

# **Network Anomaly Detection with Incomplete Audit Data**

Animesh Patcha

Dissertation submitted to the Faculty of the  
Virginia Polytechnic Institute and State University  
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy  
in  
Computer Engineering

Dr. Jung-Min Park, Chair

Dr. Luiz A. DaSilva

Dr. Y. Thomas Hou

Dr. Sandeep K. Shukla

Dr. Christopher L. North

July 6, 2006

Blacksburg, Virginia

Keywords: Anomaly detection, denial-of-service, expectation-maximization,  
high speed networks, weighted sampling

© Copyright 2006, Animesh Patcha

# **Network Anomaly Detection with Incomplete Audit Data**

Animesh Patcha

(ABSTRACT)

With the ever increasing deployment and usage of gigabit networks, traditional network anomaly detection based intrusion detection systems have not scaled accordingly. Most, if not all, systems deployed assume the availability of complete and clean data for the purpose of intrusion detection. We contend that this assumption is not valid. Factors like noise in the audit data, mobility of the nodes, and the large amount of data generated by the network make it difficult to build a normal traffic profile of the network for the purpose of anomaly detection.

From this perspective, the leitmotif of the research effort described in this dissertation is the design of a novel intrusion detection system that has the capability to detect intrusions with high accuracy even when complete audit data is not available. In this dissertation, we take a holistic approach to anomaly detection to address the threats posed by network based denial-of-service attacks by proposing improvements in every step of the intrusion detection process. At the data collection phase, we have implemented an adaptive sampling scheme that intelligently samples incoming network data to reduce the volume of traffic sampled, while maintaining the intrinsic characteristics of the network traffic. A Bloom filters based fast flow aggregation scheme is employed at the data pre-processing stage to further reduce the response time of the anomaly detection scheme. Lastly, this dissertation also proposes an expectation-maximization algorithm based anomaly detection scheme that uses the sampled audit data to detect intrusions in the incoming network traffic.

*In loving memory of my grandparents*

*late Smt. Anasuya Devi*

*and*

*late Shri. Narla Gouri Shankar Rao*

# Acknowledgements

I would like to acknowledge many people for helping me during my doctoral work. I would especially like to thank my advisor, Dr. Jung-Min Park, for his generous time and commitment. Throughout my doctoral work he has always encouraged me to develop independent thinking and research skills. His words of encouragement, quiet urgings and careful reading of all of my written work will never be forgotten.

I am also very grateful for having an exceptional doctoral committee and wish to thank Dr. Sandeep K. Shukla, Dr. Y. Thomas Hou, Dr. Luiz DaSilva, and Dr. Chris North for their continual support and encouragement.

I would also like to thank Dr. Glenda Scales and Dr. Wayne Scales, without whose affection and support I would not have made it through the last two years of graduate school. I would also like to acknowledge the guidance and support provided by the staff, at the Office of Distance Learning within the College of Engineering at Virginia Tech.

Throughout my stay at Virginia Tech. and before there have been people who have contributed to my career by inspiring, supporting, and challenging me. First and foremost, I would like to thank Mrs. and Dr. Prabhala M. Prasad for always having the faith in

my abilities to make it through graduate school; my friends Ruiliang Chen, Syed Suhaib, Madhup Chandra, Hiren Patel, Vasudha and Sastry Kompella, Kaigui Bian, Amol Goel, Phillip Stephen, Deepak Abraham Mathaikutty, Ryan Thomas, William J. Adams, Grant A. Jacoby, Sachin Lal, Deeti Dave and Anant Utgikar for making my stay in Blacksburg a memorable one. Thank you all.

I thank my family, especially, my parents Sudha and Ramachandra Rao Patcha, my sister and brother-in-law Suparna and Srinivas Dandamudi, the two little ones: my adorable niece and nephew Shreya and Ashwin, and my aunt and uncle Mary Kay and Sudhir Narla, for their continued love, support and encouragement over the years. Last but not the least, I thank my beautiful wife, Lavanya, for her patience, love, and understanding.

# Contents

|                                         |             |
|-----------------------------------------|-------------|
| <b>Dedication</b>                       | <b>iii</b>  |
| <b>Acknowledgements</b>                 | <b>iv</b>   |
| <b>List of Figures</b>                  | <b>xi</b>   |
| <b>List of Tables</b>                   | <b>xiii</b> |
| <b>1 Introduction</b>                   | <b>1</b>    |
| 1.1 Problem Statement . . . . .         | 3           |
| 1.2 Background and Motivation . . . . . | 4           |
| 1.3 Research Contribution . . . . .     | 6           |

|          |                                                    |           |
|----------|----------------------------------------------------|-----------|
| 1.4      | Document Organization . . . . .                    | 8         |
| <b>2</b> | <b>Background</b>                                  | <b>9</b>  |
| 2.1      | Intrusion Detection . . . . .                      | 10        |
| 2.2      | Anomaly Detection Techniques . . . . .             | 15        |
| 2.2.1    | Premise of Anomaly Detection . . . . .             | 15        |
| 2.2.2    | Techniques Used in Anomaly Detection . . . . .     | 16        |
| 2.2.2.1  | Statistical Anomaly Detection . . . . .            | 16        |
| 2.2.2.2  | Machine Learning based Anomaly Detection . . . . . | 22        |
| 2.2.2.3  | Data Mining based Anomaly Detection . . . . .      | 30        |
| 2.3      | Hybrid Systems . . . . .                           | 41        |
| 2.4      | Summary . . . . .                                  | 43        |
| <b>3</b> | <b>Adaptive Sampling for Anomaly Detection</b>     | <b>48</b> |
| 3.1      | Motivation . . . . .                               | 48        |
| 3.2      | Related Work . . . . .                             | 52        |

|          |                                                           |           |
|----------|-----------------------------------------------------------|-----------|
| 3.3      | Self-Similarity and Network Traffic . . . . .             | 55        |
| 3.3.1    | Properties of Network Traffic . . . . .                   | 55        |
| 3.3.2    | Self-Similarity and the Hurst Parameter . . . . .         | 57        |
| 3.4      | Proposed Sampling Algorithm . . . . .                     | 59        |
| 3.4.1    | Weighted Least Square Predictor . . . . .                 | 60        |
| 3.4.2    | Adaptive Weighted Sampling . . . . .                      | 61        |
| 3.5      | Summary . . . . .                                         | 64        |
| <b>4</b> | <b>System Design</b>                                      | <b>66</b> |
| 4.1      | Motivation . . . . .                                      | 66        |
| 4.2      | System Overview . . . . .                                 | 69        |
| 4.3      | Adaptive Sampling . . . . .                               | 71        |
| 4.4      | Flow Aggregation . . . . .                                | 71        |
| 4.5      | POTION: EM Algorithm-Based Clustering Algorithm . . . . . | 73        |
| 4.6      | Data Summaries for Data Reduction . . . . .               | 77        |

|          |                                                         |           |
|----------|---------------------------------------------------------|-----------|
| 4.7      | Anomalous Flow Detection . . . . .                      | 78        |
| 4.8      | Summary . . . . .                                       | 80        |
| <b>5</b> | <b>Experimental Analysis</b>                            | <b>83</b> |
| 5.1      | Data Set Descriptions . . . . .                         | 83        |
| 5.2      | Experimental Results . . . . .                          | 88        |
| 5.2.1    | Evaluation of the Sampling Algorithm . . . . .          | 88        |
| 5.2.1.1  | Experimental Setup . . . . .                            | 89        |
| 5.2.1.2  | Metrics for Evaluation . . . . .                        | 90        |
| 5.2.1.3  | Experimental Results . . . . .                          | 91        |
| 5.2.2    | Evaluation of the Clustering Algorithm . . . . .        | 93        |
| 5.2.2.1  | Experimental Setup . . . . .                            | 94        |
| 5.2.2.2  | Metrics for Evaluation . . . . .                        | 95        |
| 5.2.2.3  | Experimental Results . . . . .                          | 96        |
| 5.2.3    | Evaluation of the Anomaly Detection Algorithm . . . . . | 98        |

|          |                                                 |            |
|----------|-------------------------------------------------|------------|
| 5.2.3.1  | Experimental Setup . . . . .                    | 98         |
| 5.2.3.2  | Metrics for Evaluation . . . . .                | 99         |
| 5.2.3.3  | Experimental Results . . . . .                  | 99         |
| 5.3      | Summary . . . . .                               | 101        |
| <b>6</b> | <b>Summary and Future Work</b>                  | <b>105</b> |
| 6.1      | Summary . . . . .                               | 106        |
| 6.2      | Future Work . . . . .                           | 108        |
| 6.3      | Conclusion . . . . .                            | 110        |
|          | <b>References</b>                               | <b>111</b> |
|          | <b>Appendices</b>                               | <b>129</b> |
|          | <b>A The Expectation–Maximization Algorithm</b> | <b>129</b> |
|          | <b>Vita</b>                                     | <b>133</b> |

# List of Figures

|     |                                                                       |    |
|-----|-----------------------------------------------------------------------|----|
| 2.1 | Organization of a generalized intrusion detection system [6]. . . . . | 11 |
| 2.2 | Flow chart of real time operation in NIDES [4]. . . . .               | 20 |
| 2.3 | Example of a hidden Markov model. . . . .                             | 28 |
| 2.4 | The training phase of ADAM [8] . . . . .                              | 40 |
| 2.5 | The intrusion detection phase of ADAM [8] . . . . .                   | 40 |
| 3.1 | Static sampling techniques [20]. . . . .                              | 53 |
| 3.2 | Block diagram of the adaptive sampling algorithm . . . . .            | 62 |
| 4.1 | System model. . . . .                                                 | 69 |
| 4.2 | Online sampling and flow aggregation algorithm. . . . .               | 73 |

|      |                                                                                                                  |     |
|------|------------------------------------------------------------------------------------------------------------------|-----|
| 4.3  | EM algorithm for clustering. . . . .                                                                             | 81  |
| 4.4  | Data summaries. . . . .                                                                                          | 82  |
| 5.1  | Standard deviation of packet delay. . . . .                                                                      | 91  |
| 5.2  | Average percentage error for the Hurst parameter. . . . .                                                        | 92  |
| 5.3  | Average percentage error for the mean statistic. . . . .                                                         | 93  |
| 5.4  | k-means algorithm for clustering. . . . .                                                                        | 94  |
| 5.5  | ROC curve for detection of the SYN Flood attack using POTION. . . . .                                            | 97  |
| 5.6  | ROC curve for detection of the SYN Flood attack using the <i>k</i> -means clustering<br>algorithm. . . . .       | 98  |
| 5.7  | ROC curve for detection of the SSH Process Table attack using complete audit data.                               | 100 |
| 5.8  | ROC curve for SYN Flood (Neptune) attack detection using complete audit data.                                    | 101 |
| 5.9  | ROC curve for SYN Flood attack detection using varying degrees of missing data<br>(via random sampling). . . . . | 102 |
| 5.10 | Accuracy of clustering with POTION vs. percentage of missing data. . . . .                                       | 103 |
| 5.11 | Comparison of adaptive sampling and random sampling in the detection of the<br>SYN Flood attack. . . . .         | 104 |

# List of Tables

|     |                                                                                                   |    |
|-----|---------------------------------------------------------------------------------------------------|----|
| 2.1 | A summary of statistical anomaly detection systems . . . . .                                      | 45 |
| 2.2 | A summary of machine learning based anomaly detection systems. . . . .                            | 46 |
| 2.3 | A summary of data mining-based anomaly detection systems. . . . .                                 | 47 |
| 5.1 | Examples of DoS attacks in the 1999 DARPA intrusion detection evaluation dataset. . . . .         | 85 |
| 5.2 | Examples of surveillance/probes in the 1999 DARPA intrusion detection evaluation dataset. . . . . | 86 |

# Chapter 1

## Introduction

Today, the Internet along with the corporate network plays a major role in creating and advancing new business avenues. Business needs have motivated enterprises and governments across the globe to develop sophisticated, complex information networks. Such networks incorporate a diverse array of technologies that include distributed data storage systems, encryption and authentication techniques, voice and video over IP, remote and wireless access, and web services. Moreover, corporate networks have become more accessible; for instance, most businesses allow access to their services on their internal networks via extranets to their partners, enable customers to interact with the network through e-commerce transactions, and allow employees to tap into company systems through virtual private networks.

The aforementioned access points make today's networks more vulnerable to intrusions and attacks. Cyber-crime is no longer the prerogative of the stereotypical hacker. Joining ranks with the hackers are disgruntled employees, unethical corporations, and even

terrorist organizations. With the vulnerability of present-day software and protocols combined with the increasing sophistication of attacks, it comes as no surprise that network-based attacks are on the rise [84, 98, 112, 124]. The 2005 annual *Computer Crime and Security survey* [49], jointly conducted by the Computer Security Institute and the FBI, indicated that the financial losses incurred by the respondent companies due to network attacks/intrusions was in the vicinity of US \$130 million. In another survey commissioned by VanDyke Software in 2003, some 66 percent of the companies stated that they perceived system penetration to be the largest threat to their enterprises. Although 86 percent of the respondents used firewalls, their consensus was that firewalls by themselves are not sufficient to provide adequate protection. Moreover, according to recent studies, an average of twenty to forty new vulnerabilities in commonly used networking and computer products are discovered every month. Such wide-spread vulnerabilities in software add to today's insecure computing/networking environment. This insecure environment has given rise to the ever evolving field of intrusion detection and prevention. The cyberspace's equivalent to the burglar alarm, intrusion detection systems complement the beleaguered firewall.

An intrusion detection system gathers and analyzes information from various areas within a computer or a network to identify possible security breaches. In other words, intrusion detection is the act of detecting actions that attempt to compromise the confidentiality, integrity or availability of a system/network. Traditionally, intrusion detection systems have been classified as signature detection systems, anomaly detection systems or a hybrid/compound detection systems. A signature detection system identifies patterns of traffic or application data presumed to be malicious while anomaly detection systems compare activities against a "normal" baseline. On the other hand, a hybrid intrusion detection system combines the techniques of the two approaches. Both signature detection and anomaly detection systems have their share of advantages and drawbacks. The primary advantage of signature detection is that known attacks can be detected fairly reliably with a low false

positive rate. The major drawback of the signature detection approach is that such systems typically require a signature to be defined for all of the possible attacks that an attacker may launch against a network. The biggest advantage of anomaly detection systems is that profiles of normal activity are customized for every system, application and/or network, and therefore making it very difficult for an attacker to know with certainty what activities it can carry out without getting detected. However, the anomaly detection approach also has its share of drawbacks: the intrinsic complexity of the system and the difficulty of associating alarms with the specific events that triggered those alarms.

However, most intrusion detection systems have not been able to keep up with the advances in high speed networking. Intrusion detection products, currently deployed in gigabit networks, need significant improvements before they can offer adequate protection against attacks. A majority of the products in the market today can detect less than half of the attacks directed at them, even though many of those attacks are well documented [130]. Therefore the leitmotif of this dissertation is the development of a network based anomaly detection system that can detect intrusions/attacks in a large, high volume and high speed enterprise network.

## **1.1 Problem Statement**

With the ever increasing deployment and usage of gigabit networks, traditional network anomaly detection based intrusion detection systems have not scaled accordingly. Most, if not all, systems deployed assume the availability of complete and clean audit data for the purpose of intrusion detection. We contend that this assumption is not valid anymore and that existing intrusion detection systems, when applied to high speed networks and/or

wireless networks are limited in a number of ways. Firstly, factors like noise in the audit data and the mobility of the nodes make it difficult to build a normal profile of the network for the purpose of anomaly detection. Secondly, existing network-based intrusion detection sensors can barely keep up with bandwidths of a few hundred Mbps. From this perspective, this research explores a sampling based approach to intrusion detection, and, proposes efficient methods to detect intrusions in the absence of complete audit data within the framework of network anomaly detection.

## **1.2 Background and Motivation**

Intrusion detection in high speed networks has been a much debated topic in the intrusion detection community. An argument [64] that has often been made against intrusion detection in high speed networks is the inability of current techniques to process the incoming audit data at high speeds. A process that is made more difficult with the widespread use of encrypted communication. Others have proposed a distributed architecture to deal with the heavy load of network traffic. Irrespective of which viewpoint is accepted, analysis of network traffic on high-speed links still represents a fundamental need in many practical network installations.

From the research perspective, very few papers have been published that deal with the problem of intrusion detection in high-speed networks. In 1999, Sekar et al. [107] described an approach for high performance analysis of network data, but unfortunately their approach did not provide experimental data based on live traffic analysis. More recently, researchers have been looking at hardware based solutions that use field programmable gate arrays (FPGA) and network cards equipped with a NPU (Network Process Unit). The

advantage of a hardware based approach is that hardware solutions are able to compute traffic statistics directly on the card without having to move all the traffic to the computer. Kruegel et al. [64] describe a hardware architecture that splits incoming traffic into several intrusion detection sensors that work in parallel to identify intrusion attempts. An entirely different approach was proposed by Franklin et al. [59]. Their approach is to build an intrusion detection system using Non-Deterministic Finite Automata (NFA). The NFA approach has the advantage of being able to match quite complex regular expressions without the need to convert the NFA into a Deterministic Finite Automata (DFA). One drawback of this approach is the fact that it does not support dynamic reconfiguration. As a result, the FPGA designs need to be re-compiled following rule changes. The scheme proposed by Franklin et al. operates on a 8-bit data block and achieves FPGA clock rates of 30-120 MHz, dependent on the regular expression complexity.

However, hardware-based solutions such as the ones mentioned above have certain drawbacks; these include the following: (1) They are expensive; (2) They are available only for a few media types, usually for Ethernet and a few others; and (3) They have little memory on board, thus dramatically limiting the size of programs that can run on the card itself.

A number of commercial products have come out onto the market to respond to the need for high speed intrusion detection systems. A number of vendors now claim to have sensors that can operate on high-speed ATM or Gigabit Ethernet links. For example, ISS [60] offers NetICE Gigabit Sentry, a system that is designed to monitor traffic on high-speed links. ISS advertises the system as being capable of performing protocol reassembly and analysis for several application-level protocols (e.g., HTTP, SMTP, and POP) to identify malicious activities. The tool claims to be the first network based intrusion detection system that can handle full Gigabit speeds. However, testbed based experimentation [130] have shown that under real world conditions GigaSentry can only capture slightly more

than 500,000 packets/second.

Other intrusion detection system vendors (like Cisco [19]) offer comparable products with similar features. Unfortunately, no experimental data gathered on real networks is presented in their analysis. For example, TopLayer Networks [87], advertises a switch that can keep track of application-level sessions. In this product, the network traffic is split with regard to the individual sessions and forwarded to several intrusion detection sensors. Packets that belong to the same session are sent through the same link. This allows sensors to detect multiple steps of an attack within a single session. Unfortunately, the correlation of information between different sessions is not supported. This could result in undetected attacks when attacks are performed against multiple hosts (e.g., ping sweeps) or performed across multiple sessions.

For a more detailed description of related work in the domain of anomaly detection, the reader is directed to Section 2.2 in Chapter 2.

### **1.3 Research Contribution**

The overarching goal of the research presented in this dissertation was the development of an anomaly detection scheme that could detect network based intrusions in high bandwidth networks. Although there have been recent attempts [37, 39] to train anomaly detection models over noisy data, to the best of our knowledge, the research presented in this document is the first attempt to investigate the effect of incomplete data sets on the anomaly detection process.

The motivation behind our intrusion detection framework is straightforward: sampling

reduces the amount of audit data that needs to be processed, thereby enabling real time anomaly detection even in high-speed networks. In typical cases, sampling would lead to loss of information, leading to inaccurate predictions and/or false alarms. To avoid such a scenario, and, to achieve the aforementioned goal viz. detecting network based intrusions with incomplete audit data, this dissertation presents an intrusion detection scheme christened SCAN (Stochastic Clustering Algorithm for Network anomaly detection). SCAN consists of the following two modules:

- An adaptive sampling module that intelligently adapts the sampling rate to sample packets geared towards intelligent sampling for anomaly detection,
- A predictive data mining based anomaly detection module that has the capability to estimate the missing data and reduce the occurrence of false alarms,

At the core of SCAN is a stochastic clustering algorithm which is based on an improved version of the Expectation–Maximization (EM) algorithm [26]. The EM algorithm’s speed of convergence was accelerated by using a combination of Bloom filters, data summaries and associated arrays. The advantage of using the EM algorithm is that unlike other imputation techniques, EM computes the maximum likelihood estimates in parametric models based on prior information. The clustering approach that we propose can be used to process incomplete and unlabeled data. To the best of our knowledge, there have been no prior attempts to investigate the effects of incomplete audit data in anomaly detection.

The salient features of SCAN are:

- Ability to intelligently sample incoming network traffic to reduce the amount of audit data that needs to be processed without losing the inherent characteristics of

the network traffic.

- Ability to detect anomalies with a high degree of accuracy in the absence of complete audit data.

## 1.4 Document Organization

This dissertation follows the following outline. Chapter 1 introduces some basic concepts, outlines the problems in current intrusion detection systems when applied to high speed networks and provides the motivation and justification for the work described in this dissertation. Chapter 2 describes related work in the domain of anomaly detection and outlines the problems with current approaches. In Chapter 3 we give a brief background on sampling techniques and describe the adaptive sampling algorithm that forms an integral part of SCAN. Chapter 3 is followed by Chapter 4 where we present the system design of SCAN and discuss the individual modules that comprise it. Chapter 5 presents the analysis and results of the evaluation of the proposed anomaly detection scheme. Finally, Chapter 6 presents the conclusions, summarizes the contributions of this dissertation, and discusses directions for future research.

# Chapter 2

## Background

This chapter presents a comprehensive survey of recent literature in the domain of anomaly detection. In doing so, we attempt to assess the ongoing work in this area as well as consolidate the existing results. The remainder of this chapter is organized as follows. In Section 2.1, we define intrusion detection, put forth the generic architectural design of an intrusion detection system, and highlight the three main techniques for detecting intrusions/attacks in computer networks and systems. In Section 2.2, we describe the premise of anomaly detection and provide detailed discussions on the various techniques used in anomaly detection. We conclude the chapter by highlighting on recently proposed hybrid systems in Section 2.3.

## 2.1 Intrusion Detection

An intrusion detection system is a software tool used to detect unauthorized access to a computer system or network. An intrusion detection system is capable of detecting all types of malicious network traffic and computer usage. This includes network attacks against vulnerable services, data driven attacks on applications, host-based attacks-such as privilege escalation, unauthorized logins and access to sensitive files-and malware. An intrusion detection system is a dynamic monitoring entity that complements the static monitoring abilities of a firewall. An intrusion detection system monitors traffic in a network in promiscuous mode, very much like a network sniffer. The network packets that are collected are analyzed for rule violations by a pattern recognition algorithm. When rule violations are detected, the intrusion detection system alerts the administrator.

One of the earliest work that proposed intrusion detection by identifying abnormal behavior can be attributed to Anderson [5]. In his report, Anderson presents a threat model that classifies threats as external penetrations, internal penetrations, and misfeasance, and uses this classification to develop a security monitoring surveillance system based on detecting anomalies in user behavior. External penetrations are defined as intrusions that are carried out by unauthorized computer system users; internal penetrations are those that are carried out by authorized users who are not authorized for the data that is compromised; and misfeasance is defined as the misuse of authorized access both to the system and to its data.

In a seminar paper, Denning [27] puts forth the idea that intrusions to computers and networks could be detected by assuming that users of a computer/network would behave in a manner that enables automatic profiling. In other words, a model of the behavior

of the entity being monitored could be constructed by an intrusion detection system, and subsequent behavior of the entity could be verified against the entity's model. In this model, behavior that deviates sufficiently from the norm is considered anomalous. In the paper, Denning mentioned several models that are based on statistics, Markov chains, time-series etc.

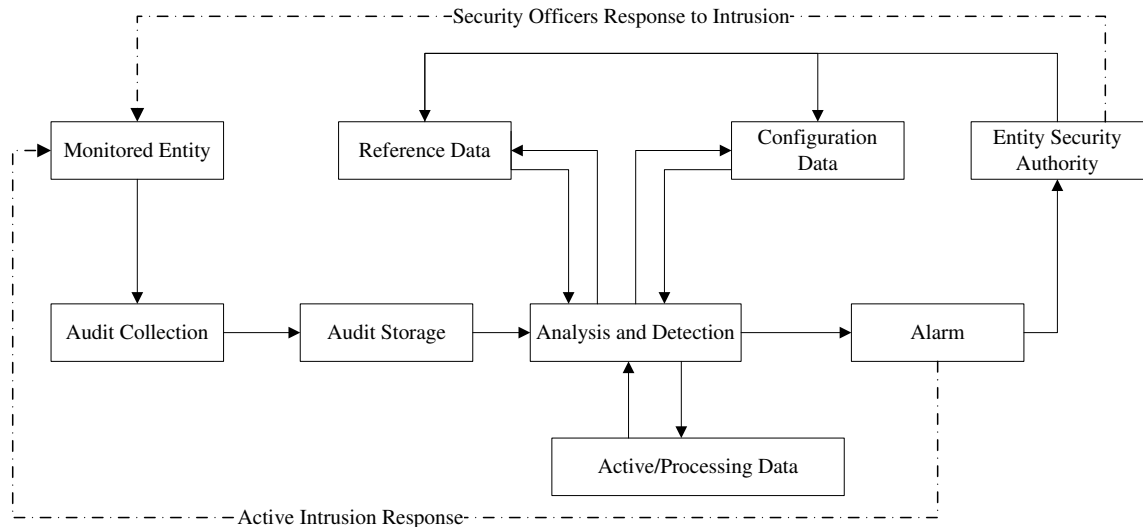


Figure 2.1: Organization of a generalized intrusion detection system [6].

In a much cited survey on intrusion detection systems, Axelsson [3] put forth a generalized model of a typical intrusion detection system. Figure 2.1 depicts such a system where solid arrows indicate data/control flow while dotted arrows indicate a response to intrusive activity. According to Axelsson, the generic architectural model of an intrusion detection system contains the following modules:

**Audit data collection:** This module is used in the data collection phase. The data collected in this phase is analyzed by the intrusion detection algorithm to find traces

of suspicious activity. The source of the data can be host/network activity logs, command-based logs, application-based logs, etc.

**Audit data storage:** Typical intrusion detection systems store the audit data either indefinitely or for a sufficiently long time for later reference. The volume of data is often exceedingly large. Hence, the problem of audit data reduction is a major research issue in the design of intrusion detection systems.

**Analysis and detection:** The processing block is the heart of an intrusion detection system. It is here that the algorithms to detect suspicious activities are implemented. Algorithms for the analysis and detection of intrusions have been traditionally classified into three broad categories: *signature* (or *misuse*) *detection*, *anomaly detection* and *hybrid* (or *compound*) *detection*.

**Configuration data:** The configuration data is the most sensitive part of an intrusion detection system. It contains information that is pertinent to the operation of the intrusion detection system itself such as information on how and when to collect audit data, how to respond to intrusions, etc.

**Reference data:** The reference data storage module stores information about known intrusion signatures (in the case of signature detection) or profiles of normal behavior (in the case of anomaly detection). In the latter case, the profiles are updated when new knowledge about system behavior is available.

**Active/processing data:** The processing element must frequently store intermediate results such as information about partially fulfilled intrusion signatures.

**Alarm:** This part of the system handles all output from the intrusion detection system. The output may be either an automated response to an intrusion or a suspicious activity alert for a system security officer.

Historically, intrusion detection research has concentrated on the analysis and detection stage of the architectural model shown in figure 2.1. As mentioned above, algorithms for the analysis and detection of intrusions/attacks are traditionally classified into the following three broad categories:

- *Signature or misuse detection* is a technique for intrusion detection that relies on a predefined set of attack signatures. By looking for specific patterns, the signature detection-based intrusion detection systems match incoming packets and/or command sequences to the signatures of known attacks. In other words, decisions are made based on the knowledge acquired from the model of the intrusive process and the observed trace that it has left in the system. Legal or illegal behavior can be defined and compared with observed behavior. Such a system tries to collect evidence of intrusive activity irrespective of the normal behavior of the system. One of the chief benefits of using signature detection is that known attacks can be detected reliably with a low false positive rate. The existence of specific attack sequences ensures that it is easy for the system administrator to determine exactly which attacks the system is currently experiencing. If the audit data in the log files do not contain the attack signature, no alarm is raised. Another benefit is that the signature detection system begins protecting the computer/network immediately upon installation. On the other hand, one of the biggest problems with signature detection systems is maintaining state information of signatures in which an intrusive activity spans multiple discrete events—that is, the complete attack signature spans multiple packets. Another drawback is that the signature detection system must have a signature defined for all of the possible attacks that an attacker may launch. This requires frequent signature updates to keep the signature database up-to-date.
- An *anomaly detection* system first creates a baseline profile of the normal system,

network, or program activity. Thereafter, any activity that deviates from the baseline is treated as a possible intrusion. Anomaly detection systems offer several benefits. First, they have the capability to detect insider attacks. For instance, if a user or someone using a stolen account starts performing actions that are outside the normal user-profile, an anomaly detection system generates an alarm. Second, because the system is based on customized profiles, it is very difficult for an attacker to know with certainty what activity it can carry out without setting off an alarm. Third, an anomaly detection system has the ability to detect previously unknown attacks. This is due to the fact that a profile of intrusive activity is not based on specific signatures representing known intrusive activity. An intrusive activity generates an alarm because it deviates from normal activity, not because someone configured the system to look for a specific attack signature. Anomaly detection systems, however, also suffer from several drawbacks. The first obvious drawback is that the system must go through a training period in which appropriate user profiles are created by defining “normal” traffic profiles. Moreover, creating a normal traffic profile is a challenging task. The creation of an inappropriate normal traffic profile can lead to poor performance. Maintenance of the profiles can also be time-consuming. The biggest drawback is the complexity of the system and the difficulty of associating alarms with the specific events that triggered them.

- A *hybrid or compound detection* system combines both approaches. In essence, a hybrid detection system is a signature inspired intrusion detection system that makes a decision using a “hybrid model” that is based on both the normal behavior of the system and the intrusive behavior of the attackers.

## 2.2 Anomaly Detection Techniques

An anomaly detection approach usually consists of two phases: a training phase and a testing phase. In the former, the normal traffic profile is defined; in the latter, the learned profile is applied to new data.

### 2.2.1 Premise of Anomaly Detection

The central premise of anomaly detection is that intrusive activity is a subset of anomalous activity [65]. If we consider an intruder, who has no idea of the legitimate user's activity patterns, intruding into a host system, there is a strong probability that the intruder's activity will be detected as anomalous. In the ideal case, the set of anomalous activities will be the same as the set of intrusive activities. In such a case, flagging all anomalous activities as intrusive activities results in no false positives and no false negatives. However, intrusive activity does not always coincide with anomalous activity. Kumar and Stafford [65] suggested that there are four possibilities, each with a non-zero probability:

- *Intrusive but not anomalous*: These are false negatives. An intrusion detection system fails to detect this type of activity as it is not anomalous. These are called false negatives because the intrusion detection system falsely reports the absence of intrusions.
- *Not intrusive but anomalous*: These are false positives. In other words, the activity is not intrusive, but because it is anomalous, an intrusion detection system reports it as intrusive. These are called false positives because an intrusion detection system

falsely reports intrusions.

- *Not intrusive and not anomalous*: These are true negatives; the activity is not intrusive and is not reported as intrusive.
- *Intrusive and anomalous*: These are true positives; the activity is intrusive and is reported as such.

When false negatives need to be minimized, thresholds that define an anomaly are set low. This results in many false positives and reduces the efficacy of automated mechanisms for intrusion detection. It creates additional burdens for the security administrator as well, who must investigate each incident and discard false positive instances.

## **2.2.2 Techniques Used in Anomaly Detection**

In this subsection, we review a number of different architectures and methods that have been proposed for anomaly detection. These include statistical anomaly detection, data-mining based methods, and machine learning based techniques.

### **2.2.2.1 Statistical Anomaly Detection**

In statistical methods for anomaly detection, the system observes the activity of subjects and generates profiles to represent their behavior. The profile typically includes such measures as activity intensity measure, audit record distribution measure, categorical measures (the distribution of an activity over categories) and ordinal measure (such as CPU usage). Typically, two profiles are maintained for each subject: the current profile and the stored

profile. As the system/network events (viz. audit log records, incoming packets etc.) are processed, the intrusion detection system updates the current profile and periodically calculates an anomaly score (indicating the degree of irregularity for the specific event) by comparing the current profile with the stored profile using a function of abnormality of all measures within the profile. If the anomaly score is higher than a certain threshold, the intrusion detection system generates an alert.

Statistical approaches to anomaly detection have a number of advantages. Firstly, these systems do not require prior knowledge of security flaws and/or the attacks themselves. As a result, such systems have the capability of detecting “zero day” or the very latest attacks. In addition, statistical approaches can provide accurate notification of portscanning activities. Typically, the distribution of portscans is highly anomalous in comparison to the usual traffic distribution. This is particularly true when a packet has unusual features (e.g., a crafted packet). With this in mind, even the portscans that are distributed over a lengthy time frame will be recorded because they will be inherently anomalous. However, statistical anomaly detection schemes also have drawbacks. Skilled attackers can train a statistical anomaly detection to accept abnormal behavior as normal. It can also be difficult to determine thresholds that balance the likelihood of false positives with the likelihood of false negatives. In addition, statistical methods need accurate statistical distributions but not all behaviors can be modeled using purely statistical methods. In fact, a majority of the proposed statistical anomaly detection techniques require the assumption of a quasi-stationary process, which cannot be assumed for most data processed by anomaly detection systems. Models that do not make this assumption are more complex and require greater processing time.

Haystack [111] is one of the earliest examples of a statistical anomaly-based intrusion detection system. It used both user and group-based anomaly detection strategies, and

modeled system parameters as independent, Gaussian random variables. Haystack defined a range of values that were considered normal for each feature. If during a session, a feature fell outside the normal range, the score for the subject was raised. Assuming the features were independent, the probability distribution of the scores was calculated. An alarm was raised if the score was too large. Haystack also maintained a database of user groups and individual profiles. If a user had not previously been detected, a new user profile with minimal capabilities was created using restrictions based on the user's group membership. It was designed to detect six types of intrusions: attempted break-ins by unauthorized users, masquerade attacks, penetration of the security control system, leakage, DoS attacks and malicious use. One drawback of Haystack was that it was designed to work offline. The attempt to use statistical analyses for real-time intrusion detection systems failed, since doing so required high-performance systems. Moreover, statistical measurements can analyze one activity only, but not a sequence of activities. This feature severely limited the usefulness of such systems in terms of the number of attacks they can detect, since most attacks consist of a sequence of actions.

One of the earliest intrusion detection systems was developed at the Stanford Research Institute (SRI) in the early 1980's and was called the Intrusion Detection Expert System (IDES) [28, 78]. IDES was a system that continuously monitored user behavior and detected suspicious events as they occurred. In IDES, intrusions could be flagged by detecting departures from established normal behavior patterns for individual users. As the analysis methodologies developed for IDES matured, scientists at SRI developed an improved version of IDES called the Next-Generation Intrusion Detection Expert System (NIDES) [3, 4]. NIDES was one of the few intrusion detection systems of its generation that could operate in real time for continuous monitoring of user activity or could run in a batch mode for periodic analysis of the audit data. However, the primary mode of operation of NIDES was to run in real-time. A flow chart describing the real time operation

of NIDES is shown in figure 2.2. Unlike IDES, which is an anomaly detection system, NIDES is a hybrid system that has an upgraded statistical analysis engine. In both IDES and NIDES, a profile of normal behavior based on a selected set of variables is maintained by the statistical analysis unit. This enables the system to compare the current activity of the user/system/network with the expected values of the audited intrusion detection variables stored in the profile and then flag an anomaly if the audited activity is sufficiently far from the expected behavior. Each variable in the stored profile reflects the extent to which a particular type of behavior is similar to the profile built for it under “normal conditions”. The way that this is computed is by associating each measure/variable to a corresponding random variable. The frequency distribution is built and updated over time, as more audit records are analyzed. It is computed as an exponential weighted sum with a half-life of 30 days. This implies that the half-life value makes audit records that were gathered 30 days in the past to contribute with half as much weight as recent records; those gathered 60 days in the past contribute one-quarter as much weight, and so on. The frequency distribution is kept in the form of a histogram with probabilities associated with each one of the possible ranges that the variable can take. The cumulative frequency distribution is then built by using the ordered set of bin probabilities. Using this frequency distribution, and the value of the corresponding measure for the current audit record, it is possible to compute a value that reflects how far away from the “normal” value of the measure the current value is. The actual computation in NIDES [4] renders a value that is correlated with how abnormal this measure is. Combining the values obtained for each measure and taking into consideration the correlation between measures, the unit computes an index of how far the current audit record is from the normal state. Records beyond a threshold are flagged as possible intrusions.

However the techniques used in [3, 4] have several drawbacks. Firstly, the techniques are sensitive to the normality assumption. If data on a measure are not normally distributed,

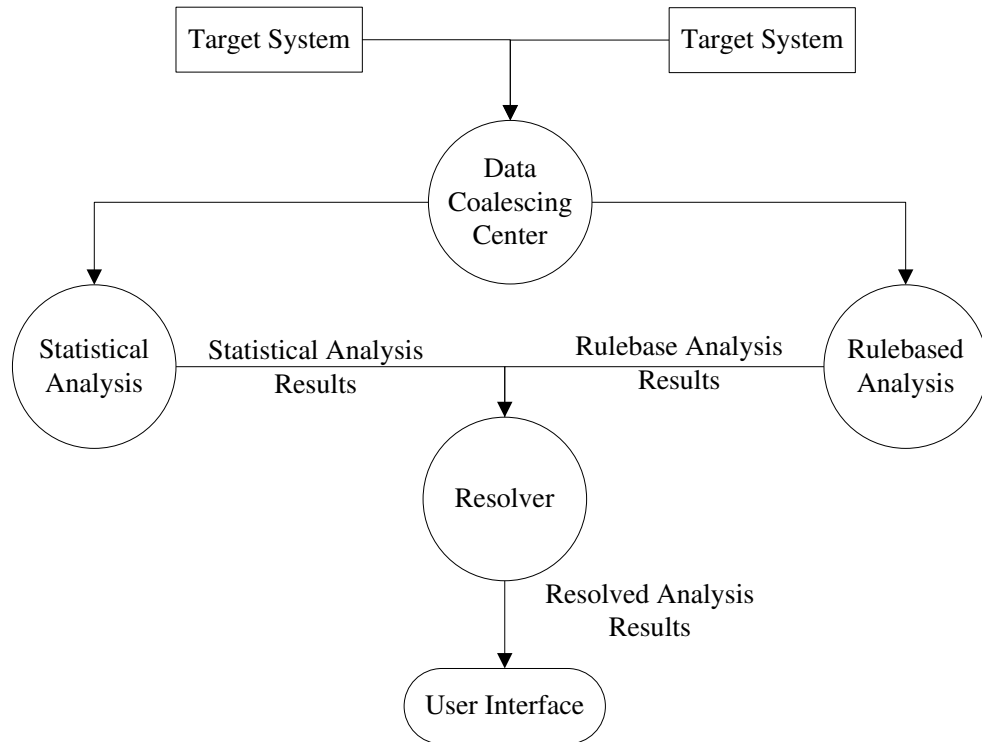


Figure 2.2: Flow chart of real time operation in NIDES [4].

the techniques would yield a high false alarm rate. Secondly, the techniques are predominantly univariate in that a statistical norm profile is built for only one measure of the activities in a system. However, intrusions often affect multiple measures of activities collectively.

Anomalies resulting from intrusions may cause deviations on multiple measures in a collective manner rather than through separate manifestations on individual measures. To overcome the latter problem, Ye et al. [126] presented a technique that used the Hotellings  $T^2$  test<sup>1</sup> to analyze the audit trails of activities in an information system and detect host

<sup>1</sup>The Hotelling's  $T^2$  test statistic for an observation  $x_i$  is determined as  $T^2 = n(x_i - \bar{x})' W^{-1} (x_i - \bar{x})$

based intrusions. The assumption is that host based intrusions leave trails in the audit data. The advantage of using the Hotellings  $T^2$  test is that it aids in the detection of both counterrelationship anomalies as well as mean-shift anomalies. In another paper, Kruegel et al. [62] show that it is possible to find the description of a system that computes a payload byte distribution and combines this information with extracted packet header features. In this approach, the resultant ASCII characters are sorted by frequency and then aggregated into six groups. However, this approach leads to a very coarse classification of the payload.

The primary problem with network based anomaly detection is that a globally accepted norm for what constitutes a normal profile does not exist. Maxion et al. [83] characterized the normal behavior in a network by using different templates that were derived by taking the standard deviations of Ethernet load and packet count at various periods in time. An observation was declared anomalous if it exceeded the upper bound of a predefined threshold. However, Maxion et al. did not consider the non-stationary nature of network traffic which would have resulted in minor deviations in network traffic to go unnoticed.

Mahoney et al. in [80–82] experimented with anomaly detection over the DARPA network data [76] by range matching network packet header fields. PHAD (Packet Header Anomaly Detector) [80], LERAD (LEarning Rules for Anomaly Detection) [81] and ALAD (Application Layer Anomaly Detector) [82] use time-based models in which the probability of an event depends on the time since it last occurred. For each attribute, they collect a set of allowed values and flag novel values as anomalous. PHAD, ALAD, and LERAD differ in the attributes that they monitor. PHAD monitors 33 attributes from the Ethernet, IP and transport layer packet headers. ALAD models incoming server TCP requests: source and destination IP addresses and ports, opening and closing TCP flags, and the

---

where  $\bar{x}$  denote an observation of  $p$  measures on a processor system at time  $i$  and  $W$  is the sample variance. A large value of  $T^2$  indicates a large deviation observation  $x_i$  from the in-control population.

list of commands (the first word on each line) in the application payload. Depending on the attribute, it builds separate models for each target host, port number (service), or host/port combination. LERAD also models TCP connections. Even though, the data set is multivariate network traffic data containing fields extracted out of the packet headers, the authors break down the multivariate problem into a set of univariate problems and sum the weighted results from range matching along each dimension. While the advantage of this approach is that it makes the technique more computationally efficient and effective at detecting network intrusions, breaking multivariate data into univariate data has significant drawbacks especially at detecting attacks. For example, in a typical SYN flood attack an indicator of the attack, is having more SYN requests than usual, but observing a lower than normal ACK rate. Because higher SYN rate or lower ACK rate alone can both happen in normal usage (when the network is busy or idle), it is the combination of higher SYN rate and lower ACK rate that signals the attack.

More recently, analytical studies on anomaly detection systems were conducted. Lee and Xiang [71] used several information-theoretic measures, such as entropy and information gain, to evaluate the quality of anomaly detection methods, determine system parameters, and build models. These metrics help one to understand the fundamental properties of audit data. The highlighting features of some of the schemes surveyed in this section are presented in Table 1.

### **2.2.2.2 Machine Learning based Anomaly Detection**

Machine learning can be defined as the ability of a program and/or a system to learn and improve their performance on a certain task or group of tasks over time. Machine learning aims to answer many of the same questions as statistics or data mining. However,

unlike statistical approaches which tend to focus on understanding the process that generated the data, machine learning techniques focus on building a system that improves its performance with experience.

**System Call and Sequence Analysis** One of the widely used machine learning techniques for anomaly detection involves learning the behavior of a program and recognizing significant deviations from the normal. In a seminal paper, Forrest et al. [42] established an analogy between the human immune system and intrusion detection. They did this by proposing a methodology that involved analyzing a program's system call sequences to build a normal profile. In their paper, they analyzed several UNIX based programs like sendmail, lpr etc. and showed that correlations in fixed length sequences of system calls could be used to build a normal profile of a program. Therefore, programs that show sequences that deviated from the normal sequence profile could then be considered to be victims of an attack. The system they developed was only used off-line using previously collected data and used a quite simple table-lookup algorithm to learn the profiles of programs. Their work was extended by Hofmeyr et al. [55], where they collected a database of normal behavior for each program of interest. Once a stable database is constructed for a given program in a particular environment, the database was then used to monitor the program's behavior. The sequences of system calls formed the set of normal patterns for the database, and sequences not found in the database indicated anomalies.

Another machine learning technique that has been frequently used in the domain of machine learning is the sliding window method. The sliding window method, a sequential learning methodology, converts the sequential learning problem into the classical learning problem. It constructs a window classifier  $h_w$  that maps an input window of width  $w$  into an individual output value  $y$ . Specifically, let  $d = (w - 1)/2$  be the "half-width" of the win-

dow. Then  $h_w$  predicts  $y_{i,t}$  using the window  $\langle x_{i,t-d}, x_{i,t-d+1}, \dots, x_{i,t}, \dots, x_{i,t+d-1}, x_{i,t+d} \rangle$ . The window classifier  $h_w$  is trained by converting each sequential training example  $(x_i, y_i)$  into windows and then applying a standard machine learning algorithms. A new sequence  $\mathbf{x}$  is classified by converting it to windows, applying  $h_w$  to predict each  $y_t$  and then concatenating the  $y_t$ 's to form the predicted sequence  $\mathbf{y}$ . The obvious advantage of this sliding window method is that permits any classical supervised learning algorithm to be applied. However, the sliding window method gives adequate performance in many applications, it does not take advantage of correlations between nearby  $y_t$  values. To be more precise, the only relationships between nearby  $y_t$  values that are captured are those that are predictable from nearby  $x_t$  values. If there are correlations among the  $y_t$  values that are independent of the  $x_t$  values, then these are not captured. The sliding window method has been successfully used in a number of machine learning based anomaly detection techniques [21, 39, 121]. Warrender et al. [121] proposed a method that utilized sliding windows to create a database of normal sequences for testing against test instances. Eskin et al. in [39], improved the traditional sliding window method by proposing a modeling methodology that uses dynamic length of a sliding window depending on the context of the system-call sequence.

However, system call based approaches for host based intrusion detection system suffer from two drawbacks. Firstly, the computational overhead that is involved in monitoring every system call is very high. This high overhead leads to a performance degradation of the monitored system. The second problem is that system calls themselves are irregular by nature. This irregularity leads to high false positive rate as it becomes difficult to differentiate between normal and anomalous system calls.

**Bayesian Networks** A Bayesian network is a graphical model that encodes probabilistic relationships among variables of interest. When used in conjunction with statistical techniques, Bayesian networks have several advantages for data analysis [52]. Firstly, because Bayesian networks encode the interdependencies between variables, they can handle situations where data is missing. Secondly, Bayesian networks have the ability to represent causal relationships. Therefore, they can be used to predict the consequences of an action. Lastly, because Bayesian networks have both causal and probabilistic relationships, they can be used to model problems where there is a need to combine prior knowledge with data. Several researchers have adapted ideas from Bayesian statistics to create models for anomaly detection [63, 118, 127]. Valdes et al. [118] developed an anomaly detection system that employed naïve Bayesian<sup>2</sup> networks to perform intrusion detection on traffic bursts. While the model, which is a part of EMERALD [100], has the capability to potentially detect distributed attacks in which none of the attack sessions are individually suspicious enough to generate an alert, it has a few disadvantages. First, as pointed out in [63], the classification capability of naïve Bayesian networks is identical to a threshold based system that computes the sum of the outputs obtained from the child nodes. Secondly, because the child nodes do not interact between themselves and their output only influences the probability of the root node, incorporating additional information becomes difficult as the variables that contain the information cannot directly interact with the child nodes.

Another area, within the domain of anomaly detection, where Bayesian techniques have

---

<sup>2</sup>A naïve Bayesian network is a restricted network that has only two layers and assumes complete independence between the information nodes (i.e., the random variables that can be observed and measured). These limitations result in a tree-shaped network with a single hypothesis node (root node) that has arrows pointing to a number of information nodes (child nodes). All child nodes have exactly one parent node, that is, the root node, and no other causal relationship between nodes are permitted.

been frequently used is the classification and suppression of false alarms. Kruegel et al. [63] proposed a multi-sensor fusion approach where the outputs of different intrusion detection systems sensors were aggregated to produce a single alarm. This approach is based on the assumption that any anomaly detection technique cannot classify a set of events as intrusion with sufficient confidence. Although using the Bayesian for the intrusion detection or intruder behavior prediction can be very appealing, there are some issues that one should be concerned about them. Since the accuracy of this method is dependent on certain assumptions that are typically based on the behavioral model of the target system, deviating from those assumptions will decrease its accuracy. Selecting an inaccurate model will lead to an inaccurate detection system. Therefore, selecting an accurate model is the first step towards solving the problem. Unfortunately selecting an accurate behavioral model is not an easy task as typical systems and/or networks are complex.

**Principal Components Analysis** Typical datasets for intrusion detection are typically very large and multidimensional. With the growth of high speed networks and distributed network based data intensive applications storing, processing, transmitting, visualizing and understanding the data is becoming more complex and expensive. To tackle the problem of high dimensional datasets, researchers have developed a dimensionality reduction technique known as *Principal Component Analysis* (PCA) [17, 57, 119]. In mathematical terms, PCA is a technique where  $n$  correlated random variables are transformed into  $d \leq n$  uncorrelated variables. The uncorrelated variables are linear combinations of the original variables and can be used to express the data in a reduced form. Typically, the first principal component of the transformation is the linear combination of the original variables with the largest variance. In other words, the first principal component is the projection on the direction in which the variance of the projection is maximized. The second principal component is the linear combination of the original variables with the second largest vari-

ance and orthogonal to the first principal component, and so on. In many data sets, the first several principal components contribute most of the variance in the original data set, so that the rest can be disregarded with minimal loss of the variance for dimension reduction of the dataset.

PCA has been widely used in the in the domain of image compression, pattern recognition and intrusion detection. Shyu et al. [110] proposed an anomaly detection scheme, where PCA was used as an outlier detection scheme and was applied to reduce the dimensionality of the audit data and arrive at a classifier that is a function of the principal components. They measured the Mahalanobis distance of each observation from the center of the data for anomaly detection. The Mahalanobis distance is computed based on the sum of squares of the standardized principal component scores. Shyu et al. evaluated there method over the KDD CUP99 data and have demonstrated that it exhibits better detection rate than other well known outlier based anomaly detection algorithms such as the Local Outlier Factor “LOF” approach, the Nearest Neighbor approach and the  $k^{th}$  Nearest Neighbor approach. Other notable techniques that employ the principal component analysis methodology include the work done by Wang et al. [119], Bouzida et al. [12] and Wang et al. [120].

**Markov Models** A hidden Markov model, like the one shown in figure 2.3, is a statistical model where the system being modeled is assumed to be a Markov process with unknown parameters. The challenge is to determine the hidden parameters from the observable parameters. Unlike a regular Markov model, where the state transition probabilities are the only parameters and the state of the state of the system is directly observable, in a hidden Markov model the only visible elements are the variables of the system that are influenced by the state of the system, and the state of the system itself is hidden. A hidden Markov

model's states represent some unobservable condition of the system being modeled. In each state, there is a certain probability of producing any of the observable system outputs and a separate probability indicating the likely next states. By having different output probability distributions in each of the states, and allowing the system to change states over time, the model is capable of representing non-stationary sequences.

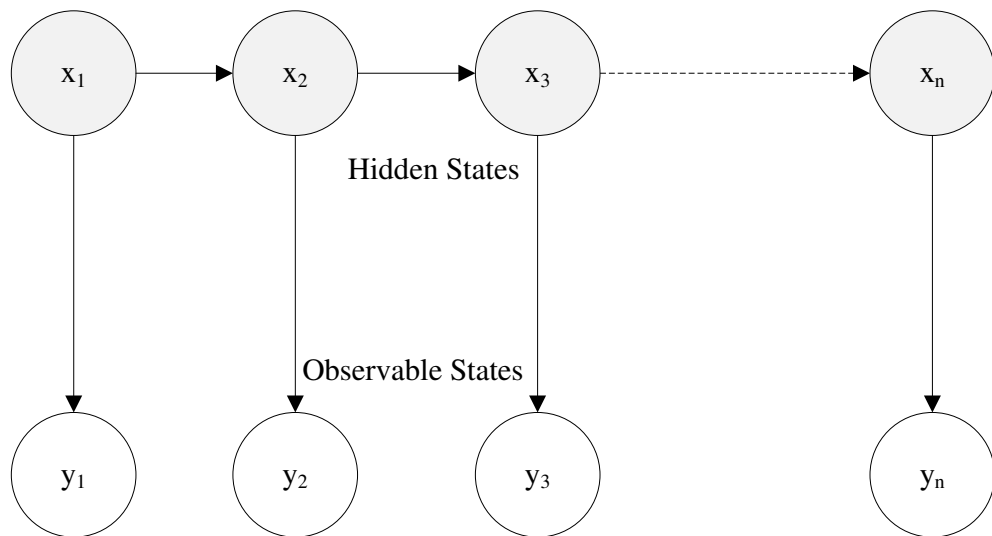


Figure 2.3: Example of a hidden Markov model.

To estimate the parameters of a hidden Markov model for modeling normal system behavior, sequences of normal events collected from normal system operation are used as training data. An expectation-maximization (EM) algorithm is used to estimate the parameters. Once a hidden Markov model has been trained, when confronted with test data, probability measures can be used as thresholds for anomaly detection. In order to use hidden Markov models for anomaly detection, three key problems need to be addressed. The first problem, also known as the *evaluation problem*, is to determine given a sequence of observations, what is the probability that the observed sequence was generated by the

model. The second is the *learning problem* which involves building from the audit data a model, or a set of models, that correctly describes the observed behavior. Given a hidden Markov model and the associated observations, the third problem, also known as the *decoding problem*, involves determining the most likely set of hidden states that have led to those observations.

Warrender et al. [121] compare the performance of four methods viz. simple enumeration of observed sequences, comparison of relative frequencies of different sequences, a rule induction technique, and hidden Markov models at representing normal behavior accurately and recognizing intrusions in system call datasets. The authors show that while hidden Markov models outperform the other three methods, the higher performance comes at a greater computational cost. The training algorithm for hidden Markov models, is very expensive i.e. it runs in time  $O(nm^2)$ , where  $n$  is the number of states in the HMM and  $m$  is the size of the trace. In another paper, Yeung et al. [129] describe the use of hidden Markov model's for anomaly detection based on profiling system call sequences and shell command sequences. On training, their model computes the sample likelihood of an observed sequence using the forward or backward algorithm. A threshold on the probability, based on the minimum likelihood among all training sequences, was used to discriminate between normal and anomalous behavior. One major problem with this approach is that it lacks generalization and/or support for users who are not uniquely identified by the system under consideration.

Markov chains, another popular Markov technique, have also been employed extensively for anomaly detection. Ye et al. [128], present an anomaly detection technique that is based on Markov chains. In their paper, system call events sequences from the recent past were studied by opening an observation window of size  $N$ . The type of each audit events,  $E_{t-N+1}, \dots, E_t$  in the window at time  $t$  was examined and the sequence of

states  $X_{t-N+1}, \dots, X_t$  obtained. Subsequently, the probability that the sequence of states  $X_{t-N+1}, \dots, X_t$  is normal was obtained. The larger the probability, the more likely the sequence of states results from normal activities. A sequence of states from attack activities is presumed to receive a low probability of support from the Markov chain model of the normal profile.

The one major drawback of many of the machine learning techniques, like the system call approach and the hidden Markov model approach mentioned above, is that they are resource expensive. For example, an anomaly detection technique that is based on the Markov chain model is computationally expensive because it uses parametric estimation techniques based on the Bayes algorithm for learning the normal profile of the host/network under consideration. If we consider the large amount of audit data and the relatively high frequency of events that occur in computers and networks of today, such a technique for anomaly detection is not scalable for real time operation. The highlighting features of some of the schemes surveyed in this section are presented in Table 2.2.

### **2.2.2.3 Data Mining based Anomaly Detection**

To eliminate the manual and ad-hoc elements from the process of building an intrusion detection system, researchers are increasingly looking at using data mining techniques for anomaly detection [66, 67, 70]. Grossman [50] defines data mining as being “concerned with uncovering patterns, associations, changes, anomalies, and statistically significant structures and events in data”. Simply put data mining is the ability to take data as input, and pull from it patterns or deviations which may not be seen easily to the naked eye. Another term sometimes used is knowledge discovery. Data mining can help improve the process of intrusion detection by adding a level of focus to anomaly detection. By

identifying bounds for valid network activity, data mining will aid an analyst in his/her ability to distinguish attack activity from common everyday traffic on the network.

**Classification-based Intrusion Detection** An intrusion detection system that classifies audit data as normal or anomalous based on a set of rules, patterns or other affiliated techniques can be broadly defined as a classification-based intrusion detection system. The classification process typically involves the following steps:

- 1 Identify class attributes and classes from training data;
- 2 Identify attributes for classification;
- 3 Learn a model using the training data;
- 4 Use the learned model to classify the unknown data samples.

A variety of classification techniques have been proposed in the literature. These include inductive rule generation techniques, fuzzy logic, genetic algorithms and neural networks-based techniques.

*Inductive rule generation algorithms* typically involve the application of a set of association rules and frequent episode patterns to classify the audit data. In this context, if a rule states that “if event **X** occurs, then event **Y** is likely to occur”, then events **X** and **Y** can be described as sets of (*variable, value*)-pairs where the aim is to find the sets **X** and **Y** such that **X** “implies” **Y**. In the domain of classification, we fix **Y** and attempt to find sets of **X** which are good predictors for the right classification. While supervised classification typically only derives rules relating to a single attribute, general rule induction techniques, which are typically unsupervised in nature, derive rules relating to any or all

the attributes. The advantage of using rules is that they tend to be simple and intuitive, unstructured and less rigid. As the drawbacks they are difficult to maintain, and in some cases, are inadequate to represent many types of information. A number of inductive rule generation algorithms have been proposed in literature. Some of them first construct a decision tree and then extract a set of classification rules from the decision tree<sup>3</sup>. Other algorithms (for eg. RIPPER [21], C4.5 [103]) directly induce rules from the data by employing a divide-and-conquer approach. A post learning stage involving either discarding (C4.5 [103]) or pruning (RIPPER [21]) some of the learnt rules is carried out to increase the classifier accuracy. RIPPER has been successfully used in a number of data mining based anomaly detection algorithms to classify incoming audit data and detect intrusions. One of the primary advantages of using RIPPER is that the generated rules are easy to use and verify. Lee et al. [66, 67, 69] used RIPPER to characterize sequences occurring in normal data by a smaller set of rules that capture the common elements in those sequences. During monitoring, sequences violating those rules are treated as anomalies.

*Fuzzy logic* techniques have been in use in the area of computer and network security since the late 1990's [56]. Fuzzy logic has been used for intrusion detection for two primary reasons [16]. Firstly, several quantitative parameters that are used in the context of intrusion detection, for e.g. CPU usage time, connection interval etc., can potentially be viewed as fuzzy variables. Secondly, as stated by Bridges et al. [16], the concept of security is

---

<sup>3</sup>Decision trees are powerful and popular tools for classification and prediction. The attractiveness of tree-based methods is due in large part to the fact that, in contrast to neural networks, decision trees represent rules. A decision tree is a tree that has three main components: nodes, arcs, and leaves. Each node is labeled with a feature attribute which is most informative among the attributes not yet considered in the path from the root, each arc out of a node is labeled with a feature value for the node's feature and each leaf is labeled with a category or class. A decision tree can then be used to classify a data point by starting at the root of the tree and moving through it until a leaf node is reached. The leaf node would then provide the classification of the data point.

fuzzy in itself. In other words, the concept of fuzziness helps to smooth out the abrupt separation of normal behavior from abnormal behavior. That is, a given data point falling outside/inside a defined “normal interval”, will be considered anomalous/normal to the same degree regardless of its distance from/within the interval. Dickerson et al., [29] developed the Fuzzy Intrusion Recognition Engine (FIRE) using fuzzy sets and fuzzy rules. FIRE uses simple data mining techniques to process the network input data and generate fuzzy sets for every observed feature. The fuzzy sets are then used to define fuzzy rules to detect individual attacks. FIRE does not establish any sort of model representing the current state of the system, but instead relies on attack specific rules for detection. Instead, FIRE creates and applies fuzzy logic rules to the audit data to classify it as normal or anomalous. Dickerson et al. found that the approach is particularly effective against port scans and probes. The primary disadvantage to this approach is the labor intensive rule generation process.

*Genetic algorithms*, a search technique used to find approximate solutions to optimization and search problems, have also been extensively employed in the domain of intrusion detection to differentiate normal network traffic from anomalous connections. The major advantage of genetic algorithms is their flexibility and robustness as a global search method. In addition, a genetic algorithm search converges to a solution from multiple directions and is based on probabilistic rules instead of deterministic ones. In the domain of network intrusion detection, genetic algorithms have been used in a number of ways. Some approaches [74, 99] have used genetic algorithms directly to derive classification rules, while others [16, 48] use genetic algorithms to select appropriate features or determine optimal parameters of related functions, while different data mining techniques are then used to acquire the rules. The earliest attempt to apply genetic algorithms to the problem of intrusion detection was done by Crosbie et al. [24] in 1995, when they applied multiple agent technology to detect network based anomalies. While the advantage of the

approach was that it used numerous agents to monitor a variety of network based parameters, lack of intra-agent communication and a lengthy training process were some issues that were not addressed.

*Neural network* based intrusion detection systems have traditionally been host based systems that focus on detecting deviations in program behavior as a sign of an anomaly. In the neural network approach to intrusion detection, the neural network learns to predict the behavior of the various users and daemons in the system. The main advantage of neural networks is their tolerance to imprecise data and uncertain information and their ability to infer solutions from data without having prior knowledge of the regularities in the data. This in combination with their ability to generalize from learned data has shown made them an appropriate approach to intrusion detection. However, the neural network based solutions have several drawbacks. First they may fail to find a satisfactory solution either because of lack of sufficient data or because there is no learnable function. Secondly, neural networks can be slow and expensive to train. The lack of speed is partly because of the need to collect and analyze the training data and partly because the neural network has to manipulate the weights of the individual neurons to arrive at the correct solution. There are a few different groups advocating various approaches to using neural networks for intrusion detection. Ghosh et al. [45–47] used the feed-forward back propagation and the Elman recurrent network [34] for classifying system-call sequences. Their experimental results with the 1998 and 1999 DARPA intrusion detection evaluation dataset verified that the Elman networks were superior to the standard multilayer perceptron based neural networks in program-based intrusion detection. However, training the Elman network was expensive and the number of neural networks required was large. In another paper, Ramadas et al. [104] present the Anomalous Network-Traffic Detection with Self Organizing Maps (ANDSOM). ANDSOM is the anomaly detection module for the network based intrusion detection system, called INBOUNDS, being developed at Ohio University. The

ANDSOM module creates a two dimensional Self Organizing Map or SOM<sup>4</sup> for every network service (in the paper they test for DNS and HTTP traffic) that is being monitored. Neurons are trained with normal network traffic during the training phase to capture characteristic patterns. When real time data is fed to the trained neurons, then an anomaly is detected if the distance of the incoming traffic is more than a preset threshold.

Anomaly detection schemes also involve other data mining techniques such as support vector machines (SVM) and other types of neural network models [68, 115]. Because data mining techniques are data driven and do not depend on previously observed patterns of network/system activity, some of these techniques have been very successful at detecting new kinds of attacks. However, these techniques often have a very high false positive rate. For example, as pointed out in [106], the approach adopted by Sung et al. [114] that use a SVM technique to realize an intrusion detection system for class-specific detection is flawed because they totally ignore the relationships and dependencies between the features.

**Clustering and Outlier Detection** Clustering is a technique for finding patterns in unlabeled data with many dimensions<sup>5</sup>. Clustering has attracted interest from researchers in the context of intrusion detection [101, 105, 108]. The main advantage that clustering provides is the ability to learn from and detect intrusions in the audit data, while not requiring the system administrator to provide explicit descriptions of various attack classes/types. As a result, the amount of training data that needs to be provided to the anomaly detection system is also reduced. Clustering and outlier detection are closely related. From the viewpoint of a clustering algorithm, outliers are objects not located in the clusters of a data set,

---

<sup>4</sup>A self organizing map is a method for unsupervised learning based on a grid of artificial neurons whose weights are adapted to match input vectors in a training set.

<sup>5</sup>The number of dimensions is equivalent to the number of attributes.

and in the context of anomaly detection, they may represent intrusions/attacks. The statistics community has studied the concept of outliers quite extensively [9]. In these studies, data points are modeled using a stochastic distribution and points are determined to be outliers based on their relationship with this model. However, with increasing dimensionality, it becomes increasingly difficult to accurately estimate the multidimensional distributions of the data points [1]. Recent outlier detection algorithms [15,61,105] are based on the full dimensional distances between the points as well as the densities of local neighborhoods.

There exist at least two approaches to clustering based anomaly detection. In the first approach, the anomaly detection model is trained using unlabelled data that consists of both normal as well as attack traffic. In the second approach, the model is trained using only normal data and a profile of normal activity is created. The idea behind the first approach is that anomalous or attack data forms a small percentage of the total data. If this assumption holds, anomalies and attacks can be detected based on cluster sizes-large clusters correspond to normal data, and the rest of the data points, which are outliers, correspond to attacks.

The distances between points play an important role in clustering. The most popular distance metric is the Euclidean distance. For a distance between two  $n$ -dimensional observations  $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$  and  $\mathbf{y} = (y_1, y_2, \dots, y_n)^T$ , a Euclidean metric is defined as:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y})}.$$

Since each feature contributes equally to the calculation of the Euclidean distance, this distance is undesirable in many applications. This is especially true when features have very different variability or different features are measured on different scales. The effect of the features that have large scales of measurement or high variability would dominate

others that have smaller scales or less variability. A distance-based approach which incorporates this measure of variability is the *Mahalanobis distance*<sup>6</sup> [79, 123] based outlier detection scheme. In this scheme, the threshold is computed according to the most distant points from the mean of the “normal” data, and it is set to be a user defined percentage, say  $m\%$ , of the total number of points. In this scheme, all data points in the audit data that have distances to the mean (calculated during the training phase) greater than the threshold are detected as outliers. The Mahalanobis distance utilizes group means and variances for each variable, and the correlations and covariances between measures.

The notion of distance-based outliers was recently introduced in the study of databases [105]. According to this notion, a point,  $P$ , in a multidimensional data set is an outlier if there are less than  $p$  points from the data in the  $\varepsilon$ -neighborhood of  $P$ , where  $p$  is a user-specified constant. In [105], Ramaswamy et al. described an approach that is based on computing the Euclidean distance of the  $k^{\text{th}}$  nearest neighbor from a point  $O$ . In other words, the  $k$ -nearest neighbor algorithm classifies points by assigning them to the class that appears most frequently amongst the  $k$  nearest neighbors. Therefore, for a given point  $O$ ,  $d_k(O)$  denotes the Euclidean distance from the point  $O$  to its  $k^{\text{th}}$  nearest neighbor and can be considered as the “*degree of outlierness*” of  $O$ . If one is interested in the top  $n$  outliers, this approach defines an outlier as follows: Given values for  $k$  and  $n$ , a point  $O$  is an outlier, if the distance to its nearest neighbor is smaller than the corresponding value for no more than  $(n - 1)$  other points. In other words, the top  $n$  outliers with the maximum  $D_k(O)$  values are considered as outliers. While the advantage of the  $k$ -nearest neighbors

---

<sup>6</sup>The basic Euclidian distance treats each variable as equally important in calculating the distance. An alternative approach is to scale the contribution of individual variables to the distance value according to the variability of each variable. This approach is illustrated by the Mahalanobis distance which is a measure of the distance between each observation in a multidimensional cloud of points and the centroid of the cloud. In other words, the Mahalanobis distance between a particular point  $x$  and the mean  $\mu$  of the “normal data” is computed as:  $D_M(x) = \sqrt{(x - \mu)^T \Sigma^{-1} (x - \mu)}$ , where  $\Sigma$  is the covariance matrix.

approach is that it is robust to noisy data, the approach suffers from the drawback that it is very difficult to choose an optimal value for  $k$  in practice. Several papers [51, 75] have proposed using the  $k$ -nearest neighbor outlier detection algorithms for the purpose of anomaly detection.

The Minnesota Intrusion Detection System (MINDS) [36] is another network-based anomaly detection approach that utilizes data mining techniques. The MINDS anomaly detection module assigns a degree of outlierness to each data point, which is called the local outlier factor (LOF) [15]. The LOF takes into consideration the density of the neighborhood around the observation point to determine its outlierness. In this scheme, outliers are objects that tend to have high LOF values. The advantage of the LOF algorithm is its ability to detect all forms of outliers, including those that cannot be detected by the distance-based algorithms.

**Association Rule Discovery** Association rules [2, 54] are one of many data mining techniques that describe events that tend to occur together. The concept of association rules can be understood as follows: Given a database  $D$  of transactions where each transaction  $T \in D$  denotes a set of items in the database, an *association rule* is an implication of the form  $X \Rightarrow Y$ , where  $X \subset D$ ,  $Y \subset D$  and  $X \cap Y = \emptyset$ . The rule  $X \Rightarrow Y$  holds in the transaction set  $D$  with confidence  $c$  if  $c\%$  of transactions in  $X$  also contain  $Y$ . Two important concepts when dealing with association rules are *rule confidence* and *rule support*. The probability of rule confidence is defined as the conditional probability  $P(Y \subseteq T | X \subseteq T)$ . The rule  $X \Rightarrow Y$  has support  $s$  in the transaction database  $D$  if  $s\%$  of transactions in  $D$  contain  $X \cup Y$ . Association rules have been successfully used to mine audit data to find normal patterns for anomaly detection [8, 67, 69]. They are particularly important in the domain of anomaly detection because association rules can be used to construct a summary

of anomalous connections detected by the intrusion detection system. There is evidence that suggests program executions and user activities exhibit frequent correlations among system features. These consistent behaviors can be captured in association rules.

Lee et al. [67, 69] proposed an association rule-based data mining approach for anomaly detection where raw data was converted into ASCII network packet information, which in turn was converted into connection-level information. These connection level records contained connection features like service, duration etc. Association rules were then applied to this data to create models to detect intrusions. In another paper, Barbará et al. [8] describe ADAM (Audit Data Analysis and Mining), a real-time anomaly detection system that uses a module to classify the suspicious events into false alarms or real attacks. ADAM's training and intrusion detection phases are illustrated in figures 2.4 and 2.5, respectively. ADAM was one out of seven systems tested in the 1999 DARPA evaluation [76]. It uses data mining to build a customizable profile of rules of normal behavior and then classifies attacks (by name) or declares false alarms. To discover attacks in TCPdump audit trail, ADAM uses a combination of association rules, mining and classification. During the training phase, ADAM builds a database of "normal" frequent itemsets using attack free data. Then it runs a sliding window online algorithm that finds frequent item sets in the last connections and compares them with those stored in the normal item set repository. With the remaining item sets that have deemed suspicious, ADAM uses a classifier which has previously been trained to classify the suspicious connections as a known attack, unknown attack, or a false alarm. Association rules are used to gather necessary knowledge about the nature of the audit data. If the item set's support surpasses a threshold, then that item set is reported as suspicious. The system annotates suspicious item sets with a vector of parameters. Since the system knows where the attacks are in the training set, the corresponding suspicious item set along with their feature vectors are used to train a classifier. The trained classifier will be able to, given a suspicious item set and a vector of features,

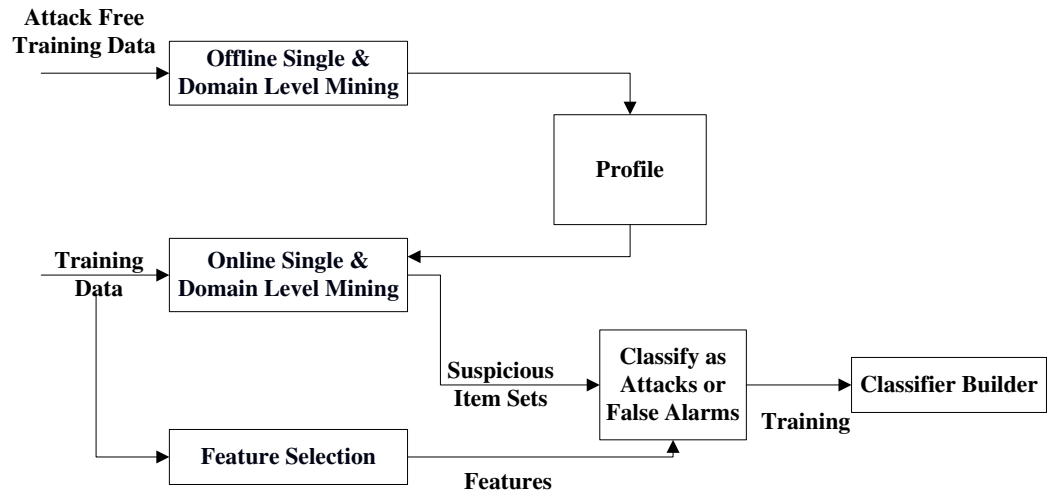


Figure 2.4: The training phase of ADAM [8]

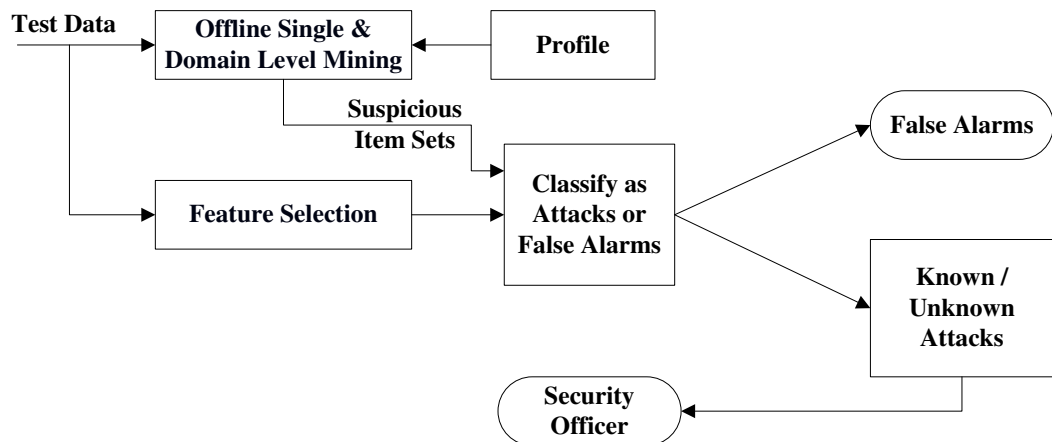


Figure 2.5: The intrusion detection phase of ADAM [8]

classify the item set as a known attack (and label it with the name of attack), an unknown attack, or a false alarm. The highlighting features of some of the schemes surveyed in this section are presented in Table 2.3.

## 2.3 Hybrid Systems

It has been suggested in the literature [3, 78, 100, 117, 131] that the monitoring capability of current intrusion detection systems can be improved by taking a hybrid approach that consists of both anomaly as well as signature detection strategies. In such a hybrid system, the anomaly detection technique aids in the detection of new or unknown attacks while the signature detection technique detects known attacks. The signature detection technique will also be able to detect attacks launched by a patient attacker who attempts to change the behavior patterns with the objective of retraining the anomaly detection module so that it will accept attack behavior as normal. Tombini et al. [117] used an approach wherein the anomaly detection technique is used to produce a list of suspicious items. The classifier module which uses a signature detection technique then classified the suspicious items into false alarms, attacks, and unknown attacks. This approach works on the premise that the anomaly detection component would have a high detection rate, since missed intrusions cannot be detected by the follow-up signature detection component. In addition it also assumed that the signature detection component will be able to identify false alarms. While the hybrid system can still miss certain types of attacks, its reduced false alarm rate increases the likelihood of examining most of the alerts.

EMERALD [100] was developed in the late 1990's at SRI. It is an extension of the seminal work done in [3, 28, 78]. EMERALD is a hierarchical intrusion detection system that moni-

tors systems at a variety of levels viz. individual host machines, domains and enterprises to form an analysis hierarchy. EMERALD uses a subscription-based communication scheme both within and between monitors. However, inter monitor subscription methodology is hierarchical and therefore limits the access to events and/or results from the layer immediately below. The system has a built-in feedback system that enables the higher layers to request more information about particular anomalies from the lower layers. To achieve a high rate of detection, the architects of EMERALD employed an ensemble of techniques like statistical analysis engines and expert systems. The single most defining feature of EMERALD is its ability to analyze system-wide, domain-wide and enterprise-wide attacks like Internet worms, DDoS attacks etc. at the top level.

In another paper [131], Zhang et al. employed the random forests algorithm [14] in the signature detection module to detect known intrusions. Thereafter, the outlier detection provided by the random forests algorithm is utilized to detect unknown intrusions. Approaches that use signature detection and anomaly detection in parallel have also been proposed. In such systems, two sets of reports of possible intrusive activity are produced and a correlation component analyzes both sets to detect intrusions. An example of such a system is NIDES [4, 78].

Lee et al. [68] extended the work done by them in [66] and proposed a hybrid detection scheme that utilized the Common Intrusion Detection Framework (CIDF) to automatically get audit data, build models, and distribute signatures for novel attacks to ensure that the time required to detect them is reduced. The advantage of using CIDF was that it enabled different intrusion detection and response components to interoperate and share the information and resources in a distributed environment.

Although it is true that combining multiple intrusion detection technologies into a single

system can theoretically produce a much stronger intrusion detection system, the resulting hybrid systems are not always better. Different intrusion detection technologies examine system and/or network traffic and look for intrusive activity in different ways. Therefore, the major challenge to building an operational hybrid intrusion detection system is getting these different technologies to interoperate effectively and efficiently.

## 2.4 Summary

In the last twenty years, intrusion detection systems have slowly evolved from host- and operating system-specific applications to distributed systems that involve a wide array of operating systems. The most important challenge that lies ahead for the next generation of intrusion detection systems and, more specifically, for anomaly detection systems will be their ability to adapt to new network paradigms like wireless and mobile networks, and, also be scalable to meet the requirements of high-speed (gigabit and terabit) networks. As mentioned above, factors like noise in the audit data, constantly changing traffic profiles, and the large amount of network traffic make it difficult to build a normal traffic profile of a network for the purpose of intrusion detection. The implication is that, short of some fundamental redesign, today's intrusion detection approaches will not be able to adequately protect tomorrow's networks against intrusions and attacks. Therefore, the primary challenge that needs to be addressed is designing next-generation intrusion detection systems that have the ability to

- manage and analyze the large quantity of audit data in real time or near real time in high speed networks; and
- track users as they move throughout the various access points in a wireless network.

The following chapters, Chapter 3 and Chapter 4, discuss a proposed solution to these problems, developed through this research. A detailed system design of an anomaly detection scheme, that involves an adaptive sampling scheme and a predictive clustering based anomaly detection scheme, is presented as a component of a networks security mechanism.

Table 2.1: A summary of statistical anomaly detection systems

| <b>Reference</b> | <b>Highlighting Feature</b>                                                                                                                                | <b>Methodology</b>                          |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| Haystack [111]   | It uses descriptive statistics to model user behavior. Also modeled acceptable behavior for a generic user within a particular user group.                 | Host based statistical anomaly detection    |
| NIDES [3]        | A distributed intrusion detection system that had both anomaly as well as signature detection modules.                                                     | Network based statistical anomaly detection |
| Ye et al. [126]  | It uses the Hotellings $T^2$ test to analyze the audit trails of activities in an computer system and detect host-based intrusions.                        | Host based statistical anomaly detection    |
| PHAD [80]        | Examines IP headers, connections to various ports as well as packet headers of Ethernet, IP and transport layer and other transport layer packet headers.  | Network based statistical anomaly detection |
| ALAD [82]        | It detects anomalies in inbound TCP connections to well known ports on the server.                                                                         | Network based statistical anomaly detection |
| LERAD [81]       | It detects TCP stream anomalies like ALAD, but uses a learning algorithm to pick good rules from the training set, rather than using a fixed set of rules. | Network based statistical anomaly detection |

Table 2.2: A summary of machine learning based anomaly detection systems.

| <b>Reference</b>    | <b>Highlighting Feature</b>                                                                                                                                                                                                                                       | <b>Methodology</b>                                    |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------|
| Forrest et al. [42] | It finds correlations in fixed length sequences of system calls in the UNIX operating system, and uses them to build a normal profile for anomaly detection.                                                                                                      | Sequence analysis based statistical anomaly detection |
| Eskin et al. [39]   | It employs dynamic window sizes to improve system call modeling.                                                                                                                                                                                                  | System call modeling and sequence analysis.           |
| Valdes et al. [118] | It employs Bayesian inference techniques to determine if traffic bursts contain attacks. It also provides distributed attack detection capability.                                                                                                                | Bayesian networks based anomaly detection             |
| Shyu et al. [110]   | It uses principal component analysis as an outlier detection scheme to detect intrusions.                                                                                                                                                                         | Principal component analysis based anomaly detection  |
| Yeung et al. [129]  | It presents a dynamic modeling approach based on hidden Markov models and a static modeling approach based on event occurrence frequency distribution for modeling system calls. It showed that the dynamic modeling approach is better for system call datasets. | Host based anomaly detection system                   |

Table 2.3: A summary of data mining-based anomaly detection systems.

| <b>Reference</b> | <b>Highlighting Feature</b>                                                                                                                                                             | <b>Methodology</b>                                           |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------|
| Lee et al. [67]  | It uses inductive rule generation to generate rules for important, yet infrequent events.                                                                                               | Classification based anomaly detection                       |
| FIRE [29]        | It generates fuzzy sets for every observed feature which are in turn used to define fuzzy rules to detect individual attacks.                                                           | Classification based anomaly detection                       |
| ANDSOM [104]     | It uses a pre-processor to summarize certain connection parameters (source and destination host and port) and then adds several values to track and classify the connection's behavior. | Classification based anomaly detection                       |
| MINDS [36]       | It clusters data and uses the density-based local outliers to detect intrusions.                                                                                                        | Classification based anomaly detection                       |
| ADAM [8]         | It performs anomaly detection to filter out most of the normal traffic, then it uses a classification technique to determine the exact nature of the remaining activity.                | Association rules and classification based anomaly detection |

## **Chapter 3**

# **Adaptive Sampling for Anomaly Detection**

### **3.1 Motivation**

The Internet today continues to grow and evolve as a global infrastructure for new services. Business needs have dictated that corporations and governments across the globe should develop sophisticated, complex information networks, incorporating technologies as diverse as distributed data storage systems, encryption and authentication mechanisms, voice and video over IP, remote and wireless access, and web services. As a result, Internet service providers and network managers in corporate networks are being motivated to gain a deeper understanding of the network behavior through monitoring and measurement of the network traffic flowing through their networks.

However, the development of enabling technologies for network-based security systems, like intrusion detection systems (IDS), have not kept pace with the increasing usage of high-speed networking technologies such as Gigabit Ethernet. The repeated occurrences of large-scale attacks (such as distributed denial-of-service (DDoS) attacks and worms) that exploit the bandwidth and connectivity of networks made possible by such technologies are a case in point.

The single biggest reason that can be attributed to the incapability of current solutions to detect intrusions in high-speed networks is the prohibitively high cost of using traditional network monitoring schemes like host and/or router based monitoring solutions. These schemes typically measure network parameters based on every packet that passes through a network based monitoring device. This approach has the drawback that it becomes extremely difficult to monitor the behavior of a large number of sessions in high-speed networks. In other words, traditional network monitoring schemes are not *scalable* to high-speed networks.

To alleviate the aforementioned problem, sampling algorithms have been proposed. Over the years, network managers have predominantly relied on static sampling algorithms for network monitoring and management. In general, these sampling algorithms employ a strategy where the samples are taken either randomly or periodically at some fixed interval. The major advantage of using a sampling algorithm is that it reduces bandwidth and storage requirements. Traditional sampling algorithms typically use a static or fixed rule to determine when to sample the next data. Static sampling of network traffic was first proposed by Claffy et al. [20] in the early 1990's for traffic measurement on the NSFNET backbone. In their much cited paper, Claffy et al. describe the application of event and timed-based sampling for network traffic measurement.

Static sampling algorithms like *simple random sampling* employ a random distribution function to determine when each sample should be taken. The distribution may be uniform, exponential, Poisson, etc. In random sampling, all items have some chance of selection that can be calculated. The advantage of utilizing a random sampling algorithm is that it ensures that bias is not introduced regarding which entity is included in the sampled population. In other words, with simple random sampling, each item in a population has an equal chance of inclusion in the sample.

However, given the dynamic nature of network traffic, static sampling does not always ensure the accuracy of estimation, and tends to over sample at peak periods when efficiency and timeliness are most critical. More generally, static random sampling algorithms do not take into account traffic dynamics. As a result, they cannot guarantee that the sampling error in each block falls within a prescribed error tolerance level.

In the commercial world, *NetFlow* [86] is a widely deployed general purpose measurement feature of Cisco and Juniper routers. The volume of data produced by NetFlow is a problem in itself. To handle the volume and traffic diversity of high speed backbone links, InMon Inc. developed *SFlow*, a plugin for NetFlow, that resorts to 1 in  $N$  packet sampling. The sampling rate is a configuration parameter that is set manually and is seldom adjusted. Setting it too low, causes inaccurate measurement results; setting it too high, can result in the measurement module using too much memory and processing power, especially when faced with increased volume of traffic or unusual traffic patterns.

Under dynamic traffic load conditions, simple periodic sampling may be poorly suited for network monitoring. During periods of idle activity or low network loads, a long sampling interval provides sufficient accuracy at a minimal overhead. However, bursts of high activity require shorter sampling intervals to accurately measure network status

at the expense of increased sampling overhead. To address this issue, *adaptive sampling* algorithms have been proposed to dynamically adjust the sampling interval and optimize accuracy and overhead.

In this chapter we put forth an adaptive sampling algorithm that is based on *weighted least squares prediction*. The proposed sampling algorithm uses previous samples to estimate or predict a future measurement. The algorithm is used in conjunction with a set of rules which defines the sampling rate adjustments to make when the prediction is inaccurate.

With the aforementioned expansive growth in the domain of Internet and networking technologies, combined with the vulnerability of present day networks and the better equipped hacker/attacker, it comes as no surprise that network based attacks are on the rise [84]. However, as pointed out above, current security mechanisms especially in the domain of attack detection have not scaled to handle the higher network throughputs.

Several approaches for either sampling or attack detection have been proposed in the research community. However, to the best of our knowledge, none of the proposed algorithms for network traffic sampling have taken an approach that is tailored to meet the needs of attack detection. From this perspective, in this chapter, we attempt to answer one key question: Is it possible to design a low cost packet sampling algorithm that will enable accurate characterization of the IP traffic variability for the purpose of detecting DoS attacks in high throughput networks?

We believe that the proposed sampling algorithm is tailored to enhance the capability of network-based IDS at detecting a DoS attack. The proposed sampling algorithm, that would ideally precede the IDS and sample the incoming network traffic for it to analyze, has the key characteristic that not only does it adaptively reduce the volume of data that

would be analyzed by the network IDS, but it also preserves the intrinsic self-similar characteristic of network traffic. We believe the latter characteristic of the proposed sampling algorithm can be used by an IDS to detect traffic intensive DoS attacks by leveraging the fact that a significant change in the self-similarity (See Section 3.3.2 for details on self-similarity) of network traffic is a known indicator of a DoS attack [73,91].

The remainder of this chapter is organized as follows. In Section 3.2, we review the related work in the area of packet sampling, while Section 3.3 discusses the properties of network traffic and reviews the principles of self-similarity. Section 3.3 is followed by Section 3.4 where we present the weighted least square predictor and the proposed adaptive weighted sampling algorithm. Section 3.5 concludes the chapter by summarizing the chapter's contributions and suggesting possible areas for the application of the proposed sampling algorithm.

## **3.2 Related Work**

The biggest challenge in employing a sampling algorithm on a given network is scalability. The increasing deployment of high-speed networks, the inherently bursty nature of Internet traffic, and the storage requirements of large volume of sampled traffic have a major impact on the scalability of a sampling algorithm. In the context of packet sampling, this implies that either the selected sampling strategy should take into account the trends in network traffic or the selected sampling algorithm should sample most if not all the packets that are flowing through the network. The major impediment towards adopting the latter approach is that a higher sampling rate would imply greater memory and space requirements for the sampling device. In addition, a higher sampling rate would run the risk of not being

scalable to high-speed networks.

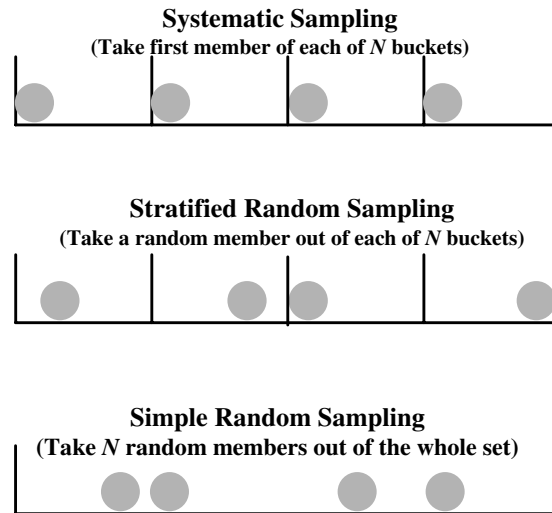


Figure 3.1: Static sampling techniques [20].

Packet sampling has been previously proposed for a variety of objectives in the domain of computer networking. Sampling network traffic was advocated as early as 1994. As mentioned above, Claffy et al. [20] compared three different sampling strategies (See figure 3.1) to reduce the load on the network parameter measurement infrastructure on the NSFNET backbone. The three algorithms studied in [20] were, systematic sampling (deterministically taking one in every  $N$  packets), stratified random sampling (taking one packet in every bucket of size  $N$ ), and simple random sampling (randomly taking  $N$  packets out of the whole set). The results showed that event-driven algorithms were more accurate than time-driven ones, while the differences within each class were small. This was attributed to trends in network traffic.

Drobisz et al. in [30] proposed an rate adaptive sampling algorithm to optimize the resource usage in routers. In their paper, Dobisz et al. propose using packet inter-arrival rates and/or CPU usage as the two methodologies to control resource usage and vary the

sampling rate. They also showed that adaptive algorithms produced more accurate estimates than static sampling under a given resource constraint. In [23], Cozzani et al. used the simple random sampling algorithm to evaluate the ATM end-to-end delays. In the SRED scheme in [90], Ott et al. use packet sampling to estimate the number of active TCP flows in order to stabilize network buffer occupancy for TCP traffic. The advantage of this scheme is that only packet headers need to be examined. The scheme proposed in [92], also uses packet sampling and it is used for fair link-bandwidth allocation.

Another approach taken by Estan and Varghese in [40], involved a random sampling algorithm to identify large flows. In the algorithm, proposed in [40], the sampling probability is determined according to the inspected packet size. In another study, Cheng et al. [18] proposed a random sampling scheme based on Poisson sampling to select a sample that is representative of the whole dataset. They contend that using Poisson sampling is better as it does not require the packet arrival to conform to a particular stochastic distribution. Sampling strategies were also used in [58] for the detection of DoS attacks. Sampling has also been proposed to infer network traffic and routing characteristics [33]. In [31], Duffield et al. focused on the issue of reducing the bandwidth needed for transmitting traffic measurements to a remote server for later analysis, and devised a size-dependent flow sampling algorithm. In another paper, Duffield et al. [32] investigated the consequences of collecting packet sampled flow statistics. They found that flows in the original stream whose length is greater than the sampling period tend to give rise to multiple flow reports when the packet inter arrival time in the sampled stream exceeds the flow timeout.

Sampling for intrusion detection entails a more thorough examination of the sampled packets. In addition, unlike some of the sampling applications mentioned above, sampling for intrusion detection and more specifically for anomaly detection requires near line-speed packet examination. This is especially because a *store-and-process* approach towards sam-

pled packets or packet-headers for off-line analysis is not sufficient to prevent intruders. Hence, in the design of an intrusion detection algorithm, sampling costs are of paramount importance.

### **3.3 Self-Similarity and Network Traffic**

In the last decade, most of the studies on network traffic argued convincingly that Internet traffic is very far from being regular, and presents large variations in its throughput at all scales [95]. These studies have shown that Internet traffic exhibits characteristics such as self-similarity [96], multi-fractality [41], and long-range dependence [35], which implies that in all cases network traffic can vary significantly. In addition, given the highly variable nature of Internet traffic, anomaly based intrusion detection systems are raising alarms for many disruptions that are not attacks. The high rate of false positives is one of the major shortcomings of current IDS and the current evolution of Internet traffic with larger and larger variations among time continues to limit the efficiency of anomaly based IDS.

#### **3.3.1 Properties of Network Traffic**

A network intrusion detection must distinguish between hostile and benign traffic, and must do so quickly to keep up with a high speed network. Depending on whether the intrusion detection system uses signature or anomaly detection, it must either model attacks (of which there are thousands) or normal traffic. There are two main challenges for modeling normal traffic for anomaly detection. First, network traffic is very complex and unpredictable, and second, the model changes over time.

Bellovin [10] and Paxson [97] found that wide area network traffic contains a wide range of anomalies and uncharacteristic data that cannot be easily explained for. Examples include private IP addresses, storms of packets routed in a loop until their TTLs expire, TCP acknowledgments of packets never sent, TCP retransmissions with inconsistent payloads, SYN packets with urgent data, and so on. Bellovin found broadcast packets (255.255.255.255) from foreign sites, ICMP packets with invalid code fields, and packets addressed to nonexistent hosts and ports. Many of these were investigated and found to be not hostile. Instead, many of the aforementioned errors were caused by mis-configured routers and/or DNS servers.

As a result, a confounding problem that has been faced by researchers in the field of Internet traffic modeling, is that it is not possible to determine the *average rate* of many types of events (for example, bytes per second or packets per second for some packet type), regardless of the duration of the sampling period. The long held paradigm in the communication and performance analysis communities has been that voice traffic and, by extension, data traffic could be modeled as a Markovian (e.g. Poisson) process, which could then be accurately analyzed and efficiently controlled. Therefore, if data traffic were to follow Markovian arrival process, it would have a characteristic burst length which would tend to be smoothed by averaging over a long enough time scale. However, in the mid nineties, it was discovered that this was not the case. In a seminal paper, Leland et al. [72] discovered that measurements on real Internet traffic traces indicated that a significant traffic variance (burstiness) is present on a wide range of time scales. It was found that if we graphed packets per second or packets per month, they would both show bursts of high traffic rates separated by gaps of low activity. Furthermore, both graphs would look the same. A burst or gap could last for a fraction of a second or for months. The distribution of traffic rates would be independent of time scale. This behavior was akin to the behavior displayed by *self-similar* or *fractal* processes.

### 3.3.2 Self-Similarity and the Hurst Parameter

Self-similarity, a term borrowed from fractal theory, implies that an object (in our case network traffic) appears the same regardless of the scale at which it is viewed. In a seminal paper published in 1994, Leland et al. [72] showed that the traffic captured from corporate networks as well as the Internet exhibits *self-similar* behavior. Prior to the publication of [72], network traffic was assumed to be Poisson in nature. However, modeling network traffic using the Poisson distribution implied that it would have a characteristic burst length which would tend to be smoothed by averaging over a long enough time scale. This was in contrast to the measured values, which indicated that there was a significant *burstiness* in network traffic over a wide range of time intervals.

The self-similar nature of network traffic can be explained by assuming that network workloads are described by a power-law distribution; e.g., file sizes, web object sizes, transfer times, and even users think times have heavy-tailed distributions which decay according to a power-law distribution. A possible explanation for the self-similar nature of Internet traffic was given in [25], where the authors suggest that many *ON/OFF* sources with heavy-tailed *ON* and/or *OFF* periods resulting in core network traffic to be self-similar. The main properties of self-similar processes include *slowly decaying variance* and *long-range dependence*. An important parameter of a self-similar process is the *Hurst parameter*,  $H$ , that can be estimated from the variance of a statistical process. Self-similarity is implied if  $0.5 < H < 1$ .

The Hurst parameter is defined as follows: For a given set of observations  $X_1, X_2, \dots, X_n$  with sample mean,  $M_n$  defined as  $(1/n) \sum_j X_j$ , adjusted range  $R(n)$  and sample variance  $S^2$ , the rescaled adjusted range or the *R/S* statistic is given by

$$\frac{R(n)}{S(n)} = \frac{1}{S(n)} \cdot A, \quad (3.1)$$

where

$$A = \left( \text{Max} \sum_{j=1}^k (X_j - M_n) - \text{Min} \sum_{j=1}^k (X_j - M_n) \right).$$

Hurst discovered that many naturally occurring time series are well represented by the relation

$$E \left[ \frac{R(n)}{S(n)} \right] \sim cn^H, \text{ as } n \rightarrow \infty \quad (3.2)$$

with the Hurst parameter  $H$  normally around 0.73, and a finite positive constant,  $c$ , independent of  $n$ . On the other hand, if the  $X_k$ 's are Gaussian pure noise or *short range dependent*, then  $H = 0.5$  in equation (3.2).

Li, et al. [73], demonstrated mathematically that a significant change in the Hurst parameter can be used to detect a DoS attack, but their algorithm requires an accurate baseline model of the normal (non-attack) traffic. In another paper, Xiang et al. [125] contend that DDoS attacks can be detected by adopting a modified version of the rescaled range statistic.

### 3.4 Proposed Sampling Algorithm

Traffic measurement and monitoring serves as the basis for a wide range of IP network operations and engineering tasks such as trouble shooting, accounting and usage profiling, routing weight configuration, load balancing, capacity planning, etc. Traditionally, traffic measurement and monitoring is done by capturing every packet traversing a router interface or a link. With today's high-speed (e.g., Gigabit or Terabit) links, such an approach is no longer feasible due to the excessive overheads it incurs on line-cards or routers. As a result, packet sampling has been suggested as a scalable alternative to address this problem.

Early packet sampling algorithms assumed that the rate of arrival of packets in a network would average out in the long term. However, it has been shown [93] that network traffic exhibits periodic cycles or trends. The main observation of [93] and other studies have been that not only does network traffic exhibit strong trends in the audit data but these trends also tend to be long term.

This section presents the proposed sampling algorithm. In Section 3.4.1, we describe the *weighted least squares* predictor that is utilized for predicting the next sampling interval. This predictor has been adopted because of its capability to follow the trends in network traffic. Thereafter, in Section 3.4.2 we describe the sampling algorithm itself.

### 3.4.1 Weighted Least Square Predictor

Let us assume that the vector  $\mathbf{Z}$  holds the values of the  $N$  previous samples, such that  $Z_N$  is the most recent sample and  $Z_1$  is the oldest sample. Having fixed a window size of  $N$ , when the next sampling occurs, the vector is right shifted such that  $Z_N$  replaces  $Z_{N-1}$  and  $Z_1$  is discarded. The weighted prediction model therefore predicts the value of  $Z_N$  given  $Z_{N-1}, \dots, Z_1$ . In general, we can express this predicted value as a function of the  $N$  past samples i.e.,

$$\hat{Z}_N = \boldsymbol{\alpha}^T \tilde{\mathbf{Z}}, \quad (3.3)$$

where  $\hat{Z}_N$  is the new predicted value,  $\tilde{\mathbf{Z}}$  is the vector of past  $N - 1$  samples, and  $\boldsymbol{\alpha}^T$  is a vector of predictor coefficients distributed such that newer values have a greater impact on the predicted value  $\hat{Z}_N$ . A second vector,  $\mathbf{t}$ , records the time that each sample is taken and is shifted in the same manner as  $\mathbf{Z}$ . The objective of the weighted prediction algorithm is to find an appropriate coefficient vector,  $\boldsymbol{\alpha}^T$ , such that the following summation is minimized

$$S = \sum_{i=1}^{N-1} w_i \left( Z_i - \hat{Z}_i \right)^2, \quad (3.4)$$

where  $w_i$ ,  $Z_i$ , and  $\hat{Z}_i$  denote the weight, the actual sampled value, and the predicted value in the  $i^{th}$  interval, respectively.

The coefficient vector is given by:

$$\boldsymbol{\alpha}^T = \left( \tilde{\mathbf{Z}}^T \mathbf{W} \tilde{\mathbf{Z}} \right)^{-1} \tilde{\mathbf{Z}}^T \mathbf{W}, \quad (3.5)$$

where  $\mathbf{W} = \mathbf{w}^T \mathbf{w}$  is a  $(N-1) \times (N-1)$  diagonal weight matrix and  $\mathbf{w}$  is a  $N \times 1$  weight vector with weight co-efficient's  $w_i$  that are determined according to two criteria:

1. The ‘‘freshness’’ of the past  $N - 1$  samples. A more recent sample has a greater weight.
2. The similarity between the predicted value at the beginning of the time interval and the actual value. The similarity between the two values is measured by the distance between them. The smaller the Euclidean distance is, the more similar they are to each other.

Based on the above two criteria, we define a weight coefficient as

$$w_i = \frac{1}{(t_N - t_i)} \left( \frac{1}{|Z_i - \hat{Z}_i|^2 + \eta} \right), 1 \leq i \leq N - 1, \quad (3.6)$$

where  $\eta$  is a quantity introduced to avoid division by zero.

### 3.4.2 Adaptive Weighted Sampling

Adaptive sampling algorithms dynamically adjust the sampling rate based on the observed sampled data. A key element in adaptive sampling is the prediction of future behavior

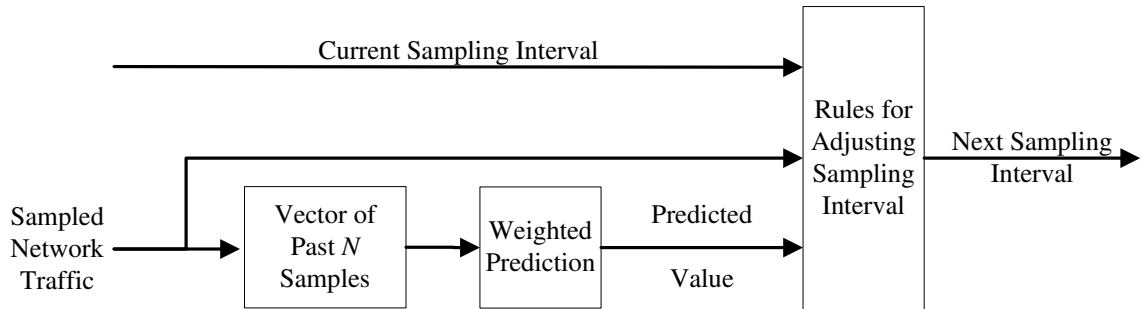


Figure 3.2: Block diagram of the adaptive sampling algorithm

based on the observed samples. The weighted sampling algorithm described in this section utilizes the weighted least squares predictor (see section 3.4.1) to select the next sampling interval. Inaccurate predictions by the weighted least squares predictor indicates a change in the network traffic behavior and requires a change in the sampling rate.

The proposed adaptive sampling algorithm consists of the following steps (see figure. 3.2):

1. Fix the first  $N$  sampling intervals equal to  $\tau$ . (In our simulations we used  $\tau = 60$  sec. and  $N = 10$ )
2. Apply the weighted least squares predictor to predict the anticipated value,  $\hat{Z}_N$ , of the network parameter.
3. Calculate the network parameter value at the end of the sampling time period.
4. Compare the predicted value with the actual value.
5. Adjust sampling rate according to the predefined rule set if the predicted value differs from the actual value.

$$\Delta T_{New} = \begin{cases} (1 + R) \times \Delta T_{Curr} & \text{if } R > R_{MAX} \\ \beta_1 \times \Delta T_{Curr} & \text{if } R_{MIN} < R < R_{MAX} \\ R \times \Delta T_{Curr} & \text{if } R < R_{MIN} \\ \beta_2 \times \Delta T_{Curr} & \text{if } R \text{ is Undefined} \end{cases} \quad (3.8)$$

The predicted output  $\hat{Z}_N$  which has been derived from the previous  $N$  samples, is then compared with the actual value of the sample,  $Z_N$ . A set of rules is applied to adjust the current sampling interval,  $\Delta T_{Curr} = t_N - t_{N-1}$ , to a new value,  $\Delta T_{New}$ , which is used to schedule the sampling query. The rules used to adjust the sampling interval compare the rate of change in the predicted sample value,  $\hat{Z}_N - Z_{N-1}$ , to the actual rate of change,  $Z_N - Z_{N-1}$ . The ratio,  $R$ , between the two rates is defined as:

$$R = \left| \frac{\hat{Z}_N - Z_{N-1}}{Z_N - Z_{N-1}} \right|. \quad (3.7)$$

Based on the value of  $R$ , which ranges from  $R_{MIN}$  to  $R_{MAX}$ <sup>1</sup>, we define the next sampling interval  $\Delta T_{New}$  as shown in Equation (3.8). The variables  $\beta_1$  and  $\beta_2$ , in Equation 3.8, are tunable parameters. When determining the values for  $\beta_1$  and  $\beta_2$ , one needs to consider the rate of change of the network parameter under consideration. As in [53], we used the values  $\beta_1 = 2$  and  $\beta_2 = 2$  in our simulations.

---

<sup>1</sup>Based on the results obtained from simulations performed by us, we selected a value of  $R_{MIN} = 0.82$  and  $R_{MAX} = 1.21$ . These values were selected because they provided good performance over a wide range of traffic types.

The value of  $R$  is equal to 1 when the predicted behavior is the same as the observed behavior. If the value of  $R$  is greater than  $R_{MAX}$ , it implies that the measured value is changing more slowly than the predicted value and this means that the sampling interval needs to be increased. On the other hand, if  $R$  is less than  $R_{min}$ , it implies that the measured value of the network parameter is changing faster than the predicted value. This indicates more network activity than predicted, so the sampling interval should be decreased to yield more accurate values for future predictions of the network parameter. The value of  $R$  may be undefined. This case arises when both the numerator and denominator of Equation (5) are zero. This condition is generally indicative of an idle network or a network in steady state. In such a scenario, the sampling interval is increased by a factor of  $\beta_2 (> 1)$ .

### 3.5 Summary

In this chapter, we have presented an adaptive sampling algorithm which uses weighted least squares prediction to dynamically alter the sampling rate based on the accuracy of the predictions. Results, analyzed in Chapter 5, have shown that compared to simple random sampling, the proposed adaptive sampling algorithm performs well on random, bursty data. Our simulation results will show that the proposed sampling scheme is effective in reducing the volume of sampled data while retaining the intrinsic characteristics of the network traffic. In addition we will also demonstrate that the proposed sampling scheme preserves the self-similar property of network traffic in the sampling process.

We believe that the proposed adaptive sampling scheme can be used for a variety of applications in the domain of network monitoring and network security. For example, our sampling algorithm can be used to collect and forward the sampled data to an IDS and in

the process reduce the volume of inspected traffic while still being able to detect variations in the self-similarity and long range dependence of network traffic, which are generally indicative of a traffic-intensive DoS attack [91].

# Chapter 4

## System Design

This chapter presents the design philosophy behind anomaly detection based intrusion detection system, christened SCAN, developed through this research. As outlined in Section 1.3, the overarching idea behind the development of SCAN, was to design, implement and test an anomaly detection system that has the ability to handle large quantities of data in high bandwidth environments. By employing an intelligent sampling algorithm, SCAN reduces the quantity of audit data it needs to analyze while maintaining the inherent properties of the original network flow(s).

### 4.1 Motivation

Intrusion detection is an important component of a network's security system. It complements existing security technologies (such as firewalls) by providing crucial information

to the network administrators about attacks and intrusions that may be undetected by existing security technologies. They also provide invaluable forensic information that will allow organizations to trace back the origins of attacks and aid in the prosecution of the attackers.

However, with the Internet having evolved in leaps and bounds over the past decade, most IDS's have not been able to keep up with the advances in high speed networking. IDS products, currently deployed in gigabit networks, need significant improvements before they can offer adequate protection against attacks. A majority of the products in the market today can detect less than half of the attacks directed at them, even though many of those attacks are well documented.

Another assumption that most deployed IDS's work under is the assumption that they have available for analysis, *clean, complete* and *labeled* data. Traditionally, both anomaly detection and misuse techniques, have traditionally relied on the availability of *clean* and *complete* data for analysis. This, however, is not a valid assumption any more due to the emergence of new types of networks such as Mobile and Ad hoc Networks (MANETs). In MANETs, mobility, sleep patterns of mobile devices and other characteristics specific to mobile and wireless networks prevent the collection of complete audit data for analysis. However, it should not be assumed that incomplete and noisy data is a problem that is relegated only to MANET's. With the advent and wide spread usage of high speed gigabit networks, the large amount of network data that such a network produces poses new challenges in anomaly detection. Prohibitively large volume of network data makes the tasks of storing, classifying, and labeling the data almost infeasible. We can, of course, obtain labeled data by simulating intrusions in a network. However, then we would be limited to the set of known attacks and we would not be able to detect new attacks.

Therefore, the focus of this chapter is the analysis of data generated in high speed IP networks, from the perspective of anomaly detection. We are particularly concerned with mining network data in such a dynamically changing, high volume environment for the express purpose of network anomaly detection in the presence of missing information. We contend that, network monitoring and data mining in high speed networks operates under a few constraints. For example, as pointed out in [22], capturing per packet information at the network edge and transporting it to a central data warehouse is unrealistic because of the storage and transportation costs. A more realistic approach would be to monitor the network traffic at key locations (network gateways and other egress and ingress points) and thereafter possibly share newly discovered intrusion details amongst the key locations. Sekar et al. [107] describe an approach to perform high performance analysis of network data, but unfortunately they do not provide experimental data based on live traffic analysis. Their claim of being able to perform real-time intrusion detection at 500 Mbps is based on the processing of off-line traffic log files.

Although there have been recent attempts [37, 39] to train anomaly detection models over noisy data, to the best of our knowledge, the research presented in this dissertation is the first attempt to investigate the effect of incomplete datasets on the anomaly detection process. The anomaly detection approach we present in this chapter follows the typical anomaly detection paradigm. We assume attack-free training data, but the outlier detection method we chose, is robust over small amount of missing audit data and noise.

The motivation behind our intrusion detection framework is straightforward: sampling reduces the amount of audit data that needs to be processed, thereby enabling real time anomaly detection even in high-speed networks. In typical cases, sampling would lead to loss of information, leading to inaccurate predictions and/or false alarms. To avoid such a scenario, the proposed intrusion detection scheme, christened SCAN (Stochastic Cluster-

ing Algorithm for Network anomaly detection), includes two distinguishing modules:

- An adaptive sampling module that intelligently adapts the sampling rate to sample packets geared towards intelligent sampling for anomaly detection.
- A predictive data mining based anomaly detection module that has the capability to estimate the missing data and reduce the occurrence of false alarms.

## 4.2 System Overview

To achieve the goal of detecting intrusions in high bandwidth environments, SCAN's architecture combines intelligent sampling and flow aggregation with data reduction and anomaly detection to achieve a high degree of accuracy in detecting intrusions with partial audit data. The design requirements for such a network based IDS were (a) stateless inspection of packets, protocols and/or packet headers at wire speed, (b) low occurrence of false alarms and high detection rate, (c) ability to track TCP states, and (d) ability to report events and/or alarms. Based on these requirements SCAN (see figure 4.1), is composed of the following five modules:

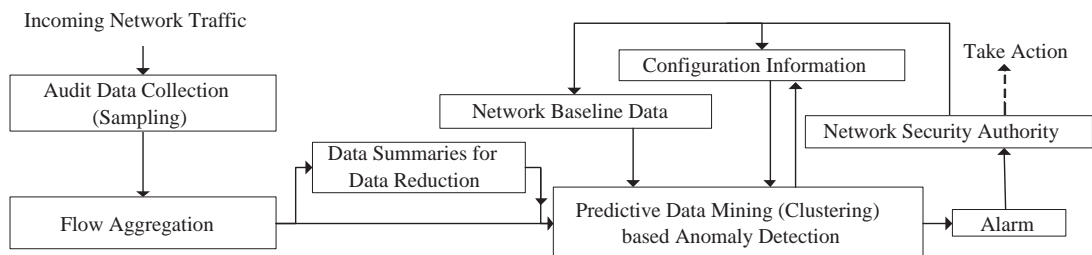


Figure 4.1: System model.

**Adaptive weighted packet sampling:** The *adaptive weighted sampling* module intelligently samples incoming network traffic to reduce the amount of network traffic that has to be processed without losing the inherent characteristics of the network traffic.

**Flow aggregation:** The *flow aggregation* module aggregates the sampled network traffic into flows based on the destination host and port information. The aggregated flows are fed to the *clustering* module and the *data reduction* module simultaneously.

**Data reduction:** For every time slice of operation, several data summaries are extracted from the aggregated flows. Data summarization has two advantages. Firstly the summaries are used to increase the speed of convergence of the clustering algorithm and secondly data summarization enables *data reduction*.

**Clustering:** At the core of SCAN is a stochastic clustering algorithm, christened PORTION, which is based on an improved version of the Expectation–Maximization (EM) algorithm [26]. The EM algorithm’s speed of convergence was accelerated by using a combination of Bloom filters, data summaries and associated arrays. The advantage of using the EM algorithm is that unlike other imputation techniques, EM computes the maximum likelihood estimates in parametric models based on prior information. The clustering approach that we propose can be used to process incomplete and unlabeled data.

**Anomaly detection:** The last step in SCAN involves performing *anomalous flow detection* on the clustered data.

### 4.3 Adaptive Sampling

To develop an accurate metric for network monitoring, the key issue is clearly the obtaining of relevant data about traffic by using end-to-end measurements. However, collecting this kind of data from a network requires the use of relatively expensive measurement techniques/equipment. On the other hand, examining every single packet traversing a given router/gateway is infeasible because of the required processing resources and the storage cost. However, to the best of our knowledge, none of the proposed algorithms for network traffic sampling have taken an approach that is tailored to meet the needs of attack detection. In this direction, we propose an intelligent sampling algorithm, the details are found in Chapter 3, that reduces the amount of audit data that needs to be analyzed while maintaining the self-similar properties of the original network flow. We will also show in Chapter 5, that the proposed sampling technique outperforms the traditional simple random sampling technique at many levels.

### 4.4 Flow Aggregation

In the context of SCAN, a *flow* is all the connections with a particular destination IP address and a port number combination. The measurement and characterization of the sampled data proceeds in time slices. We process the sampled packets into *connection records*. Because of its compact size as compared to other types of records (e.g., packet logs), connection records are appropriate for data analysis. A connection record provides the following fields:

$$(SrcIP, SrcPort, DstIP, DstPort, ConnStatus, Proto, Duration),$$

where

- SrcIP* and *DstIP* : Source and destination IP addresses
- SrcPort* and *DstPort* : Source and destination port numbers
- ConnStatus* : Status of connection (closed/open)
- Proto* : Network layer protocol identifier
- Duration* : Duration of connection

Each field can be used to correlate network traffic data and extract intrinsic features of each connection.

The flow aggregation algorithm employed by SCAN is shown in figure. 4.2. Bloom filters [11] and associated arrays are used to store, retrieve and run membership queries on incoming flow data in a given time slice. When a packet is sampled, we first identify the flow the packet belongs to by executing a membership query in the Bloom filter based on the flow's *flow ID*. The *flow ID* is a unique ID that is generated from a combination of the destination IP address and the destination port number. If an existing *flow ID* cannot be found, a new *flow ID* is generated based on the information from the connection record. This is called the *insertion phase*. If the *flow ID* exists, the corresponding array entry that is being used to track the flow is updated. If the *flow ID* does not exist, we run the *flow ID* through each of the  $k$  hash functions (that form a part of the Bloom filter), and use the result as an offset into the bit vector, turning on the bit we find at that position. If the bit is already set, we leave it on.

Bloom filters [11] were first proposed as a space efficient data structure for answering approximate membership queries over a given set. One drawback of Bloom filters is the risk of false positives (i.e. returning a “yes” indicating that a given element is in the set when it is not). The false positive probability is a function of the length of the filter and

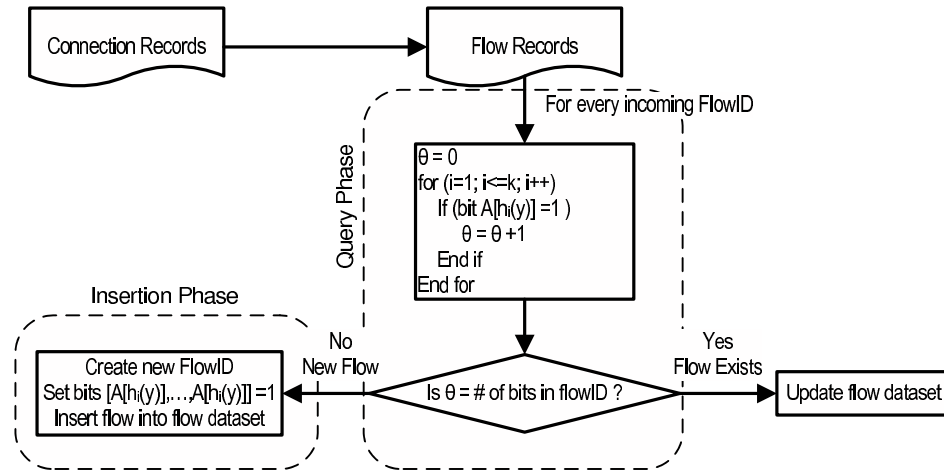


Figure 4.2: Online sampling and flow aggregation algorithm.

the number of items stored in it.

To succinctly represent a set  $S = \{x_1, x_2, \dots, x_n\}$  of  $n$  elements and support membership queries, we first select an array  $A$  consisting of  $m$  1-bit elements and initialize them to zero. In addition, we select  $k$  distinct hash functions  $h_1, \dots, h_k$  each with a range of  $[1, m]$ . During the *insertion phase*, for each element  $x \in S$ , the bits at the position  $h_1(x), \dots, h_k(x)$  in  $A$  are set to 1. In the *query phase*, to check if an element  $y$  is in  $S$ , we check the value of the bits at positions  $h_1(y), \dots, h_k(y)$  in  $A$ . The answer to the query is a *yes* if all of these bits are one and *no* otherwise.

## 4.5 POTION: EM Algorithm-Based Clustering Algorithm

Generally speaking, clustering techniques can be divided into two categories: *pairwise clustering* and *central clustering*. The former, also called similarity-based clustering, works on the guiding principle that “similar objects are grouped in the same cluster”. To

judge whether two objects are similar, a similarity measure must be given in advance. Examples of this category include graph partitioning-type methods. The latter category, also called centroid-based or model-based clustering, represents each cluster by a model, i.e., its “centroid”. Central clustering algorithms are often more efficient than similarity-based clustering algorithms. For example, similarity-based algorithms usually have a complexity of at least  $O(N^2)$  (for computing the similarity/proximity measure), where  $N$  is the number of data instances. In contrast, centroid-based algorithms are more scalable, with a complexity of  $O(NKM)$ , where  $K$  is the number of clusters and  $M$  the number of batch iterations. Regular  $k$ -means and the Expectation Maximization (EM) clustering algorithms fall into this category. In addition, all the centroid-based clustering techniques have an online version which can be suitably used for adaptive attack detection in a dataflow environment. In this section, we describe POTION (exPectatiOn maximizaTIon alGoriThm for aNomaly detection), SCAN’s improved EM-based clustering algorithm which has advantages over the  $k$ -means clustering algorithm in the context of anomaly detection.

A stochastic clustering method (such as the EM algorithm) assigns a data point to each group with a certain probability. Such statistical approaches make sense in practical situations where no amount of training data is sufficient to make a completely firm decision about cluster memberships. Stochastic clustering algorithms aim to find the most likely set of clusters given the training data and prior expectations. The underlying model, used in the stochastic clustering algorithm, is called a *finite mixture model*. A mixture is a set of probability distributions—one for each cluster—that models the attribute values of the members of that cluster.

In the finite mixture model under consideration, we consider a scenario where we assume that the distribution  $\mathbf{T}$  that describes incoming network traffic<sup>1</sup> to a server  $s$  is distributed

---

<sup>1</sup>It should be pointed out here that the incoming traffic could be composed of either packets, flows or connections.

as a mixture of two populations [39]: a *normal* (**N**) distribution and an *anomalous* (**A**) distribution such that the  $i^{th}$  traffic element<sup>2</sup>,  $x_i$ , in a given time slice  $t$  is generated from the *normal* distribution **N** with a probability of  $p$  or from an *anomalous* distribution **A** with a probability of  $1 - p$ . In other words,

$$\mathbf{T} = p\mathbf{N} + (1 - p)\mathbf{A}. \quad (4.1)$$

During the training phase, we set the baseline activity profile assuming that all incoming traffic elements are generated from the normal distribution **N**.

The EM algorithm used by SCAN to cluster data is shown in figure 4.3. The EM algorithm is a general method of finding the maximum-likelihood estimate of the parameters of a distribution from a given data set when the dataset has missing values. For this discussion, let us assume that we have a density function  $P(Z|\Theta)$  that is governed by the set of parameters  $\Theta$ . In addition, we have a dataset of size  $n$  drawn from this distribution,  $Z = \{z_1, \dots, z_n\}$ . The function  $L(\Theta|Z)$  is called the likelihood function and is defined as the likelihood of the parameters given the dataset, i.e.

$$L(\Theta|Z) = P(Z|\Theta) = \prod_{i=1}^n P(z_i|\Theta) \quad (4.2)$$

In a maximum likelihood problem, our goal is to find  $\Theta$  that maximizes  $L$ .

POTION starts with an initial guess for the parameters of the mixture model for each cluster and then iteratively applies the *Expectation Step* (*E*) and the *Maximization Step* (*M*) in order to converge to the maximum likelihood fit. In the E-step, the EM algorithm first finds the expected value of the complete-data log-likelihood, given the observed data and the parameter estimates. In the M-step, the EM algorithm maximizes the expectation computed in the E-step. The two steps of the EM algorithm are repeated until an increase in

---

<sup>2</sup>The term *traffic element* is used to denote any of the elements in one connection record.

the log-likelihood of the data, given the current model, is less than the accuracy threshold  $\epsilon$ .

The EM algorithm typically converges to a local maximum which may or may not be the same as the global maximum. In practical implementations, the EM algorithm is executed multiple times with different initial settings for the parameter values. Then a given data point is assigned to the cluster with the largest log-likelihood score.

The EM Algorithm extends the traditional  $k$ -means algorithm in two important ways. First, instead of assigning cases or observations to clusters to maximize the differences in the value of means for continuous variables, the EM clustering algorithm computes probabilities of cluster memberships based on one or more probability distributions. The goal of the clustering algorithm is to maximize the overall probability or likelihood of the data, given the final clusters. Second, unlike the classic implementation of  $k$ -means clustering, the general EM algorithm can be applied to both continuous and categorical variables.

The classical EM algorithm does have a few disadvantages. In general, the algorithm is hard to initialize and the quality of the final solution depends on the quality of the initial solution. It may also converge to a poor locally optimal solution. This means that a solution may be acceptable, but is far from the optimal one. The EM algorithm needs an unknown number of iterations to converge to a good solution. That is, it is hard to set a threshold on the maximum number of iterations that is sufficient. The EM algorithm usually requires several passes over a dataset.

## 4.6 Data Summaries for Data Reduction

Our aim was to design a clustering algorithm which is fast and requires only a few passes over the data. The data space is assumed to contain data points with both numerical and categorical attributes. The attributes of each data point are the *words* from the corresponding audit file line. Note that we are using the term *word* to loosely describe a monitored attribute (e.g., IP addresses, port numbers, etc.). From the flow records that have been hashed and stored in the Bloom filters and the associated arrays, we calculate *data summaries* for every time slice. A list of data summary items are shown in figure 4.4. Data reduction techniques like sufficient statistics and data summaries are often used to enhance the speed of convergence of the EM algorithm [89].

Our enhancements to the classical EM algorithm consist of three stages:

- Stage I: At the end of each time slice, we make a pass over the connection record dataset and build data summaries. Data summaries are particularly important in the EM algorithm framework as they reduce the I/O time by avoiding repeated scans over the data. In addition, data summaries allow periodic parameter estimation as the records are being read. This enhances the speed of convergence of the EM algorithm.
- Stage II: We make another pass over the dataset to build cluster candidates using the data summaries collected in the first stage. After the frequently appearing words have been identified, the algorithm builds all cluster candidates during this pass. The cluster candidates are hashed and stored in Bloom filters and associated arrays. The dataset is processed line by line. When a line is found to belong to one or more dense

regions (i.e. one or more frequently appearing words have been discovered in the line), a cluster candidate is formed. This is the E-step of the EM algorithm, where the cluster membership is determined. If the cluster candidate is not present in the Bloom filter, it is inserted into the filter with the value of one. If the cluster candidate is present in the Bloom filter, the corresponding value in the array is incremented

- Stage III: Clusters are selected from the set of candidates.

Learning steps (periodic M-steps) are used to accelerate convergence. By using data summaries, the clustering algorithm can run the M-step at different frequencies while scanning the connection records. We can minimize the convergence time by running the M-step after every point. Unfortunately, the EM algorithm would not produce the globally optimal solution in such a scenario. On the other hand, we can produce a solution that is closer to the globally optimum solution by executing the M-step after all the  $n$  points have been read (as in the classical EM algorithm). This, however, would require a longer convergence time. Therefore, it is desirable to set the frequency of M-step execution in between the two aforementioned cases, but closer to the latter scenario. When a good approximation to the solution has been reached, we can execute normal EM iterations until the algorithm converges. During the final step of the clustering algorithm, the Bloom filter and the associated arrays are inspected, and the regions with values equal or greater than the threshold value are reported by the algorithm as clusters.

## 4.7 Anomalous Flow Detection

The first task in an anomaly detection process is *network baselining*. Network baselining can be defined as the act of measuring and rating the performance of a network. Provid-

ing a network baseline requires evaluating the normal network utilization, protocol usage, peak network utilization, and average throughput of the network usage over a period of time. Our approach to network baselining recognizes the fact that any parametric model of network traffic is an approximation to reality. Since our ultimate goal is anomaly detection, which we formulate as detecting a marked change in the baseline model parameters, we will not be addressing the more general problem of parameter estimation as an intermediate step to anomaly detection.

In our model, elements of the network flow that are anomalous are generated from the anomalous distribution ( $\mathbf{A}$ ), and normal traffic elements are generated from the normal distribution ( $\mathbf{N}$ ). Therefore, the process of detecting anomalies involves determining whether the incoming flow belongs to distribution  $\mathbf{A}$  or distribution  $\mathbf{N}$ . We use an approach similar to [39] and calculate the likelihood of the two cases to determine the distribution to which the incoming flow belongs to. We assume that for normal traffic, there exists a function  $F_N$  which takes as parameters the set of normal elements  $N_t$ , at time  $t$ , and outputs a probability distribution  $P_{N_t}$  over the data  $T$  generated by distribution  $\mathbf{T}$ . Similarly, we have a function  $F_A$  for anomalous elements which takes as parameters the set of anomalous elements  $A_t$  and outputs a probability distribution  $P_{A_t}$ . Assuming that  $p$  is the probability with which a flow  $x_t$  belongs to  $\mathbf{N}$ , the likelihood  $L_t$  of the distribution  $\mathbf{T}$  at time  $t$  is defined as

$$\begin{aligned}
 L_t(\mathbf{T}) &= \prod_{k=1}^n P_T(x_k) \\
 &= \left[ (p)^{|N_t|} \prod_{x_i \in N_t} P_{N_t}(x_i) \right] \left[ (1-p)^{|A_t|} \prod_{x_j \in A_t} P_{A_t}(x_j) \right] \quad (4.5)
 \end{aligned}$$

The likelihood values are often calculated by taking the natural logarithm of the likelihood rather than the likelihood itself because likelihood values are often very small numbers

that are close to zero. We calculate the log-likelihood as follows:

$$\begin{aligned}
 LL_t(\mathbf{T}) = & |N_t| \ln(p) + \sum_{x_i \in N_t} \ln P_{N_t}(x_i) \\
 & + |A_t| \ln(1-p) + \sum_{x_j \in A_t} \ln P_{A_t}(x_j)
 \end{aligned} \tag{4.6}$$

We measure the likelihood of each flow,  $x_t$ , belonging to a particular group by comparing the difference  $LL_t(\mathbf{T}) - LL_{t-1}(\mathbf{T})$ . If this difference is greater than some value  $\alpha$ , we declare the flow anomalous. We repeat this process for every flow in the time interval. Note that we have to recompute the probability distributions  $P_{N_t}(x_t)$  and  $P_{A_t}(x_t)$  at every step because of changes in their distributions.

## 4.8 Summary

In this chapter, we have presented the system design for an anomaly detection scheme—called SCAN—that has the ability to detect network based denial-of-service attacks in gigabit-speed networks with a relatively high degree of accuracy in the absence of complete audit data. By employing an intelligent sampling scheme, SCAN reduces the computational complexity [20] by reducing the volume of audit data that is processed without losing the intrinsic characteristics of the network traffic. In addition, SCAN also employs an improved Expectation-Maximization algorithm based clustering technique to impute the missing values and further increase the accuracy of anomaly detection.

**Input:** Dataset D and initial estimates

**Output:** Clustered set of data points.

**Algorithm:**

(a) **Initialize:** Set an initial guess for the probability  $p$  to an arbitrary non-singular choice  $p_j^{[0]}$

(b) **Iterate:** At step  $k$

i. **E Step:** Determine cluster memberships. Compute the sufficient statistics, assuming the parameter values  $p_j^{[k]}$ .

ii. **M Step:** Derive  $p_j^{[k+1]}$  as a maximum likelihood estimate based on the sufficient statistics collected in the E-Step

Score clustering and save if best seen so far

(c) **Terminate:** Terminate when

1. The difference in the log-likelihood between two consecutive steps

$$L(T, p_j^{[k+1]}) - L(T, p_j^{[k]}) \leq \epsilon.$$

where  $L()$  is the likelihood function

and  $\epsilon$  is the accuracy threshold (We use  $\epsilon=10^{-3}$ )

**OR**

2. Number of iterations = N (We use N=100),  
whichever comes first

Figure 4.3: EM algorithm for clustering.

- *Flow Concentration Factor*: Total number of TCP flows that have  $c$  as the source,  $s$  as the destination server and  $p$  as the destination port in the  $i^{th}$  time slice.
- *Total number of data points per cluster*: Number of points per cluster.
- *Percentage of control packets*: Percentage of control packets is specific to an application. A change in the rate should identify an anomaly.
- *Percentage of data packets*
- *Average flow duration over all flows*
- *Maximum number of flows to a particular service*
- *Average flow duration per destination*
- *Sum of points*: Sum of all the points  $y_i$  in the  $j^{th}$  cluster  $Y_j$  i.e.

$$S_j = \sum_{\forall(y_i \in Y_j)} y_i \quad (4.3)$$

- *Sum of square of points*: Sum of square of all the points  $y_i$  in the  $j^{th}$  cluster  $Y_j$  i.e.

$$SS_j = \sum_{\forall(y_i \in Y_j)} (y_i y_i^T) \quad (4.4)$$

- *Percentage of same service to same host*: Percentage of incoming traffic that originates from a particular source port and terminates at a particular destination IP address.
- *Percentage of same host to same service*: Percentage of incoming traffic that originates from a particular source IP address and terminates at a particular destination port.
- *Resent rate*: Number of bytes that have been resent. Control packets count as one byte.
- *The variance of the count of packets*: This variance is an indicator of the distribution of hosts contacted in a time interval. During the event of a port scan, a high variance in the packet counts across all source-destination pairs should reveal an unusual spread in the number of machines contacted in an interval
- *Wrong resent rate*: Number of bytes that were sent even after being acknowledged.
- *Duplicate ACK rate*: Number of duplicate acknowledgements received.
- *Data bytes sent in either direction*: Tracks the number of data bytes exchanged per flow.

Figure 4.4: Data summaries.

# Chapter 5

## Experimental Analysis

In order to study the feasibility and effectiveness of our intrusion detection system and validate our detection model, we carried out extensive simulation experiments. We first describe the datasets that have been used for the evaluation in Section 5.1. Then we explain and analyze the experimental results in Section 5.2. Lastly, we conclude the chapter in Section 5.3.

### 5.1 Data Set Descriptions

We utilized two datasets, the 1998 DARPA/MIT Lincoln labs intrusion detection dataset and network traffic logs from the WIDE Project in Japan, to evaluate the proposed intrusion detection system.

The DARPA/MIT Lincoln Lab evaluation data set has been used to test a large number

of intrusion detection systems [8, 37–39, 45–47, 75, 80–82, 100], both in the 1998 and 1999 blind evaluations, and in subsequent development work after the evaluation data was released. The data can be used to test both host based and network based systems as well as both signature and anomaly detection systems.

In the 1999 DARPA intrusion detection evaluation program, an environment was set up to acquire raw TCP/IP data for a network by simulating a typical US Air Force local area network. The local area network was operated like a real environment, and was blasted with multiple attacks. The dataset consists of five weeks of *tcpdump* data. Data from week 1 and 3 consist of normal attack-free network traffic. Week 2 data consists of network traffic with labelled attacks. The week 4 and 5 data are the “*test data*” and contain 201 instances of 58 different unlabelled attacks, 177 of which are visible in the inside *tcpdump* data<sup>1</sup>. The 1999 DARPA dataset has five classes of attacks that we describe below. In this research, we focus our attention on the first two.

1. *Denial-of-service*: A denial-of-service attack is an incident in which a user or organization is deprived of the services of a resource they would normally expect to have. Typically, the loss of service is the inability of a particular network service, such as e-mail, to be available or the temporary loss of all network connectivity and services. In the worst cases, for example, a Web site accessed by millions of people can occasionally be forced to temporarily cease operation. A denial-of-service attack can also destroy various programs and files in a computer system. Although usually intentional and malicious, a denial-of-service attack can sometimes happen accidentally. A denial-of-service attack is a type of security breach to a computer system that does not usually result in the theft of information or other security loss. However, these attacks can cost the targetted person or company a great deal of

---

<sup>1</sup>The inside *tcpdump* data is the data collected by a sniffer located inside a simulated military base.

time and money. Examples of different types of denial-of-service attacks that were present in the 1999 DARPA intrusion detection evaluation dataset are shown in Table 5.1.

Table 5.1: Examples of DoS attacks in the 1999 DARPA intrusion detection evaluation dataset.

| <b>Name of Attack</b> | <b>Service/Protocol</b> | <b>Effect of Attack</b>                                                         |
|-----------------------|-------------------------|---------------------------------------------------------------------------------|
| back                  | http                    | Slows down webserver response.                                                  |
| Neptune               | TCP                     | Denies access to one/more ports                                                 |
| Ping of death         | ICMP                    | Crash or reboot a system.                                                       |
| Smurf                 | ICMP                    | spoofed host will not be able to receive or distinguish real traffic.           |
| Syslogd               | Syslog Daemon           | Kills the syslogd daemon.                                                       |
| land                  | IP                      | Remote DoS attack that reboots many systems.                                    |
| Apache2               | http                    | Kills the apache daemon.                                                        |
| Mailbomb              | N/A                     | Fills up disk space on the email system, denying email services to other users. |
| Process Table         | TCP                     | Denies creation of new processes.                                               |
| Udpstrom              | Echo/Chargen            | slows down the network.                                                         |

2. *Surveillance/Probes*: Surveillance/Probes are a class of attacks where an attacker scans a network to gather information or find known vulnerabilities. An attacker with a map of the machines and services that are available on a network can then use the information to look for exploits. There are different types of probes: some of them abuse the computers legitimate features while others use social engineering

techniques. This class of attacks, typically, require very little technical expertise. Examples of different types of surveillance/probe attacks that were present in the 1999 DARPA intrusion detection evaluation dataset are shown in Table 5.2.

Table 5.2: Examples of surveillance/probes in the 1999 DARPA intrusion detection evaluation dataset.

| <b>Name of Probe</b> | <b>Service</b> | <b>Effect of Attack</b>               |
|----------------------|----------------|---------------------------------------|
| IpSweep              | ICMP           | Identifies active machines.           |
| Nmap                 | Multiple       | Identifies active ports on a machine. |
| Port Sweep           | Multiple       | Identifies active ports on a machine. |
| Satan                | Multiple       | Looks for known vulnerabilities       |
| MScan                | Multiple       | Looks for known vulnerabilities.      |
| Saint                | Multiple       | Looks for known vulnerabilities.      |

3. *Remote to Local*: A remote to user attack is a class of attacks where an attacker sends packets to a machine over a network, then exploits machines vulnerability to illegally gain local access as a user. There are different types of remote to user attacks: the most common attack in this class is done using social engineering. The 1998 DARPA intrusion detection dataset had 14 types of remote to local attacks.
4. *User to Root*: User to root exploits are a class of attacks where an attacker starts out with access to a normal user account on the system and is able to exploit vulnerability to gain root access to the system. Most common exploits in this class of attacks are regular buffer overflows, which are caused by standard programming mistakes and environment assumptions. The 1998 DARPA intrusion detection dataset had 7 types of user to root attacks.

5. *Data attacks*: This was a new class of attacks that was added in 1999 dataset. The goal of a Data attack is to exfiltrate special files which the security policy specifies should remain on the victim hosts. These include “secret” attacks where a user who is allowed to access the special files copies/leakes them via common applications such as mail or FTP. It also includes attacks where privileged access to the special files is obtained using a user to root attack. The “data” part of the terminology “*data attack*” therefore specifies the goal of the attack rather than the attack mechanism.

The second dataset that we used is the dataset that is available from the Widely Integrated Distributed Environment (WIDE) project [122]. The WIDE project is a research consortium in Japan established in 1987. The members of the project include network researchers, engineers and students of universities, industries and government. The focus of the project is empirical study on a live large-scale internet. Thus, WIDE runs its own internet testbed carrying both commodity traffic and research experiments. The WIDE backbone network consists of links of various speeds, from 2Mbps CBR (Constant Bit Rate) ATM up to 10 Gbps Ethernet. Daily traces of internet data are collected from the following sampling points:

**Trans-Pacific** is a T1 line, one of the several international links of WIDE. The sampling point is on an Ethernet segment one hop before the T1 line. The incoming traffic (from U.S. to Japan) of this link is fairly congested.

**6Bone** is located on a FastEthernet segment connected to NXPIXP-6 (An IPv6 internet exchange point in Tokyo). The segment located at an AS boundary, and the traffic includes only native IPv6 and does not include IPv4 except tunneled IPv4 over IPv6. Because NXPIXP-6 is built on a FastEthernet switch, only the traffic crossing a single port of the switch can be captured.

The WIDE dataset we analyzed consisted of a 24-hour trace that was collected on September 22, 2005 from the Trans-Pacific sample point.

## **5.2 Experimental Results**

The experimental evaluation of SCAN has been divided into three steps. In Section 5.2.1, we evaluate the performance of the sampling algorithm and compare its performance with the simple random sampling algorithm. Section 5.2.1 is followed by Section 5.2.2, where we compare the performance of the EM algorithm based clustering algorithm with the traditional  $k$ -means clustering algorithm. Lastly, in Section 5.2.3 we evaluate the performance of the anomalous flow detection algorithm.

### **5.2.1 Evaluation of the Sampling Algorithm**

This section describes a set of simulations that were performed to compare the performance of the proposed adaptive sampling technique with the simple random sampling technique. The metrics used to compare the sampled data sets are also described in this section.

### 5.2.1.1 Experimental Setup

Simulations were conducted to compare and evaluate the performance of the proposed adaptive sampling algorithm with the simple random sampling algorithm<sup>2</sup>. The proposed adaptive sampling algorithm as well as the simple random sampling algorithm were implemented in Perl on an Intel<sup>®</sup> Pentium 4 laptop with 1 gigabyte of RAM, 100 gigabyte of hard disk space and running the Slackware<sup>®</sup> Linux operating system. The algorithms were evaluated using data from the Widely Integrated Distributed Environment (WIDE) project [122]. As mentioned above (Section 5.1), the WIDE backbone network consists of links of various speeds, from 2Mbps CBR (Constant Bit Rate) ATM up to 10 Gbps Ethernet.

---

<sup>2</sup>In simple random sampling, a sample of  $n$  individuals are drawn from a population of  $N$  individuals, in such a way that each individual has equal probability to be drawn. This implies that all combinations of  $n$  individuals have the same probability to be drawn. The essential characteristics of simple random sampling are:

- Each member of the population has an equal chance of being picked
- Selection are independent

There are, however, a number of potential problems with simple random sampling. Firstly, it is time-consuming to draw a random sample one individual at a time. Secondly, if the population is widely dispersed, it is extremely costly to reach them. In other words, it will be expensive to sample outliers and some sections of the population may, by chance, be sampled too heavily and others too lightly. Lastly, determining the appropriate sample size often requires a statistician.

### 5.2.1.2 Metrics for Evaluation

In order to compare the performance of the proposed adaptive sampling algorithm with the simple random sampling algorithm, a useful criterion to use is the mean square error (MSE) of the estimate or its square root, the root mean squared error, measured from the population that is being estimated. Formally we can define the mean square error of an estimator  $X$  of an unobservable parameter  $\theta$  as  $MSE(X) = E[(X - \theta)^2]$ . The root mean square error is the square root of the mean square error and the root mean square error is minimized when  $\theta = E(X)$  and the minimum value is the standard deviation of  $X$ .

In the second set of experiments, we verified whether the traffic data sampled by the proposed sampling scheme has the self similar property. For this verification, we used two different parameters: the mean of the packet count and the Hurst parameter. The peak-to-mean ratio (PMR) can be used as an indicator of traffic burstiness. PMR is calculated by comparing the peak value of the measure entity with the average value from the population. However, this statistic is heavily dependent on the size of the intervals, and therefore may or may not represent the actual traffic characteristic. A more accurate indicator of traffic burstiness is given by the Hurst parameter. The Hurst parameter (H), as mentioned above in Chapter 3, is a measure of the degree of self-similarity. In this dissertation we use the R-S statistical test to obtain an estimate for the Hurst Parameter (H). We run the test on both the original and the sampled data.

### 5.2.1.3 Experimental Results

In figure 5.1, we compare the proposed sampling scheme with the simple random sampling algorithm using the standard deviation of packet delay as the comparison criterion.

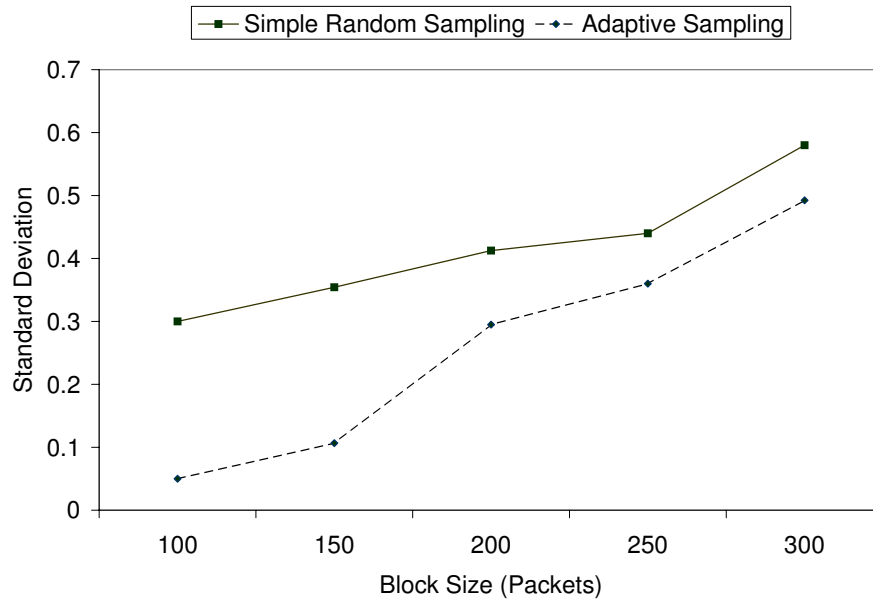


Figure 5.1: Standard deviation of packet delay.

Packet delay is an important criterion for detecting DoS attacks, especially attacks that focus on degrading the quality of service in IP networks [43]. The results show that over different block sizes, the proposed adaptive scheme has a lower standard deviation when compared with the simple random sampling algorithm. Since standard deviation is directly proportional to the root mean square error criterion, this implies that the proposed algorithm predicts the packet mean delay better than the simple random sampling algorithm while reducing the volume of traffic.

In figure 5.2 and figure 5.3 we show the average sampling error for the Hurst parameter

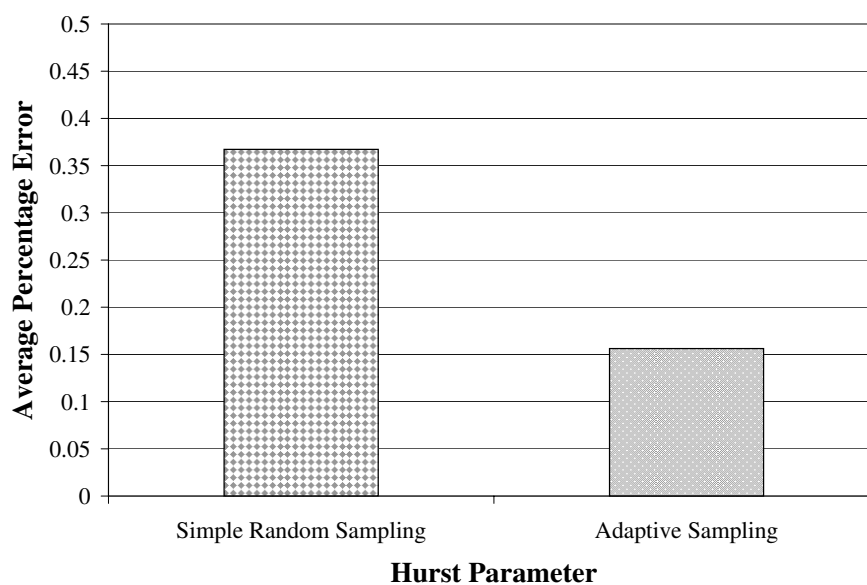


Figure 5.2: Average percentage error for the Hurst parameter.

and the sample mean respectively. As one can see from figure 5.2, the random sampling algorithm resulted in higher average percent error for the Hurst parameter when compared to adaptive sampling. This could be the result of missing data spread out over a number of sampling intervals. In figure 5.3, the average percentage error for the mean statistic was marginally higher for our sampling algorithm when compared with the simple random sampling algorithm, albeit the difference was insignificant. One possible reason for this marginal difference is the inherent adaptive nature of our sampling algorithm—i.e., the proposed sampling algorithm is more likely to miss short bursts of high network activity in periods that typically have low network traffic. The simple random sampling scheme would be less likely to have the same problem.

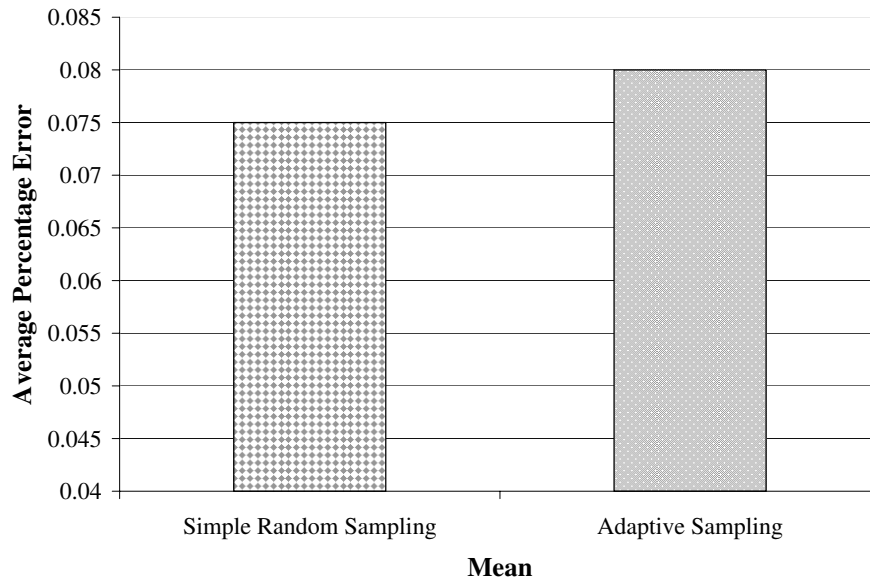


Figure 5.3: Average percentage error for the mean statistic.

## 5.2.2 Evaluation of the Clustering Algorithm

To evaluate quality of results and performance we tested our clustering algorithm with the DARPA/MIT Lincoln labs intrusion detection dataset. POTION is compared against the  $k$ -means clustering algorithm. The  $k$ -means clustering algorithm assigns multivariate observations to a pre-determined number of groups ( $k$ ). Each of the  $k$  groups consists of  $m_k$  data items and a group centroid ( $y_k$ ). In the beginning, the algorithm randomly selects  $k$  initial clusters from the dataset as the group centroids. Then the  $k$ -means algorithm calculates the arithmetic mean of each cluster formed in the dataset. In other words, the objective it tries to achieve is to minimize total intra-cluster variance

$$\sum_{i=1}^k \sum_{j \in S_i} |x_j - \mu_i|^2,$$

where there are  $k$  clusters  $S_i = 1, 2, \dots, k$  and  $\mu_i$  is the centroid or mean point of all the points  $x_j \in S_i$ . The steps outlining the basic  $k$ -means algorithm are shown in figure 5.4.

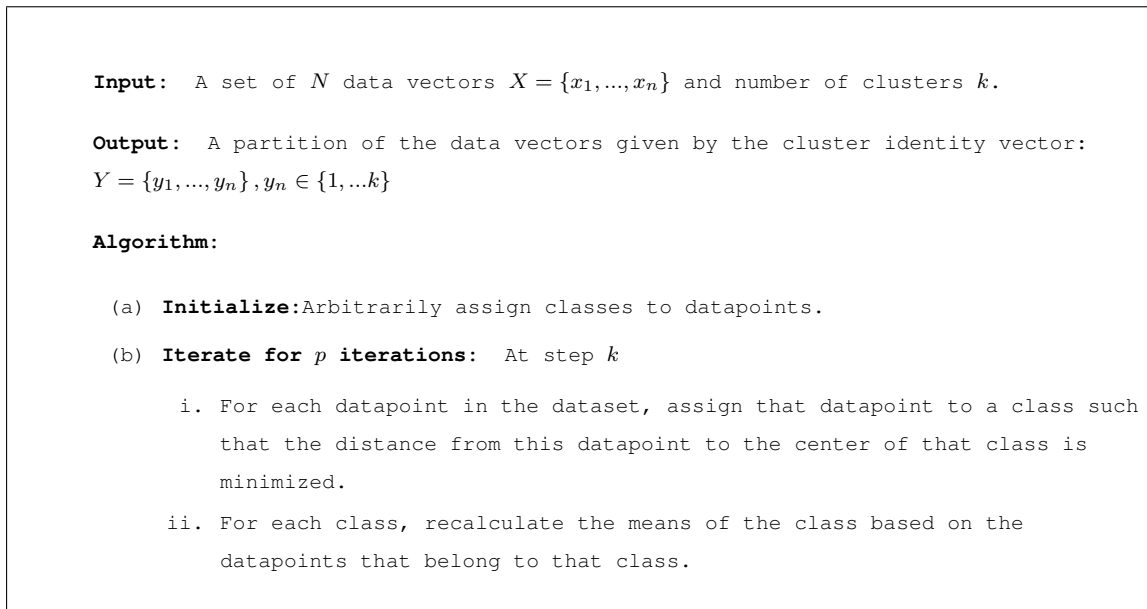


Figure 5.4:  $k$ -means algorithm for clustering.

The drawback of the  $k$ -means algorithm is that the quality of the local optimum strongly depends on the initial guess (either the centers or the assignments). If we start with a wild guess for the centers it would be fairly unlikely that the process would converge to a good local minimum. Another disadvantage of the  $k$ -means algorithm is that the answers it gives are not unique. Therefore it is difficult to compare the quality of the clusters produced.

### 5.2.2.1 Experimental Setup

Since the end goal is to accurately detect intrusions using the anomaly detection paradigm, as a first step we processed the “incoming” data as described in Section 4.4. In addition

to cluster sizes, inter-cluster distances are used in our clustering-based detection process. For the  $k$ -means algorithm, the inter-cluster distance is defined as the Euclidean distance between two cluster centroids. For the EM algorithm, it is defined as the Mahalanobis distance between two cluster centroids. Using Mahalanobis distance is particularly useful for skewed data having different sizes. The covariance matrix  $C_j$  is used to scale each dimension for distance computation. The squared Mahalanobis distance of point  $y_i$  to cluster  $j$  is

$$\delta^2(y_i, M_j, C_j) = \delta_{ij}^2 = (y_i - M_j)^T C_j^{-1} (y_i - M_j).$$

The EM and the  $k$ -means algorithms were written and compiled in Matlab. The processed datasets were stored as plain text files. All experiments were run on a Intel Pentium computer running at 1.7 GHz, having 100 gigabytes of disk space and 1 gigabyte on main memory.

### 5.2.2.2 Metrics for Evaluation

Accuracy is the main quality concern for clustering. In our case one cluster is considered accurate if there is no more than  $\epsilon$  error in its centroid/mean. If  $\mu_j$  is the correct mean of cluster  $j$  and  $M_j$  is the value estimated by the algorithm, then the cluster  $j$  is considered accurate if

$$\frac{|\mu_j - M_j|}{\mu_j} \leq e$$

where  $e$  is the estimation error and in this dissertation has been assumed to be 0.1. That is, we will consider a cluster to be accurate if it differs by no more than 10% of its “true” mean. The accuracy of clustering of POTION was also estimated by varying the percentage of missing audit data.

Since the end goal of employing the clustering algorithm is to use it for outlier detection based intrusion detection, we also evaluated the performance of the anomaly detection algorithm when the input data was clustered using POTION and  $k$ -means respectively. We use the Receiver-Operating Characteristic (ROC) curves in both cases to evaluate the efficiency and accuracy of the anomaly detection process. The ROC curve approach analyzes the tradeoff between false alarm and detection rates for detection systems. ROC analysis was originally developed in the field of signal detection. ROC curves for intrusion detection indicate how the detection rate changes as internal thresholds are varied to generate more or fewer false alarms to tradeoff detection accuracy against analyst workload. Measuring the detection rate alone only indicates the types of attacks that an intrusion detection system may detect. Such measurements do not indicate the human workload required to analyze false alarms generated by normal background traffic. False alarm rates in the range of a few hundred per day make a system almost unusable, even with high detection accuracy, because putative detections or alerts generated can not be believed and security analysts must spend many hours each day dismissing false alarms. Low false alarm rates combined with high detection rates, however, mean that the putative detection outputs can be trusted and that the human labor required to confirm detections is minimized.

### 5.2.2.3 Experimental Results

During the *training phase* we set up a baseline of normal activity. This is achieved by feeding the flow records and data summaries, obtained after processing the *normal tcp-dump* data, as inputs to the clustering algorithm. The cluster boundaries that are obtained at the conclusion of this phase gives us the “boundary” of a normal cluster.

The cluster centers, covariance matrices, and probabilities for each cluster that are obtained

during the *training phase* are used as baseline inputs when SCAN is brought online. This is the *testing phase*. As incoming network data is processed and clustered by SCAN, any points that lie beyond a cluster boundary are termed as *outliers*. Based on the tolerance level of the IDS and the relative distance of the outliers from the cluster centers, the outliers will be labelled as normal points or anomalies. In our simulations, we assume that the tolerance level is zero. This implies that during the *testing phase*, any point that lies outside a cluster boundary is considered an anomaly.

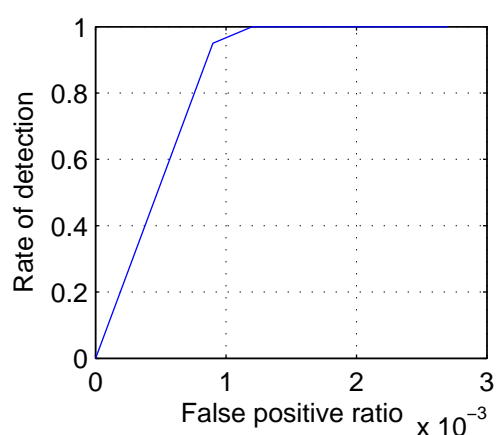


Figure 5.5: ROC curve for detection of the SYN Flood attack using POTION.

As mentioned above, we also compared the performance of POTION with the traditional  $k$ -means clustering algorithm. The  $k$ -means algorithm was trained using a “normal” dataset to cluster the normal behavior points. For the test data set, the probability of it belonging to the most probable cluster, was computed. If this was below a threshold, the instance was flagged as anomalous. After several runs, the outlier threshold for probability of belonging to a cluster was selected as 0.7. Figure 5.5 and figure 5.6 show the performance of the the traditional  $k$ -Means clustering algorithm against POTION while detecting the SYN Flood attack. The ROC curves show that POTION outperforms the traditional  $k$ -Means clustering algorithm.

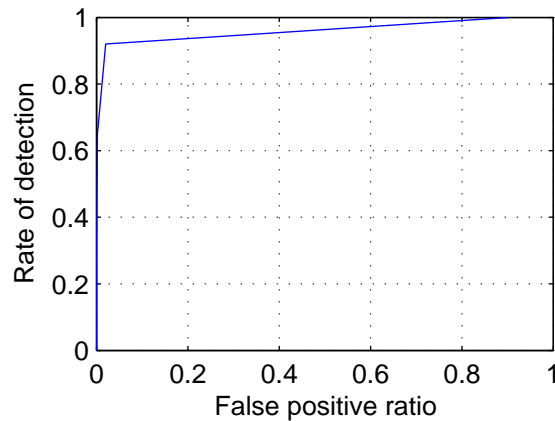


Figure 5.6: ROC curve for detection of the SYN Flood attack using the  $k$ -means clustering algorithm.

From figure 5.6 we see that the  $k$ -means clustering approach incurs a large number of false positives when all the features (figure 4.4) were used for clustering. This is in line with the known  $k$ -means clustering algorithm's weakness of poor performance in the presence of a large number of features in the dataset.

## 5.2.3 Evaluation of the Anomaly Detection Algorithm

### 5.2.3.1 Experimental Setup

In the last stage of the evaluation process, we evaluated the performance of the anomalous flow detection module in SCAN under two scenarios: when *complete* audit data is available and when only *partial* audit data is available. In the later case, to generate partial audit data, we assumed that the missing data are missing completely at random (MCAR). When we say that data are missing completely at random, we mean that the probability that an observation ( $X_i$ ) is missing is unrelated to the value of  $X_i$  or to the value of any other

variables. To introduce  $N\%$  missing features to a data set of  $D$  records, each of which has  $F$  features, we select randomly  $\lfloor (N \times D \times F) / 100 \rfloor$  records. For each selected record, we delete a randomly chosen feature to obtain a MCAR dataset.

We however contend that our scheme will work as well, in cases where we have bursty missing data. Such scenarios might occur when, for example, a network element is temporarily overwhelmed with the sudden surge in flow of network traffic and stops collecting data. Because, SCAN does not differentiate between bursty missing data and MCAR datasets, in such a case, it would determine the probable values of the network parameters based on the past information and use the thus calculated values to detect intrusions.

### 5.2.3.2 Metrics for Evaluation

As in Section 5.2.2, we use the Receiver-Operating Characteristic (ROC) curves in both cases to evaluate the efficiency and accuracy of the anomaly detection process. We use a practical assumption about the intrusion data — the number of normal instances is much larger than that of attack instances. This is usually true in reality. However, unlike in Portnoy et al. [101], we do not make the strict hypothetical requirement that the percentage of attacks has to be less than a certain threshold (e.g.,  $\tilde{1.5\%}$ ).

### 5.2.3.3 Experimental Results

The performance of SCAN at detecting a SSH Process Table attack when complete audit data is available is shown in figure 5.7. The attack is similar to the Process Table attack in that the goal of the attacker is to cause the *SSHD* daemon to spawn the maximum number

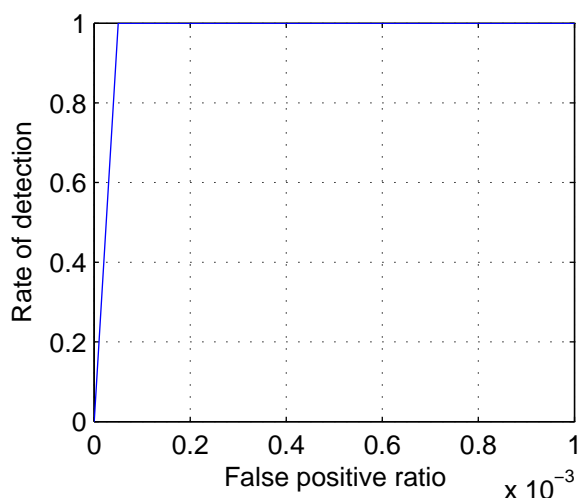


Figure 5.7: ROC curve for detection of the SSH Process Table attack using complete audit data.

of processes that the operating system will allow. In figure 5.8, the performance of SCAN at detecting a SYN flood attack is evaluated. In a SYN flood attack, an attacker makes a large number of half-open connections to a given port of a server during a short period of time. This causes the data structure in the ‘tcpd’ daemon in the server to overflow. Due to the buffer overflow, the system will be unable to accept any new incoming connections till the queue empties.

In figure 5.9, we plot SCAN’s ROC curves for the case in which complete audit data is available, and in cases when 5%, 7%, 11% and 17% of the audit data are missing. The simple random sampling algorithm was used to sample the network traffic data.

Figure 5.10 shows the accuracy of clustering<sup>3</sup> versus percentage of missing data. It can be seen that even with 10% missing data, the accuracy of clustering is in the high eighties. As expected, we see that the accuracy of clustering as well as the performance of the anomaly

<sup>3</sup>Accuracy of clustering is measured as the average percentage of correctly classified data over multiple runs.

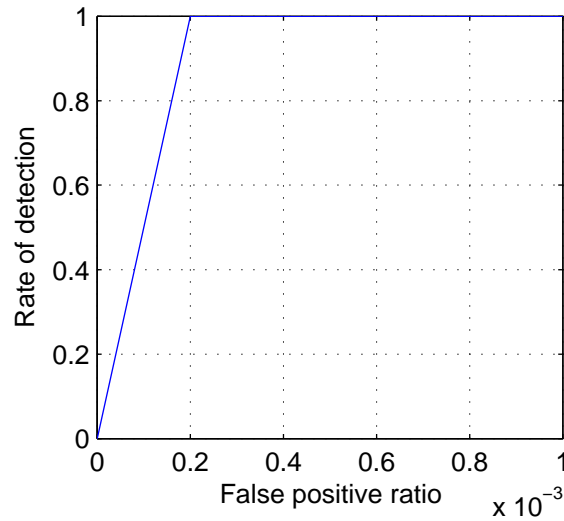


Figure 5.8: ROC curve for SYN Flood (Neptune) attack detection using complete audit data.

detection scheme degrades with increasing percentage of missing data.

Lastly, in figure 5.11, we compared the performance of the adaptive sampling technique and the random sampling technique for detecting the SYN Flood attack when clustering was done using POTION in both cases. As expected, the adaptive sampling technique showed superior performance.

### 5.3 Summary

This chapter presented the results from the simulation based analysis that was performed on SCAN. The evaluation methodology involved evaluating the components of SCAN individually before combining them and testing the system as a whole. In this direction, we first evaluated the performance of the proposed sampling algorithm, in Section 5.2.1,



Figure 5.9: ROC curve for SYN Flood attack detection using varying degrees of missing data (via random sampling).

by comparing its performance with the simple random sampling algorithm. We illustrated, through our simulation based analysis that the proposed sampling algorithm is superior to the traditionally used simple random sampling algorithm in predicting the packet mean delay while reducing the volume of traffic and maintaining the self similarity of the original network flows.

The second step of the evaluation process involves evaluating POTION, the proposed EM algorithm based clustering scheme. To do this, in Section 5.2.2, we compared the performance of the proposed algorithm with the widely used  $k$ -means clustering algorithm. Results illustrated that not only is POTION more accurate, but the anomalous flow algorithm has a better performance when the clustering algorithm that is employed is POTION.

Lastly, we evaluated the performance of the anomalous flow detection algorithm. In Section 5.2.3, our simulation results show that even with 10% missing data, the accuracy of

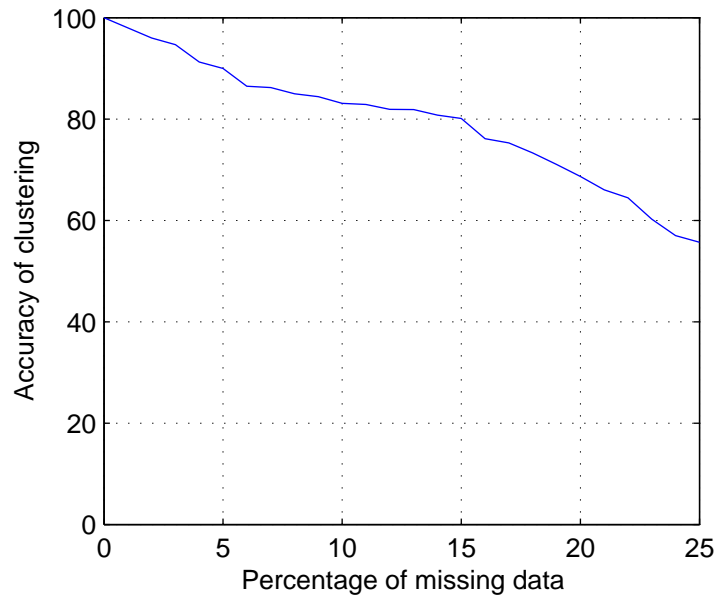


Figure 5.10: Accuracy of clustering with POTION vs. percentage of missing data.

clustering is in the high eighties. We also show, through our simulations, that the rate of detection with upto 17% missing data is very high as well.

This chapter provided ample evidence of the efficient and robust performance of the proposed anomaly detection based intrusion detection system. The results have vindicated our assertion that parametric estimation techniques like the EM algorithm, can be used effectively to detect intrusions in high bandwidth networks by employing sampling based approaches. The next chapter discusses the possible avenues of future investigation.

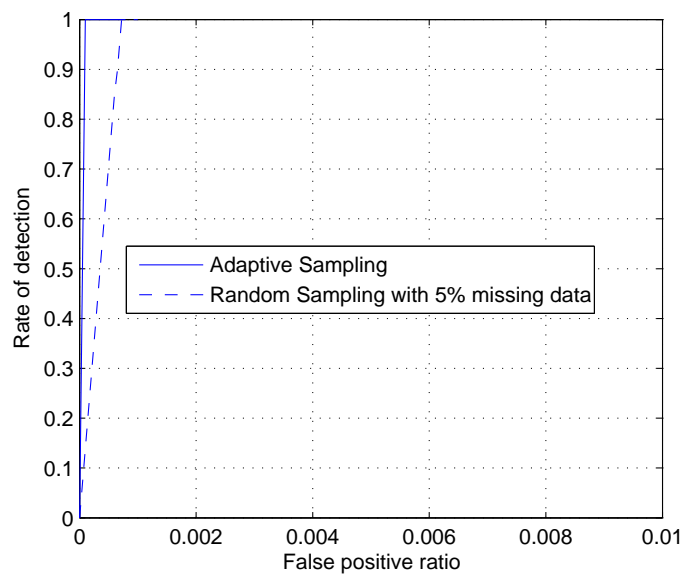


Figure 5.11: Comparison of adaptive sampling and random sampling in the detection of the SYN Flood attack.

# Chapter 6

## Summary and Future Work

The Internet has brought dramatic changes in the interaction between individuals, businesses, and governments. Moreover, global access to the Internet has become ubiquitous. With broadband networks, large amounts of data can be transferred rapidly between parties over the Internet. Users take these advances for granted until security attacks cripple the global Internet. Attacks spread rapidly through the same broadband networks that made the Internet revolution possible. The cost of these attacks to individuals, companies, and governments has increased rapidly as well. According to Trend Micro, virus attacks cost global businesses an estimated \$55 billion in damages in 2003, up from about \$20 billion in 2002.

In academia, researchers have proposed the Internet2 and Supernet networks to provide a networking test bed for all university campuses in the United States to use for real-time collaboration at speeds that are 1000's of times faster than those available on currently deployed campus networks. Many scientists predict that Internet data network speed will

increase from the 10 Gbps to the Tbps range. With this explosive growth in Internet e-commerce and the advent of the terabit network on the horizon, security experts must reevaluate current security solutions. Clearly, the critical challenge facing the Internet in the future is establishing security and trust. Many researchers argue that automatic detection and protection is the only solution for stopping fast-spreading worms such as the SQL Slammer and Code Red.

Realizing the fact that traditional intrusion detection systems have not adapted adequately to new networking paradigms like wireless and mobile networks and nor have they scaled to meet the requirements posed by high-speed (gigabit and terabit) networks, this research presents techniques to address the threats posed by network based denial-of-service attacks in such networks. Although there have been recent attempts [37, 39] to train anomaly detection models over noisy data, to the best of our knowledge, the research presented in this dissertation is the first attempt to investigate the effect of incomplete data sets on the anomaly detection process. This chapter describes the work performed during this dissertation and highlights significant achievements and possible extensions to the existing state of the art.

## **6.1 Summary**

This research provides a methodology to detect intrusions when only a subset of the audit data is available. To achieve this goal, the proposed anomaly detection scheme integrated a number of components in a unique way to overcome the limitations of previous intrusion detections systems. To tackle the problem of intrusion detection in high bandwidth networks the following modules were implemented and tested.

**Sampling for data reduction:** We believe that the key to efficient and cost-effective intrusion detection in high bandwidth networks is implementing a system that samples the incoming network traffic instead of attempting to parse and analyze all of it. From this perspective, in this dissertation, we have presented an adaptive sampling algorithm that uses the weighted least squares predictor to dynamically alter the sampling rate based on the accuracy of the predictions. Results have shown that compared to simple random sampling, the proposed adaptive sampling algorithm performs well on random, bursty data. Our simulation results have shown that the proposed sampling scheme is effective in reducing the volume of sampled data while retaining the intrinsic characteristics of the network traffic. In addition, we have also demonstrated that the proposed sampling scheme preserves the self-similar property of network traffic in the sampling process.

**Bloom filters based fast flow aggregation:** In high speed networks, flow aggregation and state maintenance becomes a problem as the number of flows increase. As a result, delays are introduced in the intrusion detection process which, in the worst case scenario, makes the intrusion detection system totally ineffective. To mitigate the delay problem and provide for a fast lookup for maintaining the state of an incoming flow, this dissertation proposes a Bloom filter based flow aggregation scheme.

**Improvements in speed of clustering:** The classical EM algorithm has many desirable features that includes a strong statistical basis, theoretical guarantees about optimality, easily explainable results, robustness to noise and to highly skewed data. Nevertheless, the classical EM algorithm also has several disadvantages. In general, the EM algorithm converges to a poor locally optimal solution and requires a number of passes over the dataset for every iteration. To address these problems, this dissertation proposed improvements in speed by employing sufficient statistics and learning steps to accelerate convergence and reduce the number of passes per

iteration.

**Anomalous flow detection** Lastly based on the improvements described above, this dissertation also proposes a novel intrusion detection scheme that is capable of detecting anomalous/intrusive flows in the incoming network traffic which would be indicative of a network based denial of service attack when only a subset of the audit data is available. Results described above, have shown that the proposed system has a high rate of detection even when only 75% of the audit data is present.

## 6.2 Future Work

In the last twenty years, intrusion detection systems have slowly evolved from host- and operating system-specific applications to distributed systems that involve a wide array of operating systems. The challenges that lie ahead for the next generation of intrusion detection systems and, more specifically, for anomaly detection systems are many.

Over the years, numerous techniques, models, and full-fledged intrusion detection systems have been proposed and built in the commercial and research sectors. However, there is no globally acceptable standard or metric for evaluating an intrusion detection system. Although the ROC curve has been widely used to evaluate the accuracy of intrusion detection systems and analyze the tradeoff between the false positives rate and the detection rate, evaluations based on the ROC curve are often misleading and/or incomplete [44,113]. Recently, several methods have been proposed to address this issue [7,44,113]. However, a majority of the proposed solutions rely on parameters values (such as the cost associated with each false alarm or missed attack instance) that are difficult to obtain and are subjective to a particular network or system. As a result, such metrics may lack the objectivity

required to conduct a fair evaluation of a given system. Therefore, one of the open challenges is the development of a general systematic methodology and/or a set of metrics that can be used to fairly evaluate intrusion detection systems.

An important aspect of intrusion detection, which has also been proposed as an evaluation metric [102, 109, 116], is the ability of an intrusion detection system to defend itself from attacks. Attacks on intrusion detection systems can take several forms. As a specific example, consider an attacker sending a large volume of non-attack packets that are specially crafted to trigger many alarms within an intrusion detection system, thereby overwhelming the human operator with false positives or crashing the alert processing or display tools. Axelsson [6], in his 1998 survey of intrusion detection systems, found that a majority of the available intrusion detection systems at that time performed very poorly when it came to defending themselves from attacks. Since then, the ability of intrusion detection systems at defending themselves from attacks has improved only marginally.

Lastly, an increasing problem in today's corporate networks is the threats posed by insiders, viz., disgruntled employees. In a survey [88] conducted by the United States Secret Service and CERT of Carnegie Mellon University, 71% of respondents out of 500 participants reported that 29% of the attacks that they experienced were caused by insiders. Respondents identified current or former employees and contractors as the second greatest cyber security threat, preceded only by hackers. Configuring an intrusion detection system to detect internal attacks is very difficult. The greatest challenge lies in creating a good rule set for detecting "internal" attacks or anomalies. Different network users require different degrees of access to different services, servers, and systems for their work, thus making it extremely difficult to define and create user- or system-specific usage profiles. Although there is some existing work in this area (e.g., [77, 94]), more research is needed to find practical solutions.

## **6.3 Conclusion**

This chapter has presented conclusions based upon the research results, and recommended areas of future research. The goal of this research was to provide a methodology to detect network based attacks with incomplete audit data. The proposed scheme, SCAN, attempts to fill a niche in the intrusion detection domain, by attempting to address the problem of detecting network based denial-of-service attacks in high performance, high availability and high speed networks. By employing an intelligent sampling scheme, SCAN reduces the computational complexity [20] by reducing the volume of audit data that is processed without losing the intrinsic characteristics of the network traffic. In addition, SCAN also employs an improved Expectation-Maximization algorithm based clustering technique to impute the missing values and further increase the accuracy of anomaly detection.

The results of this dissertation have demonstrated that the research goal was achieved. For reference details of the papers published by the author during the course of the dissertation work, the reader is directed to the Vita at the end of the dissertation.

## References

- [1] C. C. Aggarwal and P. S. Yu, “Outlier detection for high dimensional data,” in *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, T. Sellis, Ed. ACM Press, May 2001, pp. 37 – 46.
- [2] R. Agrawal, T. Imielinski, and A. Swami, “Mining association rules between sets of items in large databases,” in *In Proceedings of the ACM SIGMOD Conference on Management of Data*. ACM Press, 1993, pp. 207–216.
- [3] D. Anderson, T. Frivold, A. Tamaru, and A. Valdes, “Next generation intrusion detection expert system (nides), software users manual,” Computer Science Laboratory, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025-3493, User Manual SRI–CSL–95–07, 1994.
- [4] D. Anderson, T. F. Lunt, H. S. Javitz, A. Tamaru, and A. Valdes, “Detecting unusual program behavior using the statistical component of the next-generation intrusion detection expert system (nides),” Computer Science Laboratory, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025-3493, Technical Report SRI-CSL-95-06, May 1995.
- [5] J. P. Anderson, “Computer security threat monitoring and surveillance,” James P. Anderson Co., Fort Washington, Pennsylvania, Technical Report, April 1980.

- [6] S. Axelsson, “Research in intrusion-detection systems: A survey,” In <http://citeseer.ist.psu.edu/axelsson98research.html>, Department of Computer Engineering, Chalmers University of Technology, Technical Report 98–17, December 1998.
- [7] —, “The base-rate fallacy and its implications for the difficulty of intrusion detection,” *ACM Transactions on Information and System Security*, vol. 3, no. 3, pp. 186–205, 2000.
- [8] D. Barbará, J. Couto, S. Jajodia, and N. Wu, “ADAM: a testbed for exploring the use of data mining in intrusion detection,” in *ACM SIGMOD Record: SPECIAL ISSUE: Special section on data mining for intrusion detection and threat analysis*, vol. 30, no. 4. ACM Press, 2001, pp. 15–24.
- [9] V. Barnett and T. Lewis, *Outliers in Statistical Data*, 3rd ed., ser. Wiley Series in Probability & Statistics. John Wiley & Sons, April 1994, no. SRI-CSL–95–07.
- [10] S. M. Bellovin, “Packets found on an internet,” vol. 23, no. 3. New York, NY, USA: ACM Press, 1993, pp. 26–31.
- [11] B. H. Bloom, “Space/time trade-offs in hash coding with allowable errors,” *Communications of the ACM*, vol. 13, no. 7, pp. 422 – 426, July 1970.
- [12] Y. Bouzida, F. Cuppens, N. Cuppens-Boulahia, and S. Gombault, “Efficient intrusion detection using principal component analysis,” in *In Proceedings of the 3me Confrence sur la Scurit et Architectures Rseaux (SAR)*, 2004.
- [13] P. S. Bradley, U. M. Fayyad, and C. A. Reina, “Scaling em (expectation- maximization) clustering to large databases,” Microsoft Research, Technical Report MSR-TR-98-35, 1998.
- [14] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

- [15] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “LOF: identifying density-based local outliers,” *ACM SIGMOD Record*, vol. 29, no. 2, pp. 93 – 104, June 2000.
- [16] S. M. Bridges and R. B. Vaughn, “Fuzzy data mining and genetic algorithms applied to intrusion detection,” in *National Information Systems Security Conference*, October 2000.
- [17] R. A. Calvo, M. Partridge, and M. A. Jabri, “A comparative study of principal component analysis techniques,” in *Ninth Australian Conference on Neural Networks*, February 1998.
- [18] G. Cheng and J. Gong, “Traffic behavior analysis with poisson sampling on high-speed network,” in *ICII '01: Proceedings of the International Conferences on Info-tech and Info-net, 2001*, vol. 5, Computer Science Dept., Southeast Univ., Nanjing, China. Beijing, China: IEEE, 29 Oct.-1 Nov 2001, pp. 158–163.
- [19] CISCO, *CISCO Intrusion Detection System*, Cisco Systems, 2001.
- [20] K. C. Claffy, G. C. Polyzos, and H.-W. Braun, “Application of sampling methodologies to network traffic characterization,” in *SIGCOMM '93: Proceedings of the Conference on Communications architectures, protocols and applications*. New York, NY, USA: ACM Press, 1993, pp. 194–203.
- [21] W. W. Cohen, “Fast effective rule induction,” in *12th International Conference on Machine Learning*, A. Prieditis and S. Russell, Eds. Morgan Kaufmann, 1995, pp. 115–123.
- [22] G. Cormode and S. Muthukrishnan, “What’s new: Finding significant differences in network data streams,” in *In Proceedings of the Twenty-third Annual Joint Confer-*

- ence of the IEEE Computer and Communications Societies INFOCOM 2004*, vol. 3. IEEE Press, March 2004, pp. 1534 – 1545.
- [23] I. Cozzani and S. Giordano, “A measurement based qos evaluation through traffic sampling,” in *SICON '98: Proceedings of the 6th IEEE Singapore International Conference on Networks (SICON)*. Singapore: IEEE, June 30–July 3 1998.
- [24] M. Crosbie and G. Spafford, “Applying genetic programming to intrusion detection,” in *AAAI Symposium on Genetic Programming*, pp. 1–8.
- [25] M. Crovella and A. Bestavros, “Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes,” in *SIGMETRICS'96: Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems.*, Philadelphia, Pennsylvania, May 1996, p. 160, also, in *Performance evaluation review*, May 1996, 24(1):160-169.
- [26] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via em algorithm,” *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1–38, 1977.
- [27] D. E. Denning, “An intrusion-detection model,” *IEEE Transactions in Software Engineering*, vol. 13, no. 2, pp. 222–232, 1987.
- [28] D. E. Denning and P. G. Neumann, “Requirements and model for ides—a real-time intrusion detection system,” Computer Science Laboratory, SRI International,, Menlo Park, CA 94025-3493, Technical report 83F83-01-00, 1985.
- [29] J. E. Dickerson and J. A. Dickerson, “Fuzzy network profiling for intrusion detection,” in *In Proceedings of the 19th International Conference of the North American Fuzzy Information Processing Society (NAFIPS)*, 13-15 July 2000, pp. 301 – 306.

- [30] J. Drobisz and K. J. Christensen, "Adaptive sampling methods to determine network traffic statistics including the hurst parameter," in *IEEE LCN '98: Proceedings of the IEEE Annual Conference on Local Computer Networks*. IEEE, 1998, pp. 238–247.
- [31] N. Duffield, A. Greenberg, and M. Grossglauser, "A framework for passive packet measurement," IETF, Internet Draft draftduffield- framework-papame-01, February.
- [32] N. Duffield, C. Lund, and M. Thorup, "Charging from sampled network usage," in *IMW '01: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, November 2001, pp. 245–256.
- [33] ———, "Properties and prediction of flow statistics from sampled packet streams," in *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement*. New York, NY, USA: ACM Press, 2002, pp. 159–171.
- [34] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179–211, 1990.
- [35] A. Erramilli, O. Narayan, and W. Willinger, "Experimental queueing analysis with long-range dependent packet traffic," *IEEE/ACM Trans. Netw.*, vol. 4, no. 2, pp. 209–223, 1996.
- [36] L. Ertz, E. Eilertson, A. Lazarevic, P.-N. Tan, V. Kumar, J. Srivastava, and P. Dokas, *The MINDS - Minnesota Intrusion Detection System*. Boston: MIT Press, vol. Next Generation Data Mining.
- [37] E. Eskin, "Anomaly detection over noisy data using learned probability distributions," in *In Proceedings of the 17th International Conf. on Machine Learning*. Stanford University: Morgan Kaufmann, San Francisco, CA, June 2000, pp. 255–262.

- [38] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. J. Stolfo, "A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data," in *In Proceedings of Applications of Data Mining in Computer Security*, D. Barbara and S. Jajodia, Eds. Kluwer, 2002.
- [39] E. Eskin and W. Lee, "Modeling system calls for intrusion detection with dynamic window sizes," in *Proceedings of the DARPA Conference and Exposition on Information Survivability (DISCEX 2001)*, vol. 1. IEEE Computer Society Press, 2001, pp. 165–175.
- [40] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," in *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM Press, 2002, pp. 323–336.
- [41] A. Feldmann, A. C. Gilbert, and W. Willinger, "Data networks as cascades: investigating the multifractal nature of internet wan traffic," in *In Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*. New York, NY, USA: ACM Press, 1998, pp. 42–55.
- [42] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for unix processes," in *IEEE Symposium on Research in Security and Privacy*. IEEE Computer Society Press, 1996, pp. 120–128.
- [43] E. Fulp, Z. Fu, D. S. Reeves, S. F. Wu, and X. Zhang, "Preventing denial of service attacks on quality of service," in *DISCEX '01: Proceedings of the DARPA Information Survivability Conference and Exposition II*, vol. 2. IEEE Press, June 2001, pp. 159–172.

- [44] J. E. Gaffney and J. W. Ulvila, "Evaluation of intrusion detectors: A decision theory approach," in *In Proceedings. 2001 IEEE Symposium on Security and Privacy*. Oakland, CA, USA: IEEE Press, 2001, pp. 50–61.
- [45] A. K. Ghosh, C. Michael, and M. Schatz, "A real-time intrusion detection system based on learning program behavior," in *In Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection*, ser. Lecture Notes In Computer Science, vol. 1907. Springer-Verlag London, UK, October 02 - 04 2000, pp. 93–109.
- [46] A. K. Ghosh and A. Schwartzbard, "A study in using neural networks for anomaly and misuse detection," in *In Proceedings of the Eighth USENIX Security Symposium*, 23–26 August 1999, pp. 141–151.
- [47] A. K. Ghosh, A. Schwartzbart, and M. Schatz, "Learning program behavior profiles for intrusion detection," in *In Proceedings of the 1st USENIX Workshop on Intrusion Detection and Network Monitoring*. USENIX Association, April 9-12 1999.
- [48] J. Gomez and D. Dasgupta, "Evolving fuzzy classifiers for intrusion detection," in *In Proceedings of the IEEE Workshop on Information Assurance*. IEEE Press, 2001.
- [49] L. A. Gordon, M. P. Loeb, W. Lucyshyn, and R. Richardson, "Tenth annual 2005 CSI/FBI computer crime and security survey," In [http://i.cmpnet.com/gocsi/db\\_area/pdfs/fbi/FBI2005.pdf](http://i.cmpnet.com/gocsi/db_area/pdfs/fbi/FBI2005.pdf), 2005.
- [50] R. Grossman, S. Kasif, R. Moore, D. Rocke, and J. Ullman, "Data mining: Challenges and opportunities for data mining during the next decade," In <http://www.rgrossman.com/epapers/dmr-v8-4-5.htm>, January 1998.

- [51] V. Hautamaki, I. Karkkainen, and P. Franti, "Outlier detection using k-nearest neighbour graph," in *17th International Conference on Pattern Recognition*, vol. 03. IEEE Computer Society, 2004, pp. 430–433.
- [52] D. Heckerman, "A tutorial on learning with bayesian networks," Microsoft Research, Technical Report MSR-TR-95-06, March 1995.
- [53] E. A. Hernandez, M. C. Chidester, and A. D. George, "Adaptive sampling for network management," in *Journal of Network and Systems Management*, vol. 9, no. 4. HCS Research Laboratory, University of Florida, December 2001, pp. 409–434.
- [54] J. Hipp, U. Gntzer, and G. Nakhaeizadeh, "Algorithms for association rule mining - a general survey and comparison," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 2, 2000, pp. 58–64.
- [55] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of Computer Security*, vol. 6, no. 3, pp. 151–180, 1998.
- [56] H. H. Hosmer, "Security is fuzzy!/: Applying the fuzzy logic paradigm to the multipolicy paradigm," in *1992-1993 workshop on New security paradigms*. ACM Press, 1993.
- [57] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology*, vol. 24, pp. 417–441, 498–520, 1933.
- [58] Y. Huang and J. M. Pullen, "Countering denial-of-service attacks using congestion triggered packet sampling and filtering," in *ICCCN '01: Proceedings of the Tenth International Conference on Computer Communications and Networks.*, Dept. of Comput. Sci., George Mason Univ., Fairfax, VA, USA;. Scottsdale, AZ: IEEE, October 2001, pp. 490–494.

- [59] B. Hutchings, R. Franklin, and D. Carver, "Assisting network intrusion detection with reconfigurable hardware." in *10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, April 2002, pp. 111–120.
- [60] ISS, *BlackICE Sentry Gigabit*, Internet Security Solutions, 2001.
- [61] E. M. Knorr and R. T. Ng, "Algorithms for mining distance-based outliers in large datasets," in *Proceedings of the 24rd International Conference on Very Large Data Bases*, A. Gupta, O. Shmueli, and J. Widom, Eds. Morgan Kaufmann Publishers Inc., 1998, pp. 392 – 403.
- [62] C. Krgel, T. Toth, and E. Kirda, "Service specific anomaly detection for network intrusion detection," in *2002 ACM symposium on Applied computing*. Madrid, Spain: ACM Press, 2002, pp. 201 – 208.
- [63] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur, "Bayesian event classification for intrusion detection," in *19th Annual Computer Security Applications Conference*, Las Vegas, NV, 2003.
- [64] C. Kruegel, F. Valeur, G. Vigna, and R. Kemmerer, "Stateful intrusion detection for high-speed networks," in *IEEE Symposium on Security and Privacy*, 2002, pp. 285–294.
- [65] S. Kumar and E. Spafford, "An application of pattern matching in intrusion detection," Purdue University, Department of Computer Sciences, Purdue University, West Lafayette, Technical Report 94-013, June 1994.
- [66] W. Lee, R. A. Nimbalkar, K. K. Yee, S. B. Patil, P. H. Desai, T. T. Tran, and S. J. Stolfo, "A data mining and cidf based approach for detecting novel and distributed intrusions," in *In Proceedings of the Third International Workshop on Recent Ad-*

- vances in Intrusion Detection*, ser. Lecture Notes in Computer Science, H. Debar, L. M., and S. F. Wu, Eds., vol. 1907. Springer-Verlag, October 2000, pp. 49–65.
- [67] W. Lee and S. J. Stolfo, “Data mining approaches for intrusion detection,” in *In Proceedings of the 7th USENIX Security Symposium*. USENIX Association, January 1998.
- [68] W. Lee, S. J. Stolfo, P. K. Chan, E. Eskin, W. Fan, M. Miller, S. Hershkop, and J. Zhang, “Real time data mining-based intrusion detection,” in *In Proceedings of the Second DARPA Information Survivability Conference and Exposition*. IEEE Press, June 2001, pp. 85–100.
- [69] W. Lee, S. J. Stolfo, and K. W. Mok, “A data mining framework for building intrusion detection models,” in *IEEE Symposium on Security and Privacy*. IEEE Press, 1999, pp. 120–132.
- [70] —, “Adaptive intrusion detection: A data mining approach,” *Artificial Intelligence Review*, vol. 14, no. 6, pp. 533 – 567, December 2000.
- [71] W. Lee and D. Xiang, “Information-theoretic measures for anomaly detection,” in *Proceedings of the 2001 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 14 - 16 2001, pp. 130–143.
- [72] W. E. Leland, M. S. Taqq, W. Willinger, and D. V. Wilson, “On the self-similar nature of Ethernet traffic,” in *SIGCOMM '93: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, D. P. Sidhu, Ed., San Francisco, California, 1993, pp. 183–193. [Online]. Available: [citeseer.ist.psu.edu/leland93selfsimilar.html](http://citeseer.ist.psu.edu/leland93selfsimilar.html)
- [73] M. Li, W. Jia, and W. Zhao., “Decision analysis of network based intrusion detection systems for denial-of-service attacks,” in *Proceedings of the IEEE Conferences on*

- Info-tech and Info-net*, vol. 5, Dept. of Computer Sci., City Univ. of Hong Kong, China. IEEE, October 2001.
- [74] W. Li, "Using genetic algorithm for network intrusion detection," In <http://www.security.cse.msstate.edu/docs/Publications/wli/DOECSG2004.pdf>, pp. 1–8, 2004.
- [75] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," *Computers & Security*, vol. 21, no. 5, pp. 439–448, 2002.
- [76] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 DARPA offline intrusion detection evaluation," in *Computer Networks*, ser. Special issue on recent advances in intrusion detection systems, D. A. Frincke and M.-Y. Huang, Eds., vol. 34, no. 4. Elsevier North-Holland, Inc., October 2000, pp. 579–595.
- [77] A. Liu, C. Martin, T. Hetherington, and S. Matzner, "A comparison of system call feature representations for insider threat detection," in *6th Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop*. West Point, NY: IEEE Press, 2005, pp. 340 – 347.
- [78] T. F. Lunt, A. Tamaru, F. Gilham, R. Jagannathm, C. Jalali, P. G. Neumann, H. S. Javitz, A. Valdes, and T. Garvey, "A real-time intrusion detection expert system (ides)– final technical report," Computer Science Laboratory, SRI International," Technical Report, February 1992.
- [79] P. C. Mahalanobis, "On tests and measures of groups divergence," *International Journal of the Asiatic Society of Benagal*, vol. 25, pp. 541–548, 1930.
- [80] M. V. Mahoney and P. K. Chan, "PHAD: Packet header anomaly detection for identifying hostile network traffic," Department of Computer Sciences, Florida Institute of Technology, Department of Computer Sciences, Florida Institute of Technology, Melbourne, FL 32901, Technical Report CS-2001-4, April 2001.

- [81] ———, “Learning models of network traffic for detecting novel attacks,” Computer Science Department, Florida Institute of Technology, Tech. Rep. CS-2002-8, August 2002 2002.
- [82] ———, “Learning nonstationary models of normal network traffic for detecting novel attacks,” in *Eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM Press, 2002, pp. 376–385.
- [83] R. A. Maxion and F. E. Feather, “A case study of ethernet anomalies in a distributed computing environment,” *IEEE Transactions on Reliability*, vol. 39, no. 4, pp. 433–443, 1990.
- [84] E. Millard, “Internet attacks increase in number, severity,” In CIO Today at [http://www.cio-today.com/story.xhtml?story\\_id=003000002F13](http://www.cio-today.com/story.xhtml?story_id=003000002F13), August 2005.
- [85] R. Neal and G. E. Hinton, “A view of the em algorithm that justifies incremental, sparse and other variants,” *Learning in graphical models*, pp. 355–368, 1999.
- [86] C. NetFlow, “CISCO NetFlow,” In [http://www.cisco.com/en/US/products/ps6601/products\\_ios\\_protocol\\_group\\_home.html](http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html).
- [87] T. Networks, “Toplayer networks,” In <http://www.toplayer.com/>, 2005.
- [88] F. B. of Investigation, “Bot herder pleads guilty to fraudulent adware installs and selling zombies to hackers and spammers.” In <http://losangeles.fbi.gov/dojpressrel/pressrel06/la012306usa.htm>, 2006.
- [89] C. Ordonez and E. Omiecinski, “FREM: Fast and robust EM clustering for large data sets,” in *In Proceedings of the 11th International conference on Information and knowledge management*. ACM Press, November 2002, pp. 590 – 599.

- [90] T. J. Ott, T. Lakshman, and L. Wong, “Sred: Stabilized red,” in *INFOCOM '99: Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, Bellcore, USA. New York, NY: IEEE, March 1999, pp. 1346–1355.
- [91] P. Owezarski, “On the impact of DoS attacks on internet traffic characteristics and QoS,” in *ICCCN '05: Proceedings of the 14th International Conference on Computer Communications and Networks*, LAAS-CNRS, Toulouse, France. IEEE, October 2005, pp. 269–274.
- [92] R. Pan, B. Prabhakar, and K. Psounis, “Choke - a stateless active queue management scheme for approximating fair bandwidth allocation,” in *INFOCOM '00: Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, Stanford University. Tel Aviv: IEEE, March 2000, pp. 942–951.
- [93] K. Papagiannaki, N. Taft, Z.-L. Zhang, and C. Diot, “Long-term forecasting of internet backbone traffic: Observations and initial models,” in *INFOCOM '03: Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, Spring ATL., Burlingame, CA, USA: IEEE Press, 30 March–3 April 2003 2003, pp. 1178–1188.
- [94] J. S. Park and J. Giordano, “Role-based profile analysis for scalable and accurate insider-anomaly detection,” in *25th IEEE International Performance, Computing, and Communications Conference*. Phoenix, Arizona: IEEE Press, 2006, pp. 463 – 470.
- [95] K. Park, G. Kim, and M. Crovella, “On the relationship between file sizes, transport protocols, and self-similar network traffic,” in *ICNP '96: Proceedings of the 1996*

- International Conference on Network Protocols (ICNP '96)*. Washington, DC, USA: IEEE Computer Society, 1996, p. 171.
- [96] K. Park and W. Willinger, *Self-Similar Network Traffic and Performance Evaluation*, 1st ed. Wiley Interscience, January 2000, ch. Self-Similar Network Traffic: An Overview, pp. 1–38.
- [97] V. Paxson, “Bro: a system for detecting network intruders in real-time,” *Computer Networks (Amsterdam, Netherlands: 1999)*, vol. 31, no. 23–24, pp. 2435–2463, 1999. [Online]. Available: [citeseer.ist.psu.edu/paxson98bro.html](http://citeseer.ist.psu.edu/paxson98bro.html)
- [98] J. Phillips, “Hackers’ invasion of our data raises blizzard of questions,” In The Athens News at [http://www.athensnews.com/issue/article.php3?story\\_id=24611](http://www.athensnews.com/issue/article.php3?story_id=24611), May 2005.
- [99] M. M. Pillai, J. H. P. Eloff, and H. S. Venter, “An approach to implement a network intrusion detection system using genetic algorithms,” in *2004 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, 2004, pp. 221 – 228.
- [100] P. A. Porras and P. G. Neumann, “EMERALD: Event monitoring enabling responses to anomalous live disturbances,” in *In Proceedings of the 20<sup>th</sup> NIST-NCSC National Information Systems Security Conference*, 1997, pp. 353–365.
- [101] L. Portnoy, E. Eskin, and S. J. Stolfo, “Intrusion detection with unlabeled data using clustering,” in *ACM Workshop on Data Mining Applied to Security*, 2001.
- [102] T. Ptacek and T. Newsham, “Insertion, evasion, and denial of service: Eluding network intrusion detection,” Secure Networks Inc., Tech. Rep., 1998.
- [103] J. R. Quinlan, *C4.5: Programs for Machine Learning*, ser. Morgan Kaufmann series in machine learning. Morgan Kaufmann, 1993.

- [104] M. Ramadas, S. Ostermann, and B. Tjaden, "Detecting anomalous network traffic with self-organizing maps," in *6th International Symposium on Recent Advances in Intrusion Detection*, ser. Lecture Notes in Computer Science, vol. 2820. Springer Berlin / Heidelberg, 2003, pp. 36 – 54.
- [105] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, M. Dunham, J. F. Naughton, W. Chen, and N. Koudas, Eds. ACM Press, May 2000, pp. 427 – 438.
- [106] S. T. Sarasamma, Q. A. Zhu, and J. Huff, "Hierarchical kohonen net for anomaly detection in network security," *IEEE Transactions on Systems, Man and Cybernetics PART B: Cybernetics*, vol. 35, no. 2, pp. 302–312, 2005.
- [107] R. Sekar, Y. Guang, S. Verma, and T. Shanbhag, "A high-performance network intrusion detection system," in *In Proceedings of the 6th ACM conference on Computer and communications security*. New York, NY, USA: ACM Press, 1999, pp. 8–17.
- [108] K. Sequeira and M. Zaki, "Admit: Anomaly-based data mining for intrusions," in *8th ACM SIGKDD international conference on Knowledge discovery and data mining*. Edmonton, Alberta, Canada: ACM Press, 2002, pp. 386–395.
- [109] U. Shankar and V. Paxson, "Active mapping: Resisting nids evasion without altering traffic," in *In Proceedings of the IEEE Symposium on Research in Security and Privacy*. Oakland, CA: IEEE Press, 2003.
- [110] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier," in *IEEE Foundations and New Directions of Data Mining Workshop*, 2003, pp. 172–179.

- [111] S. E. Smaha, "Haystack: An intrusion detection system," in *In Proceedings of the Fourth Aerospace Computer Security Applications Conference*. IEEE Press, December 1988, pp. 37 – 44.
- [112] C. staff, "Hackers: companies encounter rise of cyber extortion," In Computer Crime Research Center Website <http://www.crime-research.org/news/24.05.2005/Hackers-companies-encounter-rise-cyber-extortion/>, 2005.
- [113] S. J. Stolfo, W. Fan, and W. Lee, "Cost-based modeling for fraud and intrusion detection: Results from the jam project," in *DARPA Information Survivability Conference & Exposition*, vol. 2. IEEE Press, 2000, pp. 130 – 144.
- [114] A. H. Sung and S. Mukkamala, "Identifying important features for intrusion detection using support vector machines and neural networks," in *2003 Symposium on Applications and the Internet*. IEEE Computer Society Press, 2003, pp. 209–216.
- [115] K. M. C. Tan and R. A. Maxion, "Determining the operational limits of an anomaly-based intrusion detector," *IEEE Journal on Selected Areas in Communication*, vol. 2, no. 1, pp. 96–110, 2003.
- [116] K. M. Tan, K. S. Killourhy, and R. A. Maxion, "Undermining an anomaly-based intrusion detection system using common exploits." in *In Proceedings of the Fifth International Symposium on Recent Advances in Intrusion Detection*, ser. Lecture Notes in Computer Science, A. Wespi, G. Vigna, and L. Deri, Eds., vol. 2516. Zurich, Switzerland: Springer-Verlag, Berlin, 2002., pp. 54–73.
- [117] E. Tombini, H. Debar, L. M, and M. Ducass, "A serial combination of anomaly and misuse idses applied to http traffic," in *20th Annual Computer Security Applications Conference*. Tucson, AZ, USA: IEEE Press, 2004.

- [118] A. Valdes and K. Skinner, "Adaptive model-based monitoring for cyber attack detection," in *Recent Advances in Intrusion Detection*, ser. Lecture Notes in Computer Science, vol. 1907. Toulouse, France: Springer-Verlag, 2000, pp. 80–92.
- [119] W. Wang and R. Battiti, "Identifying intrusions in computer networks with principal component analysis," in *The First International Conference on Availability, Reliability and Security*, 2006, pp. 270 – 279.
- [120] W. Wang, X. Guan, and X. Zhang, "A novel intrusion detection method based on principle component analysis in computer security," in *International Symposium on Neural Networks*, ser. Lecture Notes in Computer Science, vol. 3174. Springer Berlin / Heidelberg, 2004, pp. 657 – 662.
- [121] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models," in *IEEE Symposium on Security and Privacy*. Oakland, CA, USA: IEEE Press, 1999, pp. 133–145.
- [122] WIDE Project, "The widely integrated distributed environment project," In <http://tracer.csl.sony.co.jp/mawi/>.
- [123] Wikipedia, "Mahalanobis distance," 21 April 2006 2006.
- [124] M. Williams, "'immense' network assault takes down yahoo," In CNN.com at <http://archives.cnn.com/2000/TECH/computing/02/08/yahoo.assault.idg/index.html>, February 2000.
- [125] Y. Xiang, Y. Lin, W. L. Lei, and S. J. Huang, "Detecting DDOS attack based on network self-similarity," in *Proceedings of IEE Communications*, vol. 151, no. 3, June 2004, pp. 292–295.

- [126] N. Ye, S. M. Emran, Q. Chen, and S. Vilbert, "Multivariate statistical analysis of audit trails for host-based intrusion detection," *IEEE Transactions on Computers*, vol. 51, no. 7, pp. 810–820, 2002.
- [127] N. Ye, M. Xu, and S. M. Emran, "Probabilistic networks with undirected links for anomaly detection," in *IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, West Point, New York, 2000.
- [128] N. Ye, Y. Zhang, and C. M. Borrer, "Robustness of the markov-chain model for cyber-attack detection," *IEEE Transactions on Reliability*, vol. 53, no. 1, pp. 116–123, 2004.
- [129] D.-Y. Yeung and Y. Ding, "Host-based intrusion detection using dynamic and static behavioral models," *Pattern Recognition*, vol. 36, pp. 229–243, 2003.
- [130] B. Yocom, R. Birdsall, and D. Poletti-Metzel, "Gigabit intrusion-detection systems," In <http://www.networkworld.com/reviews/2002/1104rev.html> at Network World, November 2002.
- [131] J. Zhang and M. Zulkernine, "A hybrid network intrusion detection technique using random forests," in *In Proceedings of the First International Conference on Availability, Reliability and Security*. IEEE Press, April 20–22 2006, pp. 262 – 269.
- [132] T. Zhang, R. Ramakrishnan, and M. Livny., "Birch: An efficient data clustering method for very large databases," in *In Proceedings of the ACM SIGMOD International Conference on Management of Data*, ser. 2, H. V. Jagadish and I. S. Mumick, Eds., vol. 25. ACM Press, June 1996, pp. 103–114.

# Appendix A

## The Expectation–Maximization Algorithm

The Expectation-Maximization (EM) algorithm is a general method of finding the maximum likelihood estimate of the parameters of a distribution from a given data set when the data is incomplete or has missing values. There are two main applications of the EM algorithm. In the first application, the EM algorithm is applied when the data has missing values, due to problems or limitations of the observation process. The second occurs when optimizing the likelihood function is analytically intractable but when the likelihood function can be simplified by assuming the existence of and values for additional but missing (or hidden) parameters. The latter application of the EM algorithm is more commonly seen in pattern recognition.

Let us suppose that,

- $\mathcal{X}$ : Observed (incomplete data) that is generated by some distribution
- $\mathcal{Y}$ : Missing/Unobserved data
- $\mathcal{Z}$ : Complete data set such that  $\mathcal{Z}=(\mathcal{X}, \mathcal{Y})$
- $p(z|\Theta)$ : Joint conditional density function such that

$$p(z|\Theta) = p(x, y|\Theta) = p(y|x, \Theta)p(x|\Theta), \quad (\text{A.1})$$

where  $\Theta$  is the set of means and covariances and is also sometimes known as the *mixture model*. The density function described above arises from the marginal density function  $p(x|\Theta)$  and the the assumption of hidden variables and parameter value guesses.

With the density function, we can define a likelihood function as

$$\mathcal{L}(\Theta|\mathcal{Z}) = \mathcal{L}(\Theta|\mathcal{X}, \mathcal{Y}) = p(\mathcal{X}, \mathcal{Y}|\Theta). \quad (\text{A.2})$$

The likelihood function is also called the complete-data likelihood. The EM algorithm first finds the expected value of the complete-data log-likelihood  $\log(p(\mathcal{X}, \mathcal{Y}|\Theta))$  with respect to the unknown data  $\mathcal{Y}$  given the observed data  $\mathcal{X}$  and the current parameter estimates.

We use the EM algorithm to estimate the parameters of a mixture of  $k$  distributions. The multivariate Gaussian density function for vector  $\mathcal{X}$  on  $d$ -dimensional space for cluster  $j$ ,  $j \in \{1, \dots, k\}$ : is

$$p(x, M_j, C_j) = \left( \frac{1}{\sqrt{((2\pi)^d |C_j|)}} \right) e^{-\frac{1}{2}(x-M_j)^T C_j^{-1} (x-M_j)}, \quad (\text{A.3})$$

where  $M_j$  is the  $d$ -dimensional mean vector and  $C_j$  is a  $d \times d$  diagonal matrix representing the covariance vector. The index  $i$  is used for points and the index  $j$  for clusters.

The EM algorithm makes use of the Mahalanobis distance. Using Mahalanobis distance is particularly useful for skewed data having different sizes. The covariance matrix  $C_j$  is used to scale each dimension for distance computation. The squared Mahalanobis distance of point  $y_i$  to cluster  $j$  is given by

$$\delta^2(y_i, M_j, C_j) = \delta_{ij}^2 = (y_i - M_j)^T C_j^{-1} (y_i - M_j). \quad (\text{A.4})$$

The EM algorithm takes a dataset containing  $n$  points as an input. The output of the algorithm is the average logarithmic likelihood  $L(\theta)$  which is a measure of the quality of the solution and is defined as

$$L(\Theta) = \frac{1}{n} \sum_{i=1}^n \log(P(x_i; \Theta)), \quad (\text{A.5})$$

where  $\Theta$  is the *mixture model*. The algorithm clusters the dataset into  $k$  clusters.

EM starts from an approximation to  $\Theta$ . It has two major steps: the Expectation (E) step and the Maximization (M) step. EM iteratively executes the E step and the M step as long as the change in  $L(\theta)$  is greater than an accuracy threshold  $\epsilon$  or as long as a maximum number of iterations has not been reached. The E step computes  $P(x_i, M_j, C_j)$  and  $P(x_i, \theta)$ . The M step updates  $\Theta$  based on the probabilities computed in the E step. EM is theoretically guaranteed to monotonically increase  $L(\theta)$  in each iteration and to converge to a locally optimal solution.

The two major drawbacks of the EM algorithm are that it is slow to converge and for many real world problems, the E- or M-steps may be analytically intractable. To alleviate this and other problems, a number of strategies have been proposed. The most important

previous work comes from the machine learning community. Neal *et al.* [85] introduce the idea of using sufficient statistics<sup>1</sup>. In this work, the authors analyze several variants of EM, conceptualizing the logarithmic-likelihood optimization as an optimization of an energy function. The key idea behind one of the variants presented in their work is to use sufficient statistics. They describe a simple mixture problem with a dimensionality of one and a cluster size of two. They also show experimental evidence of the superiority of their approach. However, their study is incomplete as it does not address the problem of clustering large datasets with high dimensionality. Problems related to numerical stability, proper initialization, and outlier handling are not addressed in their work either. In [13], Bradley et al. present a scalable EM (SEM) algorithm that iterates in memory and also summarizes points through their sufficient statistics. To avoid locally optimal solutions, they re-seed empty clusters and estimate several models concurrently, sharing information across models. The scheme presented by the authors, requires the user to specify what fraction of the working buffer should be from the entire database. The value for this parameter is typically 1%. The authors, however, do not explain why this value gives the best results. BIRCH [132] and newer extensions of the classical EM algorithm [13, 89] use data summarization to accelerate convergence.

---

<sup>1</sup>A quantity  $T(X)$  that depends on the (observable) random variable  $X$  but not on the (unobservable) parameter  $\theta$  is called a statistic. A sufficient statistic captures all of the information in  $X$  that is relevant to the estimation of  $\theta$ .

# Vita

Animesh Patcha received his B.E. degree in Electrical and Electronics Engineering from Birla Institute of Technology in Mesra Ranchi in 1998, and his M.S. degree in Computer Engineering, with a certificate in Networking and Telecommunications, from the Illinois Institute of Technology in Chicago, IL in 2002.

From January 1999 to December 2000, Animesh was a senior software engineer at Zensar Technologies in Pune, India. His main job responsibilities involved designing and developing software based solutions to handle a variety of problems in the Intranet and Internet environment.

From fall 2002 to summer 2006, Animesh worked as a graduate research assistant in the Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Virginia. While at Virginia tech, Animesh has also been actively involved with the Office of Distance Learning and Computing, where he was a graduate assistant from August 2004 to January 2006. His duties, as a graduate assistant, included studying the quality of service issues involved in implementing Video over IP solutions on a campus network. He was also actively involved in making necessary recommendations on the technical as well as business aspects of such an implementation.

His current area of research is computer and network security in wired and wireless networks. Papers, book chapters and technical reports co-authored by Animesh Patcha publications, are listed below.

## List of Publications

### • Journals and Magazines

- 1 A. Patcha and J. Park, “An Overview of Anomaly Detection Techniques: Existing Solutions and Latest Technological Trends”, (*Submitted awaiting decision*) *The Elsevier Journal on Computer Networks*.
- 2 A. Patcha and J. Park, “Network Anomaly Detection with Incomplete Audit Data”, (*Submitted awaiting decision*) *The Elsevier Journal on Computer Networks*.
- 3 A. Patcha and J. Park, “A Game Theoretic Formulation for Intrusion Detection in Mobile Ad Hoc Networks”, *International Journal of Network Security*, Vol. 2, No. 2, pp. 146 -152, March 2006.
- 4 A. Mishra, K. Nadkarni, and A. Patcha “Intrusion Detection in Wireless Ad Hoc Networks”, *IEEE Wireless Communications Magazine*, Vol. 11, No. 1, pp. 48 - 60, February 2004.

### • Conferences, Symposiums and Workshop Proceedings

- 1 A. Patcha and J. Park, “Network Anomaly Detection with Incomplete Audit Data”, Poster Paper, *In Proceedings of the 16<sup>th</sup> Annual Wireless Symposium and 2nd Annual Wireless Summer School*, Blacksburg, VA, June 7-9 2006
- 2 A. Patcha and J. Park, “An Adaptive Sampling Algorithm with Applications in Denial-of-Service Detection”, *In Proceedings of the 15<sup>th</sup> IEEE International*

- Conference on Computer Communications and Networks (ICCCN'06)*, October 2006.
- 3 A. Patcha and G. Scales. "Next Generation Technologies for Distance Learning: Same Time, Anytime, Anywhere", *In Proceedings of the 2006 Annual Conference and Exposition of the American Society of Engineering Education*, Chicago, IL, June 17-21 2006.
  - 4 A. Patcha and J. Park, "Detecting denial-of-service attacks with incomplete audit data", *In Proceedings of the 14<sup>th</sup> IEEE International Conference on Computer Communications and Networks (ICCCN'05)*, pp. 263 - 268, San Diego, CA, October 2005.
  - 5 A. Patcha and G. Scales. "Development of an Internet based distance learning program at Virginia Tech", Poster Paper, *In Proceedings of the Annual Conference of the ACM Special Interest Group for Information Technology Education*, Newark, NJ, USA, October 2005.
  - 6 A. Patcha and J. Park, "A Game-Theoretic Approach to Modeling Intrusion Detection in Mobile Ad Hoc Networks", *In Proceedings of the 2004 IEEE Information Assurance Workshop, United States Military Academy, West Point, NY*, pp. 280 - 284, June 2004.
  - 7 S. J. Sane, A. Patcha, and A. Mishra, "Quality of service routing in wireless ad hoc networks", *In Proceedings of ITCOM'03, Internet Quality of Service*, M. Attiquzaman and M. Hasan, editors, Vol. 5245, pp. 125 - 130, September 2003.
  - 8 A Patcha and A Mishra, "Collaborative security architecture for Black hole attack prevention in Mobile Ad Hoc networks", Poster Paper, *In Proceedings of the 2003 IEEE Radio and Wireless Conference*, pp. 75 - 78, Boston, MA, USA, July 2003.

- **Book Chapters**

- 1 A. Patcha and J. Park, “A Survey of Anomaly Detection Techniques” (*Accepted, In Submission*), *Encyclopedia of Networked and Virtual Organizations*, Editors Goran D. Putnik and Maria Manuela Cunha, Idea Group Reference, 2007