作。因此，使用聚类和分类技术。

问题 2：罕见的流量模式被忽视或破坏

目标 1：检测不同数据的模式，是在一个小的部分发生

目标 2：提高检测率的同时，降低假阳性

解决方案：k-medoids 结合离群点检测和增量式支持向量机

解释：未能赶上罕见模式的正确结果在更多的假警报率的产生。为了解决这个问题，这里结合 **k-medoids** 聚类与孤立点检测方法。为了进一步降低误报率，应用 **SVM** 增量迭代法。

问题 3：检测的时间应该很小，而精度很高

目标 1：最大限度地检测异常入侵检测的准确率

目标 2：降低入侵检测到尽可能少的时间。

解决方案：**SVM** 选择策略的实施和设计半分隔方法

解释：采用改进的选择支持向量的策略有助于提高分类准确率和使用半分隔策略减少了检测过程中所耗费的一半的时间。

为了实现本文的研究目标，整个研究工共分为四个部分：

(1) 为入侵检测系统设计一个混合架构。本文提出的一个 4 层框架为整个研究工作所必需进行的所有实验和模拟提供了合适的框架。1.4 节介绍了该架构的结构和工作原理。

(2) 进行 **k-Means/Medoids** 聚类和贝叶斯分类实验，该实验表明，在大数据集的情况下，该 **k-medoids** 聚类优于 **k-Means** 聚类。首先进行一个结合贝叶斯分类的 **k-Means** 聚类，然后进行了一个相似的实验，但采用 **k-Medoids** 聚类。最后对两个实验在 **IDS** 中的应用进行了分析与比较。

(3) 进行一个实验来判别当结合 **k-Medoids-Outlier** 探测技术时，**SVM** 分类和贝叶斯分类的性能。在该过程中，设计了一个结合 **k-Medoid** 聚类和离群技术的算法。该算法将 **k-Medoid** 聚类扩展到 **k-Medoids-Outlier** 探测技术。使用这个新技术将贝叶斯和 **SVM** 分类结合，分别完成了两个独立的实验。和之前的实验一样，对这两个实验在中的应用进行了分析与比较。

(4) 使用 **k-Medoids** 聚类技术进行增量式 **SVM** 分类，同时提出了基于半隔离策略的支持向量选择策略，并设计了该策略的算法实现。在此，**SVM** 分类扩展为通过选择支持向量和为下一次迭代保留支持向量而迭代和增量地进行工作。由于迭代过程增加的时间通过采用半分隔策略的 **CSV-ISVM** 来减少。最后也比较了实验与其他相似方法的在 **IDS** 中的应用。

应用聚类技术，如 **k-means** 和 **k-medoids** 来处理大数据和多媒体数据已引起广泛关注。长期以来，为了保证信息，尤其是网络环境下的信息安全，已经实现了各种 **IDS** 和 **IPS**。大部分 **IDS** 和 **IPS** 系统对于常见的网络攻击和在数据



量较小或者网络流量较小的情况下工作得很好。但是，随着大数据和多媒体数据库的出现，建立一种能在可接受的时间范围内从大量数据样本中检测网络攻击的 IDS 成为亟需。

通过使用一种更好的检测技术能有效地检测异常。近年来，已提出使用数据挖掘方法作为一种检测技术来发现异常现象和未知攻击。这些方法有很高的检测精度和检测速率，但对于新的攻击存在着一定的误报率。此外，这些方法也不能对某些攻击和正常连接进行正确区分。因此，需要一种方法在互联网中准确地检测和识别正常连接与攻击。

大部分 IDS 相关工作聚焦于提高检测精度和速率上，但随之导致许多方法产生了假阳性，有时也不能检测到新的威胁，如零日攻击。因此，需要使用离群分析方法来有效地探测新的异常和帮助 IDS 在进行攻击分类时减少误报率。本文设计了一种结合聚类和离群检测的算法来实现该目标。

时间复杂度一直是 IDS 关注的主要问题。在传送大量数据和多媒体数据的情况下，在线检测通常会花更长的时间，从而降低网络速度的性能。鉴于此，正常和异常数据流量的分类应该在尽可能少的时间内完成。因此，需要一种更快的分类方法，如贝叶斯方法（NB）或支持向量机方法（SVM）。本文采用更好的 SVM 方法来解决这个问题。

NB 分类器基于一种很强的独立假设来构造，该方法非常简单。在数据分布良好的情况下，NB 工作得非常好。当 NB 和 k-Medoids 聚类结合后，其时间复杂度会随着数据量的增加而增加。因此，为了处理时间复杂度，可以使用一种更好的无监督学习方法，如 SVM 来代替 NB。由 SVM 消耗的时间也应该减少以获得额外的性能，设计一种新算法来解决这个问题被证明是可行的。本文通过提出一种新的算法，改进了增量式 SVM 分类的时间消耗。

随着新的网络威胁和攻击出现，以及新的安全场景日益形成，要求 IDS 在各种新的网络场景下必需不断学习。SVM 分类在应用于 IDS 之前也需要集成一种增量技术，因此，使用一种新的半隔离方法来减少增量式 SVM 的时间消耗。IDS 的实现也是需要考虑的问题。因为网络攻击不可预测，IDS 中的安全框架实现应该足够灵活，在必要时能集成所需的技术。因此，应该寻找一种能够提供多种选择并能结合检测技术和组件的 IDS 架构。

综上所述，本研究的主要目标是提出一种灵活的 IDS 框架，并结合 k-Medoids-Outlier 方法和增量 SVM 分类方法设计一种混合的 IDS 架构。本研究的

其它目标还包括：（1）检测实时入侵；（2）保证模型的可预测性；（3）处理非频繁模式；（4）减少误报率。

为了达到这些目标，本文着眼于模型构建和异常检测所花费的时间量对于 Web 服务器不会产生额外的代价。同时，进行模型的可预测性测试以保证模型能获得预期的检测精度。而且，本文提出的模型能处理非频繁的正常模式或者异常现象，并能通过学习进行正确的分类。该模型还使用了迭代检测模型来降低误报率。

论文的全部工作主要包括如下内容：

本文首先使用混合学习方法、基于 k-Medoids 聚类技术以及贝叶斯 (Naïve Bayes) 分类方法来研究入侵检测 IDS。由于 k-Medoids 聚类技术表示了数据的真实分布，因此基于 k-Medoids 提出的将所有数据分组到相应聚族的扩展方法比基于 k-Means 的聚类方法更精确，从而得到一个较好的分类结果。

本文进一步的工作是设计了一个基于聚类和离群探测技术的组合算法，称为“聚类-离群探测算法”。使用该算法，可以保持聚类的高度可预测性，因此可以使用该新算法来代替传统的 k-Means/Medoids 算法。然后，本文提出了一种结合 k-Medoids 聚类和 SVM 的算法，该算法能够减少聚类-离群探测算法所需的大量样本，而保持高质量的聚类结果。

最后，本文提出了改进增量 SVM 分类的方法，该方法最新提出了“半隔离”策略，该策略选择下一增量分类阶段可能成为支持向量的非支持向量作为候选支持向量 (CSV)。本文也设计了一个基于候选支持向量的增量式 SVM (CSV-ISVM) 算法，该算法实现了提出的策略和增量式 SVM 分类过程。本文还提出了对同心环方法和保留集策略进行改进的方法。

本文对不同的方法和工作进行了独立实验。实验类型包括但不限于数据预处理和抽取、聚类、离群检测、分类和交叉验证等。所有的实验均采用 Kyoto2006+数据集。聚类和离群检测实验结果使用聚类质量和执行时间指标进行评价，并与其它相似方法进行比较。结果表明，本文提出的方法获得了较好的结果。而分类实验的结果则使用性能、精度、探测率和误报率，以及在某些情况下使用执行时间指标来进行评价。

结果表明，本文的创新点有以下几个方面：

应用 k-medoids 聚类技术，而不是 k-Means，结合 Naïve Bayes 分类算法，提高了检测精度、增加了检测率同时降低了误报率。这在大型数据集的情况下尤为重要。

将支持向量机与 k-Medoids 结合获得了较高的精度、探测率和较小误报率。这种混合方法被证明优于 k-Means/k-Medoids 与贝叶斯 (Naïve Bayes) 相结合的方法。因此该方法更加有效和高效。

CSV-ISVM 结合了一个改进的同心环方法和“半隔离”策略。实验证明，与其它相似的增量式分类方法比较，该方法更加有效和快速。

本文提出的所有方法均提高了检测率，降低了误报率。所提出的算法，如聚类离群点检测算法和 CSV-ISVM，和其它的相似方法也进行了实验和比较。结果表明 IDS 在实时环境下取得了较好的应用效果。由于所提出的方法具有较高的检测率，误报率以及可接受的学习时间，因此可用于实时环境下的入侵检测。

本文的研究工作为整个研究工作的框架提出了一种实时混合入侵检测系统的体系结构。这项工作也提出了两种不同的入侵检测方法，即：（1）聚类的孤立点检测和 SVM 分类；（2）半分隔方法的增量式支持向量机分类，单独或组合使用取决于如何探测入侵检测。第一种方法的实验和分析表明，聚类和孤立点检测后并与支持向量机分类结合，不仅提高了检测性能，检测准确率，减少误报率，也保证了检测参数的可预测性。这表明，这种方法可以在实时入侵检测中使用。

第二种方法，即增量式支持向量机半分隔方法，被证明是更好的方法。除了上面提到的所有检测参数，在执行时间上也是最少的。通过提出半分隔方法和将它 CSV-ISVM 算法中实现，使入侵检测的正常时间已经减少到几乎一半。已经在 Kyoto+ 2006 数据集进行了一些实验，分别进行了第一种和第二种方法，同时与其他类似的方法进行了比较。实验分析结果表明，两种方法在性能、精度、检测率、误报率以及实时参数如时间可预测性等方面都具有优于已有的方法。

本文的研究工作对提出的两种算法，即（1）聚类的孤立点检测和（2）候选支持向量 - 增量式支持向量机也进行了实施。同时，也提出了以统一的离群点检测和保留策略，在增量型的支持向量机分类选择支持向量中的改进算法。但是本论文的研究工作尚有很多方法需要进一步改进的地方。首先，耗时的聚

类方法可以考虑通过保持其孤立点检测的地方来改进。下一步的改进可以使用多个支持向量机的核函数。此外，半分隔方法也可推广到多支持向量机分类。

关键词: 混合入侵检测; 聚类-离群点检测; 增量式支持向量机; 候选支持向量; 半隔离的方法。

# Chapter 1. Introduction

## ***1.1 Background and Significance of the Research***

The role of Intrusion Detection System (IDS) has been inevitable in the area of Information and Network Security – specially for building a good network defense infrastructure. Signature based detection and anomaly based detection techniques are the two building blocks of such a foundation. Among them, the latter has been become more important as security threats have been increasing day by day. Anomaly based intrusion detection, in the recent years, has become more dependent on learning methods, mainly on classification methods like Naïve Bayes and Support Vector Machine. To make such classification more accurate and effective, hybrid approaches of combining data mining techniques are commonly being introduced. In addition to these, incremental learning approaches are also applied so as to get better detection rates in each increment of the learning stages; and the SVM classification too does not remain as an exception. In an Incremental SVM classification, the data objects labelled as non-support vectors by the previous classification are re-used as training data in the next classification along with new data samples verified by Karush-Kuhn-Tucker (KKT) condition.

Intrusion Detection is considered to be a classification problem. So, the importance of classification is unquestionable. For a detection method to be an accurate and a real-time one, the learning must also be fast and contain high-precision knowledge. For such learning, the method obviously requires enough training data and should pass through multiple learning phases. In order to deal with big data for training purpose, the use of clustering techniques is advised. Moreover, resource-efficient and fast algorithms are needed to be built in order to make the whole detection process work in real-time. Therefore, one real-time parameter “Predictability” has been introduced in evaluating the whole detection system.

Following are some of the significances of this research work: -

1. For the detection to be an accurate and a real-time one, the learning must also be fast and contain high-precision knowledge. For such learning, the method obviously requires enough training data and should pass through multiple learning phases. In order to deal with big data for training purpose, the use of

clustering techniques is advised. Moreover, resource-efficient and fast algorithms are needed to be built in order to make the whole detection process work in real-time. One extra parameter “Predictability” has been introduced in evaluating the whole detection system so as to claim it as Real-Time Intrusion Detection System (RT-IDS).

2. IDSs can detect serious security threats as well as unwanted activities like gaining unauthorized access to files and network resources. While an IDS is also capable of sending early alarms upon risk exposure caused by any attack, at the same time it has also potential to generate high volume of false alarms.
3. Although anomaly based techniques quantitatively describe characteristics of network behaviour to distinguish normal behaviour from abnormal behaviours that are potentially intrusive, these abnormal behaviours cannot always be confirmed to be abnormal. So, abnormal activity set and intrusive activity subset should be established. And, anomaly detection has higher universality in finding out yet-undetected attacks.

The scope of the IDS with hybrid model is not limited. The approach can be implemented in any network where incoming and outgoing traffic rates are high and public users are large in number. The approach may well be used in those networks where exchange of data is considered to be high and data security and privacy need to be strictly maintained. The proposed research will have special scope in any network environment where Real-Time and Embedded Systems are used.

The whole thesis document is organized in six (6) chapters preceded by the Abstract and Table of Contents on the top and followed by the References at the bottom. The Abstract summarizes the four different aspects including background and value of the thesis topic, contents of the thesis in brief, research methodology used in the thesis work, and macro result of the thesis work.

Chapters in this document are further divided into sections and sub-sections depending upon how big chapters are and what chapters are about. The first Chapter is about the introduction of the thesis, which comprises of research background and its significance, contents of the research, research contributions, etc. Chapter 2 presents the works related to this thesis work and describes the current status of the research work around the globe.

Chapter 3, Chapter 4 and Chapter 5 focuses on the research works carried out by the author in order to meet the objectives mentioned above. Different pieces of research works are explained in detail with their experimental verifications and theoretical analyses in separate chapters. Chapter 3 presents the research work on intrusion detection based on hybrid learning method by combining k-Medoids clustering and Naïve Bayes classification. In the following chapter, the detailed research work on anomaly based intrusion detection using k-Medoids clustering and SVM classification is presented. In Chapter 5, research work on selection of candidate support vectors in incremental SVM classification is elaborated and illustrated. In the final chapter, the conclusion of the whole thesis work is presented along with the possible further enhancements and future works.

## **1.2 Research Objectives**

### **Broader Objective: -**

The broader objective of this thesis work is to propose a hybrid IDS with k-Medoids-Outlier detection and incremental SVM classification along with its suitable architecture.

### **Specific Objectives: -**

More specific objectives are as follows: -

1. *To detect intrusion in real time* : Special attention is paid to make sure that the amount of time taken to build the model and detect the anomalies does not create extra overheads to the web servers.
2. *To guarantee the predictability of the model*: Predictability of the model is tested to make sure that it can always produce the desired accuracy in detecting an attack.
3. *To handle infrequent patterns*: The proposed model handles infrequent normal patterns and learns also from them in order to carry out correct classification.
4. *To reduce false alarm rates*: Iterative detection technique is used in the proposed model to minimize the false alarm rates.

### **1.3 Solutions to Problems and their Contributions**

The problems addressed by the research work and their proposed solutions are represented below by means of a Problem-Solution diagram. As seen from Table 1-1, there are mainly three major problems that can be considered to be solved in general. They are: - (1) How to detect intrusions in large network traffic; (2) Infrequent traffic patterns are overlooked or undermined; and (3) Detection time should be very small with higher accuracy.

Each main problem is further broken down into two goals which are achieved by providing the proper solutions. The goals of the first problem are (1) To find out a predictable clustering technique for huge data and (2) To find out a quick multi-dimensional classification method. As a solution to the first goal, k-Medoids Clustering technique is selected and to the second goal, SVM classification is selected. Predictability of online classification cannot always be guaranteed. Therefore, an offline predictability analysis is also carried out to ensure that the model will perform the detection work in constantly good level. And, hence, both clustering and classification techniques.

Similarly, the second main problem also has two goals viz. (1) To detect data patterns which are occurring in a small portion; and (2) To increase detection rate by decreasing the false positives. To achieve the first goal, k-Medoids is combined with outlier detection; and an incremental SVM is applied to obtain the second goal. Failing to catch infrequent patterns correctly results in the generation of more false alarm rates. To address this problem, the k-medoids clustering is combined with outlier detection method. To further reduce it, the iterative approach of incremental SVM is applied.

Lastly, the two goals of the third main problems are: - (1) To maximize the rate of accuracy in detecting anomalies and (2) To decrease time in detecting attacks. Solution provided to achieve the first goal is the implementation of support vector selection strategy; and that for the second goal is the development of Half-partition method. By applying an improved strategy of selecting support vectors help increase the accuracy rate of classification and with the design and use of Half-partition strategy reduces the amount of time taken in the detection process.



Problems and their proposed solutions are summarized in the following table: -

**Table 1-1: Problems in the research work and Solution to them**

<i>Problems</i>	<i>Goals</i>	<i>Solutions</i>
How to detect intrusions in large network traffic?	Find out an predictable clustering technique for huge data	k-Medoids Clustering technique selected
	Find out a quick multi-dimensional classification method	SVM classification selected.
Infrequent traffic patterns are overlooked or undermined	Detect data patterns which are occurring in a small portion	k-Medoids is combined with outlier detection
	Increase detection rate by decreasing the false positives	Incremental SVM is applied
Detection time should be very small with higher accuracy	Maximize the rate of accuracy in detecting anomalies	SV selection strategy is implemented
	Decrease time in detecting attacks	Half-partition method is developed

In order to achieve the goals established by the thesis work, the entire research work has been divided into four parts: -

1. *Designing a hybrid architecture for intrusion detection system.* A 4-tier architecture is proposed in order to provide the entire thesis work a suitable infrastructure to carry out all necessary experiments and simulations. Section 1.4 describes how the architecture looks like and how it works in detail.

2. *Carrying out an experiment of intrusion detection using k-Means/Medoids clustering and NB classification*, which shows superiority of k-medoids in case of large data sets. First, an experiment of k-Means with NB classification is carried out. Secondly, a similar experiment, but with k-Medoids, is carried out. Both experiments are analysed and compared to each other for its applicability in IDS.
3. *Performing an experiment that justifies the SVM classification against NB classification when combined with k-Medoids-Outlier-Detection technique*. An algorithm that unifies k-Medoids clustering and outlier detection is also devised in the due course. This time, k-Medoid clustering is extended to k-Medoids-Outlier detection technique by unifying with clustering technique. With this new technique, NB and SVM classifications are combined, one at a time, and carry out two separate experiments. Like in previous case, both the experiments are analysed and compared for use in an IDS.
4. *Carrying out an experiment of incremental SVM classification with k-Medoids-Clustering technique*. Also, a support vector selection strategy called Half-partition strategy is developed and an algorithm to implement it is also devised. Here, the SVM classification is extended to work iteratively and incrementally by selecting support vectors and retaining them for the following iterations. The increased time due to iterative process is decreased by applying CSV-ISVM algorithm that adopts Half-partition method. Finally, the experiment also is analysed and compared with similar methods for application in IDS.

Following are the results of the thesis work: -

1. Application of k-Medoids clustering technique, instead of k-Means, followed by Naïve Bayes classification method is proven to be better in terms of detecting accuracy as well as increasing the detection rate by reducing the false alarm rate at the same time. In case of large datasets, the approach is seen to be more significant.
2. Combination of SVM classification with the better (proven in the first piece of work) k-Medoids clustering produces higher accuracy, detection and false alarm rates. This proposed hybrid approach is shown to have outperformed the

first approach of combining k-Means/k-Medoids with Naïve Bayes. Intrusion detection, thus, could be more effective and efficient.

3. The CSV-ISVM method, which incorporates an improved Concentric Circle method and the new Half-partition strategy, is proven to be more efficient and faster than other similar incremental classification techniques.

The following are the contributions of the Research: -

1. Discovers that k-Medoids clustering outperforms k-Means clustering when data samples are large:

As a part of the research work of thesis, it shows that Accuracy and Detection Rate are increased whereas False Alarm Rate is decreased in k-Medoids followed by Naïve Bayes for every size of data samples. Both increase remarkably when the size of the data samples exceeds a certain value and remain almost constant thereafter. Therefore, the fixed size can be used for training and hence the time taken can also be predicted.

2. Develops an algorithm that combines clustering technique and outlier detection:

The algorithm named as Clustering-Outlier algorithm is devised in order to perform outlier analysis and hence detect in-frequent patterns too. The algorithm is designed in such a way that it takes advantage of computational iterations of the clustering method, i.e. outlier detection is done within the time period of clustering process.

3. Develops a better time-efficient Incremental SVM Classification method:

By developing the Half-partition strategy for selecting candidate support vectors in incremental SVM classification methods, the real time applicability of the RT Hybrid model proposed in this thesis is extended. This piece of research work demonstrates that the time taken in learning and classification of attack data by normal incremental SVM classification is reduced by half, and thus making the approach appropriate for using in real-time intrusion detection.

## 1.4 Thesis Organization

### Hybrid IDS Architecture

In order to achieve the aforementioned broader and specific objectives, a Hybrid IDS System Architecture has been proposed. The proposed architecture, shown in Figure 1-1, consists of four tiers namely: -

1. Data Acquisition tier
2. Intrusion Detection tier
3. Intrusion Protection tier
4. Interfacing tier

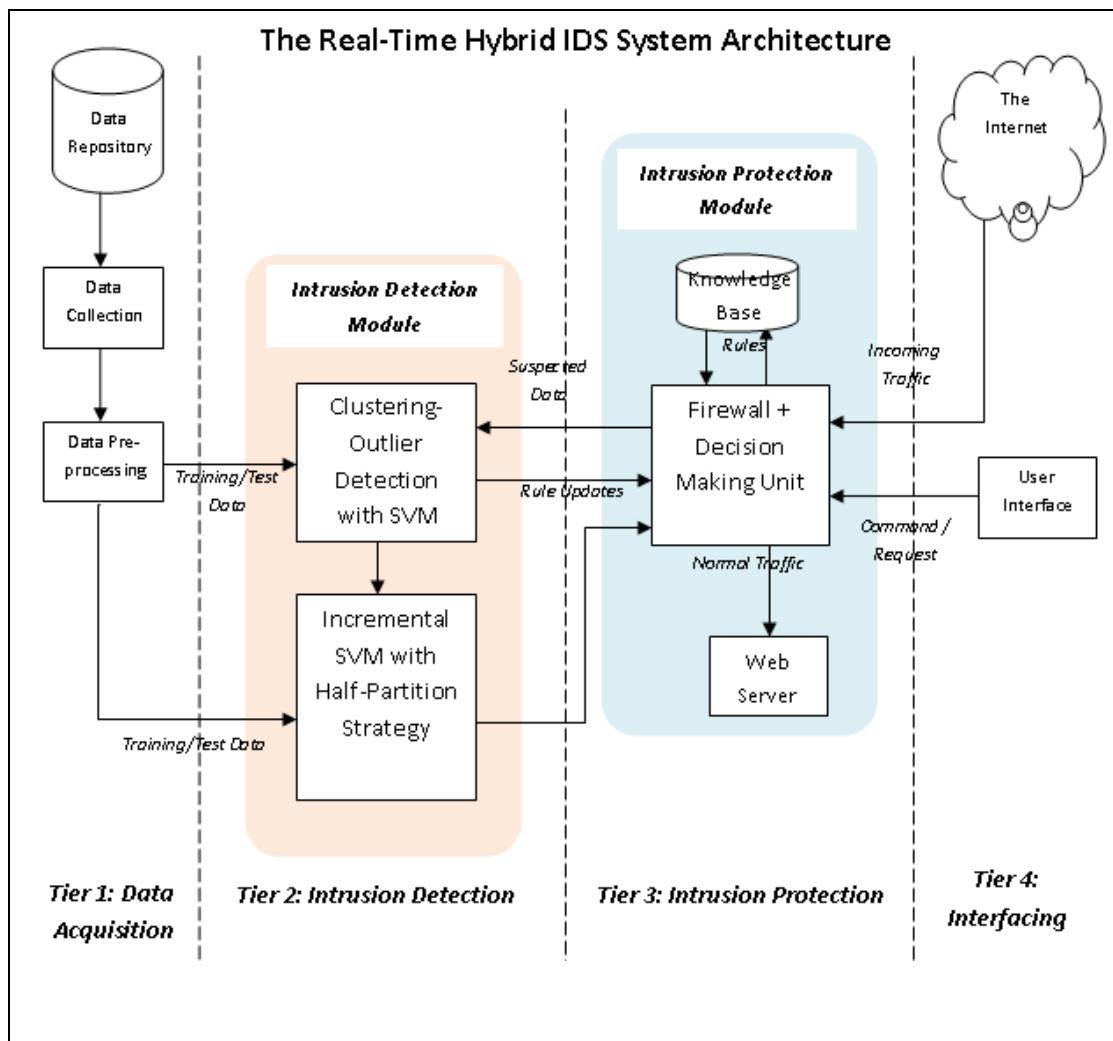


Figure 1-1: Real-Time Hybrid IDS System Architecture

The first tier i.e Data Acquisition tier collects data necessary to train, test and build the IDS model. The second tier focused in building the intrusion detection system. The tier is responsible to take necessary security decisions based on the results from tier 3 and also to maintain rules databases. Tier four provides necessary interfaces to the user and to connect to the internet.

### **Specialty of the RT Hybrid IDS System Architecture**

Among the four tiers, Tier 2: Intrusion Detection, which is the main focus of this thesis work, contains the Hybrid IDS module that further comprises of two sub-modules. The first sub- module is Clustering-Outlier-SVM that is the combination of Clustering method and Outlier detection method followed by SVM Classification. The second sub-module is an improved and modified version of Incremental SVM Classification. These two sub-modules may be used exclusively or in combination. Nevertheless, there is no doubt that two sub-modules in combination yield the best performance of Intrusion detection in real time.

Figure 1-1 further illustrates in detail the contents of both two sub-modules of Hybrid IDS Module. The Clustering-Outline-SVM sub-module combines k-Medoids clustering and Outlier detection method by using the unified Clustering-Outlier algorithm and then implements SVM classification. The second sub-module CSV-ISVM based SVM makes use of CSV-ISVM Algorithm in implementing Incremental Support Vector Machine classification. The details of these methods, algorithms, related experiments and analyses are explained one-by-one in Chapter 3, Chapter 4 and Chapter 5.

## Chapter 2. Literature Review

### **2.1 Related work in IDS developments**

According to S. R. Pampattiwar and A. Z. Chhangani, the IDSs developed so far were vulnerable to the users from inside and outside i.e. both external and internal attacks [1], which was mentioned in their research work entitled “IDS Hybrid Intrusion Detection System Using Snort” [2]. They also proposed a new anomaly detection module to design a hybrid IDS by extending the functionality of Snort IDS.

Various research works have defined IDS in different ways. There is no exact match about which security tool is considered as IDS or a part of it. One type of literatures say that not every kind of security tools are IDSs [3]. For instance, Network logging systems [4], Vulnerability assessment tools [5], Anti-virus products (although very close to intrusion detection systems, only provide a security breach detection services [6]), Firewalls (which works in combination with IDS to enhance network-wide security [7], Security/cryptographic systems and etc. are not IDSs.

Amoroso said in his article [8] that an IDS checks for suspicious activities in a network intrusion detection and defined IDS as a process of identifying and responding to malicious activity targeted at computing and networking resources [9]. He also classified attacks against computer networks and hosts into two types - misuse based IDS and anomaly based IDS [10]. While some says that IDSs deal with hacking breaches and, hence, define some dangerous activities [11]. Intrusion, Incident and Attack are taken into consideration while modeling of intrusions. An intruder initiates an attack with an introductory action. And then, he follows number of auxiliary actions or evasions in order to succeed it completely.

Phurivit Sangkatsanee et. al [12] proposed a practical real-time intrusion detection method using machine learning approaches, in which they described ways to make an IDS able to perform anomaly detectin in real-time.

Santhosh. S and Radha. R mentioned in their work that attacks, which can be either passive or active [13], are described as unauthorized access to the resources. They may be identified by the source category, namely those performed from local network, the Internet or from remote sources [14]. The following types of attacks, for example,

can be identified as one of them: - Password cracking and access violation, Trojan horses, Interceptions like man in the middle attacks[15], Spoofing [16], Scanning ports and services [17], Remote OS Fingerprinting, Network packet listening, Stealing information, Authority abuse, Unauthorized network connections, Usage of IT resources for private purposes, Unauthorized alteration of resources, Falsification of identity, Information altering and deletion, Unauthorized transmission and creation of data (sets) [18], Unauthorized configuration changes to systems and network services (servers), Denial of Service (DoS), Flooding, Distributed Denial of Service (DDoS), Buffer Overflow, Remote System Shutdown, Web Application attacks etc [19].

Phil Bandy et. al explained in their work that anomaly detection draws more attention of researchers compared to misuse detection because newer threats and an unknown attacks are appearing daily due to rapidly developing Internet technologies [20]. Majority of the researchers refer anomaly based intrusion detection as a classification problem in which ideas from machine learning is combined.

Maryam Hajizadeh and Marzieh Ahmadzadeh proposed a data mining based framework for detecting intrusions [21]. Within the framework the whole security systems related to an IDS can be implemented. They also showed that IDS architecture could be constructed by utilizing different data mining approaches.

## ***2.2 Developments in k-Means/Medoids and Outlier Techniques***

According to Adinehnia, Reza et. al [22], many recent approaches to intrusion detection have applied data-mining algorithms to large data sets of audit data collected by a system. These models have been proven effective, but the data required for training is very expensive. Data mining based IDSs collect data from sensors that monitor network activities, system calls used by user processes, or file system accesses [23]. Data gathered by sensors are evaluated by a detector using a detection model [24].

As Vineet Richhariya and Nupur Sharma explained in their works, data mining is the latest technology introduced in network security environment to find regularities and irregularities in large datasets [25] [26]. The best possible accuracy and detection rate can be achieved by using Hybrid learning approaches [27]. Different classifiers can be

used to form a hybrid learning approach such as combination of clustering and classification techniques [28].

By summarizing viewpoints of Murad Abdo Rassam et.al, it can be written that clustering is an anomaly-based detection method that is able to detect novel attack without any prior notice and is capable to find natural grouping of data based on similarities among the patterns [29]. Mining tools like k-Means and DBScan could also be used to efficiently identify a group of traffic behaviors that are similar to each other using cluster analysis [30]. Due to its simple structure, Naïve Bayes classifier is more efficient and can produce very competitive results in detecting anomaly-based network intrusion [31].

Quoting R. Luigi et.al, it is stated here that, in anomaly based IDS, clustering algorithms [32] are frequently used to detect “abnormal” behaviours. The number of clusters into which the input data may be classified is arbitrary, but as the essential goal of these systems is to distinguish between ”normal” and ”abnormal” behaviour, it is very common to partition the incoming resource access requests into two classes that correspond to these two types of behaviour [33].

T. Velmurugan and T. Santhanam [34] have analyzed the efficiency of k-Means and k-Medoids clustering algorithms by using large datasets in the cases of normal and uniform distribution; and found that the average time taken by k-Means algorithm is greater than that of k-Medoids algorithms for both the cases.

Data Mining (DM) is the technique to extract knowledge from huge data to view the hidden knowledge. It also facilitates the use of the extracted knowledge to the real time applications [35] [36] [37]. DM consists of algorithms for data analysis. Some of the major mining techniques used for analysis are Clustering, Association, Classification and etc [38].

Shalini et.al described algorithms that belong to partitioning methods and carried out comparisons [39]. According to their literature, partitioning methods creates  $k$  partitions, called clusters, from given set of  $n$  data objects. Each partition is represented by either a centroid or a medoid. A centroid is an average of all data objects in a partition, while the medoid is the most representative point of a cluster [40]. A distance measure is one of the feature space used to identify similarity or dissimilarity of patterns between data objects. The Euclidean distance has an intuitive



appeal as it is commonly used to evaluate the proximity of objects in two or three dimensional space [ 41].

In partitioning category of clustering, many techniques have been developed. Some of the popular techniques are k-means, k-medoids, Partitioning Around Medoids (PAM), Clustering LARge Applications (CLARA) and Clustering Large Applications based upon RANdomized Search (CLARANS) etc. There are hierarchical, grid-based and model-based methods too [42]. Clustering is an effective technique to search hidden patterns that exists in datasets [43].

k-means algorithm [44] is attractive, because it is simple and fast and it also minimizes the clustering error. It can also be modified to improve its efficiency [45]. However, it employs a local search procedure and, therefore, has main two limitations - one is that the number of the clusters is unknown, and the second is initial seed problem.

In the clustering analysis, each data object shares high similarity with other objects within the same cluster but at the same time, they are quite dissimilar to objects in other clusters [46]. Besides this easy, simple approach, there is also fuzzy k-Means clustering that is used for behavior pattern discovery [47].

Similary, k-medoids clustering also selects  $k$  clustering centres from data objects and sets an initial partition nearest to clustering centre for other data before iterating and moving clustering centres continuously until an optimum partition is reached [48].

K-medoids clustering algorithm can equally be used in web modelling. According to the reference [49], web model of ontology data set object is set based on algorithm analysis and selection improvement of centre point  $k$ . It shows that the improved algorithm can enhance the accuracy of clustering results under semantic web.

Petitjean, François, et al. revealed that there is also an algorithm for k-medoids clustering which works like k-means algorithm. The algorithm calculates the distance matrix once and uses it for finding new medoids at every step. The algorithm uses real and artificial data. The algorithm takes less computation time compared to PAM methods [50].

Like k-means, k-medoids can also be used in web mining, where the algorithm is applied as a reduction mechanism to partition user session data into a set of clusters.

Similar scenarios of user interactions with a web application are represented by a cluster. The algorithm then selects samples of each cluster and constructs test data. Dissimilarity data type of user sessions and their definitions are described. Number of attributes are counted in two client requests. Each client contains one basic request, none or many name-value pairs. The suite is generated by randomly selecting representative user sessions from each partitioned cluster without reconstructing or rearranging the user sessions data [51].

The algorithms - k-means and k-medoids - are examined and analyzed based on their performances [52]. According to the study, it is shown that the average time taken by k-means algorithm is greater than the time taken by k-Medoids algorithm in cases of very large data sets [53].

Sanjay Chawla and Aristides Gionis presented a unified approach for carrying out clustering and discovering outliers in data simultaneously [54]. The approach was formalized as a generalization of the k-means problem. They further extended the approach to all distance measures that can be expressed in the form of a Bregman divergence.

Rajendra Pamula, Jatindra Kumar Deka, Sukumar Nandi [55] proposed a clustering based method to capture outliers. They applied K-means clustering algorithm to divide the data set into clusters. The points which are lying near the centroid of the cluster are not probable candidate for outlier and such points could be pruned out from each cluster. A distance based outlier score is calculated for remaining points. The computations also needed to calculate the outlier score which reduces considerably due to the pruning of some points. Based on the outlier score, the top n points with the highest score are declared as outliers.

### ***2.3 Works related to NB and SVM Classification***

Xiang et. al. designed and proposed a model which contains three-level of decision tree classification to increase detection rate [56]. This model is more efficient in detecting known attacks but a serious shortcoming of this approach is the low detection rate for unknown attacks and generation of high false alarm rates. Peddabachigiri et. al. [57] proposed a model of intrusion detection system using a hierarchical hybrid intelligent system combining decision tree and support vector

machine (DT-SVM) that produces high detection rate while reduces different attacks from normal behaviour [58].

A classification task usually involves training and test sets which consist of data instances. Each instance in the training set contains one class label or a target value and several attributes or features. A classifier produces a model that is able to predict target values of data objects in the testing set, for which only the attributes are known. A typical classification problem is a two-class problem in which the classifier separates two classes by a function derived from available data samples. The best classifier is the one that generalizes well even on unseen examples [59]. Only one function maximizes the margin (i.e. distance between itself and the nearest example of each class) and this margin is called the optimal separating hyperplane. The classifier with this function generalizes better than the other options [60].

Most classification tasks are not easy, usually need complex structures to make an optimal separation that classify new objects correctly on the basis of the samples available through training data. SVMs are known as hyperplane classifiers which draw separating lines for distinguishing objects of different class memberships [61].

According to the characteristics of network intrusion data such as small sample, nonlinear and high dimension, SVM shows superior performance [62]. It is an effective, robust, efficient and accuracy in network action classification. SVM combined with kernel method has characteristics such as high generalization capability, global optimal solution and insensitive to the dimension of data [63] [64].

Z. Muda et. al. proposed hybrid approach of anomaly detection based on k-Means clustering and Naive Bayes classification. They used KDD Cup '99 dataset as evaluation data and found out that the hybrid learning approach achieved very low false alarm rate below 0.5%, while keeping the accuracy and the detection rate higher than 99% [65]. The approach was capable to correctly classify Normal data type, and also attack data types like Probe and DoS except U2R and R2L.

Fabrice Colas and Pavel Brazdil, however, observed different findings [66]. They stated that Naive Bayes is advantageous for a small number of samples, but as number of samples increases, the difference diminishes [67]. SVM is, however, disadvantageous in terms of processing times. The processing time tends to grow quadratically with the number of samples in the training set.

Huang, Lu and Ling [68] carried out a comparative study of Naive Bayes, Decision Tree and SVM in terms of classification accuracy and AUC. They found out that both Naive Bayes and SVM have a very similar predictive accuracy as well as they produce similar AUC scores.

Roshan Chitrakar and Huang Chuanhe proposed a hybrid approach to anomaly based intrusion detection by using k-Medoids clustering with Naïve Bayes classification and produced better performance compared to k-Means with Naïve Bayes classification [69]. The approach, using Kyoto 2006+ datasets, showed a maximum 4% of improvement in both Accuracy and Detection Rate while reducing 1% of False Alarm Rate.

Interchanging of support vectors and non-support vectors (or data samples) can be related to the KKT (Karush-Kuhn-Tucker theory) conditions. The necessary and sufficient condition violating the KKT is given by Wang, Zheng, Wu, and Zhang [70]. They presented and proved that if there were any new samples contrary to the KKT conditions, then the non-support vectors of the original SVM would have the chance to become support vectors.

An efficient method to sequentially eliminate the redundant support vectors at low computational cost of kernel-based SVM classification was proposed by Kobayashi and Otsu [71]. In their method, by calculating the inverse of kernel gram matrix of support vectors, up to a half of the support vectors were eliminated by preserving the same performance as that using the original set of support vectors.

Habib et.al. [72] proposed various Support Vector set reduction algorithms in order to accelerate the classification process for a generic SVM-based change detection algorithm. The adopted strategy was to obtain a reduced set of the initial SVs. The algorithms, using mechanical analogy and optimization problem, showed good performances in terms of both computational cost and classification accuracy.

Support Vector Machine (SVM) is originated from statistical learning theory and, hence, similar to probabilistic approaches. It is used as a supervised learning method for classification and regression [73]. SVM is also termed as a global classification model because it usually employs all attributes and produces non-overlapping, flat and linear partitions. SVM is based on maximum margin linear discriminants, and therefore tries to maximize between two partitions [74].

Ji-yong, Shi, et al. explained that SVMs are based on the Structural Risk Minimization (SRM) principle and are better than traditional principle of Empirical Risk Minimization (ERM) adopted by conventional Neural Networks [75]. ERM minimizes the error on the training data, while SRM minimizes an upper bound on the expected risk, thus giving SRM a greater generalization capability. According to Vapnik et.al, SVMs depends upon preprocessing the data to represent patterns in a much higher dimension than the original feature space [76]. If an appropriate nonlinear mapping to a sufficiently high dimension is used, data from two categories will always be separated by a hyperplane.

In short, SVM classifiers choose the hyperplane that has the maximum margin. The principal concept of SVM is decision planes that define decision boundaries. A decision plane is one that separates a set of objects having different class memberships [77].

Nowadays, in the field of intrusion detection, Support Vector Machine (SVM) is becoming a popular classification tool based on statistical machine learning [78]. There are two issues in machine learning - training of large-scale data sets and availability of a complete data set [79] [80]. To elaborate, computer's memory will not be enough and training time will be too long if training data set is very large. Next, we can't obtain the complete network information in the first capturing of data packets and hence a continuous online learning is required for high learning precision with increasing number of samples. The challenge of incremental learning is to decide what and how much information from the previous learning should be selected for training in the next learning phase and how to deal with the new data sets being added in the phase. So, the key of incremental learning is to cope with increasing data samples while retaining the information of original data samples in the meantime.

### **Classification and Regression SVMs**

To construct an optimal hyperplane, SVM employs an iterative training algorithm, which is used to minimize an error function. According to the form of the error function, SVM models can be classified into four distinct groups:

1. Classification SVM Type 1 (also known as C-SVM classification)
2. Classification SVM Type 2 (also known as nu-SVM classification)

3. Regression SVM Type 1 (also known as epsilon-SVM regression)
4. Regression SVM Type 2 (also known as nu-SVM regression)

### Classification SVM Type 1

For this type of SVM, training involves the minimization of the error function

$$\frac{1}{2} w^T w + C \sum_{i=1}^N \xi_i \quad (2-1)$$

subject to the constraints

$$y_i (w^T \phi(x_i) + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, i = 1, \dots, N \quad (2-2)$$

where C is the capacity constant, w is the vector of coefficients, b is a constant, and  $\xi_i$  represents parameters for handling nonseparable data (inputs). The index i labels the N training cases. Note that  $y \in \pm 1$  represents the class labels and  $x_i$  represents the independent variables. The kernel  $\phi$  is used to transform data from the input (independent) to the feature space. It should be noted that the larger the C, the more the error is penalized. Thus, C should be chosen with care to avoid over fitting.

### Classification SVM type 2

In contrast to Classification SVM Type 1, the Classification SVM Type 2 model minimizes the error function

$$\frac{1}{2} w^T w - \nu \rho + \frac{1}{N} \sum_{i=1}^N \xi_i \quad (2-3)$$

subject to the constraints

$$y_i (w^T \phi(x_i) + b) \geq \rho - \xi_i, \xi_i \geq 0, i = 1, \dots, N \text{ and } \rho \geq 0 \quad (2-4)$$

In a regression SVM, you have to estimate the functional dependence of the dependent variable y on a set of independent variables x. It assumes, like other regression problems, that the relationship between the independent and dependent variables is given by a deterministic function f plus the addition of some additive noise:

### Regression SVM

$$y = f(x) + \text{noise} \quad (2-5)$$

The task is then to find a functional form for  $f$  that can correctly predict new cases that the SVM has not been presented with before. This can be achieved by training the SVM model on a sample set, i.e., training set, a process that involves, like classification (see above), the sequential optimization of an error function. Depending on the definition of this error function, two types of SVM models can be recognized:

#### Regression SVM type 1

For this type of SVM the error function is:

$$\frac{1}{2} w^T w + C \sum_{i=1}^N \xi_i + C \sum_{i=1}^N \xi_i^* \quad (2-6)$$

which we minimize subject to:

$$\begin{aligned} w^T \phi(x_i) + b - y_i &\leq \varepsilon + \xi_i^* \\ y_i - w^T \phi(x_i) - b &\leq \varepsilon + \xi_i \\ \xi_i, \xi_i^* &\geq 0, i = 1, \dots, N \end{aligned} \quad (2-7)$$

#### Regression SVM type 2

For this SVM model, the error function is given by:

$$\frac{1}{2} w^T w - C \left( \nu \varepsilon + \frac{1}{N} \sum_{i=1}^N (\xi_i + \xi_i^*) \right) \quad (2-8)$$

which we minimize subject to:

$$\begin{aligned} (w^T \phi(x_i) + b) - y_i &\leq \varepsilon + \xi_i \\ y_i - (w^T \phi(x_i) + b) &\leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* &\geq 0, i = 1, \dots, N, \varepsilon \geq 0 \end{aligned} \quad (2-9)$$

There are number of kernels that can be used in Support Vector Machines models. These include linear, polynomial, radial basis function (RBF) and sigmoid:

#### Kernel Functions

$$K(\mathbf{X}_i, \mathbf{X}_j) = \begin{cases} \mathbf{X}_i \cdot \mathbf{X}_j & \text{Linear} \\ (\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C)^d & \text{Polynomial} \\ \exp(-\gamma |\mathbf{X}_i - \mathbf{X}_j|^2) & \text{RBF} \\ \tanh(\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C) & \text{Sigmoid} \end{cases} \quad (2-10)$$

where  $K(\mathbf{X}_i, \mathbf{X}_j) = \phi(\mathbf{X}_i) \cdot \phi(\mathbf{X}_j)$

that is, the kernel function, represents a dot product of input data points mapped into the higher dimensional feature space by transformation  $\phi$ . Gamma is an adjustable parameter of certain kernel functions.

The RBF is by far the most popular choice of kernel types used in Support Vector Machines. This is mainly because of their localized and finite responses across the entire range of the real x-axis.

SVMs can handle linearly non-separable points, where the classes overlap to some extent so that a perfect separation is not possible, by introducing slack variables  $\epsilon_i$  for each point  $x_i$  in  $D$ . If  $0 \leq \epsilon_i < 1$ , the point is still correctly classified. Otherwise, if  $\epsilon_i > 1$ , the point is misclassified. So the goal of the classification becomes that of finding the hyperplane ( $w$  and  $b$ ) with the maximum margin that also minimizes the sum of slack variables. A methodology similar to that described above is necessary to find the weight vector  $w$  and the bias  $b$ .

SVMs can also solve problems with non-linear decision boundaries. The main idea is to map the original  $d$ -dimensional space into a  $d'$ -dimensional space ( $d' > d$ ), where the points can possibly be linearly separated. Given the original dataset  $D = \{x_i, y_i\}$  with  $i = 1, \dots, n$  and the transformation function  $\Phi$ , a new dataset is obtained in the transformation space  $D\Phi = \{\Phi(x_i), y_i\}$  with  $i = 1, \dots, n$ . After the linear decision surface is found in the  $d'$ -dimensional space, it is mapped back to the non-linear surface in the original  $d$ -dimensional space. To obtain  $w$  and  $b$ ,  $\Phi(x)$  needn't be computed in isolation. The only operation required in the transformed space is the inner product  $\Phi(x_i)^T \Phi(x_j)$ , which is defined with the kernel function ( $K$ ) between  $x_i$  and  $x_j$ . Kernels commonly used with SVMs include: the polynomial kernel:

$$K(x_i, x_j) = (x_i^T x_j + 1)^q, \quad (2-11)$$

where  $q$  is the degree of the polynomial



the gaussian kernel:

$$K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}, \quad (2-12)$$

where  $\sigma$  is the spread or standard deviation.

the gaussian radial basis function (RBF):

$$K(x_i, x_j) = e^{-\gamma\|x_i - x_j\|^2}, \quad \gamma \geq 0 \quad (2-13)$$

the Laplace Radial Basis Function (RBF) kernel:

$$K(x_i, x_j) = e^{-\gamma\|x_i - x_j\|}, \quad \gamma \geq 0 \quad (2-14)$$

the hyperbolic tangent kernel:

$$K(x_i, x_j) = \tanh(x_i^T x_j + \text{offset}) \quad (2-15)$$

the sigmoid kernel:

$$K(x_i, x_j) = \tanh(ax_i^T x_j + \text{offset}) \quad (2-16)$$

the Bessel function of the first kind kernel:

$$K(x_i, x_j) = \left( \frac{\text{Bessel}_{v+1}^n(\sigma\|x_i - x_j\|)}{(\|x_i - x_j\|)^{-n(v+1)}} \right) \quad (2-17)$$

the ANOVA radial basis kernel:

$$K(x_i, x_j) = \left( \sum_{k=1}^n e^{-\sigma(x_i^k - x_j^k)^2} \right)^d \quad (2-18)$$

the linear splines kernel in one dimension:

$$K(x_i, x_j) = 1 + x_i x_j \min(x_i, x_j) - \frac{x_i + x_j}{2} \min(x_i, x_j)^2 + \frac{\min(x_i, x_j)^3}{3} \quad (2-19)$$

According to [6], the Gaussian and Laplace RBF and Bessel kernels are general-purpose kernels used when there is no prior knowledge about the data. The linear kernel is useful when dealing with large sparse data vectors as is usually the case in text categorization. The polynomial kernel is popular in image processing, and the

sigmoid kernel is mainly used as a proxy for neural networks. The splines and ANOVA RBF kernels typically perform well in regression problems.

## **2.4 Recent work in Incremental SVM Classification**

Mangai, Utthara Gosa, et al. [81] stated that the general approach to making good strong classifiers is training a static classifier that cannot incorporate new data without being fully retrained. This approach of training strong static classifiers for all data is time consuming and expensive. In such situations not all the data that was previously trained is available because it has been lost or become corrupt. This makes it necessary to have a classifier that can incrementally evolve to take on novel data and classes as they become available and to not forget previously trained data.

For a good incremental learning algorithm the classifier needs to be stable but with good plasticity [82]. A completely stable classifier would be able to preserve knowledge but will not be able to learn novel information, while a completely plastic classifier can learn the novel information presented to it, but cannot retain the previous knowledge. Support Vector Machines (SVMs) have shown to be stable classifiers with better pattern recognition performance than the traditional machine learning methods [83], yet stable SVM classifiers suffer from the lack of plasticity and are inclined to the catastrophic forgetting phenomenon [84] [85]. Therefore to fully benefit from the SVM classifier performance, an incremental learning method needs to be applied to the standard SVM which will retain its stability but make it plastic.

An incremental learning algorithm of SVM based on clustering [86] was proposed by Hongle et al. taking into account the fact that the boundary support vectors may change into support vectors after adding new samples. The method initially clusters the training data set using unsupervised clustering algorithm [87] and reconstruct the new training set with the centres of cluster particles and retrain it; then, the samples contrary to the KKT conditions are added into the support vector set again using unsupervised clustering algorithm; and finally, a new training data set is obtained and retrained.

A simple Incremental Support Vector Machine (ISVM) algorithm acquires the support vectors by training initial sample set. Both the new data sets and the previous

support vectors are merged to form a new sample set, and train them to produce new support vectors. The process is repeated till the final data set [88]. The chances of new data samples becoming support vectors can be related to KKT conditions. Samples that violate the KKT conditions may change the previous support vector set, so, they are added to previous data set. And, samples that meet KKT conditions will be discarded because they won't change the previous support vector set [89].

Most of the current intrusion detection methods use non-incremental learning algorithms. With accumulation of new samples, their training time will continuously increase, and at the same time, they have difficulties in adjusting themselves in dynamically changing network environment. On the contrary, incremental learning has the abilities of rapidly learning from new samples and modifying their original model [90]. It is clear that incremental learning can better meet the requirements of real-time intrusion detection, and improve computational accuracy of real-time applications [91].

The Incremental Batch Learning with SVM suggested by Liu et.al. [92] was based on basic ISVM and discarded all samples that were not support vectors; only kept the support vectors for the next classification. In fact, the discarded samples also carry some amount of information about the classification. And, with the addition of new datasets in the next increment of learning, non-support vectors may become support vectors or vice-versa. In such case, the classification accuracy will be seriously affected and the following learning phases will be unstable.

The Samples lying near the hyperplane have greater possibilities of becoming support vectors after adding new training set. This was explained in the redundant incremental learning algorithm that was proposed by the reference [93]. The algorithm retains the redundant samples lying near the hyperplane and adds in the next training to check whether they can become the support vectors.

Again, for incremental learning and large-scale learning problems, a fast incremental learning algorithm for SVM named the active set iteration method [94] was presented. A new L2 soft margin SVM was proposed, in which the selection of the initial active/inactive sets was also discussed.

Yuan Yao et.al. [95] proposed an incremental learning approach with SVM model to classify network data stream. The model was optimized in Support Vector Machine

(SVM) kernel functions selection and the parameters, and it was found that the model, compared to other models, improved the accuracy of classification by reducing the time cost in the mean time.

A new incremental learning method was also proposed by Na Sun and Yanfeng Guo [96], in which they focused on the structure design using multi-model and an incremental learning. The model employed SVM multi-classifier for incremental learning in structure designing. Since it improved the performance of classification and reduced the time-consumption for learning, the SVM multi-classifier could equally be used in a real-time classification of data stream.

An improved incremental SVM algorithm called RS-ISVM was proposed by Yang et al. to deal with network intrusion detection [97]. Firstly, they modified Radial Basis Function (RBF) kernel to U-RBF that reduces the noise generated by feature differences. Secondly, they developed a reserved set strategy and a concentric circle method to retain non-support vectors that are likely to become the support vectors in the next training.

Lin, Xiaojun and Chan, Patrick P.K. [98] investigated the vulnerabilities of incremental learning algorithm of SVM under the adversarial attack. In their research paper, they pointed out the problem that the performance of the conventional incremental learning algorithm of SVMs may be affected significantly in the adversarial environment as it does not consider the adversarial attack.

Nekkaa, Messaouda and Boughaci, Dalila [99] proposed a memetic algorithm combined with a support vector machine (SVM) for feature selection and classification in Data mining. The memetic algorithm (MA) is an evolutionary metaheuristic that can be viewed as a hybrid genetic algorithm combined with some kinds of local search. The proposed approach tries to find a subset of features that maximizes the classification accuracy rate of SVM. They also developed a hybrid algorithm of MA and SVM with optimized parameters.

Yin, Yingjie et.al. proposed [100] a robust state-based structured support vector machine (SVM) tracking algorithm combined with incremental principal component analysis (PCA). Different from the current structured SVM for tracking, this method directly learns and predicts the object's states and not the 2-D translation transformation during tracking.

Wang, Xuhui and Shu, Ping [101] presented an incremental model to identify aircraft land status of civil aircraft in order to support fault diagnosis and structure maintenance. They further developed a recognition model by introducing support vector method. Moreover, an incremental algorithm was also proposed to solve the problem of on line sample array.

Gu, Bin et.al. [102] presented a modified SV Learning for Ordinal Regression (SVOR) formulation based on a sum-of-margins strategy. The formulation has multiple constraints, and each constraint includes a mixture of an equality and an inequality. They also extended the accurate on-line  $\nu$ -SVC algorithm to the modified formulation.

An online fault diagnosis method based on Incremental Support Vector Data Description (ISVDD) and Extreme Learning Machine with incremental output structure (IOELM) is proposed by Yin Gang et.al [103]. The ISVDD is used to find a new failure mode quickly in the continuous condition monitoring of the equipments. The fixed structure of Extreme Learning Machine is changed into an elastic structure whose output nodes could be added incrementally to recognize the new fault mode efficiently.

Xie, Weiyi et.al. [104] proposed an incremental learning approach that greatly reduces the time consumption and memory usage for training SVMs. The proposed method is fully dynamic, which stores only a small fraction of previous training examples whereas the rest can be discarded. It can further handle unseen labels in new training batches.

Ji, Rui et.al. [105] proposed an architecture for Takagi-Sugeno (TS) fuzzy system and developed an incremental smooth support vector regression (ISSVR) algorithm to build the TS fuzzy system. ISSVR is based on the E-insensitive smooth support vector regression (E-SSVR), a smoothing strategy for solving E-SVR, and incremental reduced support vector machine (RSVM). The ISSVR incrementally selects representative samples from the given dataset as support vectors.

Gan Liangzhi Zhang, and Shicheng Liu, Haikuan [106] focused their research on ensemble learning to improve Least Squares Support Vector Machines (LS-SVMs). LS-SVMs are well known as a typical kernel method with good performance, but are subject to the curse of dimensionality. In order to get accurate and stable learning

machine with better generalization ability, they proposed the Ensemble Incremental LS-SVMs (EILS-SVMs).

Cheng Wei-Yuan and Juang Chia-Feng [107] proposed a new incremental learning approach to endow a Takagi-Sugeno-type fuzzy classification model with high generalization ability. The proposed fuzzy model is learned through incremental support vector machine (SVM) and margin-selected gradient descent learning and is called  $FM^{\{3\}}$ . An online incremental linear SVM is proposed to tune the rule consequent parameters to endow the  $FM^{\{3\}}$  with high generalization ability. The use of incremental instead of batch SVM enables the  $FM^{\{3\}}$  to handle online training problems with only one new sample available at a time.

Pan Yu-Xiong et.al. [108] proposed a time series prediction method based on the dynamic Bayesian least squares support vector machine (LS-SVM) in order to accurately predict operating parameters of the turbofan engine. By the Bayesian evidence framework theory, initial model parameters of the LS-SVM are inferred. Dynamic learning of the LS-SVM and dynamic prediction of time series are realized by the recursively incremental and decremental sample learning method.

Wang Ning et.al. [109] proposed an improved incremental learning algorithm for a large-scale data stream, which is based on SVM (Support Vector Machine) and is named DS-IILS. The DS-IILS takes the load condition of the entire system and the node performance into consideration to improve efficiency.

## ***2.5 Supervised / Unsupervised and Incremental Learning***

Quinlan, J. Ross stated in his work entitled “Programs for Machine Learning” that machine learning approach is more flexible and clearer than programming [110]. Unlike the programming field, where explicit instructions or commands are encoded to obtain solution of specific problems, the field of machine learning is focused in designing algorithms that encode inductive mechanisms where solutions to multiple classes of problems can be derived from data samples.

Di Mauro, Nicola, et al. wrote in their work that a very basic and traditional formulation of machine learning problem has been a classification problem. For instance, one is given some domain of individuals for which a general classification is

required. The classification is given by a function from this domain to some small finite set corresponding to the classes [111]. A training set that provides the class of some typical individuals play the role of examples. Moreover, some background knowledge relevant to the inductive task at hand may also be available. The general classification of the individuals will be induced from this.

Machine learning algorithms are described as either 'supervised' or 'unsupervised' [112]. According to Larose, Daniel T. and Chantal D. Larose [113], no target variable is identified as such in unsupervised methods. Unsupervised learning is similar to exploratory spirit of Data Mining [114]. where both explanatory and dependent variables are treated equally. In unsupervised learning methods, the data mining algorithm searches for patterns and structure among all the variables. In supervised learning, the data variables under learning process are divided into two groups – (1) explanatory variables and (2) dependent variables [115]. The values of the dependent variable are always known for a large part of the data set. The learning process tries to establish a relationship between these two variables.

Tsai et.al [116] and Choi et.al [117] suggested in their separate research works that most data mining methods are supervised methods meaning that (1) there is a particular pre-specified target variable, and (2) the algorithm is given many examples where the value of the target variable is provided, so that the algorithm may learn which values of the target variable are associated with which values of the predictor variables. According to P. Kavitha and M. Usha [118] and Vijayasathy et.al [119] suggested that many of the methods e.g. decision tree induction, Naive Bayes, etc. are examples of supervised learning techniques.

Wang Ding [120] explains the methodology, which most of the supervised data mining methods apply, for building and evaluating a model.

1. First, the algorithm is provided with a training set of data, which includes the pre-classified values of the target variable in addition to the predictor variables. Records in the training set need to be pre-classified [121]. A provisional data mining model is then constructed using the training samples provided in the training data set. The algorithm needs to guard against “memorizing” the training set and blindly applying all patterns found in the

training set to the future data. Such a pattern is a spurious artifact of the training set and needs to be verified before deployment.

2. The next step in supervised data mining methodology is to examine how the provisional data mining model performs on a test set of data. In the test set, a holdout data set, the values of the target variable are hidden temporarily from the provisional model, which then performs classification according to the patterns and structure it learned from the training set. The efficacy of the classifications is then evaluated by comparing them against the true values of the target variable. The provisional data mining model is then adjusted to minimize the error rate on the test set [122].
3. Finally, the adjusted data mining model is then applied to a validation data set, another holdout data set, where the values of the target variable are again hidden temporarily from the model. The adjusted model is itself then adjusted, to minimize the error rate on the validation set. Estimates of model performance for future, unseen data can then be computed by observing various evaluative measures applied to the validation set [123].

Iclal Çetin Taş [124] said that, in unsupervised learning, the results vary widely and may be completely off if the first steps are wrong.

Atish Roy et. al. [125] mentioned that unsupervised learners are not provided with classifications. In fact, the basic task of unsupervised learning is to develop classification labels automatically. Unsupervised algorithms seek out similarity between pieces of data in order to determine whether they can be characterized as forming a group. These groups are termed clusters, and there are a whole family of clustering machine learning techniques. Generally, in cluster analysis, the machine is not told how the texts are grouped. Its task is to arrive at some grouping of the data [126]. In a very common of cluster analysis (k-means), the machine is told in advance how many clusters it should form a potentially difficult and arbitrary decision to make.

Wilhelm, Adalbert [127] described in a handbook of computational statistics entitled "Data and Knowledge Mining" that the huge data present in Data Mining tasks allows splitting the data file in three groups - training cases, validation cases and test cases. But James E. Gentle et.al. [128] argued that it is not always possible to split the data into required subsets because data becomes a scarce resource due to partially available



values and other data properties. Therefore, learning must be continued over time rather than making it a one-shot experience. The idea of incrementality is, therefore, unavoidable in many situations.

Joseph, Anthony D. et al [129] stated that incremental learning research has been forgotten for a long period during the development process of machine learning. With the passing of times, Cardoso et al. [130] explained the distinction between incremental learning tasks and incremental learning algorithms. It was also clarified that the way to tackle incremental learning tasks is to apply incremental learning algorithms. Design issues for incremental algorithms were discussed and necessary incremental learning systems were described too.

Heinen et.al. [131] pointed out that the term “incremental” lead to some confusion because it was used in both learning tasks and learning algorithms. They also suggested that it could be cleared out by providing formal definitions and examples of incrementality for tasks and for algorithms.

Huang, Guang-Bin et. al. [132] wrote in their survey report entitled "Extreme learning machines: a survey" that incremental learning had been applied in different ways. The simplest of the incremental learning approaches is one of storing all the data which allows for retraining with all the data. At the other extreme is the training of the data, instance by instance, in an online learning fashion. Methods using the online learning approach for incremental learning have been implemented but have not considered all the issues of learning, particularly the learning of new classes.

According to Elwell et.al. [133] and Chen et.al. [134] suggested the necessary criteria for a classifier to be incremental. It should : -

1. be able to learn additional information from new data.
2. not require access to the original data used to train the existing classifier.
3. preserve previously acquired knowledge (that is, it should not suffer from catastrophic forgetting).
4. be able to accommodate new classes that may be introduced with new data.

One of the most recent incremental learning approaches are Learn++, which is based on AdaBoost [135] [136] introduced by Polikar et al., later on modified as Learn++.MT [137]. To overcome problems present in these algorithms, a new

incremental learning approach called Incremental Learning Using Genetic Algorithm (ILUGA) [138] is developed. ILUGA uses binary SVM classifiers that are trained to be strong classifiers using genetic algorithm [139].

## ***2.6 Shortcomings of the Current Researches***

Although many attempts have been made in the recent past regarding intrusion detection, a few shortcomings are felt prior to carry out this research work and, therefore, modifications to them are suggested in this work.

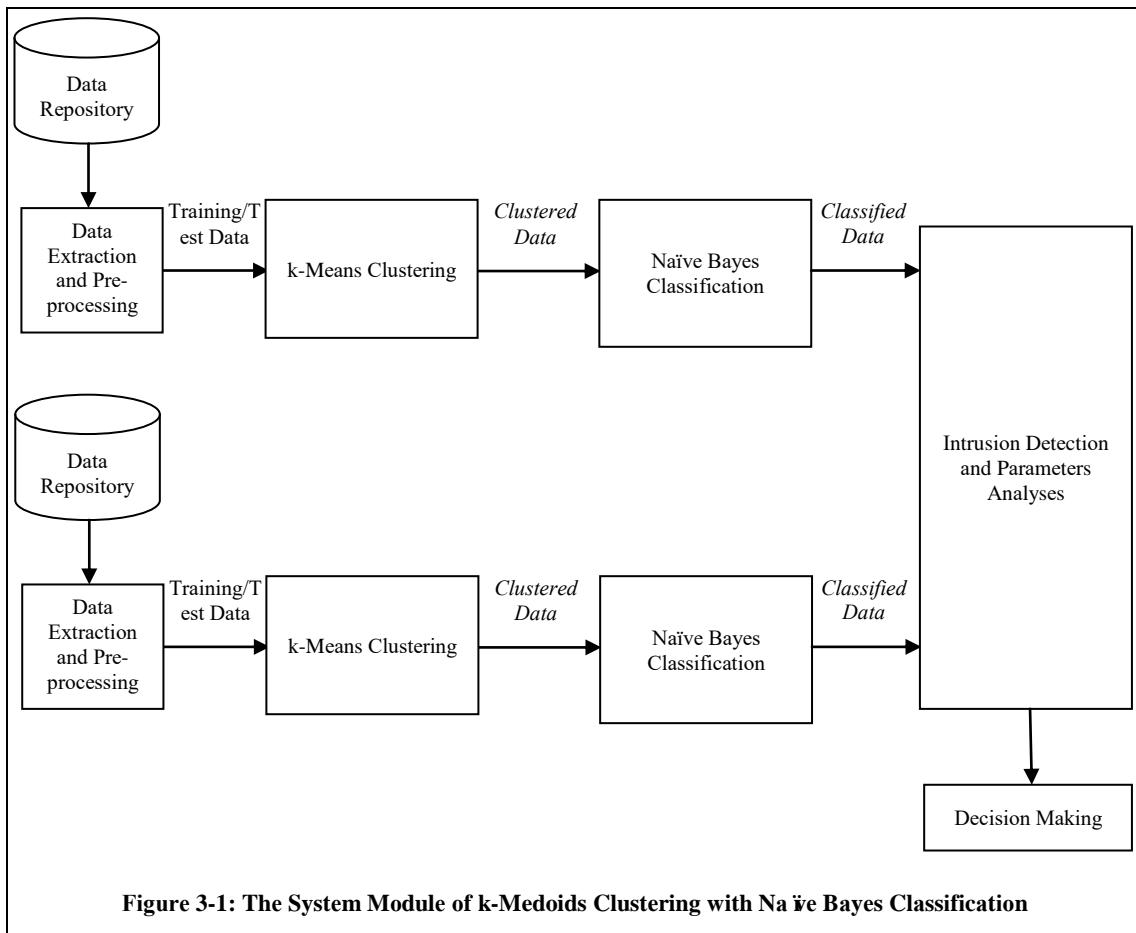
Most of the researches were successful in detecting anomalies in fairly large data traffic, but did not perform consistently well in case of large data. Some yielded greater detection rate but also increases false alarm rates; some took very long time to detect thus failing to work in real time. Some works were found to have adopted k-means to cluster the data before classification but they still did not find to provide predictability for all volume of very large data.

Algorithms used by many of the recent past works in order to classify attack and normal data were simple classification algorithms like Naïve Bayes, because it was believed that such algorithms would work in real time. Some research works used SVM classification, which improved the quality of classification. There was still problem with identifying less common attacks in short times. Very few works tried to implement iterative approach in classification but they took very long time in accomplishing the task and the detection was still slower.

## Chapter 3. k-Medoids with Naïve Bayes Classification

### 3.1 System Model and Problem Description

Some intrusion behaviours are similar to normal and also to other intrusion instances. Many clustering algorithms including k-Means are unable to correctly distinguish such intrusion instances and a few normal instances too. In order to overcome such classification problems and to enhance the detection accuracy, an additional classification technique e.g. Naïve Bayes classifier is needed to be combined. Naïve Bayes classifiers are based on a very strong independence assumption with fairly simple construction. It analyzes the relationship between independent variable and the dependent variable to derive a conditional probability for each relationship [140].



In this study, the learning approach based on combination of k-Medoid clustering and Naïve Bayes classification technique is proposed to further enhance k-Means based detection capabilities in terms of accuracy, detection rate and false positive rate. The

performance evaluation of the proposed approach is done by using Kyoto 2006+ dataset.

The advantage of the k-Means algorithm is its favorable execution time. Its drawback is that it is sensitive to outliers since an object with an extremely large value may distort the distribution of data. If the number of data points is less, then the k-Means algorithm takes lesser execution time. But when the data points are increased to maximum, the k-Means algorithm takes maximum time. Whereas k-Medoids algorithm attempts to minimize the squared error, which is the distance between points in the cluster and a point that is designated as the center or centroid (also called a medoid) of a cluster. Therefore, the k-Medoids algorithm performs reasonably better than the k-Means algorithm [141].

### ***3.2 General Description of the Solution***

The role of Intrusion Detection System (IDS) has been inevitable in the area of Information and Network Security – specially for building a good network defense infrastructure. Anomaly based intrusion detection technique is one of the building blocks of such a foundation. In this paper, the attempt has been made to apply hybrid learning approach by combining k-Medoids based clustering technique followed by Naïve Bayes classification technique. Because of the fact that k-Medoids clustering techniques represent the real world scenario of data distribution, the proposed enhanced approach will group the whole data into corresponding clusters more accurately than k-Means such that it results in a better classification. An experiment is carried out in order to evaluate performance, accuracy, detection rate and false positive rate of the classification scheme. Results and analyses show that the proposed approach has enhanced the detection rate with minimum false positive rates.

In this study, the learning approach based on combination of k-Medoid clustering and Naïve Bayes classification technique is proposed to further enhance k-Means based detection capabilities in terms of accuracy, detection rate and false positive rate. The performance evaluation of the proposed approach is done by using Kyoto 2006+ dataset.

### 3.3 Comparison between k-Means and k-Medoids

The k-means method uses centroid to represent the cluster and it is sensitive to outliers [142]. This means, a data object with an extremely large value may disrupt the distribution of data. k-medoids method overcomes this problem by using medoids to represent the cluster rather than centroid. A medoid is the most centrally located data object in a cluster. Here, k data objects are selected randomly as medoids to represent k cluster and remaining all data objects are placed in a cluster having medoid nearest (or most similar) to that data object. After processing all data objects, new medoid is determined which can represent cluster in a better way and the entire process is repeated. Again all data objects are bound to the clusters based on the new medoids. In each iteration, medoids change their location step by step. Or in other words, medoids move in each iteration. This process is continued until no any medoid move. As a result, k clusters are found representing a set of n data objects.

Generally, arbitrarily distributed input data points are used to evaluate the clustering quality and performance of two clustering algorithms, e.g., k-Means and kMedoids. To evaluate the clustering quality, the distance between two data points are taken for analysis. The computational time is calculated for each algorithm in order to measure the performance of the algorithms. The experimental results show that the k-Means algorithm yields the best results compared with k-Medoids algorithm. The average execution time of the k-Means algorithm is very less than the k-Medoids algorithm [143].

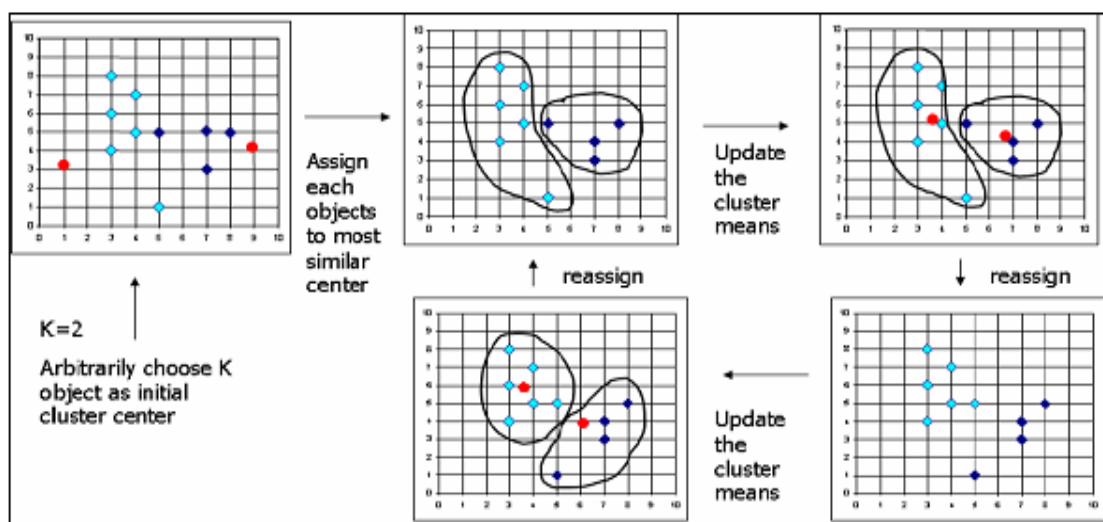


Figure 3-2: Working of k-Means

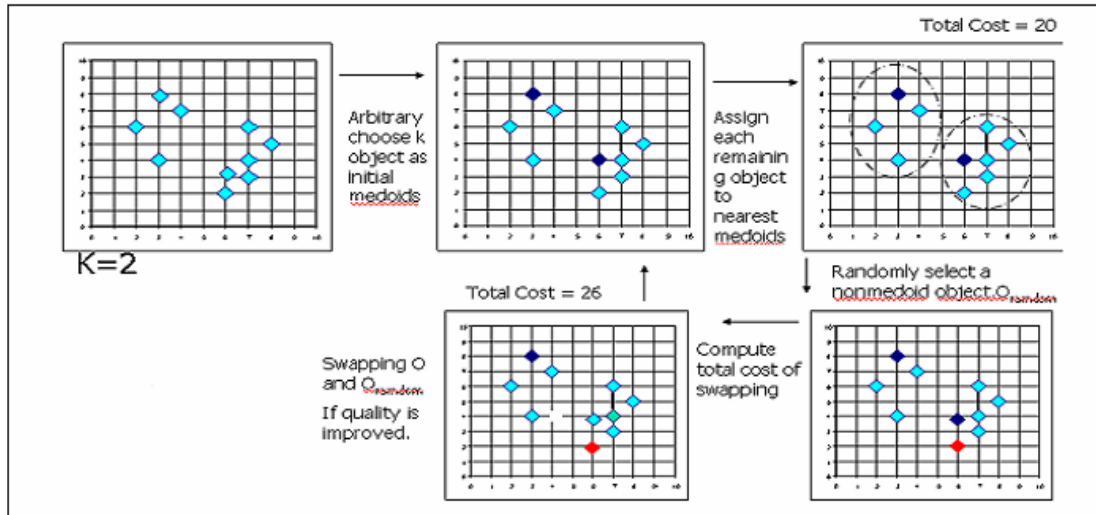


Figure 3-3: Working of k-medoids algorithm

### 3.4 The Proposed Hybrid Approach

As the first stage of the proposed approach, similar data instances are grouped based on their behaviors by using k-Medoids clustering technique. The resulting clusters are then classified into attack classes using Naïve Bayes classifiers.

#### k-Means Clustering

K-means [144] is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. This method adopts a truly simple and very easy way to classify a given data set, given a certain number of clusters,  $k$  for instance, a priori. The main idea is to define  $k$  centroids, one for each cluster. The centroids should be placed in such a way that they lie as much far away from each other as possible. Next it takes each point that belong to a given data set and assign it to the nearest centroid. The first step is completed as soon as there is no point pending. Here,  $k$  new centroids are needed to re-calculated as barycenters of the clusters resulting from the previous step. After having  $k$  new centroids, a new binding is done between the same data set points and the nearest new centroid. The process is done iteratively, as a result the  $k$  centroids change their locations step by step until no more changes are possible.

To summarize, this algorithm aims at minimizing an *objective function*, i.e. a squared error function in this case [145]. The objective function is given by: -

$$J = \sum_{j=1}^k \sum_{i=1}^n \|x_i^{(j)} - c_j\|^2 \quad (3-20)$$

Where,  $\|x_i^{(j)} - c_j\|^2$  is a chosen distance measure between a data point  $x_i^{(j)}$  and the cluster centre  $c_j$ ,  $i$  is an indicator of the distance of the  $n$  data points from their respective cluster centres.

The algorithm is composed of the following steps [ 146] : -

---

***Algorithm 3- 1: k-means Algorithm***

---

**Input** :  $k$ , the number of clusters to be partitioned;

$n$ , the number of objects with  $m$  attributes.

**Output**: A set of  $k$  clusters based on given similarity function.

**Steps**:

- 1: Arbitrarily choose  $k$  objects as the initial cluster centers;
  - 2: Repeat,
  - 3: Assign each object to the cluster to which the object is the most similar based on the given similarity function;
  - 4: Update the centroid by calculating the mean value of the objects for each cluster;
  - 5: Until no change.
- 

***Complexity of the Algorithm:***

In each loop there is distance calculation using Euclidean distance function, for which 6 operations (2 subtractions, one addition, two multiplications and one square root operation) are needed. The other operations include comparison between distances, calculation of centroids. So, each iteration requires  $6*[k*n*m] + [(k-1)*n*m]$  operations +  $[k*((n-1) + 1)*m]$  operations. If the algorithm converges in  $i$  iterations, the whole algorithm will require  $6*[i*k*n*m] + [i*(k-1)*n*m] + [i*k*((n-1) + 1)*m]$  operations. Hence, the time complexity can be termed as  $O(i*k*n*m)$ . In case of very

large data sets, where  $k$  and  $m$  are almost fixed/constant; and also  $l \ll n$ , the complexity is approximately given by  $O(n)$ .

The k-means algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. The algorithm is also significantly sensitive to the initial randomly selected cluster centres. The k-means algorithm can be run multiple times to reduce this effect.

### **k-Medoids Clustering**

k-Medoids is a clustering-by-partitioning algorithm that is related to the k-Means algorithm. Instead of taking the mean value of the objects in a cluster, the most centrally located object (data point) in a cluster is considered as the reference point, which is called a medoid or a centroid. So, the partitioning can be performed based on their distance metric (i.e. minimizing the sum of the dissimilarities between each object and its corresponding reference point.)

Prior to begin the algorithm, the k-Medoids algorithm takes an input integer  $k$ , which determines how many clusters the entire dataset of  $n$  objects has to be partitioned into [147]. The algorithm actually starts by arbitrarily finding a representative object (the medoid) for each cluster. Each remaining object is clustered with the medoid to which it is the most similar.

The cost for the current clustering configuration is calculated using the following equation: -

$$Cost(M, X) = \sum_{i=1}^n \min_{j=1}^k (d(m_j, x_i)) \quad (3-21)$$

Where  $M$  is the set of medoids,  $X$  is the data set,  $n$  is the number of events,  $k$  is the number of clusters,  $m_j$  is the  $j^{\text{th}}$  medoid,  $x_i$  is the  $i^{\text{th}}$  event, and  $d$  is a distance function. The distance function can be any distance metric, Euclidean distance for instance. The equation basically calculates the total cost across the entire data set. The function  $min$  is meant to find the medoid that a given event is closest to. This is done by calculating the distance from every medoid to a given event and then adding the smallest distance to the total cost.



---

**Algorithm 3- 2: k-Medoids Algorithm**

---

**Input:**  $k$ : The number of clusters;

$D$ : Data set containing  $n$  objects

**Output:** A set of  $k$  clusters. (Each cluster will have minimum dissimilarities of all the objects to their nearest medoids.)

**Let:**

$O_j$  : Set of medoid objects

$O_i$  : Set of data objects

// Step 1: choose  $k$  objects in  $D$  as initial medoids  $O_j$  //

1: For  $j \leftarrow 1$  to  $k$

2:     Set  $O_j$  to *random*( $O_i$ );

3: EndFor

4: Repeat

    // Step 2: Assign each object to the nearest cluster and compute the total distance

    //

5:     For  $i \leftarrow 1$  to  $n$

6:         For  $j \leftarrow 1$  to  $k-1$

7:             If  $|O_i - O_j| < |O_i - O_{j+1}|$  then

8:                  $min\_distance = |O_i - O_j|$ ;

9:                  $O_i\_cluster = j$ ;

10:             Else

11:                  $min\_distance = |O_i - O_{j+1}|$ ;

12:                  $O_i\_cluster = j+1$ ;

13:             EndIf

14:         EndFor

15:         Add  $min\_distance$  into  $total\_distance$ ;

16:     EndFor

```

// Step 3: Swap objects and medoids temporarily and compute total cost of
// swapping//
17:   For  $j \leftarrow 1$  to  $k$ 
18:     For  $i \leftarrow 1$  to  $n$ 
19:        $C = \text{Cost}(O_j, O_i)$ ; // Using Eqn. 1 //
20:     EndFor
// Step 4: If cost is negative, swap objects and medoids permanently //
21:   If  $C < 0$  then
22:     swap( $O_j, O_i$ );
23:   EndIf
24: EndFor
25: Until no_change

```

---

### ***Complexity of the Algorithm:***

The algorithm has three loops – the outermost, the middle and the innermost loops. The outermost loop iterates through each medoid object; there are  $k$  medoids. The middle loop iterates through each non-medoid object; and there are  $(n-k)$  non-medoid objects. The innermost loop iterates through each non-medoid object to compute the swapping cost of medoids and non-medoids. The complexity of each iteration of the algorithm can be expressed as  $O(k*(n-k)*k*(n-k))$  i.e.  $O((k(n-k))^2)$ .

The problem of k-medoids method is that it does not generate the same result with each run, because the resulting clusters depend on the initial random assignments. It is more robust in presence of noise and outliers; however it's processing is more costly than the k-means method. Lastly, the optimal number of clusters  $k$  is hard to be predicted, so it is difficult for a user without any prior knowledge to specify the value of  $k$  [ 148].

### **Naïve Bayes Classifier**

A Naïve Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong (naïve) independence assumptions. A Naïve Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. It analyzes the relationship between independent variable and the dependent variable to derive a conditional probability

for each relationship [149]. Depending on the precise nature of the probability model, Naïve Bayes classifiers can be trained very efficiently in a supervised learning setting.

Bayes Theorem can be expressed as:

$$P(H|X) = \frac{P(X|H) P(H)}{P(X)} \quad (3-22)$$

Let  $X$  be the data record,  $H$  be some hypothesis representing data record  $X$ , which belongs to a specific class  $C$ . For classification, we would like to determine  $P(H|X)$ , which is the probability that the hypothesis  $H$  holds, given an observed data record  $X$ .  $P(H|X)$  is the posterior probability of  $H$  conditioned on  $X$ . In contrast,  $P(H)$  is the prior probability. The posterior probability  $P(H|X)$ , is based on more information such as background knowledge than the prior probability  $P(H)$ , which is independent of  $X$ . Similarly,  $P(X|H)$  is posterior probability of  $X$  conditioned on  $H$ .

Bayes theorem is useful because it provides ways to calculate the posterior probability  $P(H|X)$  from  $P(H)$ ,  $P(X)$ , and  $P(X|H)$ .

### ***Algorithm 3- 3: NB Classification Algorithm***

***Input:***  $D$ : Data set having  $n$  data objects

$C$ : Set of classes e.g. {Normal; Attack}

$X$ : Data record to be classified

$H$ : Hypothesis (that  $X$  is classified into  $C$ )

***Output:*** The predicted class  $C_{NB}$  where  $X$  should be classified into.

```

// Learning //
1: For  $j \leftarrow 1$  to no. of classes
   // Step 1: Calculate prior probabilities of  $C$  //
2:    $C_j\_count \leftarrow$  no. of  $D_i$  where  $D_i.class\_label = j$ ;
3:    $P(C_j) \leftarrow C_j\_count / n$ ;
   // Step 2: Calculate prior probabilities of  $X$  //
4:   For each attribute value  $X_l$  in  $X$ 
5:      $X_l\_count \leftarrow$  no. of  $X_l$  in  $C_j$ ;
6:      $P(X_l|C_j) \leftarrow X_l\_count / C_j\_count$ ;
7:   EndFor

```

```

// Step 3: Calculate posterior probability of X //
8:   P(X) ← average (P(Xi/Cj));
9: Endfor

// Classifying //

// Step 4: Determine required Naïve Bayes probability //
10: For j ← 1 to no_of_classes
11: P(Cj|X) ← P(Cj/H) * P(Cj) / P(X)
12: Endfor

// Step 5: Get the class with maximum probability //
13: CNB = max(P(Cj|X))

```

---

### ***Complexity of the Algorithm:***

There are altogether two levels of loops in this algorithm. The outer loop iterates through each class. There are  $j$  classes and it is constant (i.e. 2) in this case. The inner loop iterates through number of records in each class. Its maximum value is the number of records itself i.e.  $n$ . So, the complexity of the algorithm can be expressed as  $O(j*n)$  or  $O(2n)$  by replacing  $j$  with 2.

### **3.5 Experimental Results and Analysis**

The entire set of experiments performs cross-validation of the two approaches with their corresponding program codes: -

- (1) k-Means clustering followed by Naïve Bayes classification and
- (2) k-Medoids clustering followed by Naïve Bayes classification.

Since the main purpose is to evaluate the proposed approach i.e. k-Medoids based approach with k-Means based approach, both the programs are carried out using the same datasets and in the same operating and hardware environment. Finally, the results from the both programs were compared, evaluated and analyzed.

Following is the details of platform for experiments carried out for this piece of research work: -

**Table 3-1: Experimental platform for k-Means/k-Medoids with NB classification**

Operating System :	Linux (Ubuntu)
Clustering and Classification Tool:	Rapidminer 5.0
Performance Analysis Tool:	Rapidminer 5.0
Algorithm coding and testing platform:	Matlab
Data Analysis and Graphical Representation Tool:	LibreOffice Calc
Data Source:	Kyoto 2006+ Data

### *3.5.1 Selection of Experimental Data*

Mostly two benchmark datasets are used for performance evaluation of any experiment related to intrusion detection. They are KDD Cup 1999 dataset [150] and Kyoto 2006+ dataset [151].

In this work, Kyoto 2006+ dataset is used as evaluation data for the experiments. After examination of the data and preliminary experiments, the following datasets were chosen for the use in the experiments: -

- (2) 2007 Nov. 1, 2 and 3 with 238179 records;
- (3) 2007 Dec. 1, 8, 15 and 22 with 360726 records;
- (4) 2008 Dec. 1, 9, 15 and 22 with 381358 records;
- (5) 2009 July 1, 8, 15 and 22 with 500865 records.

### *3.5.2 Description of Experimental Data*

The required experimental data are taken from Kyoto 2006+ dataset [152]. It consists of 14 conventional features and 10 additional features. The first 14 features were extracted based on KDD Cup 99 data set. From this dataset, only 14 significant and essential features have been extracted from the raw traffic data obtained by honeypot systems [153] that are deployed in Kyoto University. Additional 10 features have also been extracted in order to help investigate more effectively what happens on the

networks. Moreover, they can also be used as training and testing data along with 14 conventional features. The details of features are presented in Appendix I: Features of Kyoto 2006+ dataset.

Duration	Src. ice	Source bytes	Destination bytes	Count	Same src rate	Server rate	Sv. server rate	Dst host count	Dst host sv count	Dst host same src port rate	Dst host server rate	Dst host sv server rate	Flag	IDS detection	Malware detection	Admin detection	Label	Source IP Address
46.71	8	4124	40	0	0	0	0	1	100	1	0	0	RSTO	0	0	0	1	fd58:d212:b798:88ed:364d:285b:1fbc:2
21.82	8	3878	244	1	1	0	0	0	100	0	0	0	RSTO	0	0	0	1	fd58:d212:b798:dff6:376e:28ef:47fa:13

{See Appendix for more...}

In order to provide the rough idea about the experimental data, a snapshot of a particular day's experimental data is provided below. For more data, please refer to the Appendix II: Experimental Data Samples.

### 3.5.3 Data Pre-processing

A few simple and basic data pre-processing techniques like sampling and filtering are applied for the sake of easy and smooth operation of the experiments [154]. Samples with known and unknown attacks are merged together so as to render two types of data only viz. Normal and Attack data.

The categorical attributes of the data are treated differently for the use in k-Means clustering only as it cannot handle this data type. They are converted into numerical values e.g. {REJ, S0, RST0, SF} is encoded as {1,2,3,4}.

### 3.5.4 The Experimental Procedure

Among the selected Kyoto2006+ dataset, a number of different sized data samples are extracted. The number of huge data samples is less than that of small data samples. For example, there are 20 (twenty) different samples of the smallest size having 10000 (ten thousand) records; and only one sample of the largest size consisting of about 500000 (five hundred thousand) records.

Using every data samples, one by one, both programs are executed. The number of true positive, true negative, false positive and false negative values of both the programs are recorded and used in the performance evaluation of both the programs.

### 3.5.5 Performance Evaluation

The performance evaluation of the experiment is carried out in terms of accuracy ( $A$ ), detection rate ( $DR$ ) and false alarm rate ( $FAR$ ) by using the following equations:-

$$A = (TP+TN) / (TP+TN+FP+FN) \quad (3-23)$$

$$DR = (TP) / (TP+FP) \quad (3-24)$$

$$FAR = (FP) / (FP+TN) \quad (3-25)$$

Where,

$TP$  = True Positive (attack detected as attack)

$TN$  = True Negative (normal detected as normal)

$FP$  = False Positive (normal detected as attack)

$FN$  = False Negative (attack detected as normal)

The followings are the results showing the improvement of accuracy, increase in detection rate and decrease in false alarm rates of the proposed approach compared to k-Means based approach.

**Table 3-2: Comparison of Accuracy of k-Means and k-Medoids clustering followed by NB Classification**

No. of Data Samples	k-Means +NB	k-Medoids +NB	Improvement (%)
10000	96.08	96.55	0.47
100000	85.55	86.18	0.63
238179	73.86	78.29	3.92
360726	91.64	95.57	4.21
381358	85.26	89.47	4.22
500865	87.79	92.02	4.20

**Table 3-3: Comparison of DR of k-Means and k-Medoids clustering followed by Naïve Bayes Classification**

No. of Data Samples	k-Means +NB	k-Medoids +NB	Increase (%)
10000	96.21	96.77	0.56
100000	96.95	97.67	0.72
238179	66.51	70.69	4.18

360726	84.62	88.35	3.73
381358	84.98	88.73	3.75
500865	88.53	92.39	3.86

**Table 3-4: Comparison of FAR of k-Means and k-Medoids clustering followed by NB Classification**

No. of Data Samples	k-Means +NB	k-Medoids +NB	Decrease (%)
10000	3.76	3.11	-0.65
100000	2.29	1.61	-0.68
238179	5.29	4.34	-0.95
360726	6.52	5.46	-1.05
381358	3.08	2.05	-1.03
500865	3.29	2.24	-1.06

### 3.5.6 Analysis of the Results

From the experimental results and the performance evaluation, it is seen that Accuracy and Detection Rate have been increased whereas False Alarm Rate has been decreased in k-Medoids followed by Naïve Bayes for every size of data samples. Hence, the proposed approach of replacing k-Means with k-Medoids clustering has been justified.

With the varying sizes of data (i.e. from 10 thousand to 500 thousand records), k-Medoids combined with NB shows better accuracy. It can be seen from Table 3-2 that the average improvement in accuracy in case of the smallest data sets i.e. ten thousand (10,000) is not remarkable which means k-means method works as better as k-medoids in this case. As seen from the later observations, the improvements in accuracy goes on remarkably increasing, which implies that k-medoids method works relatively much better when data samples are really big – more than 200 thousand in this case. To be specific, upto one hundred thousand (100,000) samples has been found to be less than 1 percent, whereas it has been found to be about 4 percent in case of large data sets.

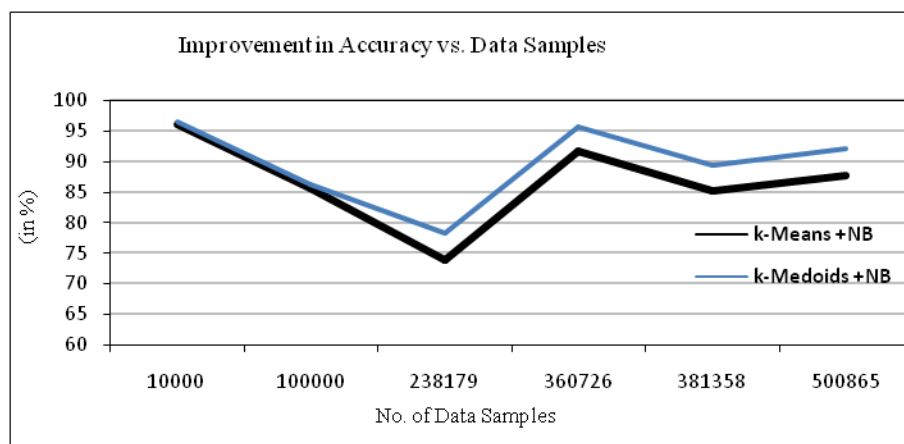
Similarly, Table 3-3 illustrates that the similar condition holds true for Detection Rate too. Here also, detection rate in case of 10000 (ten thousand) records, the



improvement is not significant but which means k-means and k-medoids perform equally good. But as the data samples go on increasing, as shown by the following rows in the table, the improvement obtained due to k-medoids in case of large data samples become more and more remarkable.

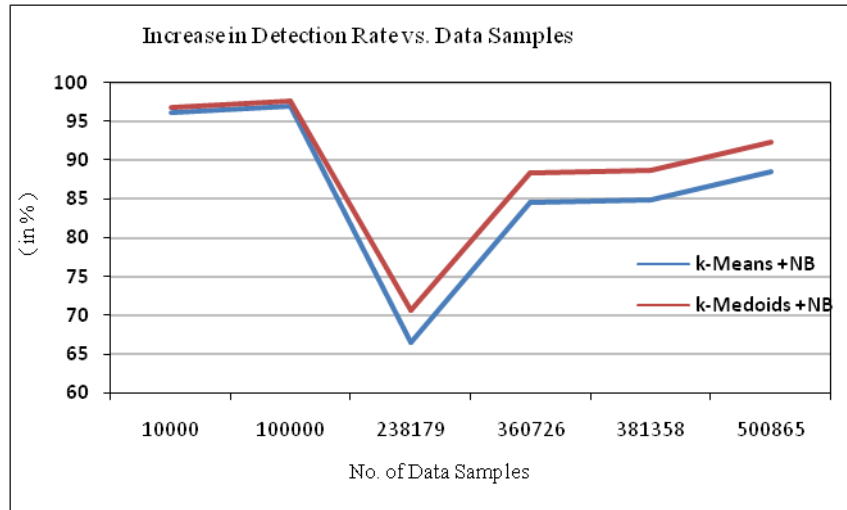
False Alarm Rate also is found to be decreased, as shown in Table 3-4, by minimum of 0.65 percent in case of small sized datasets and by more than 1 percent in case of large data samples. Here also, it is once again proved that k-medoid suppresses the false alarm rate remarkably when the data samples exceeds 200 thousand. Below this level, k-medoids works just a little better than k-means.

Naïve Bayes classification in both the cases works simply to classify the clustered data produced by k-means and medoids as attack or as normal. Therefore, it does not have any influence in performance of either k-means or k-medoids technique.



**Figure 3-4: Improvement in Accuracy achieved by k-Medoids followed by Naïve Bayes Classification**

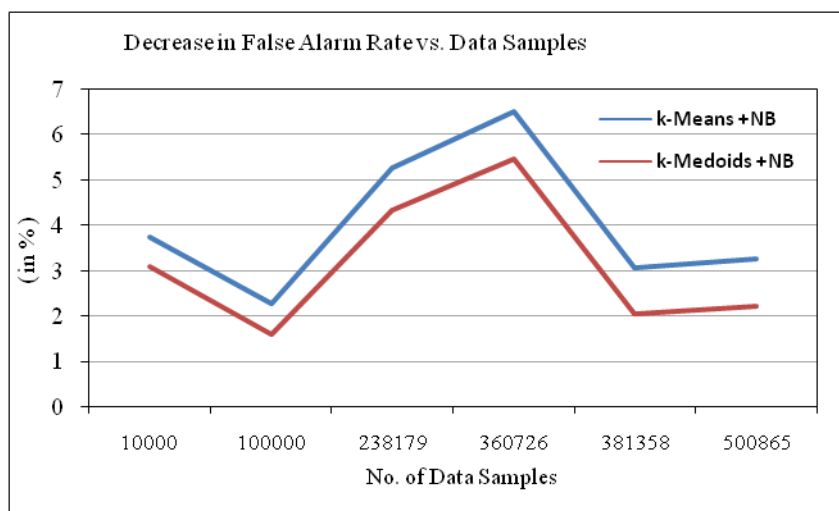
In this work, the size of data samples may also be considered as a contributing factor for the overall enhancement. Figure 3-4 and Figure 3-5 show that Accuracy and Detection Rate increase remarkably when the size of the data samples exceeds a certain value and remain almost constant thereafter. To be more specific, it can be said that with data samples more than two hundred records, the increase in accuracy is maintained by k-medoids methods implying that this methods guarantees the predictability of the intrusion detection. Similarly, detection rates also is guaranteed to be increased with almost the same value as accuracy. The margin between two lines, as shown in the chart, are almost constant after the value of data samples passes 200 thousand.



**Figure 3-5: Increase in Detection Rate achieved by k-Medoids followed by Naïve Bayes Classification**

Figure 3-6 shows the similar but declining curve indicating that False Alarm Rate also decreases after that value and tends to be constant. Looking at it, we can further claim that the detection rate is remarkably reduced by the k-medoids method as the margin between two curves are almost constant throughout all size of data samples. This is because of the reason that the clustering quality of k-medoids is better than that of k-means in the sense that the former minimizes the squared error of the distances between the attributes of a data object and the cluster medoid.

From the above analysis, it is clear that once the learning is saturated, the proposed approach starts showing its best performance. The saturation point depends upon the attack ratio present in the datasets and may be determined by hit-and-trial basis.



**Figure 3-6: Decrease in False Alarm Rate achieved by k-Medoids clustering followed by NB Classification**

### **3.6 Chapter Summary**

Generally, it has been observed that the application of k-Medoids clustering technique, instead of k-Means, followed by Naïve Bayes classification method is better in terms of the Detection Accuracy as well as in increasing the Detection Rate by reducing the False Alarm Rate in the mean time. It is also learnt that the betterment of the proposed approach becomes more remarkable in case of large datasets.

Naïve Bayes Classification is a basic classification scheme and works fine with good data distribution. But the data distribution model of network intrusion data differs from environment to environment and hence is very hard to predict. Moreover, with the k-Medoids based clustering techniques, time complexity increases as the size of the data grows. Therefore, in order to address the time complexity of the proposed approach, an attempt to replace Naïve Bayes classification with a better unsupervised learning method that can produce high Detection Rate with a small-sized data distribution e.g. Support Vector Machine [155] could be carried out as a future work of this work.

## Chapter 4. k-Medoids-Outlier with SVM Classification

### 4.1 System Model and Problem Description

Continuing from the research work described in the previous section, the next problem in this becomes the selection of the proper classification method. As it has been already proved that k-medoids clustering is already the best method for making good quality clusters for intrusion detection system, it is no doubt chosen in this stage of work as well. But, It is still felt necessary to chose the best algorithm to detect normal and attack data from big network traffic. Moreover, an outlier analysis is also felt important to be implemented in order to detect small portions of anomalies like infrequent patterns that occurs in very less interval of time.

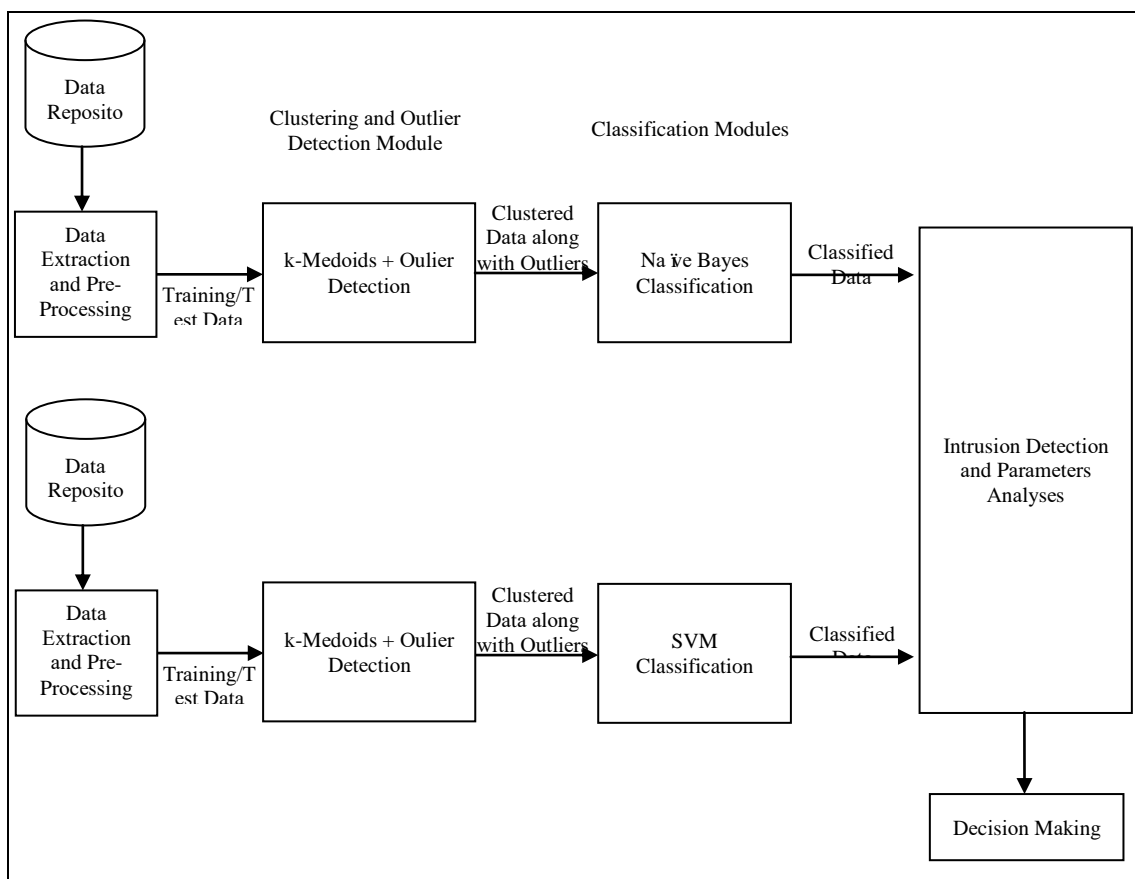


Figure 4-1: The System Module of Comparing between Naïve Bayes and SVM Classification

Anomaly based intrusion detection, in the recent years, has become more dependent on learning methods – specially on classifications schemes. To make the classification

more accurate and effective, hybrid approaches of combining with data mining techniques are commonly being introduced. In this section, a better combination is proposed to address problems of the previously proposed hybrid approach of combining k-Means/k-Medoids clustering technique with Naïve Bayes classification. In this new approach, the need of large samples by the previous approach is reduced by using Support Vector Machine while maintaining the high quality clustering of k-Medoids intact. Further, a unified clustering-outlier algorithm is developed by combining k-medoids clustering and outlier analysis. This algorithm performs k-medoids clustering and outlier analysis simultaneously without adding any computing complexity. Experiments have also been carried out by using Kyoto2006+ data sets in order to evaluate performance, accuracy, detection rate and false positive rate of the classification scheme. Experiments and analyses show that the new approach is better in increasing the detection rate as well as in decreasing the false positive rate.

#### ***4.2 General Description of the Solution***

In this study, the learning approach based on combination of k-medoid-clustering and Support Vector Machine classification technique is proposed and found out to be better than the hybrid approach of k-Means/k-Medoids combined with Naïve Bayes classification scheme in terms of accuracy, detection rate and false positive rate.

#### ***4.3 The Proposed Approach***

As the first stage of the proposed approach, similar data instances are grouped based on their behaviors by using k-Medoids-Outlier clustering technique. The resulting clusters are then classified into normal and attack classes using SVM classifiers.

##### ***k-Medoid-Outlier Detection***

The proposed algorithm, given a name as k-Medoids-Outlier algorithm, is extended from the k-medoids algorithm. The pseudocode is shown below.

The working the the algorithm is described here. The lines 1 to 6 represent the initialization section that is similar to k-medoids. All the data objects are ranked in decreasing order by their distance to their nearest cluster center in line 7. Lines 8 and 9, the top  $l$  points of this ranked list are inserted into  $L_i$  which is then removed from

the set to form the set  $X_i$ . The rest of the lines are identical to k-medoids algorithm. So, like in k-medoids algorithm, the process is repeated till the solution stabilizes.

---

***Algorithm 4- 1: k-Medoids-Outlier Algorithm***

---

**Input:**  $k$ : No. of clusters;

$l$ : No. of outliers

$X$ : Data set  $\{X_1, \dots, X_n\}$

$d$ : Distance metric ( $X \times X \rightarrow \mathbf{R}$ )

**Output:**

A set of  $k$  clusters having  $O$  as their cluster centers

A set of  $l$  outliers  $L \subseteq X$

**Let:**

$O_j$ : Set of medoid objects

$X_i$ : Set of data objects

- 1: For  $j \leftarrow 1$  to  $k$
- 2:     Set  $O_j$  to  $\text{random}(X_i)$ ;
- 3: EndFor
- 4: While (no convergence achieved) Do
- 5: For  $i \leftarrow 1$  to  $n$
- 6:     Compute  $d(x/O_{i-1})$ , for  $x \in X$ ;
- 7:     Re-order the points in  $X$  such that  $d(x_1/O_{i-1}) \geq \dots \geq d(x_n/O_{i-1})$ ;
- 8:      $L_i \leftarrow \{x_1, \dots, x_l\}$ ;
- 9:      $X_i \leftarrow X \setminus L_i = \{x_{l+1}, \dots, X_n\}$ ;
- 10:     For  $j \leftarrow 1$  to  $k-1$
- 11:         If  $|X_i - O_j| < |X_i - O_{j+1}|$  then
- 12:              $\text{min\_distance} = |X_i - O_j|$ ;
- 13:              $X_i\text{\_cluster} = j$ ;
- 14:         Else

```

15:          $min\_distance = |X_i - O_{j+1}|;$ 
16:          $X_i\_cluster = j+1;$ 
17:     EndIf
18: EndFor
19:     Add  $min\_distance$  into  $total\_distance;$ 
20: EndFor
21: For  $j \leftarrow 1$  to  $k$ 
22:     For  $i \leftarrow 1$  to  $n$ 
23:          $C = Cost(O_j, X_i); // Using Eqn. 1 //$ 
24:     EndFor
25:     If  $C < 0$  then
26:         swap( $O_j, X_i$ );
27:     EndIf
28: EndFor
29: EndDo

```

---

### ***Complexity of the Algorithm:***

The algorithm has three loops – the outermost, the middle and the innermost loops. The outermost loop iterates through each medoid object; there are  $k$  medoids. The middle loop iterates through each non-medoid object; and there are  $(n-k)$  non-medoid objects. The innermost loop iterates through each non-medoid object to compute the swapping cost of medoids and non-medoids. The complexity of each iteration of the algorithm can be expressed as  $O(k*(n-k)*k*(n-k))$  i.e.  $O((k(n-k))^2)$ . In this case, the lines 7, 8 and 9, which are responsible to carry out outlier analysis, are all linear operations and do not add any extra complexity in the algorithm.

### **Support Vector Machine**

SVM method can be applied to solve classification problems. In case of linear separable problems, SVM method supposes that training data  $\{(x_1, y_1), \dots, (x_i, y_i), \dots, (x_n, y_n), x_i \in \mathbb{R}^m, y_i \in \{+1, -1\}\}$  can be separated by a hyperplane without error. There are  $m$ -dimensional vector  $w$  and constant  $b$ , then:

$$\begin{cases} (w \cdot x_i) + b > 0, & y_i = 1 \\ (w \cdot x_i) + b < 0 & y_i = -1 \end{cases} \quad (4-1)$$

The classification problems are actually to seek suitable  $(w, b)$  to optimally separate two types of data to seek optimal classification hyperplane. Intuitively, the classification hyperplane farthest from two types of sample points may acquire optimal classifying ability. The optimal classification hyperplane depends on a small number of sample points, referred to as Support Vectors, nearest to it, and bears no relation to other samples. When the classification hyperplane is normalized, then:

$$y_i ((\mathbf{w} \cdot \mathbf{x}_i) + b) - 1 \geq 0, \quad i = 1, \dots, n \quad (4-2)$$

The classification interval is equal to  $2/\|\mathbf{w}\|$ . SVM method is to maximize the classification interval, which is equivalent to minimize  $\frac{1}{2}\|\mathbf{w}\|^2$ . The classification hyperplane that meets condition of the Equation 4-2 and minimizes  $\frac{1}{2}\|\mathbf{w}\|^2$  is called the optimal classification hyperplane.

Now, Let us Introduce Lagrange function:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \alpha_i (y_i (\mathbf{w} \cdot \mathbf{x}_i) + b), \quad \alpha_i \geq 0 \quad (4-3)$$

Set

$$\begin{cases} \frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i y_i = 0 \\ \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = \mathbf{w} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i = 0 \end{cases} \quad (4-4)$$

From Karush-Kuhn-Tucker theorem, it is seen that the optimal solution should meet KKT condition:

$$\alpha_i (y_i ((\mathbf{w} \cdot \mathbf{x}_i) + b) - 1) = 0, \quad i = 1, 2, \dots, n \quad (4-5)$$

Where  $\alpha_i$  is Lagrange multiplier corresponding to each sample, and the sample  $x_i$  corresponding to non-zero  $\alpha_i$  is Support Vector. Introduce system of Equation 4-5 into Equation 4-4, and then obtain maximum objective function:

$$\begin{aligned} Q(\boldsymbol{\alpha}) &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j), \\ \alpha_i &\geq 0, \quad \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned} \quad (4-6)$$



Suppose that the optimal parameter in Equation 4-6 is  $\alpha^* = (\alpha_1^*, \dots, \alpha_n^*)$ , and then the parameters of the optimal classification hyperplane are:

$$\begin{aligned} \mathbf{w}^* &= \sum_{i=1}^n \alpha_i^* y_i \mathbf{x}_i \\ b^* &= y_i - \mathbf{w}^* \cdot \mathbf{x}_i \end{aligned} \quad (4-7)$$

The optimal classification hyperplane is  $\sum_{i=1}^n \alpha_i^* y_i \mathbf{x}_i \cdot \mathbf{x}_j + b^* = 0$ , so the optimal classification function can be obtained as:

$$f(\mathbf{x}) = \text{sgn} \left\{ \sum_{SV} \alpha_i^* y_i (\mathbf{x}_i \cdot \mathbf{x}_j) + b^* \right\} \quad (4-8)$$

### **SVM Training Algorithm**

SVM training algorithm can be divided into two major types. One type is “chunking algorithm”, which is based on the fact that the solution of the original problem is not affected if the training sample whose Lagrange multiplier is equal to zero is removed [156]. The aim of “chunking algorithm” is to gradually exclude non-support vectors through certain iterative pattern.

The specific process is as follows: select some samples to constitute work sample set for training, eliminate non-support vectors therein, test the remaining samples with training result, combine the samples (or part of these samples) failing to conform to the training result (generally refer to violating KKT condition) and the support vectors in the training result into a new work sample set, and re-train the new set. The process is repeated until the optimal result is achieved. When the number of support vectors is far smaller than that of training samples, it is obvious that “chunking algorithm” can greatly improve operational rate. However, if the number of support vectors is comparatively larger, the work sample set will become increasingly larger with increase in iterative times of algorithm, and the algorithm will still grow into a complex one [157].

The other one is to divide the problem into sub-problems with fixed number of samples: the size of work sample set is controlled within allowable limit of algorithmic rate, and the iterative process is just to conduct equal quantity exchange between some “state-worst samples” among the remaining samples and the samples in

the work sample set [158]. Even though the number of support vectors outweighs the size of the work sample set, the scale of the work sample set is not changed, and only part of support vectors are optimized.

**Procedure of SVM Algorithm**

Let D be a classification dataset with n points in a d-dimensional space  $D = \{(x_i, y_i)\}$ , with  $i = 1, 2, \dots, n$  and let there be only two class labels such that  $y_i$  is either +1 or -1. A hyperplane  $h(x)$  gives a linear discriminant function in d dimensions and splits the original space into two half-spaces:

$$h(x) = w^T x + b = w_1x_1 + w_2x_2 + \dots + w_dx_d + b, \quad (4-9)$$

where  $w$  is a d-dimensional weight vector and  $b$  is a scalar bias. Points on the hyperplane have  $h(x) = 0$ , i.e. the hyperplane is defined by all points for which  $w^T x = -b$ .

If the dataset is linearly separable, a separating hyperplane can be found such that for all points with label -1,  $h(x) < 0$  and for all points labeled +1,  $h(x) > 0$ . In this case,  $h(x)$  serves as a linear classifier or linear discriminant that predicts the class for any point. Moreover, the weight vector  $w$  is orthogonal to the hyperplane, therefore giving the direction that is normal to it, whereas the bias  $b$  fixes the offset of the hyperplane in the d-dimensional space.

Given a separating hyperplane  $h(x) = 0$ , it is possible to calculate the distance between each point  $x_i$  and the hyperplane by:

$$\delta_i = \frac{y_i h(x_i)}{\|w\|} \quad (4-10)$$

The margin of the linear classifier is defined as the minimum distance of all n points to the separating hyperplane.

$$\delta^* = \min_{x_i} \left\{ \frac{y_i h(x_i)}{\|w\|} \right\} \quad (4-11)$$

All points (vectors  $x^*i$ ) that achieve this minimum distance are called the support vectors for the linear classifier. In other words, a support vector is a point that lies precisely on the margin of the classifying hyperplane.

In a canonical representation of the hyperplane, for each support vector  $x_i^*$  with label  $y_i^*$  we have that  $y_i^* h(x_i^*) = 1$ . Similarly, for any point that is not a support vector, we have that  $y_i h(x_i) > 1$ , since, by definition, it must be farther from the hyperplane than a support vector. Therefore we have that  $y_i h(x_i) \geq 1, \forall x_i \in D$ .

The fundamental idea behind SVMs is to choose the hyperplane with the maximum margin, i.e. the optimal canonical hyperplane. To do this, one needs to find the weight vector  $w$  and the bias  $b$  that yield the maximum margin among all possible separating hyperplanes, that is, the hyperplane that maximizes  $\frac{1}{\|w\|}$ . The problem then becomes that of solving a convex minimization problem (notice that instead of maximizing the margin  $\frac{1}{\|w\|}$ , one can obtain an equivalent formulation of minimizing  $\|w\|$ ) with linear constraints, as follows:

*Objective Function:* -

$$\min \frac{\|w\|^2}{2} \quad (4-12)$$

*Linear Constraints:* -

$$y_i h(x_i) \geq 1, \forall x_i \in D \quad (4-13)$$

This minimization problem can be solved using the Lagrange multiplier method, which introduces a Lagrange multiplier  $\alpha$  for each constraint:

$$\alpha_i (y_i h(x) - 1) = 0 \text{ with } \alpha_i \geq 0 \quad (4-14)$$

This method states that  $\alpha_i = 0$  for all points that are at a distance larger than  $\frac{1}{\|w\|}$  from the hyperplane, and only for those points that are exactly at the margin, i.e. the support vectors,  $\alpha_i > 0$ . The weight vector of the classifier is obtained as a linear combination of the support vectors, while the bias is the average of the biases obtained from each support vector.

#### **4.4 Experimental Results and Analysis**

A set of experiments consists of three phases – clustering, classification including training and prediction, and finally cross validation of the classification scheme. So, a simulation program is built for the following three hybrid approaches: -

- (1) k-Means clustering followed by Naïve Bayes classification,
- (2) k-Medoids clustering followed by Naïve Bayes classification, and
- (3) k-Medoids clustering followed by Support Vector Machine classification.

Since the main purpose is to evaluate the proposed approach i.e. SVM classification with k-Medoids clustering, all the programs are carried out using the same datasets and in the same operating and hardware environment. The program (1) and program (2) confirm that NB combined with k-Medoids is better than that combined with k-Means. Finally, the results from the both programs were compared, evaluated and analyzed.

In order to carry out experiments related to this research work, following platforms are used: -

**Table 4-1: Experimental platform for k-Medoids-Clustering with SVM classification**

Operating System :	Linux (Ubuntu)
Clustering and Classification Tool:	Rapidminer 5.0
Algorithm coding and testing platform:	Matlab
Performance Analysis Tool:	Rapidminer 5.0
Data Analysis and Graphical Representation Tool:	LibreOffice
Data Source:	Kyoto 2006+ Data

#### **4.4.1 Selection of Experimental Data**

In this work, Kyoto 2006+ dataset is used as evaluation data for the experiments. After examination of the data and preliminary experiments, the datasets of the following dates were chosen for the use in the experiments: -

- (1) 2006 Nov. 1-3 containing 238179 records;

- (2) 2008 Dec. 1, 15 and 22 with 260634 records;
- (3) 2009 July 1, 8, 15 and 22 having 500865 records;
- (4) 2009 Aug 25-31 consisting of 900924 records.

#### 4.4.2 Description and Samples of Experimental Data

The detailed description of data used in the simulation and experiment works in this piece of research work is mentioned in Appendix I: Features of Kyoto+ 2006 dataset.

In order to provide the rough idea about the experimental data, a tiny portion of a snapshot of typical one day's experimental data is shown below. For more data, please refer to the Appendix II: Experimental Data Samples.

Duration	Service	Source bytes	Destination bytes	Count	Same svr rate	Server rate	Svr server rate	Dst host count	Dst host svr count	Dst host same src port rate	Dst host server rate	Dst host svr server rate	Flag	IDS detection	Malware detection	Estimula detection	Label
3.07	S	0	0	0	0	0	0	0	100	0	0	0	SH	0	0	0	1
0	S	0	0	1	1	0	0	0	100	0	0	0	SH	0	0	0	1

{See Appendix II for more...}

#### 4.4.3 Data Pre-processing

A data pre-processing techniques like sampling and filtering are applied for easy and smooth operation of the experiments. After an exhaustive examination on hit-and-trial basis, data samples are extracted such that each set contains 1% of attacks including unknown attacks too.

Attributes of the datasets are converted to appropriate types and normalized according to the need of the SVM kernel. Samples with both known and unknown attacks are treated as the single attack class and labeled as -1. Thus, the entire selected data can be categorized into {-1, 1} where 1 represents normal class. This makes the application of SVM very simple and almost any type of SVM classification may be implemented. Moreover, the categorical attributes of the data are treated differently

for the use in k-Means clustering only as it cannot handle this data type. They are converted into numerical values e.g. {REJ, S0, RST0, SF} is encoded as {1, 2, 3, 4}.

#### 4.4.4 The Experimental Procedure

From the selected data sets listed above, small data sets of 10,000 (ten thousand) and large data sets of 100,000 (one hundred thousand) records are extracted. A complete set of experiments is carried out both on small and large data sets. The idea of experimenting on such a differing sized data helps examine whether NB and SVM classifiers produce different performance measures.

Using every data samples, one by one, both programs are executed. The number of true positive, true negative, false positive and false negative values of both the programs are recorded and used in the performance evaluation of both the programs.

#### 4.4.5 Performance Evaluation

The performance evaluation of the experiment is carried out in terms of accuracy ( $A$ ), detection rate ( $DR$ ) and false alarm rate ( $FAR$ ) by using the Equation 3-4, Equation 3-5 and Equation 3-6 respectively.

The followings tables show Accuracy, Detection Rate and False Alarm Rates of the proposed SVM approach compared to NB based approaches. Only a portion of the complete list of results is presented here but the average values have been calculated using the complete results.

**Table 4-2: Comparison of accuracy of Naïve Bayes classification with SVM (with 10,000 records)**

Data Set	NB (with k-Means)	NB (with k-Medoids)	SVM (with k-Medoids)
1	97.40	97.73	99.33
2	90.15	90.68	99.29
3	95.56	97.28	99.48
4	79.51	79.17	99.51
5	97.56	96.83	99.79
6	96.86	97.08	99.41
7	94.37	94.65	99.34
8	96.02	96.32	99.26
9	95.21	95.39	99.43
10	96.12	97.36	99.46

Table 4-2 and Table 4-3 show comparisons of Accuracy between NB and SVM classification in case of small datasets and large datasets respectively. Values in Table 4-2 show that accuracies of NB approaches range from 79% to 97%. It is seen that NB combined with k-Medoids has improved it a little but the proposed SVM approach has greater accuracy rate and is almost constant for every set of data sample. Again, the average accuracies of NB approaches are 93.87% and 94.25% respectively whereas that of SVM is 99.43%.

**Table 4-3: Comparison of Accuracy of Naïve Bayes Classification with SVM (with 100,000 Records)**

Data Set	NB (with k-Means)	NB (with k-Medoids)	SVM (with k-Medoids)
1	85.55	86.18	99.34
2	73.86	78.29	99.19
3	91.64	95.57	99.23
4	85.26	89.47	99.24
5	87.79	92.02	99.34

Similarly, in case of large datasets, as seen in the Table 4-3, the average accuracies of NB approaches are 84.82% and 88.30%. But with the proposed SVM method shows the equally constant accuracy with an average value of 99.21%.

**Table 4-4: Comparison of detection rate of Naïve Bayes classification with SVM (with 10,000 records)**

Data Set	NB (with k-Means)	NB (with k-Medoids)	SVM (with k-Medoids)
1	96.77	97.43	99.75
2	86.90	87.73	99.87
3	96.66	96.75	99.87
4	96.47	96.54	99.81
5	98.69	98.69	99.97
6	96.35	96.68	99.83
7	98.06	98.19	99.72
8	95.83	95.88	99.77
9	96.46	96.52	99.67
10	97.37	98.22	99.53

Comparisons of Detection Rates among NB and SVM approaches in case of small data sets are presented in Table 4-4; and Table 4-5 presents the same in case of large datasets. Table 4-4 shows the average Detection Rates of NB approaches for small data sets viz. 95.96% and 96.26% respectively whereas the Detection Rate of SVM is found to be 99.78%, which is again much better than the former approach.

**Table 4-5: Comparison of detection rate of Naïve Bayes classification with SVM (with 100,000 records)**

<b>Data Set</b>	<b>NB (with k-Means)</b>	<b>NB (with k-Medoids)</b>	<b>SVM (with k-Medoids)</b>
1	96.95	97.67	99.61
2	66.51	70.69	99.19
3	84.62	89.35	99.35
4	84.98	88.43	99.31
5	88.53	92.39	99.03

A similar result is seen in Table 4-5 too. The average Detection Rate of SVM approach is 99.30% that is more than that of NB approaches viz. 84.32% and 87.71%.

**Table 4-6: Comparison of false alarm rate of Naïve Bayes classification with SVM (with 10,000 records)**

<b>Data Sets (10000 each)</b>	<b>NB (with k-Means)</b>	<b>NB (with k-Medoids)</b>	<b>SVM (with k-Medoids)</b>
1	2.52	1.99	0.52
2	9.43	8.78	0.30
3	4.08	4.10	0.30
4	2.99	2.90	0.73
5	2.40	2.37	0.05
6	2.11	1.92	0.42
7	1.93	1.80	0.82
8	2.81	2.80	0.51
9	2.78	2.74	0.45
10	4.50	3.06	0.45

Again, it is seen from Table 4-6 and Table 4-7 that False Alarm Rates are also improved with SVM approach. False Alarm Rates of NB range from 1% to 9% approximately whereas that of SVM is less than 1% and is found almost constant throughout all datasets.

**Table 4-7: Comparison of false alarm rate of Naïve Bayes classification with SVM (with 100,000 records)**

<b>Data Sets (100000 each)</b>	<b>NB (with k-Means)</b>	<b>NB (with k-Medoids)</b>	<b>SVM (with k-Medoids)</b>
1	2.29	1.61	0.46
2	5.29	4.34	0.97
3	6.52	5.46	0.58
4	2.98	2.05	0.77
5	3.29	2.24	0.99



#### 4.4.6 Analysis of the Results

From the experimental results and the performance evaluation, it is seen that Accuracy and Detection Rate have been increased whereas False Alarm Rate has been decreased by SVM approach combined with k-Medoids in case of every size of data samples.

In the following section, analyses of both NB and SVM approaches in case of small and large datasets are discussed with the help of Receiver Operating Characteristics (ROC) curves. Figure 4-2 shows ROC curve produced by NB with k-Means and k-Medoids in case of small datasets. The curve produced by the latter seems to be slightly better but both are not smooth though.

Naïve Bayes classification is generally independent of past data samples and, therefore, is considered to be faster. But it is also true that the independence of NB is mostly satisfied by the attributes of sample dataset. In the Figure 4-2, the reason why the ROC curve is not smooth may be because of the data samples. The initial portion of the curve is not smooth at all, which indicates that small size data samples (like 10 thousand records) do not always produce smooth curves while combined with k-means or medoids clustering method.

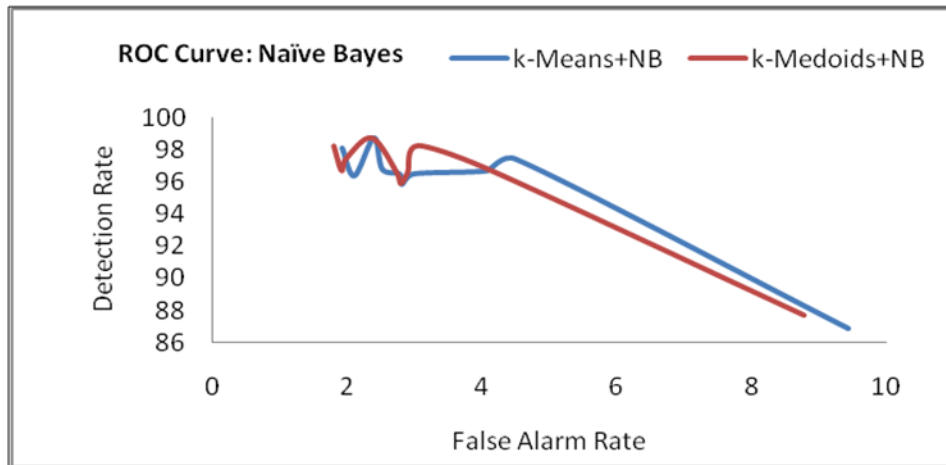


Figure 4-2: ROC Curve produced by NB Classification with k-Means and k-Medoids (with 10,000 records)

Unlike NB approach, the ROC curve produced by SVM in case of small data sets is smoother as shown in Figure 4-3, which explains the decreasing tendency of False Alarm Rate while Detection Rate is increasing. Since SVM has capability of produce good classified data even from small data samples, the ROC curve produced by SVM

is smoother than that of NB classification. Moreover, SVM preceded by k-medoids guarantees the continuous improvement in terms of detection rate and false alarm rate. But even the ROC of k-medoids followed by SVM is not ideologically smooth; it may be because of the nature of the distribution of attack data in the data sample.

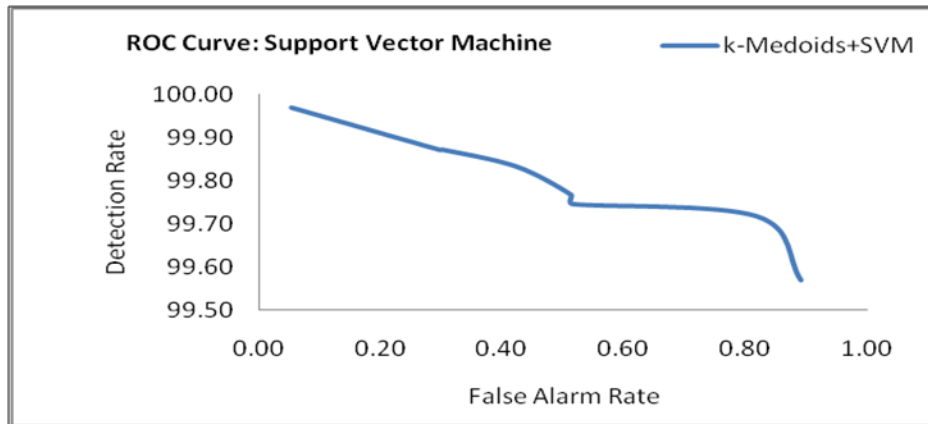


Figure 4-3: ROC Curve produced by SVM with k-Medoids Clustering (with 10,000 records)

Figure 4-4 and Figure 4-5 show ROC curves produced in case of large datasets respectively by NB and SVM classifications. Figure 4-4 shows a relatively smooth ROC curve of NB in case of large datasets compared to that in case of small datasets but the curves are not declining, which means NB does not guarantee the inversely proportional relationship of Detection Rate with False Alarm Rate.

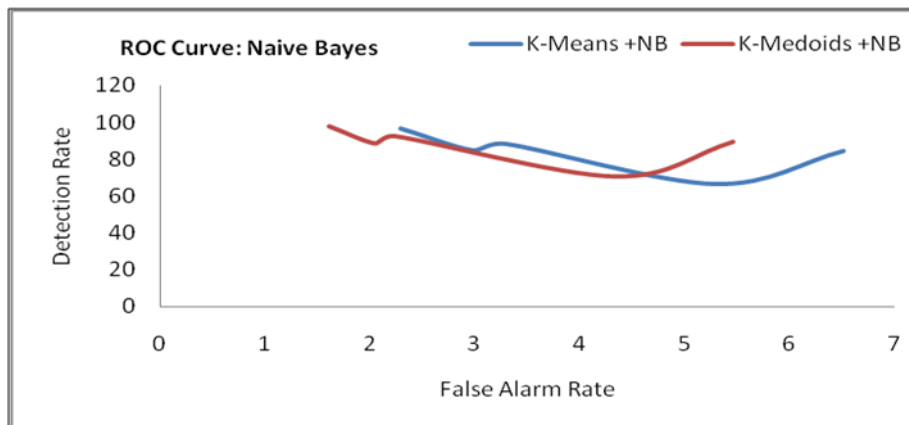


Figure 4-4: ROC Curve produced by NB Classification with k-Means and k-Medoids (with 100,000 records)

But in Figure 4-5, ROC curves produced in case of large datasets by SVM classification seems relatively smoother and declining.

Looking at all tables and ROC curves presented above, it has become obvious that k-Medoids help NB classification in improving its overall performance but does not

contribute in maintaining the ideological ROC curve. The proposed SVM with k-Medoids approach not only increases the overall performance of classification but also produces smoother and more declining ROC curve.

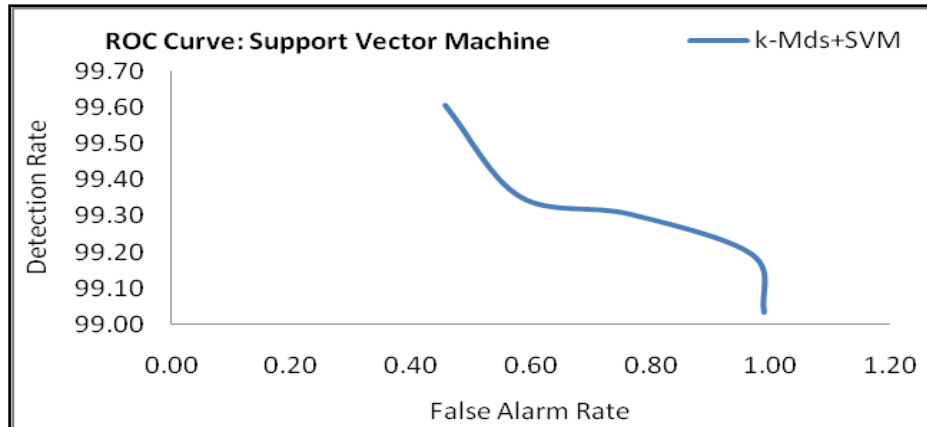


Figure 4-5: ROC Curve produced by SVM with k-Medoids Clustering (with 100,000 records)

#### 4.5 Chapter Summary

A comparatively better clustering method k-Medoids is combined with Support Vector Machine classification to produce better performance in terms of Accuracy, Detection Rate and False Alarm Rate. From the experiments and analyses, it has been shown that the proposed hybrid approach has outperformed the previous approach of combining Naïve Bayes classification with k-Means/k-Medoids clustering. Therefore, intrusion detection can be more effective and efficient with the proposed approach.

The proposed approach can further be made more efficient and effective, by applying multiple kernel based SVM classification [159] schemes, in detecting various known and unknown attacks and then separating them into correct categories. Moreover, the time complexity of k-Medoids clustering can still be reduced by implementing more efficient and accurate clustering for large data sets like MapReduce or OptiGrid [160] clustering techniques.

## Chapter 5. Incremental Support Vector Machine with CSV-ISVM

### 5.1 System Model and Problem Description

The problem, here, is to improve the SVM in such a way that it should classify normal an attack data very accurately – with high detection rate and lower false positive. An iterative approach is one of the options but it incurs more time complicity and therefore, there is also a challenge to reduce the time taken due to the incremental approach. How to make minimum possible iterations is one issue while how to select minimum possible support vectors is the another issue.

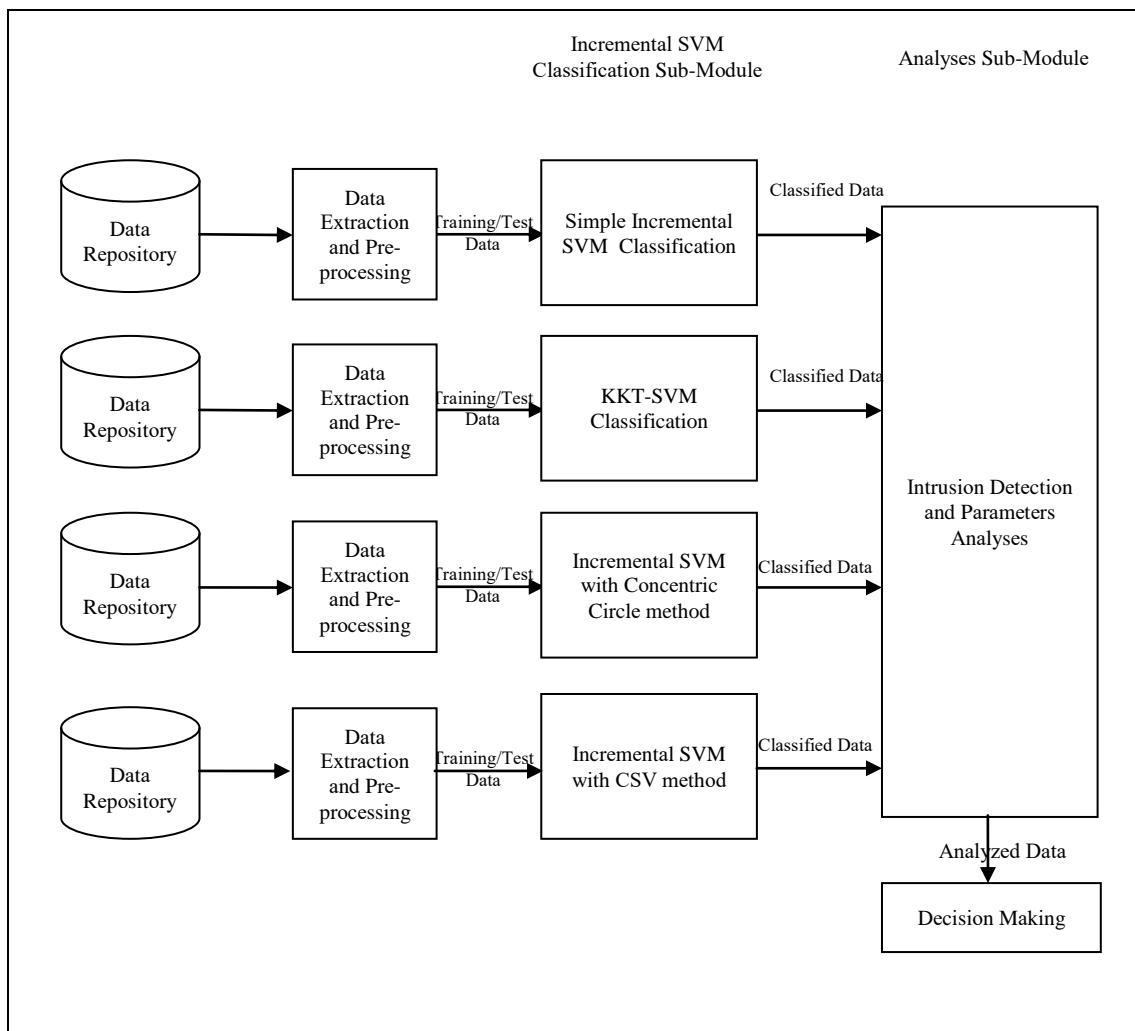


Figure 5-1: The System Module of Incremental SVM with Support Vector Selection

In an Incremental Support Vector Machine classification, the data objects labelled as non-support vectors by the previous classification are re-used as training data in the

next classification along with new data samples verified by Karush-Kuhn-Tucker (KKT) condition. This piece of work proposes Half-partition strategy of selecting and retaining non-support vectors of the current increment of classification - named as Candidate Support Vectors (CSV) - which are likely to become support vectors in the next increment of classification. This research work also designs an algorithm named the Candidate Support Vector based Incremental SVM (CSV-ISVM) algorithm that implements the proposed strategy and materializes the whole process of incremental SVM classification.

The work also suggests modifications to the previously proposed concentric-ring method and reserved set strategy. The performance of the proposed method is evaluated with experiments and also by comparing it with other ISVM techniques. The experimental results and performance analyses show that the proposed algorithm CSV-ISVM is better to be used in real-time network intrusion detection than general ISVM classifications.

## ***5.2 General Description of the Solution***

This paper proposes an improved and efficient learning approach of Incremental Support Vector Machine (ISVM) based on the idea of retaining the original and current data samples throughout the whole learning process. In this proposed approach, data points, that are not the support vectors in the current increment of classification but have chances of becoming support vectors in the next increment of classification, are selected and retained as the Candidate Support Vectors (CSVs) so that they can be combined with the new training data set that will be added in the next training. This approach also introduces Half-Partition strategy as a part of the CSV selection method and devices an algorithm, named CSV-ISVM.

## ***5.3 Candidate Support Vectors based Incremental SVM***

The key of the proposed Incremental Support Vector Machine is the selection of Candidate Support Vectors. Here, two methods of selecting CSVs are presented: – (1) Improved Concentric Circle method; and (2) Half-Partition strategy.

Since the latter method is proven to be better, the algorithm designed in this work i.e. the CSV Selection algorithm makes use of the Half-Partition method. New data samples used for training in each increment is verified if they can become support vector by using the decision function suggested by KKT-theory.

### **KKT conditions for ISVM**

In SVM classification, a decision function is obtained by solving a quadratic program [161]. To get an optimal solution, the following Karush–Kuhn–Tucker (KKT) conditions must be satisfied: -

$$\begin{cases} \alpha_i = 0 \Rightarrow f(x_i) > 1 \text{ or } f(x_i) < -1 \\ 0 < \alpha_i < C \Rightarrow f(x_i) = 1 \text{ or } f(x_i) = -1 \\ \alpha_i = C \Rightarrow -1 < f(x_i) < 1 \end{cases} \quad (5-1)$$

Where  $\alpha_i$  is the Lagrange multiplier corresponding to the samples, and  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_i, \dots)$  is the optimal solution, if and only if every sample  $x_i$  meets these conditions.

Here,  $f(x) = 0$  is the optimum separating hyperplane,  $f(x) = \pm 1$  are the boundaries of the separating margin. Therefore, for a training sample  $x_i$ , if  $\alpha_i = 0$ , then it lies outside the boundaries; if  $0 < \alpha_i < C$ , then it lies on either of the boundaries; and if  $\alpha_i = C$ , then it lies inside the boundaries of the separating margin.

### **Rotations of the SV Hyperplane**

The idea of Half-Partition method is arisen from the phenomenon of SV hyperplane rotations. So, here, the possible geometric rotations of the SV hyperplane are illustrated. As seen from Figure 5-2, the hyperplane can rotate either to clockwise or to anti-clockwise direction with respect to the current position of the hyperplane [162]. No change in the rotation implies that the new samples satisfy the KKT conditions. To list the possible SV hyperplane rotations: -

- (1) Towards the anti-clockwise direction.
- (2) Towards the clockwise direction.
- (3) No Rotation

In Figure 5-2, solid circles and solid squares represent the original samples, while the circles enclosed by hollow squares represent new incremental samples. The original

hyperplane is  $f(x) = 0$ . Samples  $S_1$  and  $S_2$  are support vectors. Say, new samples  $i$  and  $j$  are introduced for incremental SVM learning purpose. Since they violate the KKT conditions, they lead to the rotation of the optimal hyperplane in clockwise and anti-clockwise directions respectively in the next iteration of the SVM learning stages. The non-support vectors  $\{b, c, j\}$  and  $\{e, i\}$ , then, change to the support vectors.

It has been obvious that if we consider new samples and original support vectors only, and discard original non-support vectors, some valuable information will be lost, which may result in obtaining a bad classifier.

The later learning increments will lead to oscillation if the initial samples are not enough. However, if all of non-support vectors are included in the learning process, then there will be more unwanted data instead of the real candidate support vectors. Meanwhile, with new samples coming in continuously, the size of the training set will eventually become too large. So, a method to select CSVs is explained to make ISVM more efficient.

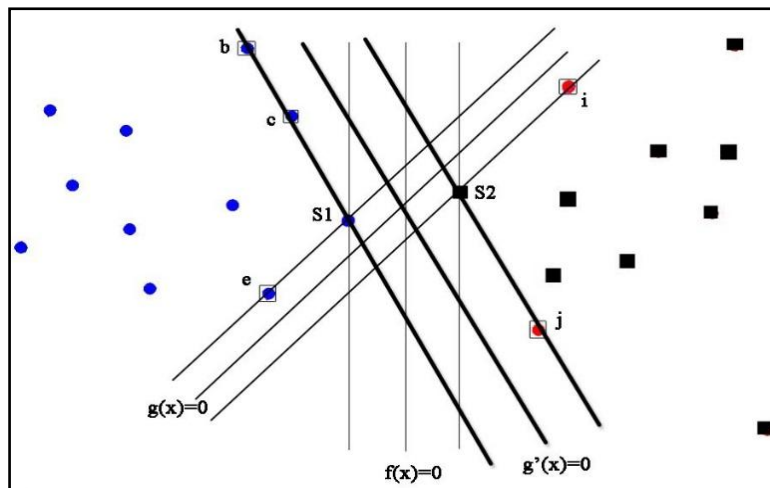


Figure 5-2: Hyperplane rotations due to new sample sets  $\{b,c,j\}$  &  $(e,i)$

### 5.3.1 The Improved Concentric Circle Method

The decision function of SVM is determined by the support vectors. To determine which samples are most likely to be support vectors and thus to make the CSV set, this paper proposes Improved Concentric Circle method, which actually is a modification to the Concentric Circle method proposed by Yang et al. [163]. It explained that most of the marginal samples lie in the ring region between the two

concentric circles, as shown in Figure 5-3, and therefore, the Concentric Circle method retains the samples lying in the ring region. But, the process of constructing the ring i.e. determining radii  $R_i$  and  $R_o$  of the inner and the outer circle respectively needs separate experiments that keep on changing with the data samples. So, the proposed Improved Concentric Circle method suggests a fixed way of determining  $R_i$  and  $R_o$ .

Firstly, the centres of all classes are calculated as follows:

$$m_{ij} = \frac{1}{n_i} \sum_{k=1}^{n_i} x_{kj} \quad (5-2)$$

Where  $j = 1, 2, \dots, d$ ;  $d$  is the dimension of the mean vector.  $n_i$  is the number of samples belonging to class  $i$  in sample set, and  $x_{kj}$  is the  $j^{\text{th}}$  attribute of sample  $k$ .  $i = 1, 2$ , stands for the sample class.

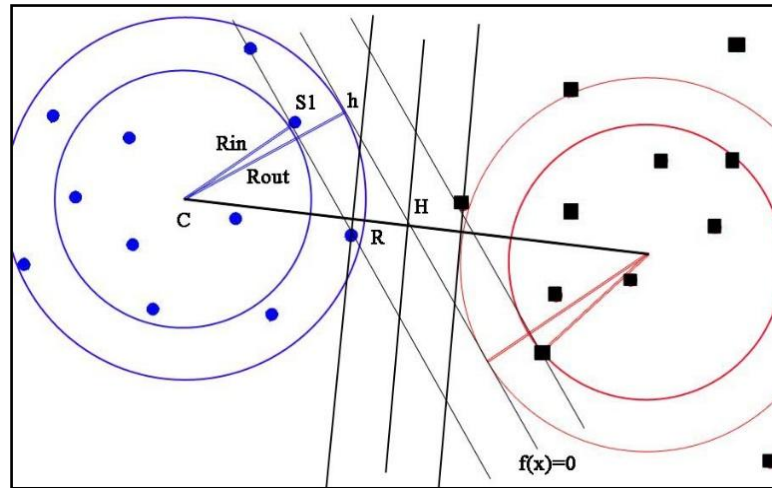


Figure 5-3: The Improved Concentric Circle method for selecting CSVs

Secondly, calculate the Euclidean distance  $R$  between two class centres. For the binary classification problem, it can be denoted as:

$$R = r(m_1, m_2) = \sqrt{\left(\sum_{k=1}^d |m_{1k} - m_{2k}|^2\right)} \quad (5-3)$$

Where  $m_1$  and  $m_2$  are mean vectors,  $m_{1k}$  and  $m_{2k}$  are the  $k^{\text{th}}$  dimension of the mean vectors. The Euclidean distance between two samples is calculated by the following equation: -

$$d_{ij} = \sqrt{\left(\sum_{k=1}^m |x_{ik} - x_{jk}|^2\right)} \quad (5-4)$$



### Selection of Radii $R_i$ and $R_o$

Selection of  $R_i$  and  $R_o$  is vital and sensitive too. The method should be such that data points lying inside the inner circle has the least probabilities of being support vectors and that lying outside the outer circle do not have any. So, we take  $R_i$  and  $R_o$  as following measurements: -

$R_i$  = the distance between the class centre  $C$  and the nearest support vector  $S_i$

$R_o$  = the distance between the class centre  $C$  and the nearest point at the hyperplane  $h$ .

Here,  $R_i$  can be calculated using Equation 5-3. We know that the mid-point of two class centres, say  $H$ , lies on the hyperplane. If  $R$  is the distance between two class centres, then  $R$  is given by Equation 5-2. Once we know  $R$ ,  $R_o$  may be calculated as  $R_o = R/2$ . But this paper recommends an alternate method of calculation as given in the following section.

### Calculation of $r$

Suppose,  $r$  is the Euclidean distance from a point  $\vec{x}$  to the decision boundary, as shown in Figure 5-4, say at the point  $x'$ . We know that the shortest distance between a data point and a hyperplane is perpendicular to the plane [164], and hence, parallel to  $\vec{w}$ . A unit vector in this direction is  $\vec{w}/|\vec{w}|$ . The dotted line in the diagram is then a translation of the vector  $\vec{w}/|\vec{w}|$ . Let us label the point on the hyperplane closest to  $\vec{x}$  as  $\vec{x}'$ . Then:

$$\vec{x}' = \vec{x} - yr \frac{\vec{w}}{|\vec{w}|} \quad (5-5)$$

where multiplying by  $y$  just changes the sign for the two cases of  $\vec{x}$  being on either side of the decision surface. Moreover,  $\vec{x}'$  lies on the decision boundary and so satisfies  $\vec{w}^T \vec{x}' + b = 0$ . Hence:

$$\vec{w}^T (\vec{x} - yr \frac{\vec{w}}{|\vec{w}|}) \quad (5-6)$$

Solving for  $r$  gives:

$$r = y \frac{\vec{w}^T \vec{x} + b}{|\vec{w}|} \quad (5-7)$$

The advantage of this calculation method is that we do not need to calculate distance between two class centres,  $R$ , which makes the calculation dependent only on the current class.

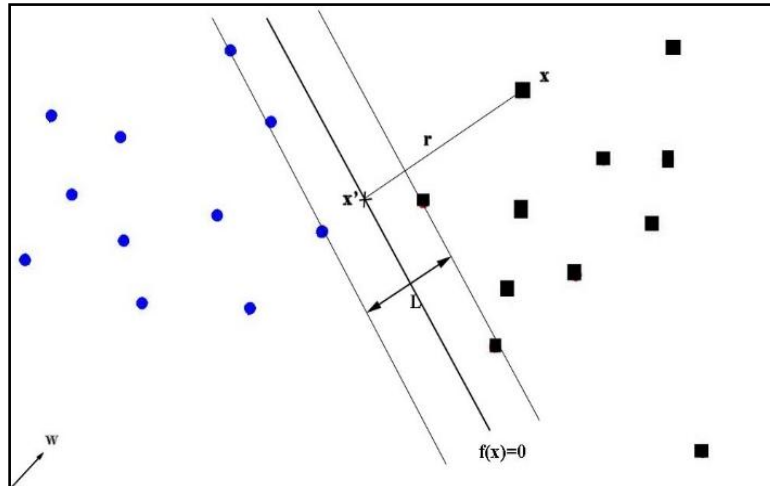


Figure 5-4: Distance ( $r$ ) of a data point ( $x$ ) from the hyperplane

### 5.3.2 The Half-Partition Strategy

In Figure 5-2, there are data points even outside the ring-region. These data points are not being considered as the Candidate Support Vector by the Concentric Circle methods. Unless we prove they are real outliers (i.e. do not belong to the same class) or do not contribute in classification process, they cannot be excluded from the CSV set. So, this research work considers removing the outer circle (then the ring no longer exists as shown in Figure 5-5) so as to maximize the CSV set. By so doing, all non-support vectors lying outside the inner circle (having radius  $R_i$ ) and also the outliers belong to the greater CSV set, thus providing them the equal opportunity to become a CSV.

Furthermore, looking at the possible rotations of the SV hyperplanes in *Figure 15*, it is seen that the outer half of the entire data space covered by each class is not touched by the rotated hyperplane i.e.  $g(x) = 0$  and  $g'(x) = 0$  respectively, which means that outer halves of the classes do not contain data points that could be selected as CSVs for the next increment. And, this situation is very true in case of two-class classification. Since network intrusion detection is a binary classification problem, the

outer-partitions of the both classes can be avoided for CSV selection. Hence, this paper proposes the Half-Partition method illustrated in Figure 5-5, which considers data points lying only at the inner half partitions for the CSV selection.

If a non-support vector, say a data point  $x$  in Figure 5-5, satisfies the following condition, then it will be considered as a Candidate Support Vector: -

$$d \geq R_i \text{ AND } r \leq R_o \quad (5-8)$$

where  $d$  is the distance between  $x$  and the class centre  $C$ , and  $r$  is the distance between  $x$  and the hyperplane.

---

**Algorithm 5- 1: CSV Selection**

---

**Input:** Sample set  $X_0$

**Output:** The CSV set

//Calculate the mean vector of each class//

1: **For** every sample  $x_i$  ( $i=1,2, \dots, l$ ) in  $X_0$

2:     **If**  $x_i$  belongs to the normal class

3:         **For**  $k=1, \dots, d$

4:              $m_1(k) = m_1(k) + x_i(k);$

5:         **EndFor**

6:          $n_1 = n_1 + 1;$

7:     **Else** // $x_i$  belongs to attack class//

8:         **For**  $k=1, \dots, d$

9:              $m_2(k) = m_2(k) + x_i(k);$

10:         **EndFor**

11:          $n_2 = n_2 + 1;$

12:     **EndIf**

13: **EndFor**

14: **For**  $k=1, \dots, d$

```

15:    $m_1(k) = m_1(k)/n_1;$ 
16:    $m_2(k) = m_2(k)/n_2;$ 
17: EndFor
18: Calculate the radius  $R_i$  using Equation 5-3;
      //Select samples to construct the CSV set//
19: For every sample  $x_i$  ( $i = 1, 2, \dots, l$ )
20:   Calculate distance  $d$  from  $x_i$  to the mean vector of the class that  $x_i$  belongs to;
21:   Calculate the distance  $r$  from  $x_i$  to the hyperplane at  $h$ ;
22:   If  $d \geq R_i$  and  $r \leq R_o$ 
23:     Add  $x_i$  into the CSV set;
24:   EndIf
25: EndFor

```

---

### ***Complexity of the Algorithm:***

SVM classification algorithm has two complexities – one is at training time and another at the testing time. For a SVM classification algorithm that uses RBF kernel, support vectors are selected during the training time. In the testing time, time complexity is linearly based on the number of the support vectors, which is given by training set size \* training set error rate. It is also linearly based on the number of features. Therefore,  $O(n\_samples^2 * n\_features)$  is the complexity of RBF kernel based SVM classification algorithm.

In case of incremental SVM, the selection and retaining the support vectors are done as long as there is new data sample. Now, the complexity of incremental SVM becomes  $O(n\_samples^2 * n\_features)$  multiplied by no of iterations, which is fixed/constant for a typical implementation. It means the complexity of an incremental SVM remains the same as the non-incremental SVM classification. The CSV Selection algorithm is an improvement to the SVM algorithm, which do not affect the overall complexity of the SVM classification algorithm. So, the complexity still is given by  $O(n\_samples^2 * n\_features)$ .

### Calculation of Weights of CSV

Probability of being a CSV depends on two parameters: -

$d$  = the Euclidean distance between Candidate Support Vector,  $x$  in Figure 5-4, and the class centre  $C$ ; and

$r$  = the distance between the Candidate Support Vector  $x$  and the SVM hyperplane,  $x'$ .

Here,  $d$  and  $r$  can be calculated using Equation 5-4 and Equation 5-7 respectively.

The data samples which lie far away from the hyperplane has less probabilities of becoming support vectors. Similarly, the data points lying inside the circle also has equally less probability of being support vectors. So, we can say that data points lying outside the circle and nearer to the hyperplane have maximum probability.

The greater is  $d$ , higher will be the possibility; and the smaller is  $r$ , higher will be the chance of being support vector. Therefore,  $d \geq R_i$  and  $r \leq L$  are the two criteria for formulating the equation for weight calculation. Here  $L$  is the distance between the support vector  $S_i$  and the hyperplane, which is given by  $1/\bar{w}$  (We know that  $2/\bar{w}$  is the margin of the decision boundary).

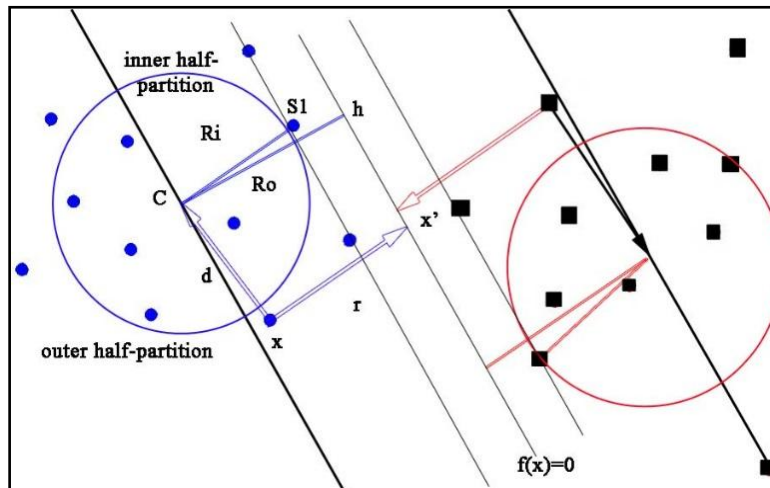


Figure 5-5: The Half-Partition strategy and Weight calculation of CSVs

For weight calculation, the following expression is proposed: -

$$W_c = \{d/(d+R_i) + L/(L+r)\} \quad (5-9)$$

where  $W_c$  is the total weighted distance of the Candidate Support Vector. For the Support Vector nearest from the class centre  $C$ ,  $d=R_i$  and  $r=L$ , therefore,  $d/(R_i+d)=0.5$

and  $L/(L+r)=0.5$  and  $W_C=1$ . So, all the non-support vectors which yields the value of  $W_C$  close to 1 (one) are selected as CSVs.

### ***Threshold Assignment***

From the previous section, we learnt that threshold  $T$  may be defined as  $W_C-1$  such that  $T = 0$  for a typical support vector. And, the equation for the threshold  $T$  may be expressed as

$$W_C-1 = \{d/(d+R_i)+L/(L+r)\}-1 \quad (5-10)$$

Depending upon the distribution of data points and nature of occurring new data samples, the threshold  $T$  may be set variably. For instance, we can take  $T = \pm 0.25$  could be a decision factor to consider a data point that yields the weighted value  $W_C$  either equal to 1.25 or 0.75.

### ***5.3.3 CSV Selection Algorithm and CSV-ISVM Algorithm***

The first-ever CSV Selection algorithm triggers as soon as the first SVM learning is finished. Using the CSV Selection algorithm, described in Algorithm 5-1, CSVs are selected and retained for the next iteration of the SVM classification. During the CSV selection process, the weights of CSVs are also preserved along with the CSV sets. These CSV weights may also be determined by a threshold value.

Every time a new incremental sample appears, simple incremental SVM checks whether all the samples meet the KKT conditions. If there is none, it will add these samples to the support vectors to form a new training set. Meanwhile, in each incremental learning step, select samples from the CSV set according to their weights, and combine them with the samples that violate the KKT conditions in the new sample set and the original support vectors to form a new training set. After incremental learning, update the CSV set and the weights of the samples in it. The whole process of incremental learning is described in Algorithm 5-2.

---

#### ***Algorithm 5- 2: CSV-ISVM***

---

**Input:** Sample set  $X_0$ , New incremental sample set  $X_I$ ,

**Output:** Updated CSV set and the classifier SVM

- 1: Get classifier  $SVM_0$  and support vector set  $SV_0$  by training on  $X_0$ ;
- 2: Calculate class centres  $X_{0+}$  and  $X_{0-}$  using Equation 5-2;
- 3: Compute the distance between samples and class centres using Equation 5-4;
- 4: Construct CSV set  $N_0$  by selecting samples that satisfy Equation 5-8;
- 5: Get weights  $\Theta_i$  and thresholds  $T_i$  using Equation 5-9 and Equation 5-10 respectively;
- 6: Take new sample set  $X_I$ ;
- 7: **For** every sample  $x_i$  in  $X_I$
- 8:     **If**  $x_i$  violates the KKT conditions of  $SVM_0$ ;
- 9:         Add  $x_i$  to  $X_{Is}$ ;
- 10:     **Else**
- 11:         Add  $x_i$  to  $X_{Id}$ ;
- 12:     **EndIf**
- 13: **EndFor**
- 14: **If**  $X_{Is}$  is not null //Select samples according to their thresholds//
- 15:     **For** every sample  $x_i$  in  $X_{Is}$
- 16:         **If**  $T \geq$  given threshold value
- 17:             Add sample  $x_i$  to the set  $NSV_0$ ;
- 18:         **EndIf**
- 19:     **EndFor**
- 20:     Set  $N_I = SV_0 \cup NSV_0 \cup X_{Is}$ ;
- 21:     Select samples in  $N_I$  to construct  $N_I'$ ;
- 22:     Obtain classifier  $SVM_I$  and  $SV_I$  by training on  $X_0$ ;
- 23:     Set  $SVM_I$  as the output;
- 24:     Update the CSV set and also weights  $\Theta_i$  and thresholds  $T_i$ ;
- 25: **Else**

26: Set  $SVM_0$  as the output;

27: **EndIf**

---

***Complexity of the Algorithm:***

The CSV-ISVM algorithm is a modified version of incremental SVM algorithm, which is designed in such a way that it does not affect the overall complexity of the incremental SVM classification algorithm. So, the complexity still is given by  $O(n\_samples^2 * n\_features)$ .

### **5.4 Experimental Results and Analysis**

For the purpose of simulation and experiment works, the following platforms are used in this piece of research: -

**Table 5-1: Experimental platform for incremental SVM classification with CSV-ISVM**

Operating System :	Linux (Ubuntu)
SVM Classification Tool:	LibSVM
Algorithm coding and testing platform:	LibSVM, Matlab
Simulation and Performance Analysis Tool:	Matlab, LibSVM
Data Analysis and Graphical Tool:	LibreOffice
Data Source:	Kyoto 2006+ Data

#### **5.4.1 Experimental Data**

In this work, Kyoto 2006+ dataset is used as data sets for the experiments. After examination of the data - specially the attack ratio - and preliminary experiments, the datasets of the days 2007 Nov. 1, 2 and 3 are chosen for the final experiments. Please refer to the Chapter 3 for description and the samples of the experimental data.



### 5.4.2 Description and Samples of Experimental Data

The detailed description of data used in the simulation and experiment works in this piece of research work is mentioned in the Section 3.

In order to provide the rough idea about the experimental data, a part of a snapshot of one day's experimental data is provided below. For more data, please refer to the Appendix I: Experimental Data Samples.

Duration	Service	Source bytes	Destination bytes	Count	Source error rate	Source rate	Destination error rate	Destination rate	Exploit count	Exploit type count	Exploit frame rate per min	Exploit error rate	Exploit type error rate	Exploit	ID# detection	Malware detection	OS/patch detection	Label	
754.15	5	69677	135318	0	0	0	0	0	0	0	0	0	0	RSTO50	203027 (600)	0	0	-1	5
910.55	8	29358	39	0	0	0	0	0	0	0	0	0	0	RSTRH	0	0	0	1	5
0.00	8	0	0	0	0	0	0	0	0	0	0	0	0	RSTO	0	0	0	1	5

{See Appendix II for more...}

### 5.4.3 Data Pre-processing

A data pre-processing techniques like sampling and filtering are applied for easy and smooth operation of the experiments. As suggested by Chitrakar and Huang [165], data samples are extracted such that each set contains 1% of attacks including unknown attacks too. Attributes of the datasets are converted to appropriate types and normalized according to the need of the SVM kernel. Samples with both known and unknown attacks are treated as a single attack class and labeled as -1. Thus, the entire selected data can be categorized into {-1, 1} where 1 represents normal class. This makes the application of SVM very simple and almost any type of SVM classification may be implemented.

### 5.4.4 The Experimental Detail

The experimental data thus obtained from Kyoto 2006+ dataset is randomly divided into two non-overlapping subsets - training and test set. Then, extract 10 (ten) separate data sets from the training subset as the incremental training sets, each set containing 1000 samples having both normal and abnormal data. Similarly, extract 10 data sets from the test data set too.

Using each data sample, experiments are carried out for Simple-ISVM, KKT-ISVM, RS-ISVM and CSV-ISVM one by one. In all four methods, the RBF kernel is used because SVM produces better performance with RBF [166].

The number of true positive, true negative, false positive and false negative values of all the methods are recorded and used for performance evaluation.

#### 5.4.5 Performance Evaluation and Analysis

The performance evaluation of the experiment is carried out in terms of accuracy ( $A$ ), detection rate ( $DR$ ) and false alarm rate ( $FAR$ ) by using the Equation 3-4, Equation 3-5 and Equation 3-6 respectively.

**Table 5-2: Comparison of DR and FAR**

Increment Count	Simple-ISVM		KKT-ISVM		RS-ISVM		CSV-ISVM	
	DR(%)	FAR(%)	DR(%)	FAR(%)	DR(%)	FAR(%)	DR(%)	FAR(%)
1	79.646	4.535	79.646	4.535	79.646	4.535	80.646	4.41
2	86.826	7.245	77.451	4.449	84.113	4.099	84.724	4.025
3	81.435	6.169	82.608	5.494	85.069	4.059	85.576	3.915
4	76.871	6.076	81.829	5.501	85.917	3.948	86.082	3.802
5	77.513	6.393	81.941	4.518	86.407	3.805	86.814	3.54
6	77.474	6.403	80.69	5.153	87.352	3.476	87.238	3.33
7	75.618	6.552	86.518	4.327	87.54	3.372	88.082	3.027
8	71.882	3.216	84.221	4.141	87.968	3.225	88.095	3.012
9	80.606	6.762	87.419	4.162	89.144	3.124	89.643	2.52
10	79.576	5.699	87.706	4.16	89.817	3.015	90.147	2.314

Evaluation of CSV-ISVM is done by comparing it with Simple-ISVM, the KKT-ISVM and RS-ISVM, on detection rate and false alarm rate. Evaluation has also been done for the training and testing time of all four methods.

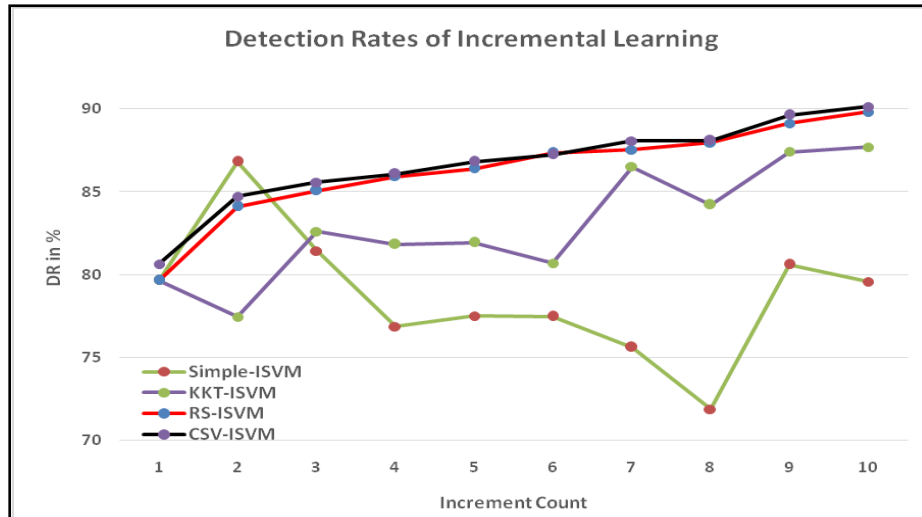


Figure 5-6: Comparison of DR

Table 5-2 shows that the values of DRs and FARs in case of Simple-ISVM keep increasing and decreasing in the subsequent Increments, e.g. the DR has increased to 86.826% in the 2<sup>nd</sup> increment but decreased to 81.435% in the 3<sup>rd</sup> Increment; and similar in case of FAR too. This implies that they depend heavily upon the new data sets of each increment.

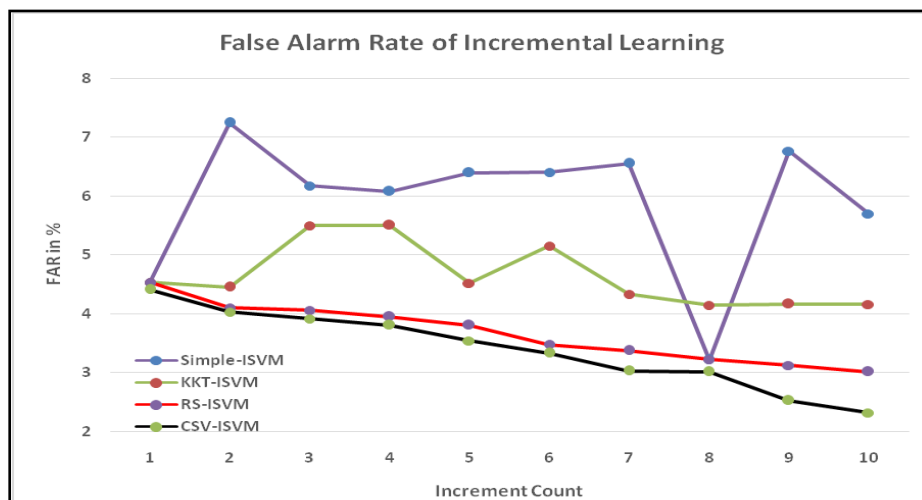


Figure 5-7: Comparison of FAR

KKT-ISVM produces higher DR values (and lower FAR values) in the last couple of increments compared to the initial few increments, but produces increasing and decreasing values over middle increments, e.g. in Increments 5, 6 and 7, DRs are 81.941%, 80.69% and 86.518% respectively and FARs are 4.518%, 5.153% and 4.327% respectively. It is seen from Figure 5-6 and Figure 5-7 that KKT-ISVM compared to Simple-ISVM has a better growing trend of DR and a better falling trend of FAR, in general, but still has inconsistencies in some increments.

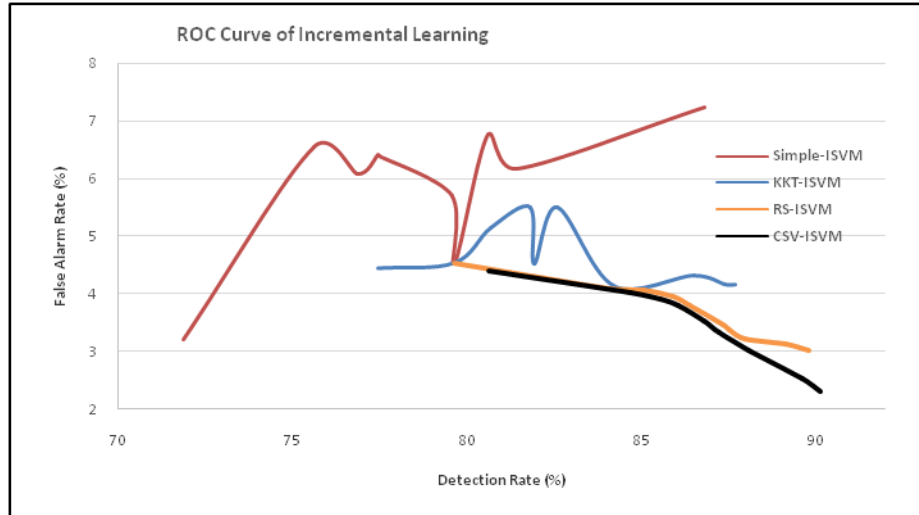
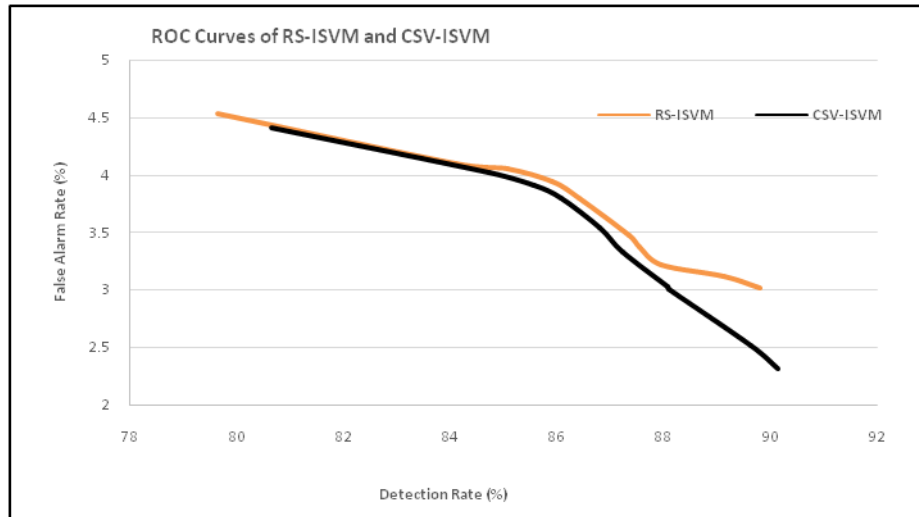


Figure 5-8: Comparison of ROC Curves

Again from Table 5-2, it is seen that both RS-ISVM and CSV-ISVM produce continuously increasing DRs and decreasing FARs and also have better rates compared to Simple-ISVM and KKT-ISVM. For example, RS-ISVM has 79.646% and 89.817% of lowest and highest values of DR respectively; and 4.535% and 3.015% of maximum and minimum FAR respectively. CSV-ISVM shows even better values yielding 80.646% of DR and 4.41% of FAR in the first Increment; and 90.147% of DR and 2.314% of FAR in the last increment.

As seen from Figure 5-6 and Figure 5-7, both RS-ISVM and CSV-ISVM show growing DR lines and falling FAR lines as increments go on. Moreover, comparing RS-ISVM and CSV-ISVM methods, DR and FAR of CSV-ISVM have seem to have better rates and more perfect decreasing trend than RS-ISVM. This clearly indicates that the idea of retaining support vectors as prior-knowledge for incremental learning yields better results.



**Figure 5-9: RS-ISVM and CSV-ISVM Compared**

The ROC curves of DR vs. FAR are shown in Figure 5-8, where we can see smoother and more declining ROC curves produced by CSV-ISVM compared to the other ISVM methods. Needless to say, the ROC curve of the Simple-ISVM is not acceptable whereas that of KKT-ISVM does not maintain the ratio DR/FAR all the time. The figure also illustrates that the CSV-ISVM yields the highest possible DR while maintaining the lowest FAR.

**Table 5-3: Comparison of training and testing time**

Increment Count	Simple-ISVM		KKT-ISVM		RS-ISVM		CSV-ISVM	
	Train(s)	Test(s)	Train(s)	Test(s)	Train(s)	Test(s)	Train(s)	Test(s)
1	1.843	9.4	1.842	9.48	1.833	8.58	1.823	7.76
2	6.407	24.688	12.578	27.005	4.935	17.5255	3.463	10.363
3	9.375	34.27	26.667	39.973	7.3175	24.8565	5.26	15.443
4	15.155	39.87	50.733	49.427	12.4085	29.701	9.662	19.532
5	31.433	44.427	63.833	57.162	21.3675	33.581	11.302	22.735
6	30.078	58.412	81.15	64.245	21.8355	42.1495	13.593	25.887

7	37.787	70.157	119.4	74.272	26.0675	49.1935	14.348	28.23
8	55.157	79.948	138.667	83.36	36.316	55.7815	17.475	31.615
9	46.9	73.022	160.45	88.412	34.96	54.2845	23.02	35.547
10	67.023	79.978	168.75	99.912	47.14	60.0375	27.257	40.097

When compared specially with RS-ISVM, the CSV-ISVM seems to have maintained producing lesser FAR rates when DR exceeds about 85% throughout the subsequent increments as illustrated by further down-warded curve of CSV-ISVM as shown in Figure 5-9. This indicates that even with the new data sample sets which keep coming in each new increments, the CSV-ISVM still produces increasing DR and decreasing FAR because the Half-Partition strategy prevents the learning method from retaining potentially unwanted data points.

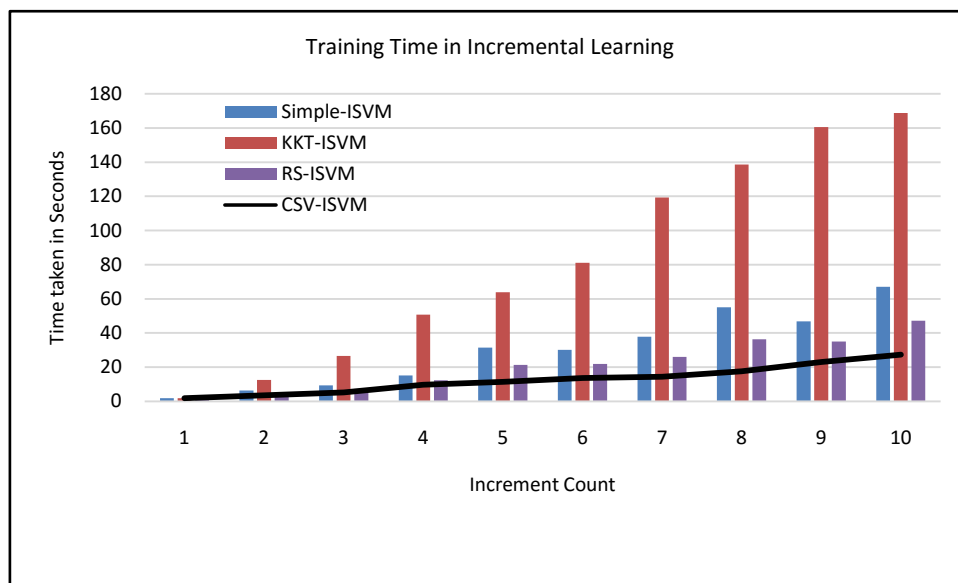


Figure 5-10: Comparison of Training Time

Table 5-3 compares time taken by the four methods in training and testing data samples in each increment. The amount of time taken for training data sets in the initial increment is almost same for all methods i.e. 1.8xx seconds, whereas that for testing has been reduced in CSV-ISVM to 7.76 seconds from 9.4 seconds in Simple-ISVM by saving 1.64 seconds.

Similarly, in the last increment, 39.766 seconds of training time and 39.881 seconds of testing time have been saved by CSV-ISVM. Unlike other methods, KKT-ISVM, though has better DRs and FARs compared to Simple-ISVM, takes longer learning time, illustrated also by Figure 5-10 and Figure 5-11, which is obvious because it has to do cross-judging and more training. It is clear that the time taken by CSV-ISVM for both training and testing are reduced at least by half compared to Simple-ISVM and KKT-ISVM. Moreover, thanks to the Half-partition strategy, time taken both in training and testing are also reduced remarkably compared to Concentric Circle method of RS-ISVM

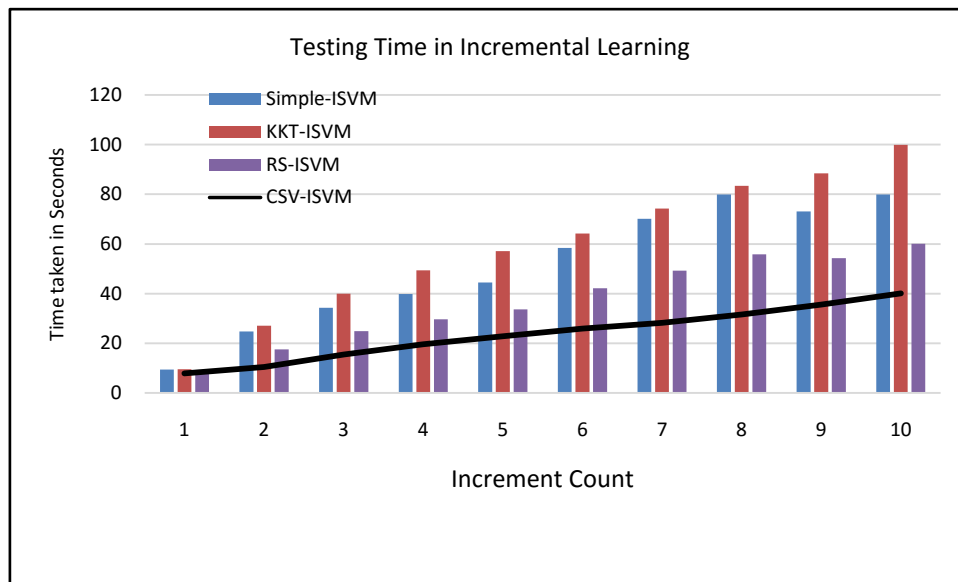


Figure 5-11: Comparison of Testing Time

The amounts of time taken by all methods are obviously increasing as increments go on, but RS-ISVM and CSV-ISVM takes less time to learn in each increment compared to other two methods.

Table 5-4: Comparison of time difference between two subsequent increments

Increment Count	Simple-ISVM		KKT-ISVM		RS-ISVM		CSV-ISVM	
	Train(s)	Test(s)	Train(s)	Test(s)	Train(s)	Test(s)	Train(s)	Test(s)
1	{ This is the very first increment }							
2	4.564	15.288	10.736	17.525	3.102	8.9455	1.64	2.603

3	2.968	9.582	14.089	12.968	2.3825	7.331	1.797	5.08
4	5.78	5.6	24.066	9.454	5.091	4.8445	4.402	4.089
5	16.278	4.557	13.1	7.735	8.959	3.88	1.64	3.203
6	-1.355	13.985	17.317	7.083	0.468	8.5685	2.291	3.152
7	7.709	11.745	38.25	10.027	4.232	7.044	0.755	2.343
8	17.37	9.791	19.267	9.088	10.2485	6.588	3.127	3.385
9	-8.257	-6.926	21.783	5.052	-1.356	-1.497	5.545	3.932
10	20.123	6.956	8.3	11.5	12.18	5.753	4.237	4.55
Average	7.242222	7.842	18.54533	10.048	5.034111	5.7175	2.826	3.593

In other words, as shown in Table 5-4, the time difference between two subsequent increments for both training and testing data sets in one method is remarkably different from the other. The average training and testing time difference in case of Simple-ISVM are calculated to be 7.242 seconds and 7.842 seconds respectively, whereas those in case of CSV-ISVM are 2.826 seconds and 3.593 seconds respectively. This time difference between two increments may be regarded as the time taken to handle the new data set added in the later increment. This means that CSV-ISVM also handles the new data sets most efficiently of all other ISVM methods.

All afore-mentioned experimental results and in-depth analyses show clearly that the proposed CSV-ISVM has surpassed the Simple-ISVM, the KKT-ISVM and the RS-ISVM too in terms of DR, FAR as well as in terms of training and testing time.

### **5.5 Chapter Summary**

An improved approach to Incremental Support Vector Machine, called CSV-ISVM, has been proposed and applied to the incremental learning to SVM based network intrusion detection. The approach has contributed in two ways. Firstly, it has suggested modifications to the previously proposed Concentric Circle method.



Secondly, it has proposed a more efficient Half-Partition strategy and also designed an algorithm to implement [167]. The experiment works and the analyses have shown that the proposed method has better detection rate and false alarm rate as well as acceptable amount of learning time and, therefore, can be used for network intrusion detection in real-time.

The Half-Partition strategy actually discards (almost half) the data points lying farther than the class centre from the hyperplane. And, such strategy cannot be implemented in other than binary classification. Modifying the strategy in order to make it work with multi-class classification might be considered as a future work of this research.

## Chapter 6. Concluding Remarks

### 6.1 Conclusion

This thesis research work has presented a Real-Time Hybrid Intrusion Detection Architecture as a framework for the entire research work. This work has also proposed two different intrusion detection methods viz. (1) Clustering-Outlier-Detection with SVM classification and (2) Incremental SVM classification with Half-partition method, which could be used either exclusively or in combination depending upon how deep and thorough the intrusion detection should be. The experiments and analyses with the very first method has shown that the combined approach of Clustering and Outlier detection followed by SVM classification not only increases the detection performance, accuracy, detection rate by reducing false alarm rate but also guarantees the predictability of the detection parameters, This indicates that this approach can be implemented in real-time intrusion detection.

The second method i.e. Incremental SVM with Half-partition method is proven to be better, in addition to all detection parameters mentioned above, in terms of execution time also. By proposing the Half-partition method and thereby implementing it in the CSV-ISVM algorithm, the normal time taken for intrusion detection has been reduced almost to half. Several experiments have been carried out using one method or both methods at a time using Kyoto+ 2006 data sets, also by comparing them with other similar methods. The analysis results of the experiments show that both the methods have outperformed their competitive methods in terms of performance, accuracy, detection rate, false alarm rate as well as in terms of real-time parameters like time and predictability.

In accomplishing the research work, two algorithms namely – (1) Clustering-Outlier Detection and (2) Candidate Support Vector – Incremental SVM have also been proposed and implemented. In due course, improvements to unified Outlier detection and Reserved Set strategy of selection support vectors in incremental SVM classification have also been proposed.

## **6.2 Future Work**

This thesis work, however, can be further improved in a number of ways. Firstly, the time consuming clustering method may be considered improving by keeping its outlier detection in place. Next improvement could be using multiple SVM kernels. Additionally, the Half-partition method may also be extended to multiple class SVM classification.

## References

---

- [1] Sylvester Ngoma, Vulnerability of IT Infrastructures: Internal and External Threats, *Advances in Intelligent Systems and Computing, Volume 216*, 2012.
- [2] Sonal R. Pampattiwar, Prof. Anil Z. Chhangani, Hybrid Intrusion Detection System Using Snort, *Sonal-International Journal of Computer Science information and Engg. Technologies, ISSN 2277-4408*, 2014.
- [3] Przemyslaw Kazienko & Piotr Dorosz, Intrusion Detection Systems (IDS) Part I - (network intrusions; attack symptoms; IDS tasks; and IDS architecture), *WindowSecurity.com*, 7 March 2015.
- [4] Sunil Gupta, Logging and Monitoring to detect Network Intrusions and Compliance Violations in the Environment, *SANS Institute Reading Room*, 2012.
- [5] Nunnally, Troy, et al. 3DSVAT: A 3D Stereoscopic Vulnerability Assessment Tool for network security. *LCN*. 2012.
- [6] Singh, B., Network Security and Management, *PHI Learning, ISBN 9788120344976*, 2011.
- [7] Intrusion detection vs. firewall, <http://www.dhk.com/services/managed-security/intrusion-detection-and-prevention/288-2/> on 12/25/2014.
- [8] Intrusion Detection For Dummies, IT Security, <http://www.itsecurity.com/features/intrusion-detection-for-dummies-072906/>, September 21, 2014
- [9] Phil Bandy, Michael Money & Karen Worstell, Intrusion Detection FAQ: Why is intrusion detection required in today's computing environment?, [http://www.sans.org/security-resources/idfaq/id\\_required.php](http://www.sans.org/security-resources/idfaq/id_required.php), 27 March 2014.
- [10] Rajni Ranjan Singh, Deepak Singh Tomar, Network Forensics: Detection and Analysis of Stealth Port Scanning Attack, *International Journal of Computer Networks and Communications Security, Vol. 3, no. 2*, February 2015, 33–42.
- [11] More, Sumit, et al. A knowledge-based approach to intrusion detection modeling, *IEEE Symposium on Security and Privacy Workshops (SPW)*, 2012.
- [12] Phurivit Sangkatsanee, Naruemon Wattanapongsakorna, Chalernpol Charnsripinyo, Practical real-time intrusion detection using machine learning approaches, *Computer Communications 34*, pp. 2227–2235, 2011.

- 
- [13] Santhosh.S, Radha.R, A Novel Security Model for Preventing Passive and Active Attacks In WSNS, *International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 10*, October 2013.
- [14] Swamidoss Sathiakumar, Lalit Kumar Awasthi, Roberts Masillamani, S S Sridhar, *Proceedings of International Conference on Internet Computing and Information Communications: ICICIC Global*, 2012.
- [15] The Frame of Deception: Wireless Man-in-the-Middle Attacks and Rogue Access Points Deployment, <http://books.msspace.net/mirrorbooks/wireless/0321202171/ch08lev1sec7.html> on 12/25/2014
- [16] Chris Sanders, Understanding Man-In-The-Middle Attacks – Part2: DNS Spoofing, *WindowSecurity.com*, 2010.
- [17] Gordon Lyon, The Art of Port Scanning, [http://nmap.org/nmap\\_doc.html](http://nmap.org/nmap_doc.html), 1/4/2015.
- [18] Matsumoto, T.; Hata, M.; Tanabe, M.; Yoshioka, K.; Oishi, K., A Method of Preventing Unauthorized Data Transmission in Controller Area Network, *Vehicular Technology Conference (VTC Spring), 2012 IEEE 75th*, vol., no., pp.1,5, 6-9 May 2012.
- [19] Schellekens, C. H., Sandro Etalle, and Damiano Bolzoni. Alert Classification of Web Application Attacks Using Bayesian Networks to Classify Alerts from Anomaly Based Intrusion Detection Systems, *A Master's Thesis, Technische Universiteit Eindhoven*, 2013.
- [20] Phil Bandy, Michael Money & Karen Worstell, Intrusion Detection FAQ: Why is intrusion detection required in today's computing environment?, [http://www.sans.org/security-resources/idfaq/id\\_required.php](http://www.sans.org/security-resources/idfaq/id_required.php), 27 March 2014.
- [21] Maryam Hajizadeh and Marzieh Ahmadzadeh, A Novel Hybrid Intrusion Detection Framework Based On Data Mining Techniques, *International Journal of Scientific Research and Management Studies (IJSRMS), Volume 1 Issue 5*, pg: 183-191, Aug 2014.
- [22] Adinehnia, Reza, et al. Effective mining on large databases for intrusion detection. *International Symposium on Biometrics and Security Technologies (ISBAST)*, 2014.
- [23] Bapna, Deependra. Intrusion detection system for wireless sensor network. *Diss.* 2014.
- [24] Xiaoyun, L. I. U., et al. Data Mining Intrusion Detection in Vehicular Ad Hoc Network. *IEICE Transactions on Information and Systems* 97.7: 1719-1726. 2014.

- 
- [25] Vineet Richhariya, Nupur Sharma, Optimized Intrusion Detection by CACC Discretization Via Naïve Bayes and K-Means Clustering, *International Journal of Computer Science and Network Security*, vol.14 No.1, January 2014.
- [26] Snehal S. Somwanshi, Soniya N.Madavi, Review Paper on Intrusion Detection Using Data Mining Technique, *International Journal of Emerging Technology and Advanced Engineering*, Volume 4, Issue 9, September 2014.
- [27] Ujwala Ravale, Nilesh Marathe, Puja Padiya, Attribute Reduction based Hybrid Anomaly Intrusion Detection using K-Means and SVM Classifier, *International Journal of Computer Applications (0975-8887) Volume 82 - No 15*, November 2013.
- [28] Yousef Emami, Marzieh Ahmadzadeh, Mohammad Salehi, Sajad Homayoun, Efficient Intrusion Detection using Weighted K-means Clustering and Naïve Bayes Classification, *Journal of Emerging Trends in Computing and Information Sciences*, Vol. 5, No. 8, August 2014.
- [29] Murad Abdo Rassam, Mohd. Aizaini Maarof, and Anazida Zainal, Intrusion Detection System Using Unsupervised Immune Network Clustering with Reduced Features, *Int. J. Advance. Soft Comput. Appl.*, Vol. 2, No. 3, November 2010.
- [30] Ali Asghar Yarifard and Mohammad Hossein Yaghmaee, The Monitoring System Based on Traffic Classification, *World Applied Sciences Journal 5 (2): 150-160*, 2008.
- [31] Shyara Taruna R., Mrs. Saroj Hiranwal, Enhanced Naïve Bayes Algorithm for Intrusion Detection in Data Mining, *International Journal of Computer Science and Information Technologies*, Vol. 4(6), 960-962, 2013.
- [32] R. Luigi, T.E. Anderson, and N. McKeown, Traffic Classification using Clustering Algorithms. *ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Pisa, Italy, ACM Press, pp. 281-286, Sep. 11-15, 2011.
- [33] Choi, Jai Joon, Grubel, Brian Christopher, Reid, Dion Stephen David, System And Method For Discovering Optimal Network Attack Paths, *United States Patent Application Publication*, Feb 26, 2015
- [34] T. Velmurugan and T. Santhanam, Computational Complexity between k-Means and k-Medoids Clustering Algorithms for Normal and Uniform Distributions of Data Points, *Journal of Computer Science*, 6 (3): 363-368, 2010.

- 
- [35] Velmurugan, Dr T. Efficiency of k-Means and K-Medoids Algorithms for Clustering Arbitrary Data Points. *Int. J. Computer Technology & Applications* 3.5: 1758-1764. 2012.
- [36] Chen, CL Philip, and Chun-Yang Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences* 275: 314-347. 2014.
- [37] Chen, Hsinchun, Roger HL Chiang, and Veda C. Storey. Business Intelligence and Analytics: From Big Data to Big Impact. *MIS quarterly* 36.4 : 1165-1188. 2012.
- [38] Velmurugan, T. Performance based analysis between k-Means and Fuzzy C-Means clustering algorithms for connection oriented telecommunication data. *Applied Soft Computing* 19 : 134-146, 2014.
- [39] Shalini S Singh, N C Chauhan and A.D.Patel, K-means v/s K-medoids: A Comparative Study, *National Conference on Recent Trends in Engineering & Technology*, 13-14 May, 2011
- [40] Phurivit Sangkatsanee, Naruemon Wattanapongsakorna, Chalernpol Charnsripinyo, Practical real-time intrusion detection using machine learning approaches, *Computer Communications* 34, pp. 2227–2235, 2011.
- [41] El Agha, Mohammed, and Wesam M. Ashour. Efficient and fast initialization algorithm for k-means clustering. *International Journal of Intelligent Systems and Applications (IJISA)* 4.1: 21, 2012.
- [42] Agrawal, Akshay S., and Sachin Bojewar. Comparative Study Of Various Clustering Techniques. *International Journal of Computer Science and Mobile Computing, Vol.3 Issue.10*, , pg. 497-504, October- 2014.
- [43] Khanmohammadi, M., et al. Clustering Decision Making Units in Data Envelopment Analysis by the Common Set of Weights. *World Applied Sciences Journal* 28.9 : 1262-1274. 2013.
- [44] Chitta, Radha, et al. Approximate kernel k-means: Solution to large scale kernel clustering. *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM*, 2011.
- [45] Niknam, Taher, et al. An efficient hybrid algorithm based on modified imperialist competitive algorithm and K-means for data clustering. *Engineering Applications of Artificial Intelligence* 24.2 : 306-317. 2011.

- 
- [46] Satish, Belsare, and Patil Sunil. Study and Evaluation of user's behavior in e-commerce Using Data Mining. *Research Journal of Recent Sciences*, ISSN 2277 : 2502. 2012.
- [47] Ghosh, Soumi, and Sanjay Kumar Dubey. Comparative analysis of k-means and fuzzy c-means algorithms. *International Journal of Advanced Computer Science and Applications (IJACSA) 4.4*, 2013.
- [48] Ji, Wen Tian, Qing Ju Guo, and Sheng Zhong. The Improvement of K-Medoids Clustering Algorithm under Semantic Web. *Applied Mechanics and Materials 380 : 1286-1289*, 2013.
- [49] Noor Kamal Kaur<sup>1</sup>, Usvir Kaur, Dr.Dheerendra Singh, K-Medoid Clustering Algorithm- A Review, *International Journal of Computer Application and Technology (IJCAT) Volume 1 Issue 1, ISSN: 2349-1841*, April 2014.
- [50] Petitjean, François, et al. Dynamic Time Warping averaging of time series allows faster and more accurate classification. *IEEE International Conference on Data Mining*. 2014.
- [51] Li, Jinhua, Hengxiang Tian, and Dandan Xing. Clustering user session data for web applications test. *Journal of Computational Information Systems 7.9 : 3174-3181*, 2011.
- [52] Shalini S, Singh Vallabh, N C Chauhan, K-means v/s K-medoids: A Comparative Study, *National Conference on Recent Trends in Engineering & Technology*, 13-14 May, 2011.
- [53] Xie, Juanying, et al. An efficient global K-means clustering algorithm. *Journal of computers 6.2 : 271-279*. 2011.
- [54] Chawla, Sanjay, and Aristides Gionis. k-means-: A Unified Approach to Clustering and Outlier Detection. *SDM*, 2013.
- [55] Rajendra Pamula, Jatindra Kumar Deka, Sukumar Nandi, An Outlier Detection Method based on Clustering, *Second International Conference on Emerging Applications of Information Technology*, 2011.
- [56] Xiang, C., M.Y. Chong and H. L. Zhu, Design of Multiple-Level Tree Classifiers for Intrusion Detection System. *IEEE Conference on Cybernetics and Intelligent Systems (CCIS 2004), Singapore, pp: 873-878*, 2004.
- [57] Peddabachigiri, S., A. Abraham, C. Grosan and J. Thomas, Modeling Intrusion Detection System using Hybrid Intelligent Systems, *J. Network Comput. Appl.*, 30: 114-132, 2007.



- 
- [58] Sundus Juma, Zaiton Muda, Warusia Yassin, Reducing False Alarm Using Hybrid Intrusion Detection Based On X-Means Clustering And Random Forest Classification, *Journal of Theoretical and Applied Information Technology*. Vol. 68 No.2. 20th October 2014.
- [59] [http://en.wikibooks.org/wiki/Data\\_Mining\\_Algorithms\\_In\\_R/Classification/SVM](http://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Classification/SVM), Sep 23, 2014.
- [60] Data Mining Algorithms In R/Classification/SVM. Wikibooks, The Free Textbook Project. [http://en.wikibooks.org/w/index.php?title=Data\\_Mining\\_Algorithms\\_In\\_R/Classification/SVM&oldid=2496635](http://en.wikibooks.org/w/index.php?title=Data_Mining_Algorithms_In_R/Classification/SVM&oldid=2496635). On December 23, 2014.
- [61] Chih-Wei Hsu, Chih-Chung Chang and Chih-Jen Lin. A Practical Guide to Support Vector Classification, April 15, 2010.
- [62] Chitrakar Roshan, Huang Chuanhe. Anomaly detection using support vector machine classification with k-medoids clustering. *In Proceedings of the third Asian Himalayan international conference on internet (AH-ICI); 2012.*
- [63] Yuping Li, Weidong Li, Guoqiang Wu, An Intrusion Detection Approach Using SVM and Multiple Kernel Method, *International Journal of Advancements in Computing Technology(IJACT) Volume4, Number1*, January 2012..
- [64] T. Velmurugan and T. Santhanam, Computational Complexity between k-Means and k-Medoids Clustering Algorithms for Normal and Uniform Distributions of Data Points, *Journal of Computer Science*, 6 (3): 363-368, 2010.
- [65] Z. Muda, W. Yassin, M.N. Sulaiman, N.I. Udzir , Intrusion Detection based on k-Means Clustering and Naïve Bayes Classification , *7th International Conference on IT in Asia (CITA)*, 2011.
- [66] Fabrice Colas and Pavel Brazdil, Comparison of SVM and Some Older Classification Algorithms in Text Classification Tasks, *Proceedings - IFIP AI Proceeding*, pp.169-178, 2006.
- [67] Zhuang Wang, Koby Crammer and Slobodan Vucetic, Breaking the Curse of Kernelization: Budgeted Stochastic Gradient Descent for Large-Scale SVM Training, *Journal of Machine Learning Research* 13 3103-3131, 2012.

- 
- [68] Jin Huang, Jingjing Lu, Charles X. Ling, Comparing Naive Bayes, Decision Trees, and SVM with AUC and Accuracy, *The Third International Conference on Data Mining*, 2003.
- [69] Roshan Chitrakar and Huang Chuanhe, Anomaly based Intrusion Detection using Hybrid Learning Approach of combining k-Medoids Clustering and Naïve Bayes Classification, *The 8th International Conference on Wireless Communication, Networking and Mobile Computing, Shanghai, China, 2012*.
- [70] Wang XD, Zheng CY, Wu CM, Zhang HD. New algorithm for SVM-based incremental learning. *Comput Appl*;26(10):2440-3, 2006.
- [71] Kobayashi Takumi, Otsu Nobuyuki. Efficient reduction of support vectors in kernel-based methods. *In: Proceedings of IEEE international conference on image processing. ICIP*; . pp. 2077-80. 2009.
- [72] Habib Tarek, Inglada Jordi, Mercier Gr egoire, Chanussot Jocelyn. Support vector reduction in SVM algorithm for abrupt change detection in remote sensing. *Proc IEEE Geosci Remote Sens Lett*;6(3). 2009.
- [73] Pradhan, Biswajeet. A comparative study on the predictive ability of the decision tree, support vector machine and neuro-fuzzy models in landslide susceptibility mapping using GIS. *Computers & Geosciences 51*: 350-365. 2013.
- [74] Khan, Naimul Mefraz, et al. SN-SVM: a sparse nonparametric support vector machine classifier. *Signal, Image and Video Processing 8.8 ()*: 1625-1637, 2014.
- [75] Ji-yong, Shi, et al. Rapid detecting total acid content and classifying different types of vinegar based on near infrared spectroscopy and least-squares support vector machine. *Food chemistry 138.1* : 192-199, 2013.
- [76] Bernabe, Sergio, et al. Spectral–spatial classification of multispectral images using kernel feature space representation. *Geoscience and Remote Sensing Letters, IEEE 11.1* : 288-292, 2014.
- [77] Rajendra Pamula, Jatindra Kumar Deka, Sukumar Nandi, An Outlier Detection Method based on Clustering, *Second International Conference on Emerging Applications of Information Technology*, 2013

- 
- [78] Muamer N. Mohammad, Norrozila Sulaiman, Emad T. Khalaf..A Novel Local Network Intrusion Detection System based on Support Vector Machine. In *Journal of Computer Science* 7 (10): 1560-1564. 2011.
- [79] Le, Giang Hoang Nguyen. Machine Learning with Informative Samples for Large and Imbalanced Data Sets. *PhD Thesis, School of Electrical, Computer and Telecommunication Engineering, School of Wollongong*. 2011.
- [80] Siddharth Gopal, Large-scale Structured Learning, *A PhD thesis, School of Computer Science, Carnegie Mellon University*, 2014.
- [81] Mangai, Utthara Gosa, et al. A survey of decision fusion and feature fusion strategies for pattern classification. *IETE Technical review* 27.4 : 293-307, 2010.
- [82] Polikar, Robi. Ensemble learning. *Ensemble machine learning. Springer US*, 1-34. 2012.
- [83] Waske, Björn, et al. Sensitivity of support vector machines to random feature selection in classification of hyperspectral data. *Geoscience and Remote Sensing, IEEE Transactions on* 48.7 : 2880-2889, 2010.
- [84] Patel, Anita J., and Joy S. Patel. Ensemble systems and incremental learning. *International Conference on Intelligent Systems and Signal Processing (ISSP), IEEE*, 2013.
- [85] Hoens, T. Ryan, Robi Polikar, and Nitesh V. Chawla. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence* 1.1: 89-101. 2012.
- [86] Hongle Du, Shaohua Teng , Mei Yang , Qingfang Zhu. Intrusion detection System Based on improved SVM Incremental Learning. *In Proceedings of International Conference on Artificial Intelligence and Computational Intelligence*. 2009.
- [87] Kamaledin Ghiasi-Shirazi, Reza Safabakhsh and Mostafa Shamsi, Learning Translation Invariant Kernels for Classification, *Journal of Machine Learning Research* 11: 1353-139. 2010.
- [88] Makili L., J. Vega, S. Dormido-Canto. Incremental Support Vector Machines for Fast Reliable Image Recognition, *In Fusion Engineering and Design*. 2012
- [89] Karasuyama, M., Takeuchi, Ichiro. Multiple Incremental Decremental Learning of Support Vector Machines, *In IEEE Transactions on Neural Networks Vol. 21, No. 7*. 2010.

- 
- [90] Yang Yi, Jiansheng Wu, Wei Xu.. Incremental SVM based on Reserved Set for Network Intrusion Detection, *In Expert Systems with Applications* 7698–7707. 2011.
- [91] Kaltiokallio, Ossi, and Maurizio Bocca. Real-time intrusion detection and tracking in indoor environment through distributed RSSI processing. *17th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), IEEE. Vol. 1.* 2011
- [92] Liu Yangguang, He Qinming, Chen Qi. Incremental batch learning with support vector machines. *In: Proceedings of the 5th world congress on intelligent control and automation;*. pp. 1857-61. 2004
- [93] Wang Wen-Jian. A redundant incremental learning algorithm for SVM. *In: Proceedings of the seventh international conference on machine learning and cybernetics;*. pp. 734-8. 2008.
- [94] Tao Liang, Fast incremental SVM learning algorithm based on active set iteration. *China Journal Syst Simul*;18(11):3305-8. 2006.
- [95] Yao Yuan, Feng Lin, Jin Bo, Chen Feng. An incremental learning approach with support vector machine for network data stream classification problem. *Proc Inf Technol J*;11:200-8. 2012
- [96] Sun Na, Guo Yanfeng. A modified incremental learning approach for data stream classification. *In: Sixth international conference on internet computing for science and engineering*; 2012
- [97] Yang Yi, Jiansheng Wu, Wei Xu.. Incremental SVM based on Reserved Set for Network Intrusion Detection, *In Expert Systems with Applications* 7698–7707. 2011.
- [98] Lin, Xiaojun, Chan, Patrick P.K., Causative attack to incremental support vector machine, *International Conference on Machine Learning and Cybernetics*, v 1, p 137-142, January 13, 2015.
- [99] Nekkaa, Messaouda, Boughaci, Dalila, A memetic algorithm with support vector machine for feature selection and classification, *Memetic Computing*, v 7, n 1, p 59-73, 2015.
- [100] Yin, Yingjie; Xu, De; Wang, Xingang; Bai, Mingran, Online State-Based Structured SVM Combined With Incremental PCA for Robust Visual Tracking, *IEEE Transactions on Cybernetics*, February 13, 2015.

- 
- [101] Wang, Xuhui; Shu, Ping, Incremental support vector machine learning method for aircraft event recognition, *Proceedings - 2nd International Conference on Enterprise Systems, ES 2014*, p 201-204, December 23, 2014.
- [102] Gu, Bin; Sheng, Victor S.; Tay, Keng Yeow; Romano, Walter; Li, Shuo, Incremental Support Vector Learning for Ordinal Regression, *IEEE Transactions on Neural Networks and Learning Systems*, 2014.
- [103] Yin, Gang; Zhang, Ying-Tang; Li, Zhi-Ning; Ren, Guo-Quan; Fan, Hong-Bo, Online fault diagnosis method based on Incremental Support Vector Data Description and Extreme Learning Machine with incremental output structure, *Neurocomputing*, v 128, p 224-231, March 27, 2014;
- [104] Xie, Weiyi; Uhlmann, Stefan; Kiranyaz, Serkan; Gabbouj, Moncef, Incremental learning with support vector data description, *Proceedings - International Conference on Pattern Recognition*, p 3904-3909, December 4, 2014.
- [105] Ji, Rui; Yang, Yupu; Zhang, Weidong, Incremental smooth support vector regression for Takagi-Sugeno fuzzy modeling, *Neurocomputing*, v 123, p 281-291, January 10, 2014.
- [106] Gan, Liangzhi Zhang, Shicheng Liu, Haikuan, Ensemble incremental least squares support vector machines for both classification and regression, *Journal of Computational Information Systems*, v 10, n 14, p 6171-6178, July 15, 2014;
- [107] Cheng, Wei-Yuan; Juang, Chia-Feng, A fuzzy model with online incremental SVM and margin-selective gradient descent learning for classification problems, *IEEE Transactions on Fuzzy Systems*, v 22, n 2, p 324-337, April 2014.
- [108] Pan, Yu-Xiong; Ren, Zhang; Li, Qing-Dong, Dynamic Bayesian least squares support vector machine, *Kongzhi yu Juece/Control and Decision*, v 29, n 12, p 2297-2300, December 1, 2014;
- [109] Wang, Ning; Yang, Yang; Feng, Liyuan; Mi, Zhenqiang; Meng, Kun; Ji, Qing, SVM-based incremental learning algorithm for large-scale data stream in cloud computing, *KSII Transactions on Internet and Information Systems*, v 8, n 10, p 3378-3393, October 31, 2014;
- [110] Quinlan, J. Ross. *C4. 5: Programs for Machine Learning*, Elsevier, 2014.
- [111] Di Mauro, Nicola, et al. Italian Machine Learning and Data Mining research: The last years. *Intelligenza Artificiale 7.2* : 77-89. 2013.

- 
- [112] <http://sarfrazjmi.blogspot.com/2012/10/supervised-vs-unsupervised-learning.html>, Sep 23, 2014
- [113] Larose, Daniel T., and Chantal D. Larose. Preparing to Model the Data. *Discovering Knowledge in Data: An Introduction to Data Mining, Second Edition* : 138-148, 2014.
- [114] [http://sfb649.wiwi.huberlin.de/fedc\\_homepage/xplore/ebooks/html/csa/node204.html](http://sfb649.wiwi.huberlin.de/fedc_homepage/xplore/ebooks/html/csa/node204.html), Sep 23, 2014.
- [115] Supervised and Unsupervised Learning, [http://sfb649.wiwi.hu-berlin.de/fedc\\_homepage/xplore/ebooks/html/csa/node204.html](http://sfb649.wiwi.hu-berlin.de/fedc_homepage/xplore/ebooks/html/csa/node204.html) on 12/25/2014.
- [116] C. F. Tsai, and C.Y Lin, A Triangle Area-Based Nearest Neighbors Approach to Intrusion Detection, *Pattern Recognition*, 43(1): 222-229, 2010.
- [117] Choi, Byoung-Jeong, et al. Variable Selection for Naive Bayes Semisupervised Learning. *Communications in Statistics-Simulation and Computation* 43.10 : 2702-2713, 2014.
- [118] P. Kavitha, M. Usha, Anomaly Based Intrusion Detection In WLAN Using Discrimination Algorithm Combined With Naïve Bayesian Classifier, *Journal of Theoretical and Applied Information Technology*, Vol. 61 No.3, 2014.
- [119] Vijayasathy, R., Ravindran, B.and Raghavan, S.V, A system approach to network modeling for DDoS detection using a Naive Bayesian classifier, *COMSNETS*, 2011.
- [120] Wang Ding, Supervised vs Unsupervised Methods, <http://dm-dingwang.blogspot.com/2007/05/supervised-versus-unsupervised-methods.html>, Sep 23, 2014
- [121] Zhou, Yinghui, et al. Pre-classification based hidden Markov model for quick and accurate gesture recognition using a finger-worn device. *Applied intelligence* 40.4 : 613-622. 2014
- [122] Supervised versus unsupervised methods, [http://ce.sharif.edu/courses/84-85/2/ce324/resources/root/SupplementaryMaterialsforFinalExam/DataMining\(Classification\)pdf.pdf](http://ce.sharif.edu/courses/84-85/2/ce324/resources/root/SupplementaryMaterialsforFinalExam/DataMining(Classification)pdf.pdf) on 12/25/2014 10:55:33 PM
- [123] Bergmeir, Christoph, and José M. Benítez. On the use of cross-validation for time series predictor evaluation. *Information Sciences* 191: 192-213, 2012
- [124] Iclal Çetin Taş, Depiction Of Hemorrhagic Regions Caused By Diabetic Retinopathy On Fundus Flouresan Angiography Images By Using Image Enhancement And Basic

- 
- Classification Methods, *A Msc Thesis, Çukurova University, Institute Of Natural And Applied Sciences*, 2012.
- [125] Atish Roy, Vikram Jayaram and Kurt J. Marfurt, Active learning algorithms in seismic facies classification, *SEG Houston 2013 Annual Meeting*, 2013.
- [126] Galit Schmueli, Nitin R. Patel and Peter C. Bruce, Data Mining for Business Intelligence: Concepts, *Techniques and Applications in Microsoft Office Excle with XLMiner*, Wiley Publication, Second Edition, 2010.
- [127] Wilhelm, Adalbert. Data and Knowledge Mining. *Handbook of Computational Statistics. Springer Berlin Heidelberg*, 825-852, 2012.
- [128] James E. Gentle, Wolfgang Karl Härdle, Yuichi Mori, Handbook of Computational Statistics, *Springer Science & Business Media*, Jul 6, 2012
- [129] Joseph, Anthony D., et al. Machine Learning Methods for Computer Security. Machine Learning Methods for Computer Security, *Dagstuhl Manifestos 3.1 : 1-30*. 2012
- [130] Cardoso, Douglas de O., et al. Clustering data streams with weightless neural networks. *ESANN*. 2011.
- [131] Heinen, Milton Roberto, and Paulo Martins Engel. An incremental probabilistic neural network for regression and reinforcement learning tasks. *Artificial Neural Networks–ICANN 2010. Springer Berlin Heidelberg 170-179*, 2010.
- [132] Huang, Guang-Bin, Dian Hui Wang, and Yuan Lan. Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics 2.2 : 107-122*. 2011.
- [133] Elwell, Ryan, and Robi Polikar. Incremental learning of concept drift in nonstationary environments. *Neural Networks, IEEE Transactions on 22.10 : 1517-1531*. 2011.
- [134] Chen, Sheng, and Haibo He. Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach. *Evolving Systems 2.1 : 35-50*. 2011.
- [135] Elwell, Ryan, and Robi Polikar. Incremental learning of concept drift in nonstationary environments. *Neural Networks, IEEE Transactions on 22.10 : 1517-1531*. 2011.
- [136] Polikar, Robi, et al. Learn++. MF: A random subspace approach for the missing feature problem. *Pattern Recognition 43.11 : 3817-3832*. 2010.
- [137] Kumari, Gt Prasanna. Learn++ Using Dynamic Weighting Ensembles. *Publications Of Problems & Application In Engineering, Vol 04, Special Issue 01*; 2013.

- 
- [138] Marwala, Tshilidzi. On-line Condition Monitoring Using Ensemble Learning. *Condition Monitoring Using Computational Intelligence Methods*. Springer London, 211-226, 2012.
- [139] T. Marwala, Economic Modeling Using Artificial Intelligence Methods, Advanced Information and Knowledge Processing, DOI 10.1007/978-1-4471-5010-7 10, © Springer-Verlag London 2013.
- [140] Z. Muda, W. Yassin, M.N. Sulaiman, N.I. Udzir , Intrusion Detection based on k-Means Clustering and Naïve Bayes Classification , *7th International Conference on IT in Asia (CITA)*, 2011.
- [141] T. Velmurugan and T. Santhanam, Computational Complexity between k-Means and k-Medoids Clustering Algorithms for Normal and Uniform Distributions of Data Points, *Journal of Computer Science*, 6 (3): 363-368, 2010.
- [142] Saxena, Pankaj, Dr BR Vineeta, and Sushma Lehri Ambedkar. Evolving Efficient Clustering Patterns In Liver Patient Data Through Data Mining Techniques. *International Journal Of Computer Applications* 66 , 2013.
- [143] Noor Kamal Kaur, Usvir Kaur, Dr.Dheerendra Singh, K-Medoid Clustering Algorithm- A Review, *International Journal of Computer Application and Technology (IJCAT)*, Volume 1 Issue 1, 2014.
- [144] J. B. MacQueen, Some Methods for classification and Analysis of Multivariate Observations, *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press, 1:281-297, 1967.
- [145] [http://home.deib.polimi.it/matteucc/Clustering/tutorial\\_html/means.html](http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/means.html), September 23, 2014
- [146] Singh, Shalini S., and N. C. Chauhan. K-means v/s K-medoids: A Comparative Study. *National Conference on Recent Trends in Engineering & Technology*. Vol. 13. 2011.
- [147] Gholap, Prachi, and Vikas Maral. Information Retrieval of K-Means Clustering For Forensic Analysis. *International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064*, Volume 4 Issue 1, January 2015
- [148] Jain, Anil K. Data clustering: 50 years beyond K-means. *Pattern recognition letters* 31.8 : 651-666. 2010.



- 
- [149] Archana.M and, Dr. Sumithra Devi K.A, Multilingual Information Retrieval Based On Knowledge Creation Techniques, *International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)*, Vol.1, No.4, October 2011.
- [150] UCI KDD. The Third International Knowledge Discovery and Data Mining Tools Competition Dataset KDD Cup 1999 Data, <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>,1999.
- [151] Traffic Data from Kyoto University's Honeypots. [http://www.takakura.com/Kyoto\\_data/](http://www.takakura.com/Kyoto_data/) 7 Aug, 2012
- [152] Jungsuk Song, Hiroki Takakura and Yasuo Okabe, Cooperation of Intelligent Honeypots to Detect Unknown Malicious Codes, *WOMBAT Workshop on Information Security Threat Data Exchange (WISTDE 2008)*, The IEEE CS Press, Amsterdam, Netherlands, 21-22 April 2008.
- [153] Jungsuk Song, Hiroki Takakura and Yasuo Okabe, Cooperation of Intelligent Honeypots to Detect Unknown Malicious Codes, *WOMBAT Workshop on Information Security Threat Data Exchange (WISTDE 2008)*, The IEEE CS Press, Amsterdam, Netherlands, 21-22 April 2008.
- [154] Roshan Chitrakar, Huang Chuanhe, Selection of Candidate Support Vectors in Incremental SVM for Network Intrusion Detection, *Computers and Security* 45, 231-241, 2014.
- [155] Carlos A. Catania, Facundo Bromberg, Carlos Garcia Garino, An Autonomous Labelling Approach to Support Vector Machine Algorithms for Network Traffic Anomaly Detection, *Expert Systems with Applications: An International Journal Archive*, vol. 39 Issue 2, February 2012.
- [156] Gurevich, Yuri, Nikolaj S. Bjorner, and Dan Teodosiu. Efficient chunking algorithm. *U.S. Patent No. 8,117,173*. 14 Feb. 2012.
- [157] Huacheng Dou, Shijun Deng, Guofei Wang, Yu Jiang, Yongjie Wang, Zebing Zhou, Li Wang, Fei Yan, Fuzzy Nonlinear Proximal Support Vector Machine for Land Extraction Based on Remote Sensing Image, *PLoS One*; 8(7): e69434, 2013.
- [158] Bao, Xiaohui, Tianqi Xu, and Hui Hou. Network intrusion detection based on support vector machine. *Management and Service Science, 2009. MASS'09. International Conference on. IEEE, 2009.*

- 
- [159] M. Ishida, H. Takakura and Y. Okabe, High-Performance Intrusion Detection Using OptiGrid Clustering and Grid-based Labelling, *IEEE/IPSJ Intl. Symposium on Applications and the Internet*, 2011.
- [160] Carlos A. Catania, Facundo Bromberg, Carlos Garcia Garino, An Autonomous Labelling Approach to Support Vector Machine Algorithms for Network Traffic Anomaly Detection, *Expert Systems with Applications: An International Journal Archive*, vol. 39 Issue 2, February 2012.
- [161] Scikit Learn, Support Vector Machines, <http://scikit-learn.org/stable/modules/svm.html>, on 6 March, 2015.
- [162] Luxburg, Ulrike von, Olivier Bousquet, and Bernhard Schölkopf. A compression approach to support vector model selection. *The Journal of Machine Learning Research* 5 : 293-323. 2004.
- [163] Yi Yang, Wu Jiansheng, Xu Wei. Incremental SVM based on reserved set for network intrusion detection. *Expert Syst Appl* 7698-707. 2011.
- [164] Manning Christopher D, Raghavan Prabhakar, Schultze Hinrich. Introduction to information retrieval. *Cambridge University Press*; 2008.
- [165] Roshan Chitrakar and Huang Chuanhe, Anomaly based Intrusion Detection using Hybrid Learning Approach of combining k-Medoids Clustering and Naïve Bayes Classification, *The 8th International Conference on Wireless Communication, Networking and Mobile Computing, Shanghai, China*, 2012.
- [166] Bhavsar Himani, Panchal Mahesh H, Patel Mittal, A comprehensive study on RBF kernel in SVM for multiclass using OAA, *Int J. Comput. Sci. Manag. Res.* 2(5), 2013.
- [167] Roshan Chitrakar, Huang Chuanhe, Selection of Candidate Support Vectors in Incremental SVM for Network Intrusion Detection, *Computers and Security* 45, 231-241, 2014.

## Published Papers

1.	Chitrakar Roshan and Huang Chuanhe, “Anomaly based Intrusion Detection using Hybrid Learning Approach of combining k-Medoids Clustering and Naïve Bayes Classification”, <i>The 8th International Conference on Wireless Communication, Networking and Mobile Computing, Shanghai, China, 2012.</i>	Related to Chapter 3
2.	Chitrakar Roshan, Huang Chuanhe, “Anomaly Detection using Support Vector Machine Classification with k-Medoids Clustering. <i>In Proceedings of the third Asian Himalayan International Conference on Internet (AH-ICI). 2012.</i>	Related to Chapter 4
3.	Chitrakar Roshan, Huang Chuanhe, “Selection of Candidate Support Vectors in Incremental SVM for Network Intrusion Detection”, <i>Computers &amp; Security, 2014.</i>	Related to Chapter 5

## **Acknowledgement**

The first and the foremost gratefulness goes to my supervisor Prof. Huang Chuanhe for his tremendous courage and support to me. During five years long period, he has put me into the right track of my research work and enriched me with the level of knowledge that I possess now.

I am equally indebted to my Chinese friends and teachers in my department and university for providing me with necessary help and information regarding the entire study and research process. In the mean time, I can not stop myself from thanking foreigner friends, who are studying and living in China, for creating a right environment to live and study in this foreign-land.

Finally but not least importantly, I am obliged to all my countrymates studying and living in Wuhan for their love and warmth like in home. Similarly, I must give a lot of thanks to my wife, daughter and other family members who allowed me to stay abroad for such a remarkable time period to accomplish my research.

## **Appendix I : Features of Kyoto 2006+ datasets**

All the experiments works carried out in this research use Kyoto 2006+ dataset for the purpose of simulating and evaluating the proposed approaches. In this section, the features / attributes are described.

### **Kyoto 2006+ Dataset**

The Kyoto 2006+ dataset consists of 14 conventional features, which were present in KDD Cup '99 dataset, and 10 additional features. Based on KDD Cup 99 data set, but unlike 41 original features of KDD Cup 99 data set, only 14 significant and essential features have been extracted from the raw traffic data obtained by honeypot systems that are deployed in Kyoto University. Additional 10 features have also been extracted in order to help investigate more effectively what happens on the networks. Moreover, they can also be used as training and testing data along with 14 conventional features.

Here are the 14 conventional features: -

1. **Duration**: the length (number of seconds) of the connection
2. **Service**: the connection's service type, e.g., http, telnet, etc
3. **Source bytes**: the number of data bytes sent by the source IP address
4. **Destination bytes**: the number of data bytes sent by the destination IP address
5. **Count**: the number of connections whose source IP address and destination IP address are the same to those of the current connection in the past two seconds
6. **Same srv rate**: % of connections to the same service in Count feature
7. **Serror rate**: % of connections that have "SYN" errors in Count feature
8. **Srv serror rate**: % of connections that have "SYN" errors in Srv count(the number of connections whose service type is the same to that of the current connection in the past two seconds) feature
9. **Dst host count**: among the past 100 connections whose destination IP address is the same to that of the current connection, the number of connections whose source IP address is also the same to that of the current connection
10. **Dst host srv count**: among the past 100 connections whose destination IP address

is the same to that of the current connection, the number of connections whose service type is also the same to that of the current connection

11. ***Dst host same src port rate***: % of connections whose source port is the same to that of the current connection in Dst host count feature
12. ***Dst host error rate***: % of connections that have “SYN” errors in Dst host count feature
13. ***Dst host srv error rate***: % of connections that “SYN” errors in Dst host srv count feature
14. ***Flag***: the state of the connection at the time the summary was written (which is usually when the connection terminated). This feature can have following values: -
  - S0**: Connection attempt seen, no reply.
  - S1**: Connection established, not terminated.
  - SF**: Normal establishment and termination.
  - REJ**: Connection attempt rejected.
  - S2**: Connection established and close attempt by originator seen (but no reply from responder).
  - S3**: Connection established and close attempt by responder seen (but no reply from originator).
  - RSTO**: Connection established, originator aborted (sent a RST).
  - RSTR**: Established, responder aborted.
  - RSTOS0**: Originator sent a SYN followed by a RST, never seen a SYN ACK from the responder.
  - RSTRH**: Responder sent a SYN ACK followed by a RST, never seen a SYN from the (purported) originator.
  - SH**: Originator sent a SYN followed by a FIN, never seen a SYN ACK from the responder (hence the connection was “half” open).
  - SHR**: Responder sent a SYN ACK followed by a FIN, never seen a SYN

from the originator.

**OTH:** No SYN seen, just midstream traffic (a “partial connection” that was not later closed).

The additional features are as follows: -

**15. IDS detection:** reflects whether IDS(Intrusion Detection System) triggered an alert for the connection; ‘0’ means any alerts were not triggered, and an arabic numeral(except ‘0’) means the different kinds of the alerts. Parenthesis indicates the number of the same alert observed during the connection.

Symantec IDS[3] was used to extract this feature.

**16. Malware detection:** indicates whether malware, also known as malicious software, was observed in the connection; ‘0’ means no malware was observed, and a string indicates the corresponding malware observed at the connection. We used ‘clamav’ software to detect malwares. Parenthesis indicates the number of the same malware observed during the connection.

**17. Ashula detection:** means whether shellcodes and exploit codes were used in the connection by using the dedicated software[4]; ‘0’ means no shellcodes and exploit codes were observed, and an arabic numeral(except ‘0’) means the different kinds of the shellcodes or exploit codes. Parenthesis indicates the number of the same shellcode or exploit code observed during the connection.

**18. Label:** indicates whether the session was attack or not; ‘1’ means the session was normal, ‘-1’ means known attack was observed in the session, and ‘-2’ means unknown attack was observed in the session.

**19. Source IP Address:** indicates the source IP address used in the session. Due to the security concerns, the original IP address on IPv4 was properly sanitized to one of the Unique Local IPv6 Unicast Addresses (private IP addresses)[5]. Also, the same private IP addresses are only valid in the same month; if two private IP addresses are the same within the same month, it means their IP addresses on IPv4 were also the same, but if two private IP addresses are the same within the different month, their IP addresses on IPv4 are also different.

**20. Source Port Number:** indicates the source port number used in the session.

**21. Destination IP Address:** indicates the source IP address used in the session. Due

to the security concerns, the original IP address on IPv4 was properly sanitized to one of the Unique Local IPv6 Unicast Addresses (private IP address)[5]. Also, the same private IP addresses are only valid in the same month; if two private IP addresses are the same within the same month, it means their IP addresses on IPv4 were also the same, but if two private IP addresses are the same within the different month, their IP addresses on IPv4 are also different.

**22. Destination Port Number:** indicates the destination port number used in the session.

**23. Start Time:** indicates when the session was started.

**24. Duration:** indicates how long the session was being established.



## Appendix II: Experimental Data Samples

Since the entire experimental data is enormously large – billions in total, only 10 (ten) data records from each simulation/experiment work is presented below just to give a glance to the data. (Starting from the left to the right, the first column represents the 1<sup>st</sup> feature, the second column represents the 2<sup>nd</sup> feature, and so on): -

**Table 6-1: 10 (Ten) records of the sample data from 2007 Nov. 1**

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host srv count	Dst host same src port rate	Dst host error rate	Dst host srv error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
754.15	5	69677	135318	0	0	0	0	0	0	0	0	0	RSTOSO	203027 (600)	0	0	-1	fd58:d212:b798:4097:3915:3bf0:1b09:0d67	40775	fd58:d212:b798:04e1:7d0f:27ea:07ab:05f7	21	0:00:02	754.1481
910.55	8	29358	89	0	0	0	0	0	0	0	0	0	RSTRH	0	0	0	1	fd58:d212:b798:4d03:0f47:274d:15d7:241f	9994	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	0:00:03	910.5477
0.22	8	0	58	0	0	0	0	0	0	0	0	0	RSTO	0	0	0	1	fd58:d212:b798:745e:2d7c:5792:4423:1fd2	37824	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	0:00:14	0.22494
17.33	8	1574	244	0	0	0	0	0	0	0	0	0	SF	0	0	0	1	fd58:d212:b798:c594:09f1:5a6b:2bb6:08f3	4665	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	0:00:17	17.32547
0	11	512	0	0	0	0	0	0	0	0	0	0	SO	0	0	0	-1	fd58:d212:b798:afdb:23e6:50ad:56de:659d	2467	fd58:d212:b798:f558:7d32:2749:619e:0f6e	53	0:00:19	0
4.2	8	4043	183	0	0	0	0	0	0	0	0	0	SF	0	0	0	1	fd58:d212:b798:f7a3:0554:0aa0:4709:2d09	2678	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	0:00:20	4.197315
59.7	8	38915	268	0	0	0	0	0	0	0	0	0	SF	0	0	0	1	fd58:d212:b798:74f6:1767:052e:03c9:135a	3671	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	0:00:22	59.7017
89.88	8	2547	166	0	0	0	0	0	0	0	0	0	SF	0	0	0	1	fd58:d212:b798:5c47:232d:1590:290e:1f8a	38956	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	0:00:24	89.8774
0	8	0	0	0	0	0	0	0	0	0	0	0	RSTRH	0	0	0	1	fd58:d212:b798:200e:0692:0f4e:33ea:075d	24877	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	0:00:29	0
0	49	376	0	0	0	0	0	0	0	0	0	0	SO	20081 (2)	0	58 (1)	-1	fd58:d212:b798:0330:3449:43b9:0c9b:4036	1060	fd58:d212:b798:ba86:7db8:27ed:60dd:0f3d	1434	0:00:36	0

... continues to record number 75,700 (seventy five thousand seven hundred)

**Table 6-2: 10 (Ten) records of the sample data from 2007 Nov. 2**

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host srv count	Dst host same src port rate	Dst host error rate	Dst host srv error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
0.3	8	6	9	0	0	0	0	1	100	1	0	0	SF	0	0	0	1	fd58:d212:b798:acbe:039d:7d37:197b:1981	1180	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	00:00:01	0.302446
0.56	8	1003	77	1	1	0	0	0	100	0	0	0	RSTO	0	0	0	1	fd58:d212:b798:1308:0717:0f61:2d10:7c5b	59572	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	00:00:01	0.564294
1.44	8	11	40	2	1	0	0	1	100	1	0	0	RSTO	0	0	0	1	fd58:d212:b798:286c:1842:0efc:03cd:0092	56793	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	00:00:02	1.441167
15.22	8	4023	40	3	1	0	0	1	100	1	0	0	RSTO	0	0	0	1	fd58:d212:b798:1530:2aed:07de:7ae7:3763	50304	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	00:00:03	15.216203
56.16	8	7887	31	2	1	0	0	1	100	1	0	0	RSTO	0	0	0	1	fd58:d212:b798:81eb:4962:41d9:4196:7dc7	1882	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	00:00:04	56.164035
0	49	376	0	0	0	0	0	0	0	0	0	0	SO	20081(2)	0	58(1)	-1	fd58:d212:b798:724d:23ec:4006:0063:3995	63628	fd58:d212:b798:f54:7d9a:2719:0792:0fd6	1434	00:00:05	0
0	49	376	0	0	0	0	0	0	0	0	0	0	SO	20081(2)	0	58(1)	-1	fd58:d212:b798:0330:3449:43b9:0c9b:4036	1060	fd58:d212:b798:db48:285e:0a26:3c67:0f3b	1434	00:00:09	0

5.92	8	7611	244	0	0	0	0	0	100	0	0	0	SF	0	0	0	1	fd58:d212:b798:f529:0f70:02f6:0365:0020	3724	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	00:00:12	5.923192
10.72	8	5148	227	1	1	0	0	0	100	0	0	0	SF	217009(2)	0	0	1	fd58:d212:b798:b5f3:49aa:0bc2:272e:7a3a	3557	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	00:00:14	10.724039
2.75	8	9502	244	1	1	0	0	0	100	0	0	0	RSTO	0	0	0	1	fd58:d212:b798:5af3:0f08:23bb:67bf:39c6	54471	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	00:00:15	2.745778
... continues to record no. 59,800 (fifty nine thousand eight hundred)																							

Table 6-3: 10 (Ten) records of the sample data from 2007 Nov. 3

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host svr count	Dst host same src port rate	Dst host error rate	Dst host svr error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
46.71	8	4124	40	0	0	0	0	1	100	1	0	0	RSTO	0	0	0	1	fd58:d212:b798:88ed:364d:285b:1fbc:2beb	58072	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	00:00:01	46.710455
21.82	8	3878	244	1	1	0	0	0	100	0	0	0	RSTO	0	0	0	1	fd58:d212:b798:dff6:376e:28ef:47fa:134b	55075	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	00:00:02	21.819117
4.87	8	178	40	2	1	0	0	1	100	1	0	0	RSTO	0	0	0	1	fd58:d212:b798:fb9e:36ec:477a:41ae:1994	50362	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	00:00:03	4.865306
19.47	8	1857	162	3	1	0	0	19	100	0.05	0	0	RSTO	0	0	0	1	fd58:d212:b798:7e00:3790:6a7a:63af:177f	50393	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	00:00:03	19.472351
924.77	8	3868	262	4	1	0	0	0	100	0	0	0	RSTR	0	0	0	1	fd58:d212:b798:6475:3481:398a:17a0:0393	50225	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	00:00:03	924.768498
2.9	8	3892	244	3	1	0	0	0	100	0	0	0	RSTO	0	0	0	1	fd58:d212:b798:d8f9:2ade:3c82:232a:1bc9	57591	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	00:00:05	2.901441
0	8	0	58	0	0	0	0	2	100	0	0	0	RSTOS0	0	0	0	1	fd58:d212:b798:88ed:364d:285b:1fbc:2beb	57448	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	00:00:08	0.000207
18.94	8	1918	244	0	0	0	0	0	100	0	0	0	RSTO	0	0	0	1	fd58:d212:b798:6656:002c:1381:0d1d:4975	48248	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	00:00:13	18.938784
10.1	8	1884	244	1	1	0	0	0	100	0	0	0	RSTO	0	0	0	1	fd58:d212:b798:a269:42cc:0249:1990:1b22	58045	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	00:00:14	10.103155
60.35	8	6933	244	0	0	0	0	3	100	0	0	0	RSTO	0	0	0	1	fd58:d212:b798:88ed:364d:285b:1fbc:2beb	58113	fd58:d212:b798:3a98:7dae:27b7:0775:0fd2	25	00:00:20	60.354551
... continues to record no. 102,679 (One hundred two thousand and six hundred seventy nine)																							

Table 6-4: 10 (Ten) records of the sample data from 2007 Dec. 1

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host svr count	Dst host same src port rate	Dst host error rate	Dst host svr error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
86394.23	49	590300	0	0	0	0	0	0	0	0	0	0	S0	0	0	0	-1	fd0d:a544:ba8c:68ca:7d75:27c4:0741:0770	1985	fd0d:a544:ba8c:7c52:1fa7:ffbd:ffd1:01ab	1985	00:00:02	86394.226142
0.78	8	0	48	0	0	0	0	0	0	0	0	0	RSTO	0	0	0	1	fd0d:a544:ba8c:1b5a:07da:15ad:30d3:353c	2882	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:03	0.777007
0.58	8	0	48	0	0	0	0	0	0	0	0	0	RSTO	0	0	0	1	fd0d:a544:ba8c:63c1:2be1:0533:4430:3cd5	2336	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:03	0.576297
42.73	8	1805	291	0	0	0	0	0	0	0	0	0	SF	0	0	0	1	fd0d:a544:ba8c:d3ff:4de0:0337:2175:4f57	3466	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:04	42.733766
0.36	8	0	48	0	0	0	0	0	0	0	0	0	RSTO	0	0	0	1	fd0d:a544:ba8c:3f45:1242:1520:660a:23ee	1214	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:06	0.360616
0.58	8	0	48	0	0	0	0	0	0	0	0	0	RSTO	0	0	0	1	fd0d:a544:ba8c:63c1:2be1:0533:4430:3cd5	2351	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:06	0.57891
0	49	376	0	0	0	0	0	0	0	0	0	0	S0	20081 (2)	0	58(1)	-1	fd0d:a544:ba8c:25bd:77da:007a:03a7:23a6	4470	fd0d:a544:ba8c:6caa:7de8:27ec:60c4:3502	1434	00:00:07	0
19412.69	49	6196	3360	0	0	0	0	0	0	0	0	0	SF	0	0	0	-1	fd0d:a544:ba8c:4bc4:36dd:282c:13c9:39cf	10324	fd0d:a544:ba8c:4c2d:285d:0aab:3fbf:0049	1666	00:00:07	19412.686673
3.32	8	2486	244	0	0	0	0	0	0	0	0	0	RSTO	0	0	0	1	fd0d:a544:ba8c:a4a6:002a:647d:15d3:0e08	53561	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:08	3.318075
0	49	376	0	0	0	0	0	0	0	0	0	0	S0	20081 (2)	0	58(1)	-1	fd0d:a544:ba8c:a9a:0b0f:5075:0c55:4930	4321	fd0d:a544:ba8c:fb04:7d08:27a6:6069:0f55	1434	00:00:10	0
... continues to record no. 92513 (ninety two thousand five hundred thirteen)																							

**Table 6-5: 10 (Ten) records of the sample data from 2007 Dec. 8**

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host srv count	Dst host same src port rate	Dst host error rate	Dst host srv error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
14.8	8	4419	127	0	0	0	0	9	100	0.11	0	0	SF	0	0	0	1	fd0d:a544:ba8c:e463:0c17:1fbf:29f8:0152	2184	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:01	14.804559
0	8	0	0	1	1	0	0	32	100	0.03	0	0	RSTRH	503014 (2)	0	0	1	fd0d:a544:ba8c:b1d4:27c2:0fc9:6496:71eb	35114	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:02	0
6.36	8	4537	40	2	1	0	0	10	100	0.1	0	0	SF	0	0	0	1	fd0d:a544:ba8c:e463:0c17:1fbf:29f8:0152	2133	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:02	6.364676
0.31	8	0	48	3	1	0	0	33	100	0	0	0	RSTO	0	0	0	1	fd0d:a544:ba8c:b1d4:27c2:0fc9:6496:71eb	35032	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:03	0.310112
19.38	8	4691	175	4	1	0	0	11	100	0	0	0	SF	0	0	0	1	fd0d:a544:ba8c:e463:0c17:1fbf:29f8:0152	2229	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:03	19.384287
86395.9	49	590340	0	0	0	0	0	0	0	0	0	0	SO	0	0	0	-1	fd0d:a544:ba8c:68ca:7d75:27c4:0741:0770	1985	fd0d:a544:ba8c:7c52:1fa7:ffbd:ffd1:01ab	1985	00:00:03	86395.90183
3.14	8	4488	244	4	1	0	0	0	100	0	0	0	RSTO	0	0	0	1	fd0d:a544:ba8c:7bea:21d7:21e4:1c41:073e	53475	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:04	3.140802
0.31	8	0	48	3	1	0	0	34	100	0	0	0	RSTO	0	0	0	1	fd0d:a544:ba8c:b1d4:27c2:0fc9:6496:71eb	33844	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:05	0.305864
0.33	8	0	48	2	1	0	0	1	100	1	0	0	RSTO	0	0	0	1	fd0d:a544:ba8c:5d84:0c2a:245e:21e0:0003	60293	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:06	0.325276
0	8	0	0	3	1	0	0	0	100	0	0	0	RSTRH	0	0	0	1	fd0d:a544:ba8c:ceea:4fb3:0e4c:0fcf:3e4c	59449	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:06	0

... continues to record no. 80,929 (eighty thousand nine hundred and twenty nine)

**Table 6-6: 10 (Ten) records of the sample data from 2007 Dec. 15**

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host srv count	Dst host same src port rate	Dst host error rate	Dst host srv error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
22.71	8	2989	317	0	0	0	0	1	100	1	0	0	SF	0	0	0	1	fd0d:a544:ba8c:750f:0099:4e89:6f9f:0cde	5922	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:01	22.710102
0	49	376	0	0	0	0	0	0	0	0	0	0	SO	20081 (2)	0	58 (1)	-1	fd0d:a544:ba8c:4c4e:35fd:3f59:5199:177e	1107	fd0d:a544:ba8c:ef90:7d79:2739:0f96:148c	1434	00:00:02	0
23.57	8	0	48	1	1	0	0	0	100	0	0	0	RSTO	0	0	0	1	fd0d:a544:ba8c:06b1:0921:2f8a:33a9:6faa	2694	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:02	23.571397
33.97	8	11	40	2	1	0	0	1	100	1	0	0	RSTO	0	0	0	1	fd0d:a544:ba8c:0ffa:377c:2d28:0d86:2d31	50744	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:02	33.965337
0.01	49	0	0	0	0	0	1	23	100	0.04	0	0	SF	0	0	0	-1	fd0d:a544:ba8c:630c:2ce0:6f80:5301:1db1	3356	fd0d:a544:ba8c:4c2d:285d:0aab:3fbf:0049	8947	00:00:03	0.005557
11.65	8	1556	196	3	1	0	0	1	100	1	0	0	RSTO	0	0	0	1	fd0d:a544:ba8c:2bd1:36c5:0836:146b:5148	61809	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:03	11.653584
29.36	49	9	0	1	1	0	0.5	24	100	0	0	0	SF	0	0	0	-1	fd0d:a544:ba8c:630c:2ce0:6f80:5301:1db1	3426	fd0d:a544:ba8c:4c2d:285d:0aab:3fbf:0049	8947	00:00:03	29.355431
86395.09	49	590400	0	0	0	0	0.33	0	0	0	0	0	SO	0	0	0	-1	fd0d:a544:ba8c:68ca:7d75:27c4:0741:0770	1985	fd0d:a544:ba8c:7c52:1fa7:ffbd:ffd1:01ab	1985	00:00:04	86395.093952
2.45	8	0	48	1	1	0	0	2	100	0.5	0	0	RSTO	0	0	0	1	fd0d:a544:ba8c:8477:21cd:5530:0503:0796	1771	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:05	2.449122
165.01	8	2075	291	1	1	0	0	0	100	0	0	0	SF	0	0	0	1	fd0d:a544:ba8c:84b1:4142:03ad:35d5:32ba	1839	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:07	165.012371

... continues to record no. 87,893 (eighty seven thousand eight hundred ninety three)

**Table 6-7: 10 (Ten) records of the sample data from 2007 Dec. 22**

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host srv count	Dst host same src port rate	Dst host error rate	Dst host srv error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
0	8	0	0	2	1	0	0	27	100	0.04	0	0	SH	0	0	0	1	fd0d:a544:ba8c:2205:39d0:0b19:1b90:1fa7	1617	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:01	0.00218
0	8	0	0	3	1	0	0	1	100	1	0	0	RSTOS0	0	0	0	1	fd0d:a544:ba8c:9559:3105:09ed:03db:40f7	1887	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:01	0
0.37	8	1958	93	4	1	0	0	0	100	0	0	0	RSTO	0	0	0	1	fd0d:a544:ba8c:09af:17fa:703b:05b9:711e	59172	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:01	0.37166
0.36	8	0	0	5	1	0	0	8	100	0.12	0	0	SH	0	0	0	1	fd0d:a544:ba8c:e2ec:0000:153b:21b0:6720	62643	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:01	0.364339
0.99	8	0	48	6	1	0	0	12	100	0.08	0	0	RSTO	0	0	0	1	fd0d:a544:ba8c:12bf:41c6:5b99:0399:43b9	3426	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:01	0.989423
1.69	8	4759	77	7	1	0	0	28	100	0.04	0	0	SF	0	0	0	1	fd0d:a544:ba8c:2205:39d0:0b19:1b90:1fa7	1662	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:01	1.687638
2.51	8	3925	40	8	1	0	0	9	100	0.11	0	0	SF	0	0	0	1	fd0d:a544:ba8c:e2ec:0000:153b:21b0:6720	62662	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:01	2.511667
2.73	8	4572	93	9	1	0	0	29	100	0	0	0	SF	0	0	0	1	fd0d:a544:ba8c:2205:39d0:0b19:1b90:1fa7	1661	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:01	2.728535
4.34	8	4130	85	10	1	0	0	10	100	0.1	0	0	SF	0	0	0	1	fd0d:a544:ba8c:e2ec:0000:153b:21b0:6720	62696	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:01	4.343112
5.28	8	4744	175	11	1	0	0	30	100	0	0	0	SF	0	0	0	1	fd0d:a544:ba8c:2205:39d0:0b19:1b90:1fa7	1715	fd0d:a544:ba8c:c958:7dd5:2763:073d:0f8e	25	00:00:01	5.277697

... continues to record no. 99,399 (ninety nine thousand three hundred ninety nine)

**Table 6-8: 10 (Ten) records of the sample data from 2008 Dec. 1**

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host srv count	Dst host same src port rate	Dst host error rate	Dst host srv error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
0.55	8	6	9	0	0	0	0	0	0	0	0	0	SF	0	0	0	1	fd07:24e2:4c3e:1c66:4207:03e2:052c:1cd6	12219	fd07:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:02	0.552304
830.74	49	5660	0	0	0	0	0	0	0	0	0	0	S0	0	0	0	-1	fd07:24e2:4c3e:93b3:7dd5:279e:07eb:07d0	1985	fd07:24e2:4c3e:281b:1f2b:ff1c:ffc9:0161	1985	00:00:02	830.74129
3.9	8	4171	245	0	0	0	0	0	0	0	0	0	SF	0	0	0	1	fd07:24e2:4c3e:39be:17aa:07f3:2733:0ded	4997	fd07:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:03	3.899372
0	49	0	0	0	0	0	0	0	0	0	0	0	S0	0	0	0	-1	fd07:24e2:4c3e:fca9:450e:637c:13c4:42b0	2326	fd07:24e2:4c3e:f886:7d64:2768:6081:093b	42405	00:00:03	0
0	49	0	0	0	0	0	0	0	0	0	0	0	S0	0	0	0	-1	fd07:24e2:4c3e:b60c:0d3b:018f:1bf1:3116	1801	fd07:24e2:4c3e:fa03:7de7:27a2:6021:0397	445	00:00:07	0
0.33	8	0	0	0	0	0	0	0	0	0	0	0	RSTO	0	0	0	1	fd07:24e2:4c3e:b256:26f6:3ada:63aa:0f6a	61053	fd07:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:15	0.325813
0.32	8	0	0	0	0	0	0	0	0	0	0	0	RSTO	0	0	0	1	fd07:24e2:4c3e:b256:26f6:3ada:63aa:0f6a	61199	fd07:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:15	0.322563
29.92	8	23262	567	0	0	0	0	0	0	0	0	0	SF	0	0	0	1	fd07:24e2:4c3e:cb31:17a6:04ea:0faa:2d6d	4199	fd07:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:15	29.920104
0	49	0	0	0	0	0	0	0	0	0	0	0	RSTOS0	0	0	0	-1	fd07:24e2:4c3e:d560:0988:7e8d:00d9:579f	9201	fd07:24e2:4c3e:9976:7acd:30ba:30a9:3992	30556	00:00:19	0
0.36	8	0	0	0	0	0	0	0	0	0	0	0	RSTO	0	0	0	1	fd07:24e2:4c3e:b256:26f6:3ada:63aa:0f6a	59833	fd07:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:19	0.357562

... continues to record no. 73,249 (seventy three thousand two hundred forty nine)

**Table 6-9: 10 (Ten) records of the sample data from 2008 Dec. 9**

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host srv count	Dst host same src port rate	Dst host error rate	Dst host srv error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
----------	---------	--------------	-------------------	-------	---------------	------------	----------------	----------------	--------------------	-----------------------------	---------------------	-------------------------	------	---------------	-------------------	------------------	-------	-------------------	--------------------	------------------------	-------------------------	------------	----------

3.92	8	3857	245	0	0	0	0	0	100	0	0	0	SF	0	0	0	1	fda7:24e2:4c3e:d065:0f6b:031d:4cdd:63de	1947	fda7:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:01	3.921487
1.25	8	12	17	1	1	0	0	0	100	0	0	0	SF	0	0	0	1	fda7:24e2:4c3e:3a0a:475e:0a34:1948:03af	6281	fda7:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:02	1.246658
1.95	8	927	184	2	1	0	0	0	100	0	0	0	SF	0	0	0	1	fda7:24e2:4c3e:3bb9:0713:0b57:4079:1d26	1570	fda7:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:02	1.946757
2.76	8	582	86	3	1	0	0	0	100	0	0	0	SF	0	0	0	1	fda7:24e2:4c3e:fe32:178a:3078:1f86:324a	19117	fda7:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:02	2.757666
14136.6	49	95280	0	0	0	0	0	0	0	0	0	0	S0	0	0	0	-1	fda7:24e2:4c3e:93b3:7dd5:279e:07eb:07d0	1985	fda7:24e2:4c3e:281b:1f2b:ff1c:ffc9:0161	1985	00:00:02	14136.603555
4.91	8	627	176	4	1	0	0	1	100	0	0	0	SF	0	0	0	1	fda7:24e2:4c3e:fe32:178a:3078:1f86:324a	19180	fda7:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:03	4.914287
5	8	640	176	5	1	0	0	2	100	0	0	0	SF	0	0	0	1	fda7:24e2:4c3e:fe32:178a:3078:1f86:324a	19181	fda7:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:03	5.001742
4.86	8	644	176	6	1	0	0	3	100	0	0	0	SF	0	0	0	1	fda7:24e2:4c3e:fe32:178a:3078:1f86:324a	19184	fda7:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:03	4.859307
1.62	8	30	82	3	1	0	0	0	100	0	0	0	RSTO	0	0	0	1	fda7:24e2:4c3e:74f5:354c:1e40:210a:0c99	49600	fda7:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:05	1.616571
4.9	8	630	176	4	1	0	0	4	100	0	0	0	SF	0	0	0	1	fda7:24e2:4c3e:fe32:178a:3078:1f86:324a	19203	fda7:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:05	4.899166
... continues to record no. 120,015 (one hundred twenty thousand and fifteen only)																							

Table 6-10: 10 (Ten) records of the sample data from 2008 Dec. 15

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host srv count	Dst host same src port rate	Dst host error rate	Dst host srv error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
3.07	8	0	0	0	0	0	0	0	100	0	0	0	SH	0	0	0	1	fda7:24e2:4c3e:5206:036d:21ca:2daf:3133	21327	fda7:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:01	3.072416
0	8	0	0	1	1	0	0	0	100	0	0	0	SH	0	0	0	1	fda7:24e2:4c3e:223f:2275:3063:1d3e:0f80	1787	fda7:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:01	0.000005
0.13	44	192	192	0	0	0	0	0	0	0	0	0	SF	0	0	0	-1	fda7:24e2:4c3e:9976:7acd:30ba:30a9:3992	123	fda7:24e2:4c3e:b50b:7d2a:2703:0f39:15ab	123	00:00:02	0.134415
1746.79	49	11800	0	0	0	0	0	0	0	0	0	0	S0	0	0	0	-1	fda7:24e2:4c3e:93b3:7dd5:279e:07eb:07d0	1985	fda7:24e2:4c3e:281b:1f2b:ff1c:ffc9:0161	1985	00:00:02	1746.790526
4.1	8	2514	208	0	0	0	0	0	100	0	0	0	RSTO	0	0	0	1	fda7:24e2:4c3e:d8f9:09df:2854:2140:3c10	2665	fda7:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:08	4.098791
16.45	8	4408	575	1	1	0	0	0	100	0	0	0	SF	0	0	0	1	fda7:24e2:4c3e:e35a:171e:0436:2d63:3769	2395	fda7:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:10	16.446759
16.62	8	2196	236	1	1	0	0	0	100	0	0	0	SF	0	0	0	1	fda7:24e2:4c3e:052e:0b6c:15ec:1ecf:7872	18384	fda7:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:11	16.619409
11.36	8	4370	330	0	0	0	0	0	100	0	0	0	SF	0	0	0	1	fda7:24e2:4c3e:3bdb:3b84:3213:3632:6807	3098	fda7:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:14	11.364307
14.61	8	2192	245	1	1	0	0	1	100	0	0	0	SF	0	0	0	1	fda7:24e2:4c3e:3bdb:3b84:3213:3632:6807	2015	fda7:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:14	14.606987
20.49	8	797	245	2	1	0	0	0	100	0	0	0	SF	0	0	0	1	fda7:24e2:4c3e:a4d0:1ed5:5193:2736:4b09	4150	fda7:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:14	20.492459
... continues to record no. 87,994 (eighty seven thousand nine hundred ninety four)																							

Table 6-11: 10 (Ten) records of the sample data from 2008 Dec. 22

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host srv count	Dst host same src port rate	Dst host error rate	Dst host srv error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
1	6	188	136	0	0	0	0	100	100	0.01	0	0	SF	0	0	0	-1	fda7:24e2:4c3e:f800:4db4:03f6:7c69:0791	48008	fda7:24e2:4c3e:2c72:7abe:0029:5f92:344f	22	00:00:02	0.65675
1.09	6	204	136	1	1	0	0	100	100	0.01	0	0	SF	0	0	0	-1	fda7:24e2:4c3e:f800:4db4:03f6:7c69:0791	48049	fda7:24e2:4c3e:2c72:7abe:0029:5f92:344f	22	00:00:02	1.093628
1.13	6	204	136	2	1	0	0	100	100	0	0	0	SF	0	0	0	-1	fda7:24e2:4c3e:f800:4db4:03f6:7c69:0791	48124	fda7:24e2:4c3e:2c72:7abe:0029:5f92:344f	22	00:00:02	1.126857
2.44	6	520	1304	3	1	0	0	100	100	0	0	0	SF	0	0	0	-1	fda7:24e2:4c3e:f800:4db4:03f6:7c69:0791	48368	fda7:24e2:4c3e:2c72:7abe:0029:5f92:344f	22	00:00:02	2.437225
3.5	8	194	171	0	0	0	0	1	100	1	0	0	SF	0	0	0	1	fda7:24e2:4c3e:cc2e:17ac:001c:01c0:337a	2717	fda7:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:02	3.502344
3.7	6	520	1325	4	1	0	0	100	100	0	0	0	SF	0	0	0	-1	fda7:24e2:4c3e:f800:4db4:03f6:7c69:0791	49169	fda7:24e2:4c3e:2c72:7abe:0029:5f92:344f	22	00:00:02	3.696002
5.31	8	732	253	1	1	0	0	0	100	0	0	0	SF	0	0	0	1	fda7:24e2:4c3e:21bd:06a0:677e:10e5:5ff1	2805	fda7:24e2:4c3e:1ae6:7d65:27b1:0714:0f2d	25	00:00:02	5.305814
2.64	6	520	1325	5	1	0	0	100	100	0	0	0	SF	0	0	0	-1	fda7:24e2:4c3e:f800:4db4:03f6:7c69:0791	48546	fda7:24e2:4c3e:2c72:7abe:0029:5f92:344f	22	00:00:03	2.636602

2.47	6	520	1325	6	1	0	0	100	100	0	0	0	SF	0	0	0	-1	fda7:24e2:4c3e:f800:4db4:03f6:7c69:0791	48660	fda7:24e2:4c3e:2c72:7abe:0029:5f92:344f	22	00:00:03	2.472075
3.34	6	520	1325	7	1	0	0	100	100	0	0	0	SF	0	0	0	-1	fda7:24e2:4c3e:f800:4db4:03f6:7c69:0791	49304	fda7:24e2:4c3e:2c72:7abe:0029:5f92:344f	22	00:00:03	3.342831

... continues to record no. 100,101 (one hundred thousand and one hundred one)

**Table 6-12: 10 (Ten) records of the sample data from 2009 July 1**

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host srv count	Dst host same src port rate	Dst host error rate	Dst host srv error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
0.09	44	192	144	0	0	0	0	0	0	0	0	0	SF	0	0	0	-1	fdcb:c552:6507:ff12:7d83:279f:0ff3:158e	123	fdcb:c552:6507:6e4d:7a29:3019:309e:39ca	123	00:00:01	0.08869
2.14	8	2865	94	0	0	0	0	0	0	0	0	0	RSTO	0	0	0	1	fdcb:c552:6507:3085:361d:0953:1731:031e	64643	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:01	2.137968
4.3	8	8760	383	0	0	0	0	0	0	0	0	0	RSTO	0	0	0	1	fdcb:c552:6507:e89e:305b:414a:57af:5567	55323	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:01	4.297924
0.04	49	216	156	0	0	0	0	0	0	0	0	0	SF	0	0	0	-1	fdcb:c552:6507:32a2:7a9d:30f4:3042:05c0	500	fdcb:c552:6507:dec6:3f28:14a5:4c53:21cf	500	00:00:02	0.039478
0	13	0	0	0	0	0	0	0	0	0	0	0	RSTRH	0	0	0	-1	fdcb:c552:6507:f2e3:7a3b:303e:30e8:6695	34329	fdcb:c552:6507:782d:49b3:0d21:4814:761a	80	00:00:02	0
2.96	8	2951	245	0	0	0	0	0	0	0	0	0	RSTO	0	0	0	1	fdcb:c552:6507:43d6:43d1:004d:0fd8:4ed6	37890	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:02	2.962333
2.84	8	2911	245	0	0	0	0	0	0	0	0	0	RSTO	0	0	0	1	fdcb:c552:6507:64da:3b48:3fc0:3a08:70bd	3824	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:02	2.842797
0	49	0	0	0	0	0	0	0	0	0	0	0	S0	0	0	0	-1	fdcb:c552:6507:0421:47ef:1900:2409:1569	3467	fdcb:c552:6507:99d1:7d81:27b4:6049:39a2	445	00:00:02	0
3.36	8	682	245	0	0	0	0	0	0	0	0	0	SF	0	0	0	1	fdcb:c552:6507:56a4:0fa1:6f74:234f:0f8e	3998	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:03	3.355988
3.59	8	2909	245	0	0	0	0	0	0	0	0	0	RSTO	0	0	0	1	fdcb:c552:6507:9402:151a:3b34:2749:03aa	3023	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:03	3.592596

... continues to record no. 125,198 (one hundred twenty five thousand and one hundred ninety eight)

**Table 6-13: 10 (Ten) records of the sample data from 2009 July 8**

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host srv count	Dst host same src port rate	Dst host error rate	Dst host srv error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
3.24	8	25991	69	0	0	0	0	1	68	1	0	0	SF	0	0	0	1	fdcb:c552:6507:3e1f:3733:21e9:5a9d:39a9	29445	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:02	3.24432
4.91	8	1232	245	1	1	0	0	0	68	0	0	0	SF	0	0	0	1	fdcb:c552:6507:ebc1:0019:27e4:5145:55c2	3025	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:02	4.910682
6.7	8	26327	32	2	1	0	0	1	69	1	0	0	SF	0	0	0	1	fdcb:c552:6507:e4a5:0754:5b6a:7aab:7291	15614	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:03	6.699002
12.54	8	5853	245	3	1	0	0	0	69	0	0	0	RSTO	0	0	0	1	fdcb:c552:6507:4146:3700:61d5:075a:0f3c	63125	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:03	12.53885
13.39	8	3820	245	4	1	0	0	1	70	1	0	0	RSTO	0	0	0	1	fdcb:c552:6507:8504:3ceb:17a8:1086:1364	1740	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:03	13.393583
0	49	0	0	0	0	0	0	0	0	0	0	0	S0	0	0	0	-1	fdcb:c552:6507:af2f:17fd:1540:1301:55de	4363	fdcb:c552:6507:ff09:7de4:2744:6047:2b8b	445	00:00:03	0
0	49	0	0	0	0	0	1	0	6	0	0	0	REJ	0	0	0	-1	fdcb:c552:6507:eec7:2cbf:5ecb:27bb:29e4	4500	fdcb:c552:6507:378a:7dd0:2706:0764:0501	445	00:00:05	0.000164
0	49	0	0	1	1	0	0.5	1	7	1	0	0	REJ	0	0	0	-1	fdcb:c552:6507:eec7:2cbf:5ecb:27bb:29e4	4500	fdcb:c552:6507:378a:7dd0:2706:0764:0501	445	00:00:05	0.000256
2.3	8	26291	597	0	0	0	0	15	15	0	0	0	SF	0	0	0	1	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	60736	fdcb:c552:6507:be10:70bc:01e8:3181:0fed	25	00:00:05	2.296249
4.48	8	2547	308	3	1	0	0	0	69	0	0	0	RSTO	0	0	0	1	fdcb:c552:6507:ad25:26cd:18ed:03d3:40c5	54080	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:05	4.478941

... continues to record no. 124,537 (one hundred twenty four thousand and five hundred thirty seven)

**Table 6-14: 10 (Ten) records of the sample data from 2009 July 15**

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host srv count	Dst host same src port rate	Dst host error rate	Dst host srv error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
2.19	8	1470	922	0	0	0	0	22	22	0	0	0	SF	0	0	0	1	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	55489	fdcb:c552:6507:bf2b:70ab:0100:3106:024a	25	00:00:00	2.185567
0.12	44	192	192	0	0	0	0	61	98	1	0	0	SF	0	0	0	-1	fdcb:c552:6507:6e4d:7a29:3019:309e:39ca	123	fdcb:c552:6507:ff12:7d83:279f:0ff3:158e	123	00:00:01	0.118537
0	49	0	0	0	0	0	0	1	3	1	1	1	S0	0	0	0	-1	fdcb:c552:6507:1648:1920:75ef:67fb:2155	36407	fdcb:c552:6507:0dc1:7d6d:278a:60af:0043	445	00:00:02	0
0.32	8	0	0	0	0	0	0	0	57	0	0	0	RSTO	0	0	0	1	fdcb:c552:6507:78bb:362f:0e8f:45f3:1beb	41895	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:03	0.319445
3.25	8	750	245	1	1	0	0	0	57	0	0	0	RSTO	0	0	0	1	fdcb:c552:6507:d848:1891:031d:69da:15c6	4820	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:03	3.252052
3.85	8	856	167	2	1	0	0	0	58	0	0	0	SF	0	0	0	1	fdcb:c552:6507:c7a1:41de:5a9d:03db:6980	4990	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:03	3.846338
0	13	0	0	0	0	0	0	0	14	0	0	0	RSTO	0	0	0	-1	fdcb:c552:6507:dcd8:7d12:27ba:0fa7:156e	32802	fdcb:c552:6507:3e14:5144:0347:4271:326c	80	00:00:05	0.001641
0	49	0	0	0	0	0	0	0	1	0	0	1	S0	0	0	0	-1	fdcb:c552:6507:fa6e:4528:0f05:0356:70c1	4803	fdcb:c552:6507:d156:7d3a:2741:6012:006f	445	00:00:05	0
1.73	8	984	601	0	0	0	0	21	21	0	0	0	SF	0	0	0	1	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	55494	fdcb:c552:6507:bf2b:70ab:0100:3106:024a	25	00:00:06	1.726716
2.52	8	1269	276	0	0	0	0	0	59	0	0	0	RSTO	0	0	0	1	fdcb:c552:6507:f448:189c:4dd6:416f:2c8e	2639	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:06	2.516369

... continues to record no. 125,688 (one hundred twenty five thousand and six hundred eighty eight)

**Table 6-15: 10 (Ten) records of the sample data from 2009 July 22**

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host srv count	Dst host same src port rate	Dst host error rate	Dst host srv error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
1.2	49	128	219	0	0	0	0.8	9	9	0	0	0	SF	0	0	0	-1	fdcb:c552:6507:ee5d:0b84:235f:11f0:03c2	2116	fdcb:c552:6507:207e:7d44:27cb:61bc:03dd	445	00:00:00	1.204293
33.56	8	9307	53	17	1	0.76	0.76	0	100	0	0	0.81	RSTO	0	0	0	1	fdcb:c552:6507:9fd1:1817:1731:5019:0347	33516	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:00	33.562506
38.89	8	2206	252	9	1	0.67	0.67	0	100	0	0	0.8	RSTO	0	0	0	1	fdcb:c552:6507:a950:42e0:2f22:5325:005c	62398	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:01	38.892701
0	49	0	0	0	0	0	0.5	1	1	1	1	1	S0	0	0	0	-1	fdcb:c552:6507:1ba1:4112:039f:13f1:5819	4312	fdcb:c552:6507:b47d:7df8:279f:60fb:1b27	445	00:00:01	0
0	49	0	0	0	0	0	0.67	1	2	1	1	1	S0	0	0	0	-1	fdcb:c552:6507:dccb:4324:0f1c:61c1:4b9c	2231	fdcb:c552:6507:1ad7:7d5f:2766:606c:1305	445	00:00:01	0
0.12	44	192	192	0	0	0	0	45	76	1	0	0	SF	0	0	0	-1	fdcb:c552:6507:6e4d:7a29:3019:309e:39ca	123	fdcb:c552:6507:ff12:7d83:279f:0ff3:158e	123	00:00:02	0.118913
3.36	6	520	1551	0	0	0	0	99	99	0	0	0	SF	0	0	0	-1	fdcb:c552:6507:8b3b:1734:28f5:515a:301f	48686	fdcb:c552:6507:163f:7dc4:27ce:07e3:0a1f	22	00:00:03	3.36325
6.06	8	26165	236	1	1	0	0	0	100	0	0	0.79	SF	0	0	0	1	fdcb:c552:6507:eb63:360e:0ac6:0365:1b33	2187	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:03	6.059758
1.5	8	2496	276	1	1	0	0	0	100	0	0	0.78	RSTO	0	0	0	1	fdcb:c552:6507:4345:350f:75c3:323c:53f3	61525	fdcb:c552:6507:9513:7d90:276f:07f8:0fc8	25	00:00:04	1.496729
0	49	0	0	0	0	0	0	0	3	0	0	1	S0	0	0	0	-1	fdcb:c552:6507:9664:41e3:0f30:7904:03be	3431	fdcb:c552:6507:1cc1:7d57:27de:60ef:05c3	445	00:00:05	0

... continues to record no. 125,442 (one hundred twenty five thousand and four hundred forty two)

**Table 6-16: 10 (Ten) records of sample data from 2009 Aug. 25**

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host srv count	Dst host same src port rate	Dst host error rate	Dst host srv error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
----------	---------	--------------	-------------------	-------	---------------	------------	----------------	----------------	--------------------	-----------------------------	---------------------	-------------------------	------	---------------	-------------------	------------------	-------	-------------------	--------------------	------------------------	-------------------------	------------	----------





Total no. of records = 126,448 (one hundred twenty six thousand and four hundred forty eight)

**Table 6-19: 10 (Ten) records of sample data from 2009 Aug. 28**

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host srv count	Dst host same src port rate	Dst host error rate	Dst host srv error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
6.19	8	6033	245	1	1	0	0	0	100	0	0	0	RSTO	0	0	0	1	fda4:45a3:3fdd:dcc0:42ec:0f7f:03ff:036a	50035	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:00	6.187172
0	49	0	0	0	0	0	1	1	1	1	1	1	S0	0	0	0	-1	fda4:45a3:3fdd:2a19:7324:2e84:794e:0da8	1553	fda4:45a3:3fdd:5ea4:7d5f:275e:60ea:243e	445	00:00:01	0
0.12	44	192	192	0	0	0	0	54	96	1	0	0	SF	0	0	0	-1	fda4:45a3:3fdd:3fbf:7aee:3008:3088:398b	123	fda4:45a3:3fdd:3cd8:7da1:27f6:0f67:15f9	123	00:00:02	0.118002
0.66	8	0	48	1	1	0	0	0	100	0	0	0	RSTO	0	0	0	1	fda4:45a3:3fdd:f597:4909:010f:01f6:013d	28272	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:02	0.65948
0	49	0	0	0	0	0	1	1	2	1	1	1	S0	0	0	0	-1	fda4:45a3:3fdd:5683:4d24:03cb:03ba:1b9e	2815	fda4:45a3:3fdd:5203:7d4a:2705:6051:18ca	445	00:00:02	0
3.36	8	2209	167	1	1	0	0	0	100	0	0	0	SF	0	0	0	1	fda4:45a3:3fdd:c64c:4211:2d4a:53a7:1249	3764	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:03	3.355292
5.05	8	4583	245	2	1	0	0	0	100	0	0	0	RSTO	0	0	0	1	fda4:45a3:3fdd:b4ff:1952:3029:0094:0f5b	61421	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:03	5.049727
0	49	0	0	0	0	0	0	0	1	0	0	1	S0	0	0	0	-1	fda4:45a3:3fdd:81e3:1825:011a:6eaa:0b76	3776	fda4:45a3:3fdd:9978:7dd4:2780:6004:1731	445	00:00:06	0
4.08	8	2172	244	0	0	0	0	0	100	0	0	0	RSTO	0	0	0	1	fda4:45a3:3fdd:5ef8:36cb:077b:12be:0301	4935	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:07	4.078934
0	49	0	0	0	0	0	1	0	2	0	0	1	S0	0	0	0	-1	fda4:45a3:3fdd:abcf:47bd:1b14:0c43:2ba4	4520	fda4:45a3:3fdd:7888:7d59:278a:60a9:17d2	445	00:00:08	0

Total no. of records = 127,608 (one hundred twenty seven thousand and six hundred eight)

**Table 6-20: 10 (Ten) records of sample data from 2009 Aug. 29**

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host srv count	Dst host same src port rate	Dst host error rate	Dst host srv error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
0.12	44	192	192	0	0	0	0	46	94	1	0	0	SF	0	0	0	-1	fda4:45a3:3fdd:3fbf:7aee:3008:3088:398b	123	fda4:45a3:3fdd:3cd8:7da1:27f6:0f67:15f9	123	00:00:01	0.118005
33.9	8	2241	167	1	1	0	0	0	100	0	0	0	SF	0	0	0	-1	fda4:45a3:3fdd:432c:413a:0cd4:2e41:0342	2691	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:01	33.903481
0	49	0	0	0	0	0	0.5	1	1	1	1	1	S0	0	0	0	-1	fda4:45a3:3fdd:e5d8:42e1:510f:27ff:030f	1634	fda4:45a3:3fdd:e3ad:7da3:27f4:604c:2168	445	00:00:01	0
0	49	0	0	0	0	0	0.67	0	0	0	0	0	S0	0	0	0	-1	fda4:45a3:3fdd:d4ec:5146:233a:61a8:03b6	3323	fda4:45a3:3fdd:323a:7dce:270d:60de:2919	445	00:00:01	0
0	49	0	0	0	0	0	1	0	3	0	0	1	S0	0	0	0	-1	fda4:45a3:3fdd:43b9:4f30:6300:180f:4b56	2115	fda4:45a3:3fdd:a562:7d9e:27fe:60e6:05d0	445	00:00:03	0
0	49	0	0	0	0	0	1	0	1	0	0	1	S0	0	0	0	-1	fda4:45a3:3fdd:d946:4370:00de:33e2:612b	4457	fda4:45a3:3fdd:a0c0:7d87:27d1:6019:1e9c	445	00:00:04	0
2.77	8	2299	245	0	0	0	0	0	100	0	0	0	SF	0	0	0	1	fda4:45a3:3fdd:2024:45f0:0c0e:00e3:0dc2	17837	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:05	2.770091
0	8	0	0	1	1	0	0	0	100	0	0	0	RSTOS0	0	0	0	1	fda4:45a3:3fdd:91b8:4917:1860:100c:4c14	4781	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:06	0
35.58	8	2174	276	2	1	0	0	1	100	0	0	0	RSTO	0	0	0	1	fda4:45a3:3fdd:91b8:4917:1860:100c:4c14	1288	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:06	35.576056
74.96	8	0	0	3	1	0	0	6	100	0	0	0	RSTR	503014(1)	0	0	1	fda4:45a3:3fdd:448d:427c:0c94:074f:1bb4	60879	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:06	74.955875

Total no. of records = 128,881 (one hundred twenty eight thousand and eight hundred eighty one)

**Table 6-21: 10 (Ten) records of sample data from 2009 Aug. 30**

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host srv count	Dst host same src port rate	Dst host error rate	Dst host srv error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
0	49	0	0	0	0	0	0.33	1	2	1	1	1	SO	0	0	0	-1	fda4:45a3:3fdd:6486:063e:02b6:6b6c:412c	4147	fda4:45a3:3fdd:1a52:7df3:275e:60e7:2f3f	445	00:00:00	0
0.12	44	192	192	0	0	0	0	58	94	1	0	0	SF	0	0	0	-1	fda4:45a3:3fdd:3fbf:7aee:3008:3088:398b	123	fda4:45a3:3fdd:3cd8:7da1:27f6:0f67:15f9	123	00:00:01	0.117878
25.13	8	2306	276	1	1	0	0	1	100	0	0	0	RSTO	0	0	0	1	fda4:45a3:3fdd:c464:0980:24f3:0935:64b3	4396	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:01	25.133814
3.48	8	4527	409	1	1	0	0	0	100	0	0	0	RSTO	0	0	0	1	fda4:45a3:3fdd:25cf:0d76:563c:19c0:0f5f	3967	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:02	3.481789
3.84	8	4263	377	2	1	0	0	1	100	0	0	0	RSTO	0	0	0	1	fda4:45a3:3fdd:25cf:0d76:563c:19c0:0f5f	3964	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:02	3.8377
3.96	8	2296	245	3	1	0	0	0	100	0	0	0	SF	0	0	0	1	fda4:45a3:3fdd:9819:4124:28bf:23f9:072a	36600	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:02	3.959251
6.59	8	9031	338	4	1	0	0	0	100	0	0	0	RSTO	0	0	0	1	fda4:45a3:3fdd:2a65:4237:3064:4140:0346	4536	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:02	6.591676
5.81	8	9059	338	5	1	0	0	1	100	0	0	0	RSTO	0	0	0	1	fda4:45a3:3fdd:2a65:4237:3064:4140:0346	4543	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:03	5.814764
17.46	8	0	48	5	1	0	0	0	100	0	0	0	RSTO	0	0	0	1	fda4:45a3:3fdd:edeb:4bec:68d2:7421:6e84	18020	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:04	17.456841
20.33	8	1284	551	6	1	0	0	0	100	0	0	0	SF	0	0	0	1	fda4:45a3:3fdd:c20f:09dc:1677:6874:051e	1027	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:04	20.329444

Total no. of records = 131,694 (one hundred thirty one thousand and six hundred ninety four)

**Table 6-22: 10 (Ten) records of sample data from 2009 Aug. 31**

Duration	Service	Source bytes	Destination bytes	Count	Same srv rate	Error rate	Srv error rate	Dst host count	Dst host srv count	Dst host same src port rate	Dst host error rate	Dst host srv error rate	Flag	IDS detection	Malware detection	Ashula detection	Label	Source IP Address	Source Port Number	Destination IP Address	Destination Port Number	Start Time	Duration
0.37	49	128	219	2	1	0	0.33	25	56	0	0	0	SF	0	0	0	-1	fda4:45a3:3fdd:730c:07fc:0790:3f0d:01dd	4917	fda4:45a3:3fdd:9bd5:7df4:2744:6149:031a	445	00:00:00	0.371577
1.18	49	128	219	3	1	0	0.25	12	57	0	0	0	SF	0	0	0	-1	fda4:45a3:3fdd:9909:4263:2a71:1415:6fd1	3106	fda4:45a3:3fdd:9bd5:7df4:2744:6149:031a	445	00:00:00	1.177594
2.37	49	2491	1518	4	1	0	0.2	25	57	0	0	0	RSTO	23179(1)	0	342(1)	-1	fda4:45a3:3fdd:730c:07fc:0790:3f0d:01dd	4918	fda4:45a3:3fdd:9bd5:7df4:2744:6149:031a	445	00:00:00	2.36788
9.9	49	2764	1750	5	1	0	0.17	11	58	0	0	0	RSTO	0	0	342(1)	-2	fda4:45a3:3fdd:a663:25cf:5698:3983:1eae	2727	fda4:45a3:3fdd:9bd5:7df4:2744:6149:031a	445	00:00:00	9.901614
0	49	0	0	0	0	0	0.14	1	1	1	1	1	SO	0	0	0	-1	fda4:45a3:3fdd:43e3:5194:2da5:5252:3c0e	4441	fda4:45a3:3fdd:3e03:7d50:279b:60be:033e	445	00:00:00	0
0.12	44	192	192	0	0	0	0	59	96	1	0	0	SF	0	0	0	-1	fda4:45a3:3fdd:3fbf:7aee:3008:3088:398b	123	fda4:45a3:3fdd:3cd8:7da1:27f6:0f67:15f9	123	00:00:01	0.119253
2.71	6	536	1551	0	0	0	0	100	100	0	0	0	SF	0	0	0	-1	fda4:45a3:3fdd:c218:4360:0fcd:31da:1e7a	35329	fda4:45a3:3fdd:ecce:7d1c:2792:072e:0a71	22	00:00:01	2.706176
7.08	8	9085	245	0	0	0	0	0	100	0	0	0	RSTO	0	0	0	1	fda4:45a3:3fdd:ee0:2bc2:7036:1161:07a1	1870	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:02	7.079902
8.65	8	18107	330	1	1	0	0	1	100	0	0	0	RSTO	0	0	0	1	fda4:45a3:3fdd:ee0:2bc2:7036:1161:07a1	1871	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:02	8.65455
9.37	8	18096	330	2	1	0	0	2	100	0	0	0	RSTO	0	0	0	1	fda4:45a3:3fdd:ee0:2bc2:7036:1161:07a1	1869	fda4:45a3:3fdd:d62c:7df2:2761:071f:0f51	25	00:00:02	9.369518

Total no. of records = 134,665 (one hundred thirty four thousand and six hundred sixty five)

# 武汉大学学位论文使用授权协议书

(一式两份，一份论文作者保存，一份留学校存档)

本学位论文作者愿意遵守武汉大学关于保存、使用学位论文的管理办法及规定，即：学校有权保存学位论文的印刷本和电子版，并提供文献检索与阅览服务；学校可以采用影印、缩印、数字化或其它复制手段保存论文；在以教学与科研服务为目的前提下，学校可以在校园网内公布部分及全部内容。

- 1、 在本论文提交当年，同意在校园网内以及中国高等教育文献保障系统（CALIS）高校学位论文系统提供查询及前十六页浏览服务。
- 2、 在本论文提交当年/一年/两年/三年以后，同意在校园网内允许读者在线浏览并下载全文，学校可以为存在馆际合作关系的兄弟高校用户提供文献传递服务和交换服务。（保密论文解密后遵守此规定）

论文作者（签名）：



学 号： 2010172110001

学 院： 计算机

日期：2015 年 6 月 1 日