PROBABILISTIC MODEL FOR DETECTING NETWORK TRAFFIC ANOMALIES

A thesis presented to

the faculty of

the College of Engineering and Technology of Ohio University

In partial fulfillment

of the requirements for the degree

Master of Science

Ramani Yellapragada

March 2004

This thesis entitled

PROBABILISTIC MODEL FOR DETECTING NETWORK TRAFFIC ANOMALIES

BY

RAMANI YELLAPRAGADA

has been approved for

the School of Electrical Engineering and Computer Science

and the Russ College of Engineering and Technology by

Shawn D. Ostermann

Chair of Electrical Engineering and Computer Science

R. Dennis Irwin

Dean, Russ College of Engineering and Technology

YELLAPRAGADA, RAMANI. M.S. March 2004
Electrical Engineering and Computer Science

Probabilistic Model for Detecting Network Traffic Anomalies (NUMBER-OF-PAGES 102)

Director of Thesis: Shawn D. Ostermann

Anomaly-based intrusion detection is a research area in Computer Security, wherein computer and network attacks are differentiated from normal computer interactions. Anomaly-based intrusion detection systems detect attacks by analyzing either computer or network data and flagging abnormalities as intrusions. The abnormalities are detected by analyzing certain parameters that are present in the data. Our approach analyzes certain network parameters, which characterize either a connection or a network service on a monitored host or a network service on a monitored network. This categorization of parameters helps detect varied classes of attacks including denial-of-service, port scan and buffer overflow attacks.

Anomaly-based systems use various analysis techniques to detect parameter anomalies. A new approach based on Bayesian Networks technique for analyzing and detecting anomalies is presented here. The advantage of Bayesian Networks lies in their ability to adaptively learn normal values of parameters without much training, which makes it suitable for real-time analysis. Bayesian Network can be used to combine current evidence and previous knowledge to obtain the probability of anomaly. This property helps in detecting previously seen attacks faster, since the previous knowledge provides strong evidence of an attack. The same property helps reduce the number of false positives, since considerable evidence needs to accumulate for the Bayesian Network to report high probability of anomaly.

Approved:

Shawn D. Ostermann

Chair of Electrical Engineering and Computer Science

ACKNOWLEDGEMENTS

I thank my advisor, Dr. Ostermann, for providing me the opportunity to work with him in the area of my research interest. I could not have perceived this thesis to my expectations without his suggestions, which have been instrumental to my research. I appreciate his valuable guidance and feedback throughout my thesis.

The Internet Research Group laboratory has provided a stimulating environment to fruitify my research ideas. I thank all the members of IRG and INBOUNDS research groups for their valuable suggestions.

I am greatly indebted to my parents, brothers Srikanth and Ravi for their constant words of encouragement and support during the course of my Masters at Ohio University. It is the unending support and understanding from my fiancée Prashant that resulted in the successful completion of my thesis.

I dedicate this thesis to my mom.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

The invention of computers has satisfied the demand for rapid access and processing of information. The development of the Internet has further aided in effective transfer and sharing of information. However, there are substantial risks with the convenience and ease of access of information using computers and the Internet. A computer connected to the Internet is more vulnerable to attacks than an isolated one. Computer systems can be compromised or misused to obtain sensitive information such as credit card numbers, passwords, etc.

The Computer Security of a system is based on the realization of the confidentiality, integrity, and availability of its information [31]. Confidentiality requires that information be accessible to only those who are authorized for it. Integrity requires that information remain unaltered by temporary system failures, accidents, or malicious attempts. Availability requires that information be accessible to authorized users when they need it. A system has to protect its data, and resources from being misused to realize the above principles. Mechanisms such as security policies, password authentication, and firewalls can be used to provide the protection. However, incidents such as the Morris Worm attack [46] in 1988 prove that security of systems has been compromised in spite of the use of one or more of these mechanisms. Further, the Internet had aided in the rapid spread of such worms and viruses, exposing computer systems to a wide variety of network attacks.

1.1 Network Security

Worms such as Slammer Worm [24], and Blaster worm [23] that attacked the Internet in 2003, provide examples of the large scale destruction such network attacks

can cause. The availability of various vulnerability assessment tools on the Internet has caused the rise of network security breaches. Tools such as Nmap [8], SAINT [10], Back Orifice [2], and SATAN [11] can be used to scan, identify, probe, and thereby attack systems. Thus, there is a need to secure computer systems from the various network attacks.

The various techniques that can be used to secure computers are stated below:

- Fixing software bugs that are being exploited, is the best way of protecting systems. However, this method is not always possible due to the unavailability of open source code to the users of certain software products.

- Authenticating users through mechanisms such as passwords could protect systems from illegitimate access. However, a weak password, or an exploit that thwarts the authentication mechanism can compromise such a system.

- Using firewalls at the point of entry into the network could prevent certain kinds of network attacks. However, a mis-configured firewall, or an insider attack can cause such a protection to fail.

1.2 Intrusion Detection Systems

Firewalls and authentication mechanisms are effective in protecting systems from attacks that emanate from outside the network but lack capabilities to monitor network traffic. Thus, these mechanisms could be compromised by an insider attack, or a breach that exploits a certain misconfiguration. Intrusion Detection Systems (IDS) complement these mechanisms by monitoring the network traffic, and flagging any suspicious behavior as an intrusion. An IDS acts as an alarm system whose alerts can be used by the System Administrator to take necessary action such as enforcing a firewall rule, in order to prevent further propagation of the attack.

Intrusion Detection (ID) is the art of detecting inappropriate, incorrect, or anomalous activity [71]. ID Systems (IDS) accomplish the task of Intrusion Detection by

collecting information from a variety of system and network sources, and analyzing the information for possible security breaches. The following capabilities are provided by an IDS, according to a SANS article by Rozenblum [69]:

1. Monitor, and analyze intrusive activity
2. Recognize, and report alterations to data
3. Automate the task of monitoring the network for well-known and novel attacks
4. Alert the system administrator in case of an attack, so that an appropriate action can be taken to prevent further propagation of the attack

An IDS needs to possess the following key features, according to a SANS article by Ho [39]:

1. Robustness: An IDS should run continually, and autonomously. It should be fault-tolerant, and impervious to attacks.
2. Flexibility and Scalability: An IDS should be easily configurable and flexible to changes in network infrastructure. Also, an IDS should be able to handle large amounts of traffic with high performance capabilities.
3. Ease of Use: An IDS should be managed with minimum amount of resources such as bandwidth, CPU power, and other network devices. It should also be easily deployable on a host or in a network, without affecting their operation.

An IDS can be classified as either host-based, or network-based depending on the source of data, and deployment point. A host-based IDS performs intrusion detection on a particular host either by monitoring system activity, or evaluating the system logs. A network-based IDS performs intrusion detection by analyzing the packets traveling across the monitored host, or network.

An IDS can be classified as either misuse-based, or anomaly-based depending on the analysis method used. In a misuse-based IDS, intrusions are identified as system activities that follow defined patterns of attack. Such systems define signatures to identify the attack patterns. Anomaly-based systems detect intrusions based on the anomalous behavior of computer components, and flag any irregular behavior as potentially intrusive. Such systems model normal behavior and detect deviations from the normal behavior. Anomaly-based systems have the advantage that they can detect novel attacks in addition to well-known attacks.

In this thesis, we develop a network-based and anomaly-based Intrusion Detection System for detecting intrusions. We apply the Bayesian Network technique to analyze network traffic, and detect anomalies. An overview of this technique is provided in section 2.2. We also provide a taxonomy of Intrusion Detection Systems in section 2.1, which helped us choose Bayesian Networks after evaluating various other techniques, their advantages and disadvantages when applied to intrusion detection.

1.3 Organization of Thesis

In chapter 2, we present previous work on intrusion detection, and an overview of Bayesian Networks. Chapter 3 elaborates on the application of Bayesian Network technique for intrusion detection in INBOUNDS. Chapter 4 presents the experimental results. In chapter 5, we assess the system that is developed, and provide the conclusions and future work for our research.

2. Background

This chapter provides the background required to understand the intrusion detection system developed for our research. A taxonomy of certain existing intrusion detection systems is presented in section 2.1. This is followed by an overview of Bayesian Network technique in section 2.2, which we applied for developing the system presented in this thesis. Finally, an overview of the techniques previously used for intrusion detection in INBOUNDS is presented in section 2.3.

2.1 Taxonomy of Intrusion Detection Systems

A knowledge of existing technologies or methods in a particular field helps the researchers keep abreast of the current research. However, a judgment on the pros and cons of those methods becomes evident only when the concepts are organized. A taxonomy provides such a platform to capture the essence of the knowledge that resides in those technologies. In this section, we attempt to classify Intrusion Detection Systems, first based on the Data Source of the IDS, and second, based on the Detection Method used by the IDS. This taxonomy is not complete since there might be certain IDSs that do not fall under the categories presented in this taxonomy. Related work that provides taxonomies of Intrusion Detection Systems can be found in Axelsson et al. [17], Kumar et al. [51], Werrett et al. [80], and Debar et al. [38].

2.1.1 Classification Based on Data Source

One of the components of an IDS is the data source, whose output is evaluated in order to detect intrusions. One approach is to classify IDSs based on the nature of the data source:

- Host-based IDS: Such an IDS is deployed on a host computer. The IDS moni-

tors host activity which might include system processes, registry entries, CPU usage, file accesses, and system audit logs and flags any suspicious activity as an intrusion. The earliest host-based IDS was developed in 1990 called Tripwire [32]. Later, other such IDSs were developed including Haystack [74]. However, a host-based IDS lacks the ability to detect network attacks. Also, the IDS becomes unreliable once the host is compromised.

- Multi host-based IDS: Since the evolution from standalone to networked computers, IDSs were adopted to evaluate data from multiple hosts. NIDES [29], AAFID [30], and CSM [33] are certain multi host-based IDSs, which collect data from multiple hosts, and evaluate them either at a centralized location or in a distributed host environment.

- Network-based IDS: Such an IDS is deployed on a network segment that monitors a particular host or, a network of hosts. The captured network data is evaluated by the IDS for any malicious network activity. Some of the network-based IDSs are NADIR [45], CyberCop [3], and SNORT [13]. A network-based IDS has to handle large amounts of traffic without suffering a significant latency between the time of attack and the time of alert.

- Hybrid/Hierarchical IDS: Host-based, and Network-based IDSs could complement each other to provide improved intrusion detection capability. A Hybrid/Hierarchical IDS combines the advantages of both host-based, and network-based IDS by analyzing both the input data from the hosts, and the network data to the hosts being monitored. Examples of such an IDS include EMERALD [63], and GrIDS [70].

2.1.2 Classification Based on Detection Method

A second approach to classify IDSs is based on the detection method used:

- Misuse-based IDS: Such an IDS detects intrusions that follow well-defined patterns of an attack. The system defines knowledge about bad or unacceptable behavior, and seeks to detect it directly. A misuse-based system is effective at detecting well-known attacks, but has the disadvantage of not being able to detect novel attacks.

- Anomaly-based IDS: Such an IDS detects intrusions by learning the normal behavior of computer components, and flagging any anomalies observed as intrusive. Thus, the system can detect new kinds of attacks without any previous knowledge of the attack. However, such a system has the drawback of flagging normal behavior as an attack in certain cases. Such a false interpretation of normal behavior is termed a *false positive*, which occurs due to incorrect representation of normal behavior. Thus, reducing false positives is one of the goals during the development of an anomaly-based IDS.

Misuse-based, and Anomaly-based IDSs can be further classified based on the detection principle, or the method that they use for intrusion detection. Table 2.1 shows the classification. There is no established terminology for the methods, or principles used by IDSs. Hence, we coined new terms based on principles common to IDS technologies. There might be terms in the field that have different definitions for the terms used here. The taxonomy presented here does not provide a standard classification of all the existing IDSs, and only provides a guideline for understanding the various IDS technologies.

2.1.2.1 Misuse-Based IDS

In a misuse-based IDS, an intrusion decision is based on the knowledge of an intrusive process. Intrusive and non-intrusive behaviors are clearly distinguished, and the current behavior is compared with this knowledge. A match with the intrusive behavior is flagged as an attack.

A misuse-based IDS is either user-programmed, or dynamic.

Table 2.1 A Taxonomy of Intrusion Detection Systems

Misuse-based IDS	
User Programmed	Simple rule-based String matching Keystroke monitoring
Dynamic	Model-based Expert system State Transition Analysis
Anomaly-based IDS	
Statistical-based	Threshold Multi-variate statistical
Pattern-based	HMM State-based Time-based
Feature selection	Genetic Data Mining
Feature relationship-based	Neural networks Correlation Probability-based

- **User-programmed:** In such a system, the user programs the detection rules with which the current observation is compared. The detection rule contains a straightforward coding of what can be expected in the event of an intrusion. The various methods that have been adopted for defining the detection rules are:

    1. Simple rule-based: Simple rules similar to firewall rules are defined by the user with respect to the state of the system. If the current state matches any of these rules, it is flagged as an intrusion. SNORT IDS [13] follows a rule-based approach by defining signatures for network patterns. If the network data matches any of these signatures, an intrusion is flagged. Other such IDSs include BRO [64], NADIR [45], and Haystack [74]. The drawback of the approach is that the detection of attack is sensitive to the rules that are defined. An incorrect rule might lead to either a false positive, or a false negative.

    2. String matching: String matching is a simple matching of suspicious characters in the text that is transmitted over the network with strings that define an attack. This method might be case-sensitive and less flexible, yet has been adopted due to its simplicity for string matching in audit logs, and network data. NSM [77] is one such IDS that uses string matching for detecting intrusions.

    3. Keystroke monitoring: This technique utilizes user keystrokes to determine the occurrence of an attack. The primary means is to match the pattern of specific keystroke sequences with sequences that indicate an attack. The disadvantage is the lack of reliable mechanisms for user keystroke capture.

- **Dynamic:** These systems combine models of misuse with evidence of misuse to support conclusions about the occurrence of a misuse event. The similarity of

an event to the defined model of misuse, and the conditions of event occurrence are combined to obtain a decision on intrusion. The various methods are:

1. Model-based: In this method, a database of scenarios that comprise a sequence of misuse behavior is defined. A search is made through this database to match the current event with a subset of the scenarios. If the first subset does not match, the next subset is considered and the search continues. Based on this search, evidence is accumulated and the likelihood of an attack is calculated. A system based on this technique is described in Garvey et al. [76].
2. Expert system: This system encodes knowledge about attacks as if-then implications, and asserts facts corresponding to events. For a particular event, when all the if rules are satisfied, the action event of flagging an intrusion is performed. For this method to be effective, the data should have an inherent ordering. One such system is described in Snapp et al. [73].
3. State transition analysis: This method encodes the intrusion as a number of different states, each of which has to be present in the observation space. The transitions between the different states are applied to the current event, and intrusion is flagged based on the final state. IDSs that use this method include USTAT [50], and IDIOT [51].

Misuse-based IDSs are effective in detecting known attacks with few false positives. However, they lack the capability of detecting novel kinds of attacks.

2.1.2.2 Anomaly-Based IDS

In an anomaly-based IDS, abnormalities in a particular host, or network behavior are flagged as intrusions. The construction of such a detector starts by defining what constitutes the normal behavior, and then looking for deviations from the defined nor-

mal behavior. The behavior is generally characterized using certain host, or network parameters. Various kinds of anomaly-based IDSs are explained below:

- **Statistical-based:** Such a system builds a profile for the parameters based on their statistics. These statistics could be simple statistics such as mean and standard deviation of the parameters, or complex statistics such as covariances. The current parameter values are compared against the parameter values in the profile, and a deviation is flagged as an intrusion. The various methods that have been used to-date are:

    1. Threshold-based: This is a statistical-based method where the statistics are the mean, and standard deviation values of the parameters under consideration. These values are used to calculate the level of anomaly and a large deviation of the level of anomaly from a pre-defined threshold is considered an intrusion. Some of the systems that use this method are NIDES [29], and statistical module of INBOUNDS [18]. However, there are certain disadvantages to systems that use this method. These systems are insensitive to the time, and order of occurrence of events. It is difficult to determine the thresholds such that there are few false positives, while effectively detecting actual intrusions.

    2. Multivariate statistical: This method is an improvement over the statistical-based method. Cluster analysis is used in this method to formulate normal traffic into distinct clusters. The attacks are then considered as those that are sufficiently anomalous to stand out from these clusters. One system that applies this method for intrusion detection is NATE [22].

- **Pattern-based:** In such a system, anomaly detection is performed by modeling the input data space as a pattern within a certain dimension such as time, or sequence of occurrence of a particular event. A deviation of the current event

from the modeled pattern is flagged as an intrusion. Following are some of the pattern-based techniques:

1. Hidden Markov Models (HMM): HMMs are modeled using parameters of the system, and their transition probabilities at every time interval. The parameters in the current time frame are estimated from the parameters in previous time frame using HMM inference methods. For intrusion detection, HMMs are constructed to model the user's or data's historical behavior, and a similarity measure is calculated for the current parameter values. If the similarity measure is deviant from the past values, an anomaly is flagged. A system that applies HMMs for intrusion detection is described in Lane et al. [52]. HMMs are suitable for one-dimension sequence classification like voice wave or spectrum. However, typical anomaly detection data are multi-dimensional sequences with many variables involved. Since the processing time complexity of HMMs is proportional to the number of variables, HMMs are not cost-effective for the purpose of intrusion detection.

2. State-based Models: This is a variation of HMM, where the modeling is done based on states of the parameters rather than parameters itself. The advantage is that a small number of state transitions would suffice to detect intrusions in contrast to HMMs that require large number of parameter transitions to encompass varied attacks. This model has been applied to a system described in Sekar et al. [67]. In this system, network protocols were incorporated as state machines, and intrusions were detected by analyzing anomalies in the state transitions. The system showed a high detection rate for well-known and novel attacks, and low false positive rate. However, the drawback of the approach was that a large amount of data was required to build complex state-based models for network protocols.

3. Time-based Models: This is a pattern-based method where the variables are temporal events, and the inter-arrival times of events are used to estimate the probability of the occurrence of a new event. If this probability is too low, an intrusion is flagged. This approach has been adopted for a system described in Chen et al. [28]. The system defines time-based rules to characterize the normal behavior patterns of users. A deviation is detected if the current observed sequence of events do not match the defined rules. The drawback of this method is that it is difficult to define all the possible event sequences, which leads to false positives when event sequences are not matched.

- **Feature Relationship-based:** Features that characterize a particular system may not be independent. For example, CPU usage is related to the number of users connected to the system. Hence, a system that combines all the features to derive the state of the system might be more effective in analyzing the system. Feature relationship-based methods use models to combine the features of a system to effectively detect intrusions. Various available methods are:

  1. Neural Networks: The basic approach is to train a neural network on certain data variables, which are represented as a vector in the data input space. The input to the neural network is a set of such input vectors, from which the neural network learns the behavior of the data input. At the end of the learning phase, the neural network is said to be trained on a set of representative data points that represent the input space. During analysis, current data is compared to the trained neural network and a deviation is flagged as an intrusion. The relationship between the input parameters is implicit in neural networks. Neural networks were first applied in 1990 to a system described in Fox et al. [48]. A neural network method called Support Vector Machines was applied for intrusion detection as described

in Mukkamala et al. [72]. Also, a neural network method Self-Organizing Maps (SOM) was applied for intrusion detection in INBOUNDS, described in Ramadas et al. [55]. The intrusion detection mechanism of SOM for INBOUNDS is briefly described in section 2.3.4.2. The disadvantage of neural networks is that the topology, and coordinate values of the neural network are learned after considerable trial and error, and hence large amounts of training data are required.

2. Probability-based: In probability-based approach, the normal behavior values of the parameters are measured as probabilities. These values are then used to estimate the probability of the new data to be either normal, or anomalous. Staniford et al. [75] outlines an approach to detect network scans by examining the packet's probability for expecting to see a particular source, or destination IP address or port. A low probability indicates the possibility of a network scan. Nong et al. [60] presents another probabilistic method for learning long term profiles of normal activities in computer, and network systems. An intrusion is flagged when a deviation is measured in the observed activities from the built profiles.

   Bayesian Networks fall under the category of probability-based methods. In addition to estimating the probabilities of parameters, it also provides a means of relating the parameters through conditional probability tables.

3. Correlation-based: Another way of relating the parameters of a system is by calculating the correlations between them. If measures $A_1, \ldots, A_n$ are the parameters and are represented by the vector A, then the compound anomaly measure is determined as:

$$A^T C^{-1} A$$

   Here, C is the covariance matrix representing the dependence between each pair of anomaly measures A_i and A_j, and A^T is the transpose of A.

NIDES [29] is one system that used a method called covariance matrices to account for the interrelationships among measures. The current trend in intrusion detection is to apply correlation methods to relate the various intrusion alerts. Based on the obtained result, an alert is either considered as an intrusion, or a false positive. Thus, this method is used to filter the false positives generated by the IDSs. Systems that applied correlation methods to IDS alerts are described in Valdes et al. [16], and Debar et al. [37].

The above taxonomy was used to evaluate the various techniques applied to existing intrusion detection systems, their drawbacks, and the purpose for which they were more suitable. The properties of Bayesian Networks in estimating the probabilities, and relating parameters was estimated to be useful for reducing the false positives, and aiding in better detection. Thus, the taxonomy has guided us in selecting Bayesian Networks for analyzing network data in order to detect intrusions. Application of Bayesian Networks for intrusion detection in INBOUNDS is explained in chapter 3. Section 2.2 gives an overview of Bayesian Network technique.

2.1.3 Feature Selection

The important criteria for effective intrusion detection is to choose appropriate features that accurately classify intrusions. In general, these measures are chosen heuristically but might not be adequate to detect all kinds of intrusions. Feature selection is a method where an exhaustive search is conducted to choose the optimal subset of features. Even though feature selection is not used for intrusion detection, they are described here as they can be used to complement IDSs in choosing features most suitable for analyzing intrusions. One such feature selection technique is described here.

Genetic algorithm: In this method, a genetic approach is applied to search through the possible subsets of initial measures. If n is the number of initial mea-

sures, there are 2^n subsets and searching through them exhaustively is not efficient. Heady et al. [68] describes a genetic algorithm to search through this space efficiently. In this method, the initial measures are taken, and genetic operations of crossover and mutation are applied to obtain the subsets. The subsets that exhibit low probability of predicting intrusions are removed, and are replaced by applying genetic operators to yield stronger measure subsets. Lee et al. [79] describes an IDS that used genetic algorithm for selecting features that were consistent, and useful for intrusion detection. Then, Data Mining technique was applied for detecting intrusions using these features.

2.2 Bayesian Networks

Bayesian Network is an artificial intelligence technique that has been an important area for research, and application in various fields. Following are certain areas in which Bayesian Networks have been applied successfully:

- Bayesian Networks was adopted by the National Aeronautic and Space Administration for the AutoClass project [42]. The project is an attempt to create Bayesian applications that can be used for deep-space exploration, and knowledge acquisition.

- Bayesian Networks have also been applied in searching solutions to certain NP-hard problems. Bayesian Networks replace traditional heuristic methods by introducing a method, where probabilities for feasible solutions are calculated and updated continually during search.

- Bayesian Networks was applied to a project Lumiere [40] by Microsoft, which was used in its "Office Assistant" for the Office 95 suite. The software interacts with its users by anticipating their goals, and needs.

- Bayesian Networks have also been used in spam filters to eliminate unwanted messages from a user's mail stream [56].

The wide usage of Bayesian Networks was recognized by Intel, which announced the release of Bayesian network software libraries in May, 2003 [41], to aid software developers. The following sections give an overview of Bayesian Networks, their methods, and algorithms that are applied to our research.

2.2.1 Overview

A Bayesian Network is a directed graphical model that encodes relationship between the parameters of a particular domain into its structure, and probability tables [65].

2.2.1.1 Representation

The basic representation of a Bayesian Network is a directed graph. Each node in the directed graph represents either a parameter of the domain under study, or a hypothesis that is accorded certain belief. For example, a node might either represent "Number of days in February", or "This year is a leap year". The numerical value at each node represents the degree of belief accorded to the parameter value, or to the hypothesis. The parameters are random variables, which possess discrete state values of the data domain. Thus, each node has discrete values equal to the number of states of the corresponding parameter, or degrees of belief of the corresponding hypothesis. The links between the nodes represent the relationships between the parameters [57]. The relationships between the parameters are quantified using conditional probabilities, which are calculated using Bayes Theorem. The conditional probability gives the probability that the parameter, or hypothesis has a certain value given the value of the parameter, or hypothesis to which it is related. There are numerous variations of Bayesian Networks, based on the model of the graph, which can be either directed or undirected, tree or a graph, etc. The various models are described in Murphy et al. [59] and the various probabilistic methods applied to these models are described

in Jensen et al. [43]. For our research, we have adopted the graphical model called *causal trees*, which is tree-structured. An example causal tree is shown in figure 2.1.

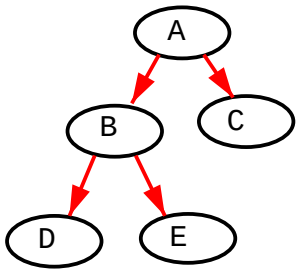

Figure 2.1. Example Causal Tree

In the tree shown, an arrow from node A to node B can be informally interpreted as "A causes B".

2.2.1.2 Bayes Theorem

The conditional probabilities at a particular node of the Causal Tree are calculated using Bayes Theorem. According to Bayes Theorem:

$$P(H/e) = \frac{P(e/H)P(H)}{P(e)}$$

The theorem states that the belief we accord to a hypothesis 'H' upon obtaining evidence 'e' can be computed by multiplying our previous belief P(H) with the likelihood P(e/H) that 'e' materializes if 'H' is true. P(H/e) is also known as posterior probability, and P(H) is called prior probability or belief. The denominator P(e) is calculated as:

$$P(e) = P(e/H) * P(H) + P(e/\neg H) * P(\neg H)$$

However, P(e) is a normalizing constant, which can be computed by requiring that P(H/e) and P(¬ H/e) sum to unity [65]. For a causal tree, evidence is the directly observable value of a parameter at the node. Hypothesis is the unknown probability of an event that we need to calculate using Bayes Theorem.

The significance of Bayes Theorem can be explained with an example. If there are two kinds of dice, fair dice, and loaded dice. Given probability of choosing fair dice is 99%, and that of loaded dice is 1%. The loaded die is loaded such that a six comes up 50% of the time. A die is picked and it has been rolled thrice to get a six. If we want to calculate the probability that the die is loaded i.e., we need to calculate the probability P(D_{loaded}/3 sixes), it can be calculated using Bayes Theorem as:

$$P(D_{loaded}/3sixes) = \frac{P(3sixes/D_{loaded}) * P(D_{loaded})}{P(3sixes)}$$

This value is calculated as

$$P(D_{loaded}/3sixes) = \frac{(0.5^3)(0.01)}{(0.5^3)(0.01) + (1/6^3)(0.99)} = 0.21$$

This implies that the probability of the die being loaded is only 0.21. However, for the same example if the six was consecutively seen six times, the probability of the die being loaded becomes,

$$P(D_{loaded}/6sixes) = \frac{(0.5^6)(0.01)}{(0.5^6)(0.01) + (1/6^6)(0.99)} = 0.88$$

The probability that the die is loaded is now a higher value i.e., 0.88. This means that as evidence is accumulated for a particular hypothesis event, the probability of the hypothesis being true increases.

The essence of Bayes Theorem is seen by using *odds* and *likelihood* ratio parameters. Consider the complementary form P(¬ H/e), which is calculated using Bayes Theorem as:

$$P(\neg H/e) = \frac{P(e/\neg H)P(\neg H)}{P(e)}$$

Then,

$$\frac{P(H/e)}{P(\neg H/e)} = \frac{P(e/H)P(H)}{P(e/\neg H)P(\neg H)}$$

Defining prior odds on H as:

$$O(H) = \frac{P(H)}{P(\neg H)}$$

Defining likelihood ratio as:

$$L(e/H) = \frac{P(e/H)}{P(e/\neg H)}$$

If posterior odds is defined as:

$$O(H/e) = \frac{P(H/e)}{P(\neg H/e)}$$

Then, posterior odds can be derived as:

$$O(H/e) = L(e/H)O(H)$$

Thus, Bayes Theorem implies that the overall belief in a hypothesis H is based on both the previous knowledge and the observed evidence, and is thus the product of prior odds O(H) and likelihood ratio L(e/H). The first term measures prior support accorded to H by background knowledge, and the second term measures current support given to H by the evidence.

Bayes Theorem is applied to calculate the numerical probability at each node of the causal tree. Since there can be multiple values to the node that represents the parameter or hypothesis, the probability is to be calculated for each of these values. As described before, links between the nodes of the tree represent the relationships between the parameters. These relationships are given by a matrix called conditional probability table (CPT), whose elements are conditional probabilities. The conditional probability is the probability that the child is in a particular state for a particular state of its parent node. Thus, the number of rows in the CPT equals the number of states of the parent and number of columns equals the number of states of the child.

2.2.2 Belief Updating in Bayesian Networks

A Bayesian Network has two kinds of nodes:

- Nodes whose values are directly observable, or already known. These known node values are termed as evidence.
- Nodes whose values are not known and need to be calculated from the evidence. These unknown node values are termed as hypothesis or belief.

Bayes Theorem is used to calculate the probability value at each of its nodes. This calculation is initiated each time new evidence is obtained. The evidence is combined and propagated, initiating calculation of beliefs at each of the nodes in the tree. The propagation between the nodes is achieved by passing messages. The steps involved in fusing the evidence, propagating them through the network, and calculating the beliefs based on the evidence is called Belief Updating. Certain rules govern the Belief Updating procedure in a causal tree, which are explained below using a simple example network in figure 2.2.

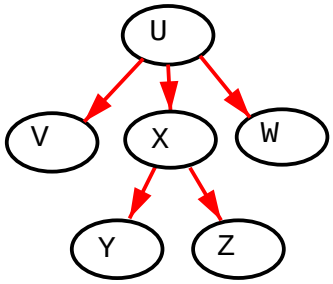

Figure 2.2. Example to Illustrate Belief Updating

As discussed in section 2.2.1.2, the parent and child nodes in a causal tree are related through a Conditional Probability Table (CPT). If X is the parent of Y, then an element of the CPT at Y is given by

CPT(i,j) = P(state of Y=j/state of parent X=i)

P is the conditional probability that the value of child node is j, given that the value of its parent node is i. For convenience we represent the CPT at node Y, whose parent is X, by the matrix $M_{y/x}$.

The belief at node X in figure 2.2 is calculated using Bayes Theorem as:

$$Bel(X) = P(X/e) = \frac{P(e/X)P(X)}{P(e)}$$

Here, e is the evidence, which is obtained from the child nodes of X in the Bayesian tree i.e., Y and Z.

The above formula can be written as

$$Bel(X) = \alpha\pi(X)\lambda(X)$$

Bel(X) is the belief at node X.

π(X) is the prior probability at X i.e., P(X), which is obtained from the belief of the parent of X (i.e. U) as $M_{x/u}$. The belief from parent of X is passed as a π message from U. A special case that applies here is for a root node, whose prior probability is given by the prior probability of the node itself rather than that of its parent.

λ(X) is P(e/X), which is obtained from the beliefs of the children of X. The beliefs at each of the children of X (i.e. Y, and Z) is given by $M_{y/x}$, and $M_{z/x}$. The beliefs at the children are fused by multiplying $M_{y/x}$, and $M_{z/x}$. The beliefs calculated at Y, and Z are passed as λ messages to node X.

α equals $\frac{1}{P(e)}$ and is merely a normalization factor.

Once the belief at node X is calculated, the new belief needs to be propagated to its parent, and child nodes. This belief is passed as π messages to nodes Y and Z (child nodes of X), and as λ message to node U (parent node of X). The belief at these nodes are then calculated in turn. Thus, beliefs are calculated at each node of the tree for every new evidence. The message propagation in the tree of figure 2.2 is shown for node X in figure 2.3.

In the figure, π_X(u) is the π message propagated from parent U to child X. λ_X(u) is the λ message propagated from child X to parent U. π_Y(x) is the π message propagated

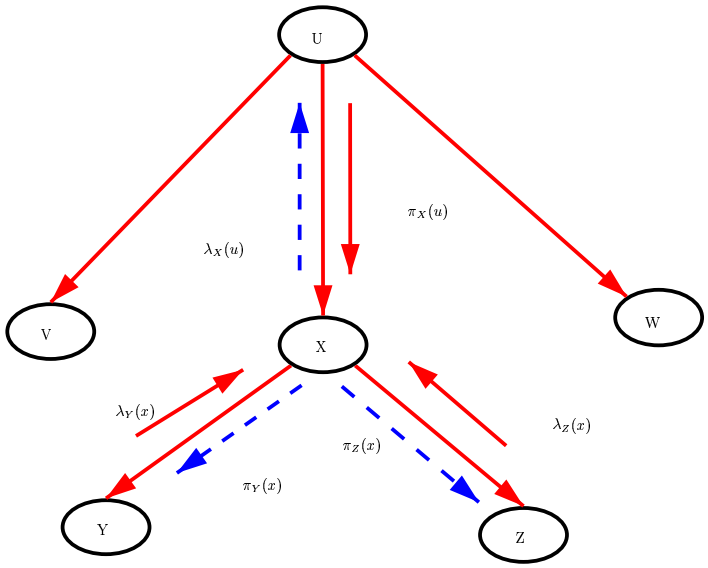

Figure 2.3. Message Passing in Causal Tree

from parent X to child Y. π_Z(x) is the π message propagated from parent X to child Z. λ_Y(x) is the λ message propagated from child Y to parent X. λ_Z(x) is the λ message propagated from child Z to parent X.

The calculation of π, and λ messages can be conveniently explained by using matrix mathematics. The π, and λ messages can be represented as matrices, and the propagation formulas can be explained as matrix operations. The mathematics for these matrix operations are called belief propagation rules, or bayesian inference rules and are explained in the next section 2.2.3.

2.2.3 Bayesian Inference Through Matrices

Bayesian Inference refers to the procedure of estimating the belief of a node given the values of observed nodes. The estimation is carried out by applying the Belief Propagation rules. There are two kinds of inferences:

- If the leaf node values are observed, we infer the values of the internal nodes in the causal tree. This is called diagnosis, or bottom-up reasoning.

- If the internal node values are observed, we infer the values of leaf nodes in the causal tree. This is called prediction, or top-down reasoning.

For our research, we adopted bottom-up reasoning, as we represent the evidence as leaf nodes, and the hypothesis as internal nodes. The graphical model that we use for our research is explained in chapter 3. The mathematical framework for Belief Propagation in this model is explained below:

- A π message can be represented as a row matrix. The number of elements of the matrix is equal to the number of states of the parent node.
- A λ message can be represented by a column matrix. The number of elements of the matrix is equal to the number of states of the child node.
- A node receives π message from its parent and λ messages from its children
- The CPT at a node Y, whose parent is X, is represented as CPT(y/x)
- The mathematical formulas of the propagation rules for Bayesian Inference at a node are given by:

  1. The π message at a node can be calculated as:

     π(node) = π(parent_node) * CPT(node/parent_node)

     π(parent_node) is the π message obtained from the parent node. For a root node, π(parent_node) is the prior probability of the node itself, since it does not have a parent node.

  2. Since a node can have multiple children, the λ messages from the children need to be fused together.

     The λ message from a child is calculated as:

     λ_from_child(child_node) = (CPT(child_node/node) * λ(child_node))

The λ(child_node) is the λ message at the child node. This λ message is calculated from directly observed evidence.

The λ message at the node can be calculated by fusing the λ_from_child messages via matrix multiplication as:

$$\lambda(node) = \prod_{c \epsilon children(node)} \lambda\_from\_child(c)$$

3. Once the π and λ values are calculated, the belief at a node can be calculated as

   Belief(node) = π(node) * λ(node)

The Belief(node) thus calculated gives the belief at a particular node. An important note to be considered is that all the resultant matrices in the above rules need to be normalized. After the calculation of the belief at each node, this belief in turn needs to be propagated to other nodes in the tree be reapplying the belief propagation rules. These rules are executed on the entire causal tree, for every new evidence that is obtained.

The propagation rules can be clearly understood through an example. Consider a simple causal tree shown in figure 2.4.

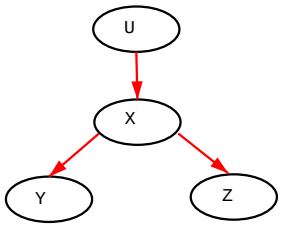

Figure 2.4. Example to Illustrate Bayesian Inference

Suppose the beliefs, and π and λ values at each node of this example tree is as shown figure 2.5. The λ matrices at the leaf nodes (Y, and Z) are calculated from

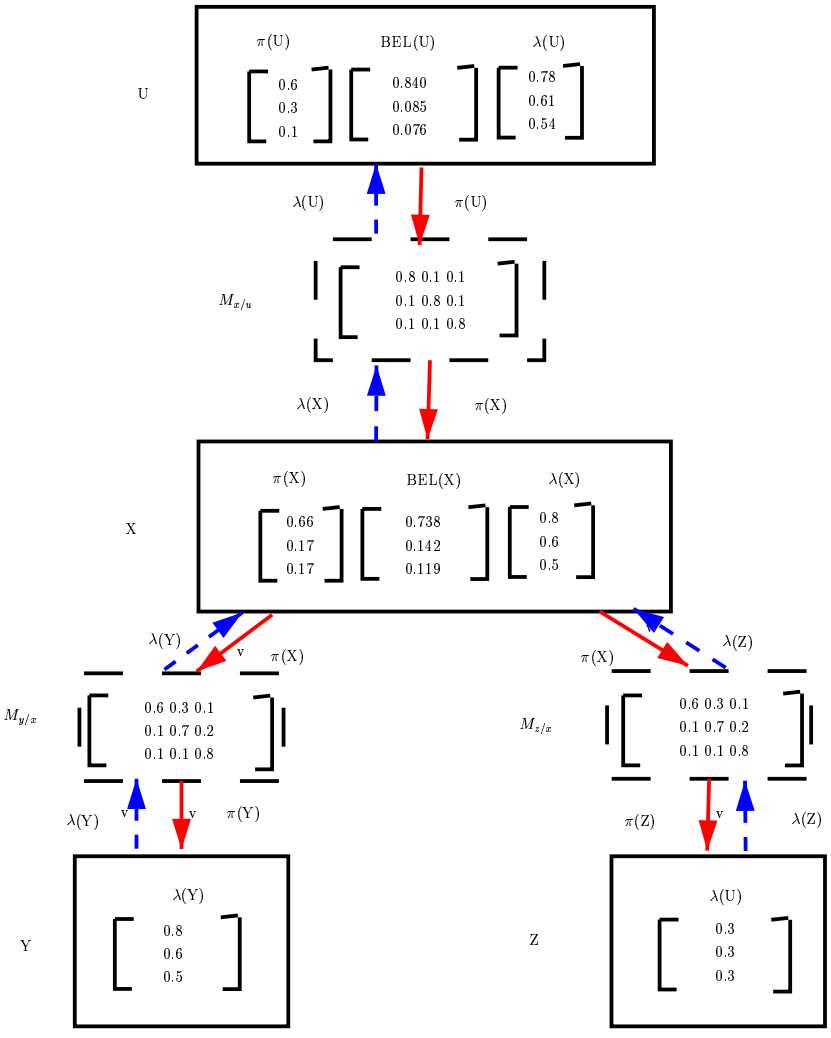

Figure 2.5. Initial Beliefs for The Nodes in Example Tree

directly observed evidence. Using the probabilities at each of the nodes, and CPT at each link of the causal tree, the belief at node X is calculated as:

1. π(X) is calculated by multiplying the matrices π(U) and $M_{x/u}$. This matrix is then normalized

2. The λ message from child Y is calculated by multiplying the matrices λ(Y) and $M_{y/x}$. The λ message from child Z is calculated similarly. The resultant matrices are then multiplied to obtain the final λ message, which is then normalized

3. The π, and λ matrices thus obtained are multiplied to calculate the belief at X, Bel(X)

4. The π, and λ messages at node X are then passed to its child, and parent nodes respectively

The resultant π, λ, and belief matrices at node X are shown in figure 2.6.

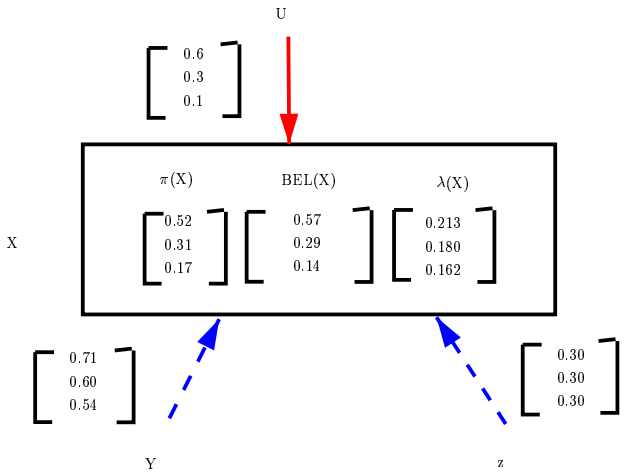

Figure 2.6. Updated Belief at Node X of Example Tree

2.3 INBOUNDS

INBOUNDS (Integrated Network Based Ohio University Network Detective Service) is a real-time, network-based, anomaly-based Intrusion Detection System that is being developed at Ohio University. The system detects anomalies by analyzing the network traffic, either off-line or in real-time. The analysis is based on an anomaly detection principle, where deviations from regular traffic patterns are characterized as intrusions. The principal goals of INBOUNDS are to detect well-known and novel attacks, generate few false positives and false negatives, operate in real-time with minimal overhead; be adaptable, be configurable, and be fool-proof.

The following sections provide an overview of the architecture, and components of the INBOUNDS system.

2.3.1 Architecture

The INBOUNDS system analyzes network data obtained from various data collection modules, and provides this information to a module that analyzes the attack and responds appropriately. Figure 2.7 depicts the INBOUNDS system, and its components.

2.3.2 Data Source

The data source module collects the network data, and provides it to the data processor of the Intrusion Detection module. This data is the raw network data, which can typically be collected using tools such as tcpdump [14], and ethereal [4]. However, INBOUNDS uses a program named *TCPurify* [20] as the data source.

TCPurify is a packet sniffer program with a focus on privacy. The tool collects raw network data, and truncates the packets by removing the data payload beyond the standard IP and TCP/UDP headers. Since INBOUNDS is based on the principle of anomaly detection, the application data is irrelevant for our research. Also, *TCPurify* protects privacy by encrypting the IP addresses, such that it would not be possible to reconstruct the original traffic data.

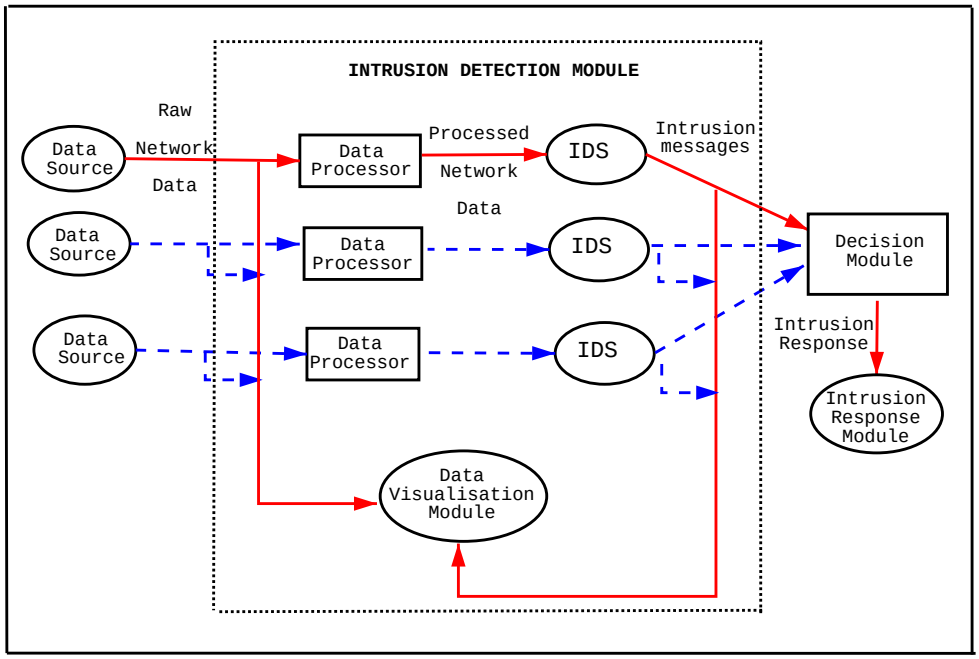

Figure 2.7. INBOUNDS Architecture

2.3.3 Data Processor

The data processor module obtains network data from the data source, and processes it. The module reports certain parameters that characterize the network data. The reported parameters are used for intrusion detection in INBOUNDS.

2.3.3.1 Existing Network Parameters

The code that performs the processing is present in the INBOUNDS module [21], which is included in a TCP/IP network traffic analysis program called TCPtrace [62]. TCPtrace reads network data captured from a data source, groups the packets into conversations, and analyzes the individual conversations. The analysis includes calculation of round trip time, calculation of throughput, and detection of retransmissions. TCPtrace uses certain modules that perform additional analysis, including the INBOUNDS module that calculates parameters for each connection to characterize

the network traffic. The parameters are reported periodically at a time interval of 60 seconds [1]. The paradigm that is used for calculating the parameters is

- A question is a packet from client to server that contains data in the payload
- An answer is a packet from server to client that contains data in the payload

The following parameters characterize each connection:

- *Interactivity:* Number of Questions per second during a particular time interval
- *ASOQ:* Average Size Of Questions in bytes during the time interval
- *ASOA:* Average Size Of Answers in bytes during the time interval
- *QAIT:* Question Answer Idle Time value per second during the time interval
- *AQIT:* Answer Question Idle Time value per second during the time interval

The output from this module are messages that are used by the intrusion detection module of INBOUNDS. The three kinds of messages reported are named O, U, and C messages. These messages are reported for each connection captured by the data source.

- The 'O' message is reported when a new connection is opened. The format for an 'O' message is:

  O Timestamp Protocol $<$ *sourcehost* $>$:$<$ *sourceport* $>$ $<$ *destinationhost* $>$:$<$ *destinationport* $>$ Status

  Timestamp is the time when the connection was opened. The current supported protocols are TCP, and UDP. The connection's 4-tuple is reported as source host, source port, destination host, and destination port. The status field is either 0 indicating that a connection's opening SYN packets were seen during

[1] This interval is tunable

the analysis, or 1 indicating that the connection was already open prior to the start of analysis i.e., the SYNs for the connection were not seen during analysis.

- The 'U' messages are update messages for each active connection, and are reported every 60 seconds. The format of a 'U' message is:

  U Timestamp Protocol $<$ *sourcehost* $>$:$<$ *sourceport* $>$ $<$ *destinationhost* $>$:$<$ *destinationport* $>$ Interactivity ASOQ ASOA QAIT AQIT

  The parameters are as explained previously, and characterize each network connection

- The 'C' message is reported when an active connection is closed. The format of the 'C' message is:

  C Timestamp Protocol $<$ *sourcehost* $>$:$<$ *sourceport* $>$ $<$ *destinationhost* $>$:$<$ *destinationport* $>$ Status

  Status field is 0 if a TCP connection was closed with two FINs seen in each direction, and 1 if a RST packet closes the connection. The status field for UDP connections is always 0. Additionally, connections are timed out, and this interval is 8 hours for TCP connections and 120 seconds for UDP connections.

The parameters described are reported for every connection, and are suitable for detecting anomalies on a particular network connection. However, certain attacks such as Denial of Service (DOS) attacks work by opening numerous connections to a particular network service, each of which are normal. Analysis of the parameters revealed that they would not capture DOS attacks. Thus, we proposed new parameters to capture various kinds of attacks, which are described in section 3.1.2.1.

2.3.4 Intrusion Detection System

Prior to the work cited in this thesis, anomaly detection in INBOUNDS used two approaches, a statistical based technique [18], and a neural network based technique [55]. The following sections briefly explain each of the approaches, and their shortcomings.

2.3.4.1 Statistical-Based Anomaly Detection

This module used the parameters explained in section 2.3.3.1. In addition, a parameter Number Of Connections (NOC) was calculated for each monitored network service (port). The statistical anomaly detection module called Network Anomaly Intrusion Detection (NAID) used two approaches to monitor network activity:

- All Connections to All Hosts on a particular port (ACAH)
- All Connections to a Single Host on a particular port (ACSH)

In the ACAH approach, the NAID module monitors the behavior of a specified port on all hosts in the network. This module enables detection of an abnormality on a particular port on all hosts. In the ACSH approach, the NAID module monitors the behavior of all connections to a specified port on a particular host in the network being monitored. This enables detecting an anomaly on a particular service running on a specific host.

The NAID module utilized two methods to monitor the behavior of a particular network service:

- *Abnormality Factor Method*

  In this method, a database is used to store the average, and standard deviation values of the dimensions for a particular key. The key could be a port, a host IP address, or a combination of both. When a new set of dimension values are obtained, these values are compared with the stored average values in the

database. The standard deviation values are used to calculate the distance of the current dimension value with those stored in the database. A distance value that is greater or equal to certain threshold is assigned an abnormality factor. Thus, the abnormality factor is the difference between the current dimension value and the threshold value. The sum of the abnormality values of all the dimensions gives the overall abnormality value. An intrusion is flagged if the resultant abnormality value is greater than a pre-defined threshold.

- *Moving Average Method*

  The abnormality factor method had the drawback that the average values for the dimensions are fairly constant over time, and do not adapt to network changes. To overcome this drawback, the moving average method was implemented, which defined a moving window for the average parameter values. The time period for which the window is active is specified by a certain value 'T'. A moving average is calculated for the current 'T' by taking the average values of all the parameters from the past time window. An intrusion is flagged if the difference between the current dimension value and the moving average is greater than a certain pre-defined threshold.

2.3.4.2 Neural Network-Based Anomaly Detection

The neural network-based method that was used in INBOUNDS for anomaly detection is Self-Organizing Maps (SOM) [55]. A SOM maps non-linear statistical relationships between data points in a high dimensional space into geometrical relationships between points in a two-dimensional map [49]. Each element in such a map is called a neuron, which represents an n-dimensional input vector. The input vector for INBOUNDS consists of the five dimensions described in section 2.3.3.1, and a sixth dimension called Duration Of Connection (DOC). Application of SOM to intrusion detection comprises a learning phase, and an operation phase. During the learning phase, SOM neurons are trained to model the input data. A different SOM

is defined for each network service whose behavior is to be monitored. During the operation phase of SOM, the current input vector is fed into the trained SOM, and associated with one of the SOM neurons. An anomaly is flagged when the current input vector does not fall within a certain vicinity of the current SOM neurons.

2.3.4.3 Drawbacks of Existing Approaches

The existing approaches used in INBOUNDS for anomaly detection had certain drawbacks. The Statistical based technique described in section 2.3.4.1 had the disadvantage that it was not adaptive to changing environments. An intruder could gradually increase the average, and standard deviation values by carrying an attack over a long period of time. This leads to an attack going unnoticed after a certain period of time. To overcome this drawback, the moving average method was used, which can detect the attacks carried out in steps over a large period of time. However, moving average method might not be able to detect attacks that increase the activity gradually in successive time windows. Also, statistical-based systems have the disadvantage that they generate more false positives, since they are dependent on an appropriate threshold being set.

The SOM-based approach described in section 2.3.4.2 overcomes the drawbacks of the Statistical-based technique, since it can store multiple data reference points for each class of network traffic. Hence, a network connection is considered normal if it falls within any of these data reference points in the SOM. However, neural networks need large amounts of training data to learn normal data values. Also, the SOM module for INBOUNDS analyzes traffic only on a per-connection basis, based on whether the connection is normal or abnormal. Hence, it cannot classify an attack, where the individual connections look normal but the anomaly is due to the presence of numerous such connections, as is the case for a Denial Of Service (DOS) attack. Another drawback of the SOM approach is that it can detect an anomaly only after the connection is closed.

We analyzed the disadvantage of not being able to detect DOS attacks, and pro-

posed new parameters, which are described in section 3.1.2.1. Also, our work is motivated by the need to design a powerful technique that overcomes the limitations of both statistical, and neural network techniques. We found Bayesian Network [65] to be a useful technique, which has the advantages of being adaptable to changing environments, and need less training. A prototype was developed using the Bayes Net Toolbox [58], and a feasibility study was conducted. We found the Bayesian technique to be promising, and hence have adopted the technique for intrusion detection in our research. Section 2.2 explained the Bayesian Network technique and chapter 3 describes the application of Bayesian Network technique to INBOUNDS.

2.3.5 Decision Module

The decision module shown in figure 2.7 is work in progress, which integrates the intrusion alerts from the various IDSs of INBOUNDS, and makes a decision accordingly. The decisions taken by the module are sent as requests to the Intrusion Response module, which implements the decisions. The decisions might be to terminate a connection, block a particular kind of traffic, or rate limit the bandwidth. The various IDSs from which the Decision Module obtains the intrusion alerts are the statistical-based IDS described in section 2.3.4.1, SOM-based IDS described in section 2.3.4.2, and the IDS presented in this thesis.

2.3.6 Intrusion Response Module

The Intrusion Response Module shown in figure 2.7 is also work in progress, which implements the decisions given by the Decision Module. This module provides the capability of taking active responses at an appropriate point in the network so as to prevent further propagation of an attack. The approaches to active intrusion response include inserting a firewall rule to block a particular connection, block a particular port, or rate limit the bandwidth for a particular kind of traffic at a gateway router, which are all done autonomously.

2.3.7 Data Visualization Module

The Data Visualization Module shown in figure 2.7 is a program called *network-graphserver*, which displays a picture of the connections made from or to the hosts in the network that is being monitored.

3. Causal Tree Anomaly-Based Network-Based Detection System (CANDES)

The architecture of INBOUNDS was depicted in figure 2.7 and elaborated in section 2.3. One of the Intrusion Detection Systems in the architecture is the Causal tree Anomaly-based Network-based DEtection System (CANDES), which uses Bayesian Network technique for intrusion detection. The following sections describe the architecture of CANDES, the graphical model used by CANDES, and the intrusion detection approach used by CANDES.

3.1 CANDES Intrusion Detection Module for INBOUNDS

A detailed architecture of the CANDES system is shown in figure 3.1. The various components in the architecture are described in the following subsections.

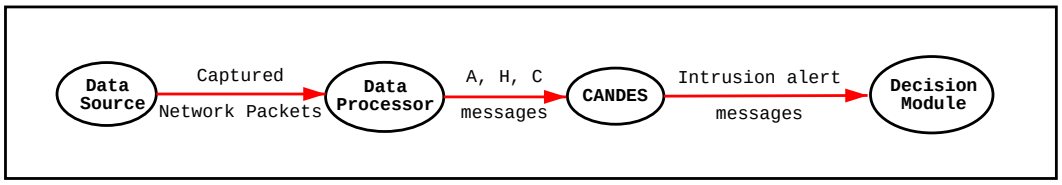

Figure 3.1. CANDES Architecture

3.1.1 Data Source

The data source is typically a network packet capturing tool such as tcpdump or ethereal, or TCPurify as described in section 2.3.2. The output of this component is the raw network data captured on the network being monitored, which is fed to the data processor.

3.1.2 Data Processor

As explained in section 2.3.3.1, the parameters reported by the existing INBOUNDS data processor would not suffice to detect certain kinds of attacks such as Denial-Of-Service (DOS) attacks. Since the parameters are reported only for each connection, an additional mechanism is needed to detect the DOS attacks. We analyzed the existing parameters, and framed new parameters that can detect varied kinds of attacks. The new parameters characterize a particular network service as well as the connections to that service. The code for calculating these parameters is a module for TCPtrace called Bayes module.

3.1.2.1 Proposed Network Parameters

The new parameters are calculated, and reported for each network service by the Bayes module. The parameters are reported every 60 seconds [2], and are elaborated below:

- Number Of Open Connections (NOOC): This parameter indicates the number of connections that have been open to a particular network service in the past 60 seconds.

- Number Of Failed Connections (NOFC): This parameter indicates the number of failed connections to a network service seen in the past 60 seconds. Failed connections include connections that have been reset, and that are half-open. The connections that have been reset are those connections that have seen a RESET packet from the client to server before the establishment of connection. Half-open connections are those connections that have not completed TCP connection establishment i.e., a SYN was sent to the server by the client, but an ACK was not seen coming back to the server. Such connections are timed out after an interval of 60 seconds and are reported as failed connections.

[2] This time interval is tunable

- Number Of Packets (NOP): This parameter indicates the total number of packets received by a monitored host, or a network on a particular network service in the past 60 seconds.

- Number Of Same client IP Connections (NOSC): This parameter indicates the total number of connections made to a particular network service from the same Client IP address. This parameter is also reported every 60 seconds, and is zeroed out after a threshold number of such connections are seen, in order to prevent the parameter from incrementing continually.

The parameters are reported as three kinds of messages by the Bayes module, and are explained below:

- 'A' message (Message for parameters per port on all hosts): This message reports the NOOC, NOFC, NOP, and NOSC parameters for a particular port to all the hosts that are being monitored. This message can help detect certain network attacks, including worms and viruses that affect all the hosts in a network. The format of the message is:

  Timestamp A $< PortNumber >$ NOOC NOFC NOP NOSC

- 'H' message (Message for parameters per port to a host): This message reports the NOOC, NOFC, NOP, and NOSC parameters for a particular port to each host in the network that is being monitored. The format of the message is:

  Timestamp H $< PortNumber >$ HostIP NOOC NOFC NOP NOSC

- 'C' message (Message for parameters for each connection): This message reports the ASOQ, and ASOA parameters for each active connection. The ASOQ, and ASOA parameters are calculated as described in section 2.3.3.1. The format of the message is:

  Timestamp C $< SourceHost >:< SourcePort > < DestinationHost >:< DestinationPort >$ ASOQ ASOA

3.1.3 CANDES

The CANDES module takes as input the messages reported by the data processor, applies the Bayesian Inference rules described in section 2.2.3, and reports any anomalies observed during the inference process as intrusions. The model used for detecting such anomalies is explained in the following sections of this chapter.

3.1.4 Decision Module

The decision module is the module that takes a decision based on the intrusion alerts obtained from the various intrusion detection systems, as described in section 2.3.5.

3.2 Graphical Model for CANDES

The Bayesian Network that is used by CANDES for intrusion detection is shown in figure 3.2.

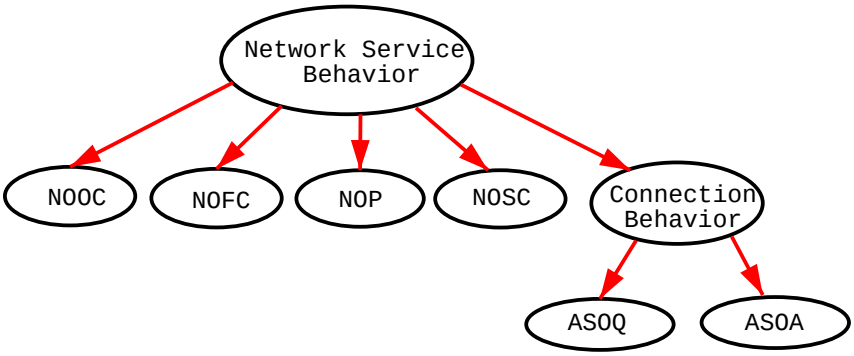

Figure 3.2. Bayesian Model for CANDES

As explained in section 2.2.1.1, each of the nodes in the tree has several discrete values. The leaf nodes are the parameters reported by the data processor: NOOC, NOFC, NOP, NOSC, ASOQ, and ASOA. These nodes are evidence nodes, and their values (probabilities) can be calculated from the reported values. Each of the evidence nodes is assumed to have four discrete values: no anomaly, low anomaly, medium

anomaly, and high anomaly, depending on the anomaly it exhibits from the normal value. The calculated values are then propagated to their parents as λ messages.

The internal nodes, Network Service Behavior, and Connection Behavior are hypothesis nodes and represent the hypotheses "Network Service behavior is anomalous", and "Connection Behavior is anomalous" respectively. The number of discrete values at these nodes is equal to the number of network services which are being monitored. The relationship between the hypothesis and evidence nodes is given by a corresponding conditional probability table. The elements of this matrix are fixed probabilities, defined before the start of analysis. The hypothesis nodes do not have parents and hence the prior probabilities at these nodes are the probabilities at the node itself. The prior probability for the current interval is the belief calculated in the previous analysis interval. The calculated prior probability is stored as a π message at the node.

When an 'A', 'H', or 'C' message is reported by the data processor, the corresponding leaf node values are calculated, and passed as λ messages. The final λ message is calculated, and combined with the π message at the corresponding hypothesis node to calculate the belief. The Bayesian Inference rules described in section 2.2.3 are applied to calculate the beliefs of the Network Service Behavior, and Connection Behavior nodes. The behavior of a particular network service, or connection is reported as either being anomalous, or not anomalous, based on the calculated beliefs.

It can be observed that in addition to the NOOC, NOFC, NOP, and NOSC parameter values, the Network Service Behavior takes as input the Connection Behavior node value. This is advantageous when numerous anomalous connections are observed for an intrusion event, which is not captured by the parameters for the network service. The next section describes the various modules that apply Bayesian Network technique to calculate, and analyze beliefs at the hypothesis nodes.

3.3 CANDES Approach for Intrusion Analysis

The Bayesian analysis method is depicted in figure 3.3

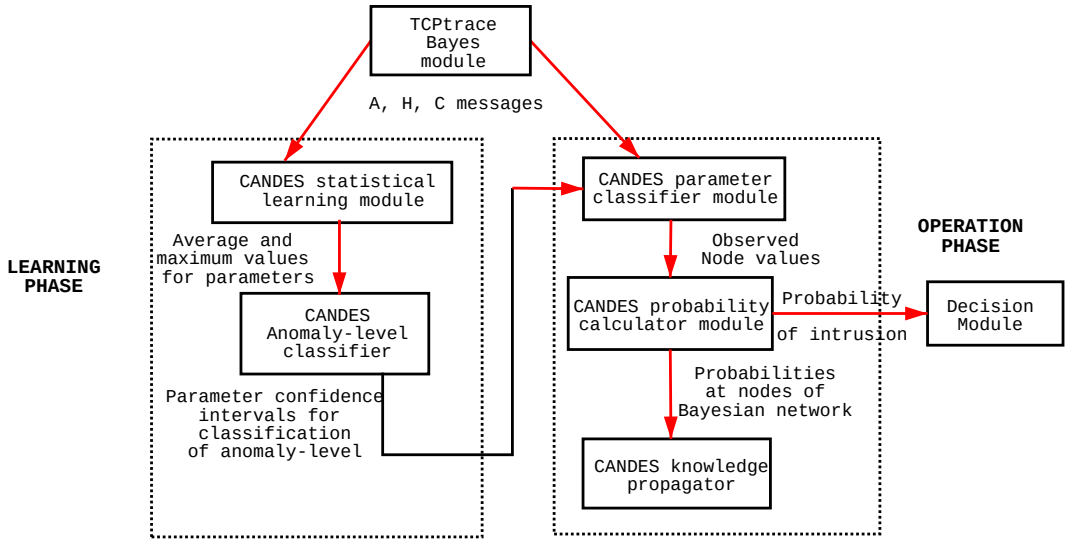

Figure 3.3. Bayesian Analysis Procedure

The various submodules in the CANDES analysis procedure are explained in the following subsections. The statistical learning, and anomaly-level classifier modules are executed during the learning phase of CANDES, while the parameter classifier, probability calculator, and knowledge propagator modules are executed during the operation phase of CANDES.

3.3.1 CANDES Statistical Learning Module

The input to this module are the A, H, and C messages from the TCPtrace Bayes module, which were described in section 3.1.2.1. The parameter values reported in these messages are used to calculate the mean, and standard deviation (SD) values for the parameters. The mean, and SD values in turn are used to calculate the confidence interval for each of the parameters. The confidence interval defines the range for the mean value of a parameter. A statistical method called Hypothesis Testing [78] provides the mathematical formulas to calculate the confidence interval

for the parameters. According to the hypothesis, for a 98% confidence that the current value is equal to the mean, the confidence interval is given by:

$$(\frac{(mean - 2.576 * SD)}{sqrt(Numberof Samples)}, \frac{(mean + 2.576 * SD)}{sqrt(Numberof Samples)})$$

However, we are interested only in the maximum of this interval, as we look for values abnormally larger than the expected value. Thus, the maximum expected value for the mean of a parameter is

Expected Mean = $\frac{(mean+2.576*SD)}{sqrt(Numberof Samples)}$

Also, the maximum value that a parameter can have, is calculated as

Maximum Allowable Parameter Value = (mean + 2.576 * SD)

The value of 2.576 in the above formulas is a constant factor, for a 98% confidence that the value is the mean. The constant factor value varies with the percentage of confidence. The statistical derivations for these values are explained in Trivedi et al. [78].

3.3.2 CANDES Anomaly-Level Classifier

Each of the evidence nodes of the Bayesian Network shown in figure 3.2 is assigned four discrete values: *no anomaly, low anomaly, medium anomaly, and high anomaly.* The output of the Statistical Learning module are the expected mean, and allowable maximum values, which are continuous. The functionality of the anomaly-level classifier is to use these values to define ranges for each of the discrete values. However, this definition of ranges was performed manually for all the parameters. An area for future work involves using a more sophisticated technique to perform the categorization. The ranges thus defined are used during the analysis phase of the CANDES module to set the level of anomaly for the current value of a parameter. The ranges are defined for only directly observable nodes of the Bayesian Network shown in figure 3.2 i.e., NOOC, NOFC, NOP, NOSC, ASOQ, and ASOA.

3.3.3 CANDES Parameter Classifier Module

The functionality of this module is to classify the current parameter value into one of the four discrete levels. These discrete values represent the levels of anomaly of the parameter. The module takes as input the current message from the TCPtrace Bayes module, and the ranges defined for the parameter by the anomaly-level classifier module. Based on the range within which a particular parameter value falls, the parameter value is categorized into one of the four anomaly levels. The λ message for the evidence node that corresponds to the parameter is then set accordingly. If the parameter value is not anomalous, the λ message is the matrix [0.25 0.25 0.25 0.25]. If the parameter value is of low anomaly, the λ message is the matrix [0.25 0.5 0.25 0.25]. If the parameter value is of medium anomaly, the λ message is the matrix [0.25 0.25 0.75 0.25]. If the parameter value is of high anomaly, the λ message is the matrix [0.25 0.25 0.25 1.0]. The values 0.25, 0.5, 0.75, and 1.0 represent the probabilities of anomaly of the parameter, corresponding to the anomaly level. Thus, if the parameter value is of medium anomaly, the probability of anomaly is considered 0.75 and the corresponding λ message entry is set to 0.75. The λ messages are calculated for all the evidence nodes, using the parameter values reported in the current 'A', 'H', or 'C' message.

3.3.4 CANDES Probability Calculator Module

The functionality of this module is to apply the Bayesian propagation rules described in section 2.2.3, in order to calculate the beliefs at the hypothesis nodes (Network Service Behavior, and Connection Behavior). The steps involved in calculating the beliefs are:

1. The λ messages at the leaf nodes are calculated by the parameter classifier module.

2. Each of these λ messages are multiplied by the CPT matrix at the corresponding evidence node, to obtain the final λ messages at these leaf nodes

3. The λ messages from all the leaf nodes are combined through matrix multiplication to obtain the λ message that is transmitted to the parent (hypothesis) node

4. The π message (prior probability) at the hypothesis node is the belief at that node itself, which was calculated in the previous inference interval

5. The λ and π messages are multiplied to calculate the belief at the hypothesis node

The belief at the hypothesis node is a matrix with number of elements equal to the number of network services that are being monitored. Each element of this matrix gives the belief that a particular network service, or connection to the network service is anomalous.

The belief calculation is illustrated with an example, for calculating the belief of the Connection Behavior node. Assume that four network services are being monitored: say, SSH, SMTP, FINGER, and HTTP. Supposing that the SSH network service parameter values are analyzed currently, and that the ASOQ value was found to be anomalous. Figure 3.4 shows the λ messages at the evidence nodes, and π message at the Connection Behavior hypothesis node.

The belief is calculated for SSH, which is the first element of belief matrix. The other elements correspond to other network services, and hence are set to 0. The anomaly for ASOQ value is set to be of low anomaly, and hence the λ at the corresponding node is [0.25 0.5 0.25 0.25]. After applying the belief propagation rules, the belief matrix is calculated to be [0.01875 0 0 0]. However, this does not directly give the probability of anomaly on the SSH connection. The actual probability of anomaly is calculated as:

$$P(Anomaly) = \frac{Belief * 0.25}{Belief\ if\ normal}$$

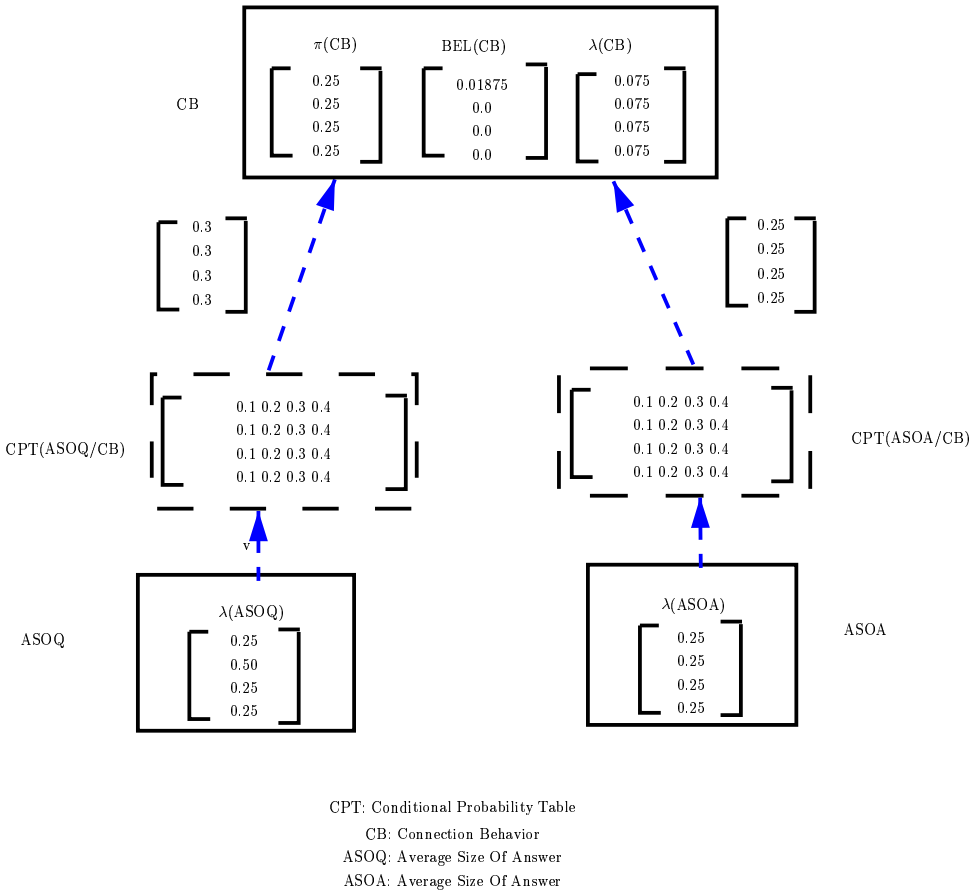

Figure 3.4. Belief Propagation for Calculating Connection Behavior

Here, P(Anomaly) is the probability of anomaly that is to be calculated. Belief is the actual belief calculated after applying the belief propagation rules for the current evidence. The denominator is the actual belief if all the evidence node values are not anomalous. The factor 0.25 represents the probability if the connection were normal. Thus, the above formula is explained as: if 0.25 is the probability for a belief to be normal, what is the probability in case of a belief that is anomalous. For the example, the "belief if normal" value is 0.015625 for connection behavior. Thus, the probability of anomaly for the belief 0.01875 is $\frac{0.01875*0.25}{0.015625}$, which is 0.30. The

probability of anomaly for a network service is calculated similarly after calculating the belief for "Network Service Behavior" hypothesis node.

3.3.5 CANDES Knowledge Propagator

As explained in section 2.2.1.2, Bayes Theorem combines previous knowledge, and current evidence to calculate the belief in a hypothesis. From an anomaly detection perspective, if an attack was seen previously, and the same attack is being carried out currently, the probability of attack is bound to increase. The propagation of previous knowledge of an attack is achieved using the Bayesian Network propagation rules, described in section 2.2.3. The previous knowledge is stored in the π messages at the "Network Service Behavior", and "Connection Behavior" hypothesis nodes. For instance, consider the belief calculation example shown in figure 3.4. The probability of anomaly on SSH connection was calculated as 0.3. This is stored in the π message for the Connection Behavior node as the matrix [0.3 0.25 0.25 0.25]. This helps using the knowledge that an anomaly of probability 0.3 was observed on SSH connection. Suppose that in the next inference, the ASOQ parameter was found to be of low anomaly. The belief calculation for the current inference interval is depicted in figure 3.5.

The probability of anomaly for the current inference is calculated to be 0.36. This implies that as an attack progresses, evidence accumulates about the anomalies, causing a rise in probability of anomaly. This property is useful for detecting an attack that exhibits low anomaly on a particular parameter value, which might go unnoticed with other intrusion detection techniques. This is because the techniques might report an alert only if the probability of anomaly is greater than a threshold value. Since the techniques might not use previous knowledge, the probability of anomaly would remain the same (0.3 for our example). If the threshold is, say 0.4, then the attack goes unnoticed. However, for the Bayesian approach, the probability increases as knowledge about the attack accumulates. The probability eventually increases beyond the threshold, and an alert would be reported.

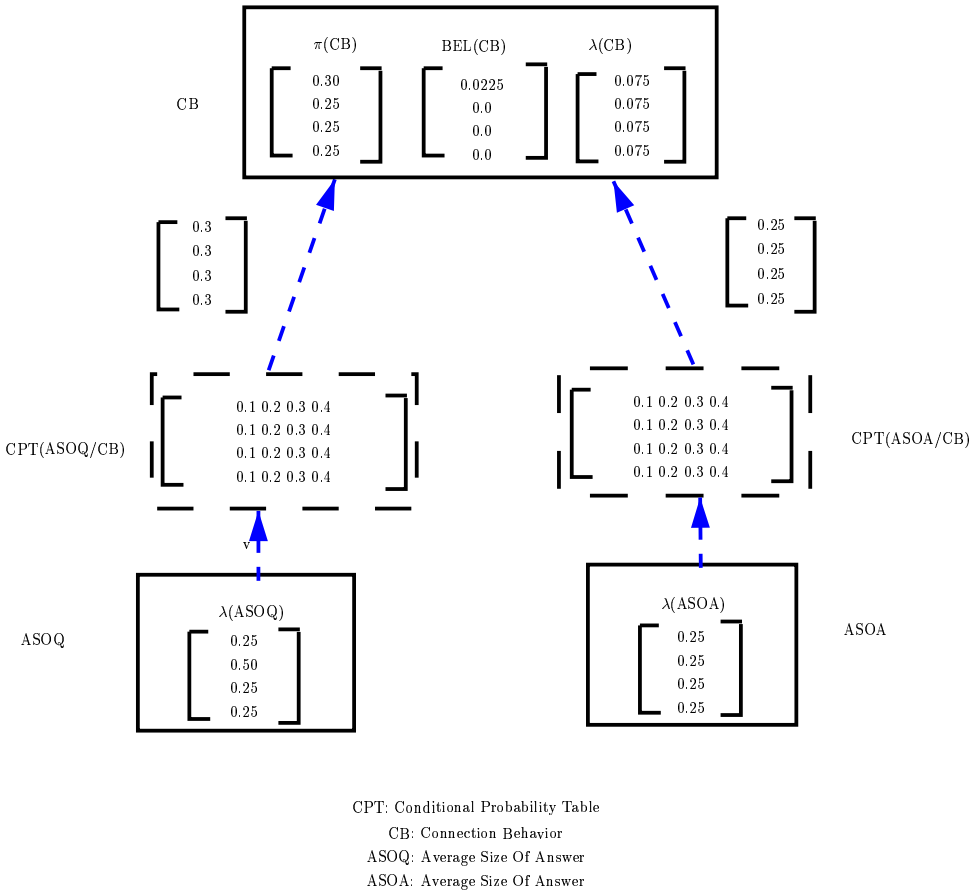

Figure 3.5. Belief Propagation Illustrating Previous Knowledge Accumulation

4. Experimental Results

According to Kendall et al. [47], computer attacks can be classified into four types:

- Denial-Of-Service (DOS) attacks, which interrupt a service to other users by exploiting a constraint on the amount of resources needed for that service. Typical examples include Neptune, Ping Of Death, Process Table attack, and UDP Storm.

- Remote-to-User (R-U) attacks, by which an attacker can gain access to a system without possessing an authorized account. Typical examples include guest, dictionary, imap, and sendmail attacks.

- User-to-Root (U-R) attacks, by which a user with nominal privileges can obtain root privileges on a system. Typical examples include ps, and eject attacks.

- Probing, which are actions taken by an attacker to gather information about the hosts in a network, so as to search for known vulnerabilities on those hosts. Typical examples include ip sweep, nmap, and saint.

CANDES is aimed at detecting most of the DOS attacks that tend to flood the network with large amounts of traffic, thereby denying legitimate access to other users of the system. CANDES also aims at detecting certain probes such as port scans, where the attacker tries to look for a vulnerable network service on the hosts in a network. The advantage of Bayesian Network is that it uses previous knowledge to gain additional information about an ongoing event. This property helped us detect a buffer overflow attack on sendmail, and a stealthy port scan on the finger

service, which might have gone unnoticed with other techniques. The same Bayesian property helps detect an ongoing attack faster, and with higher probability, which is depicted for a Process Table attack [35]. The property that Bayesian Networks can combine several parameters effectively to detect an attack is shown for an Apache buffer overflow attack. The various attacks, and their detection using CANDES module is elaborated in the following sections.

4.1 Sendmail Buffer Overflow Attack

Electronic mail is a popular application that is used to send memos across the Internet. The protocol that handles the delivery of mails is the Simple Mail Transfer Protocol (SMTP), described in RFC 2821 [44]. The exchange of mail using TCP is performed through *message transfer agents* (MTAs). SMTP is the protocol that is used for communication between the MTAs. A TCP connection is first established by the client MTA to a server MTA listening on TCP port 25. This is followed by SMTP commands from the client and responses from the server. A typical SMTP connection between a client and server MTA is depicted in figure 4.1.

Following is an explanation of the SMTP connection message exchanges:

- The SMTP client establishes a TCP connection with the SMTP server, which is listening on port 25.

- The client then waits for the "220 READY FOR MAIL" messages from the server. Upon receipt, the client sends a "HELO" message. The server responds with an OK message and also identifying itself. At this point, the client can send the mail commands to the server, in order to transfer the mail contents.

- The mail transaction begins with a "MAIL FROM" command that specifies the sender address. The server responds with a "250 OK" message in case of success. The client then issues the "RCPT TO" command that specifies the recipient address. The server responds with a "250 OK" message if there is no

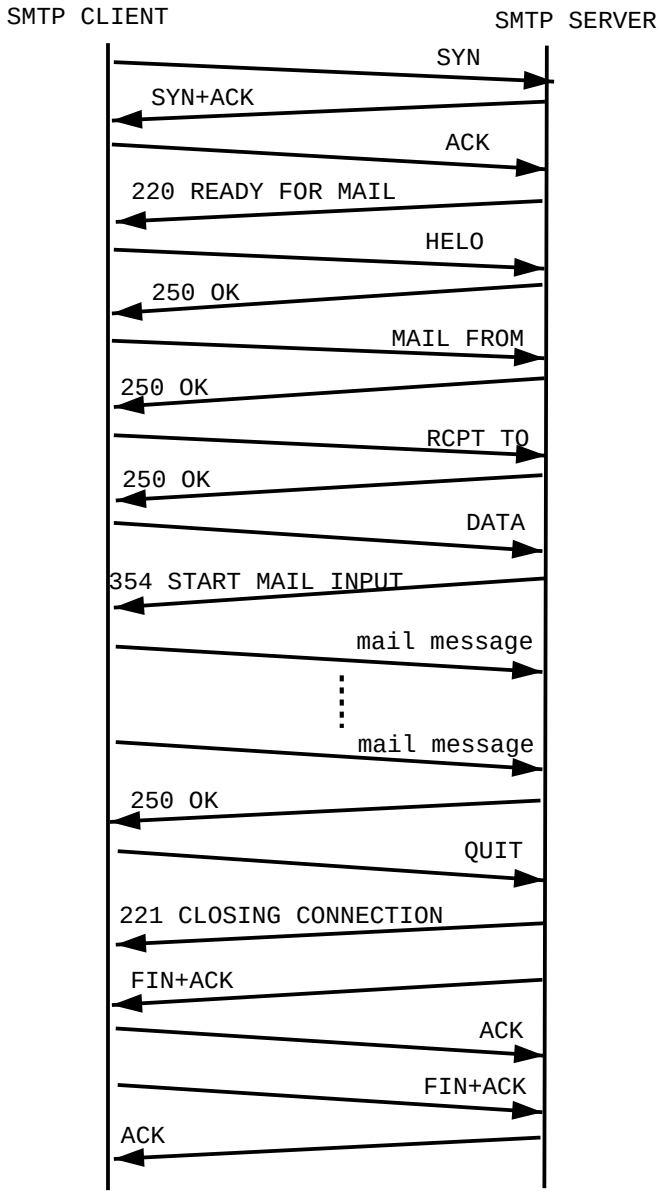

Figure 4.1. SMTP Connection Timeline

error or a "550 No such user here" for a non-existent mailbox. The client issues multiple "RCPT TO" commands for multiple recipients.

- After all the RCPT commands are acknowledged, the client issues "DATA" command to inform the server that the client is ready to send the mail message. Once the server responds with the "354 START MAIL INPUT" message, the client can transfer the mail contents.

- After the transfer, the client sends a "QUIT" command to terminate the connection. The server responds with a "221 CLOSING CONNECTION" message. The established TCP connection is then closed.

The most popular implementation of the SMTP protocol is sendmail [12] and is the most widely deployed mail transfer agent in the Internet. However, several buffer overflow vulnerabilities have been discovered in the sendmail program. One such vulnerability is described in the next section 4.1.1.

4.1.1 Attack Description

Buffer Overflow attacks work by filling a buffer with more data than it can hold. Most of the buffer overflow attacks occur by overflowing a buffer allocated on the stack, thereby modifying the return address. The modified return address points back to a code, which is present in the buffer used for the overflow. This causes the code to be executed as the next instruction. A successful attack typically spawns a shell for the attacker on the victim machine, even though the attacker does not have an authorized account on the machine. For an introduction to Buffer Overflow attacks, refer to Aleph One [61].

A buffer overflow bug was discovered in sendmail version 8.12.10 by Michal Zalewski. The specification of this bug is given in the CERT advisory [26]. The vulnerability could allow a remote attacker to execute arbitrary instructions with the privileges of sendmail daemon, which is typically root. The vulnerability exists in the address parsing code, specifically in the prescan() function, that would allow an

attacker to write past the end of a buffer stored either on the stack, or the heap. Thus a specially crafted e-mail message that exploits the vulnerability can be used to attack an unpatched sendmail daemon.

The attack code was a program obtained from an exploit archive called packet storm [9]. The code works by overflowing a buffer stored on the stack or heap, and causing the next instruction to point to the shell code, which spawns a root shell. The shell code was placed in the "RCPT TO" command of the client, such that sendmail parses the recipient address. Due to the bug in the address parsing code, a root shell can be obtained for a successful exploit. However, the attack had to be performed multiple times for a successful exploit. The attack is successful when the next instruction points exactly to the location of the shell code placed on the stack or heap. Thus, multiple attempts are made from the attacker machine to the victim machine, which is running the buggy sendmail software. While the attack is taking place, the data source (tcpdump), data processor (TCPtrace) and the CANDES intrusion detection module are executed to capture the attack.

4.1.2 CANDES Sendmail Buffer Overflow Detection

The CANDES module was trained on normal values for the connection, and port parameters as discussed in section 3.1.2.1 on port 25 (SMTP). At the end of training, expected mean and allowable maximum values for the various parameters are obtained as shown in table 4.1. The table provides the normal values for the ASOQ, and ASOA connection parameters, and also the NOOC, NOFC, NOP, and NOSC parameters for port on a particular host, for SMTP protocol. Using these values, the current parameter value can be classified into one of the anomaly levels: no anomaly, low anomaly, medium anomaly, or high anomaly. From the table, the ASOQ value is seen to be significantly large compared to the ASOA value. This is because the mail contents are sent as questions to the SMTP server, while the answers are success messages of mail receipt, which tend to be smaller.

The attack was simulated, and CANDES module was fed the attack network data

Table 4.1 SMTP Protocol Training Parameter Values

Parameter	Expected Mean	Allowable Maximum
ASOQ	1502.55	16797.54
ASOA	4.62	47.68
NOOC (per host)	1.59	3.4
NOFC (per host)	0.36	1.2
NOP (per host)	1.15	6.33
NOSC (per host)	0.76	4.45

in real-time as described in previous section 4.1.1. The data source collected the raw network data, and the data processor processed this data to report the 'A', 'H' and 'C' messages, as described in section 3.1.2.1. The CANDES module reported anomalies on connections to port 25, and also anomaly for parameters on port 25. Following are the messages displayed by CANDES intrusion detector:

- Intrusion on Connection: $< ServerIP > 25 < ClientIP > 3083$ Probability: 0.415735
- Intrusion on Connection: $< ServerIP > 25 < ClientIP > 3084$ Probability: 0.441463
- Intrusion on Connection: $< ServerIP > 25 < ClientIP > 3085$ Probability: 0.462045
- Intrusion on Connection: $< ServerIP > 25 < ClientIP > 3086$ Probability: 0.478511
- Intrusion on Connection: $< ServerIP > 25 < ClientIP > 3087$ Probability: 0.491684

- Intrusion on Port: 25 Host: $< ServerIP >$ Probability: 0.452100
- Intrusion on Connection: $< ServerIP >$ 25 $< ClientIP >$ 3089 Probability: 0.502222
- Intrusion on Port: 25 Host: $< ServerIP >$ Probability: 0.509750

The above messages were displayed during the real-time analysis of the 'A', 'H', and 'C' messages from the data processor. This shows an advantage of Bayesian Network technique over previous techniques used for INBOUNDS, which were described in section 2.3.4. The previous techniques could detect intrusions only after the connection closed. In contrast, our CANDES module could detect the anomaly before connection closure, with a delay of 60 seconds, which is the time interval for displaying the 'C' messages.

Further analysis revealed that an anomaly was reported for the ASOA parameter for the connections to the SMTP port. As described in section 4.1.1, multiple attack attempts need to be made for a successful exploit. During these attempts, the ASOA parameter was found to be anomalous, since the SMTP server returned error messages to the client such as "User Not Found" for an unsuccessful exploit. However, the ASOA value was found to be of low anomaly value and hence the initial probability is not high. But as the same kind of attack was seen repeatedly, the Bayesian network accumulated the knowledge, and reported higher probability as the attack progressed. Additionally, anomaly of NOSC parameter on port 25 increased since the attempts were made from the same host. The Bayesian model described in section 3.2 integrates the anomaly values from both the Connection Behavior node, and its parameters. Thus, the anomaly for NOSC parameter was combined with the anomaly for Connection Behavior, resulting in an alert for port 25. The system administrator can analyze the alerts generated due to connection anomaly, and also network service anomaly to conclude about the attack and take appropriate action.

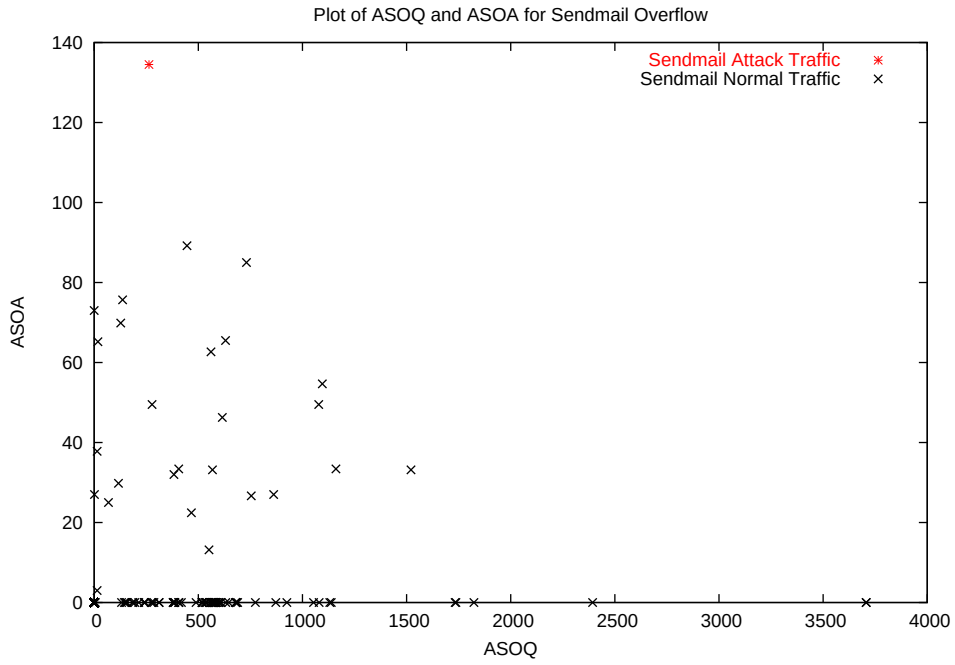

Figure 4.2. Sendmail Buffer Overflow Attack Connection Parameter Values

The graph illustrating the anomalous value for ASOA is shown in figure 4.2, and the anomalous value for NOSC on port 25 is shown in figure 4.3.

4.2 Stealthy Port Scan

There are numerous network scanning tools such as SAINT [10], and Nmap [8] that help system administrators recognize network-related security problems, and report the problems without actually exploiting them. These tools use raw IP packets to determine the hosts that are available on the network, network services that are running on the hosts, the operating system on the hosts, any firewalls that are being used, etc. One of the functionalities offered is a *port scan* [47], where the hosts in a network are scanned to see if a known vulnerable network service is running on any of the hosts. An attacker could use port scan functionality to collect, and analyze the network topology and the vulnerable ports that are open on the hosts in the network.

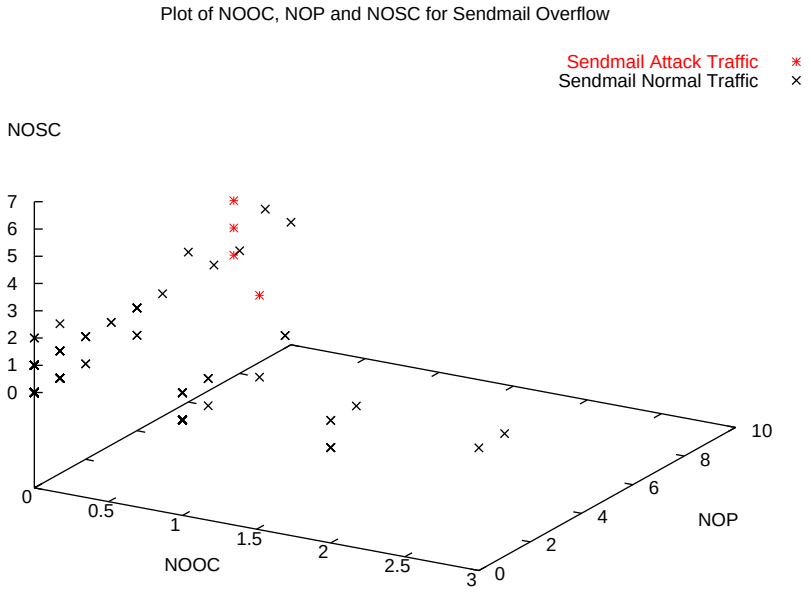

Figure 4.3. Sendmail Buffer Overflow Attack Host Parameter Values

Such scans need to be detected as they might indicate an impending attack. However, the difficulty in detecting the scans is that an attacker can try to masquerade a scan through various techniques. The techniques might be to perform a slow scan, or to perform the scan from various machines simultaneously. Such stealthy scans are hard to detect. We performed a stealthy scan to test the effectiveness of our CANDES module in detecting such a scan, and the results are presented in section 4.2.2 for a port scan on *Finger* network service (port 79) [81].

4.2.1 Attack Description

Most of the network programs offer application services through network daemons. The daemons open a particular port, and listen for incoming connections. Certain daemons create a new process for each incoming connection. The new process handles the connections, and the daemon continues to accept new connections. Finger is one

such network service that is handled by a daemon called *inetd* [27]. The daemon creates sockets on behalf of a number of services including finger, and listens on all of them simultaneously. When an incoming connection is received on any of these sockets, inetd accepts the connection and spawns the server specified for this port, passing the socket across for it to manage. The inetd daemon then returns to listening. The inetd daemon runs with root privileges.

The Finger protocol is based on the Transmission Control Protocol, using TCP port 79. This protocol displays useful information about users, such as their full names, phone numbers, office numbers, etc., which can be used by password crackers to hack into their systems. There are certain other vulnerabilities in the finger protocol such as *Finger Bomb* described in [5]. Finger Bomb is a denial-of-service attack that is achieved by forming a recursive request to the finger service. There are known buffer overflow vulnerabilities in finger protocol implementation that were exploited, the well-known exploit being the morris worm attack, described in Hafner et al. [46]. Since finger is handled by the inetd daemon, which runs with root privileges, an attacker can obtain root privileges by exploiting a finger vulnerability.

Due to the vulnerabilities in finger implementations, attackers can scan a network to see if finger port is open on any of the hosts. However, due to security issues with finger protocol, most of the network administrators either filter traffic to the finger port, or close the port. In such a case, a reset is sent on the connection made by the attacker while scanning. For our experiments, we made a stealthy port scan against port 79, making 2 connections per minute from three attacker hosts simultaneously. The scan was made onto a particular network, and the packet dumps were collected and analyzed.

4.2.2 CANDES Port Scan Detection

As discussed in section 4.2.1, finger is a vulnerable network service, and hence attackers scan networks looking for open finger port. We conducted an analysis using CANDES module, to test its ability in detecting a stealthy port scan. We trained

Table 4.2 Finger Protocol Training Parameter Values

Parameter	Expected Mean	Allowable Maximum
NOOC	5.65	13.12
NOFC	0.0	0.45
NOP	1.67	5.87
NOSC	0.56	2.13
NOOC (per host)	4.53	11.14
NOFC (per host)	0.0	0.23
NOP (per host)	1.19	4.82
NOSC(per host)	0.35	1.18

the CANDES module on the normal values for the network parameters discussed in section 3.1.2.1 on port 79. At the end of the training, the expected mean, and allowable maximum values for the various parameters are obtained, and are shown in table 4.2. The first four entries are the NOOC, NOFC, NOP, and NOSC parameter values on all hosts in the network. The last four entries are the NOOC, NOFC, NOP, and NOSC parameter values for each host in the network. A Finger Port Scan was made onto a network using Nmap [8], and packet dumps collected.

Following are the alerts reported by the CANDES module:

1. Intrusion on port: 79 Probability: 0.412500

2. Intrusion on port: 79 Probability: 0.456500

3. Intrusion on port: 79 Probability: 0.557700

Analysis of the packet dumps revealed an anomaly on port 79 for NOFC, and NOSC parameters. An important note is that the firewall on the network on which

port scan was attempted blocked incoming Finger traffic, and hence those connections were being reset. This caused the data processor to analyze the connections to the Finger server as failed connections, which resulted in the anomalous NOFC value. Also, the anomaly was reported only for the parameters on all hosts and not for the parameters on each host. Since only one connection is made to each host in the network, the anomaly was not observed on individual hosts in the network. However, the total number of connections made to all hosts in the network is high, and hence the anomaly was observed for parameters on all hosts. An anomaly was also found on the NOSC parameter, which was due to the multiple connections made from the same host to the victim network machines. Thus, the overall anomaly is a combination of the anomalous NOFC and NOSC values, and the probability of anomaly increased as the previous knowledge is used to provide stronger of evidence of the attack that is taking place.

The graph illustrating anomalous values for NOFC, and NOSC parameters on all hosts for port 79 is shown in figure 4.4.

4.3 Apache Buffer Overflow

The World Wide Web (WWW) is the most popular application of the TCP/IP technology since its inception in 1991 [6]. It provides simplified access to the information, and resources on the Internet through web browsers. The protocol that the Web commonly uses for the client-server interactions between the browser, and server is the HTTP protocol. HTTP version 1.1 is documented in RFC 2616 [66], and the previous version 1.0 is documented in RFC 1945 [19]. HTTP operates over TCP connections, usually using TCP port 80. After a successful connection, the client transmits a request message to the server, which sends a reply message back. The request consists of a single line of text that begins with the keyword *GET*, followed by the document that is requested. After the server fulfills the request, the connection is closed. This is the client-server connection paradigm used in HTTP version

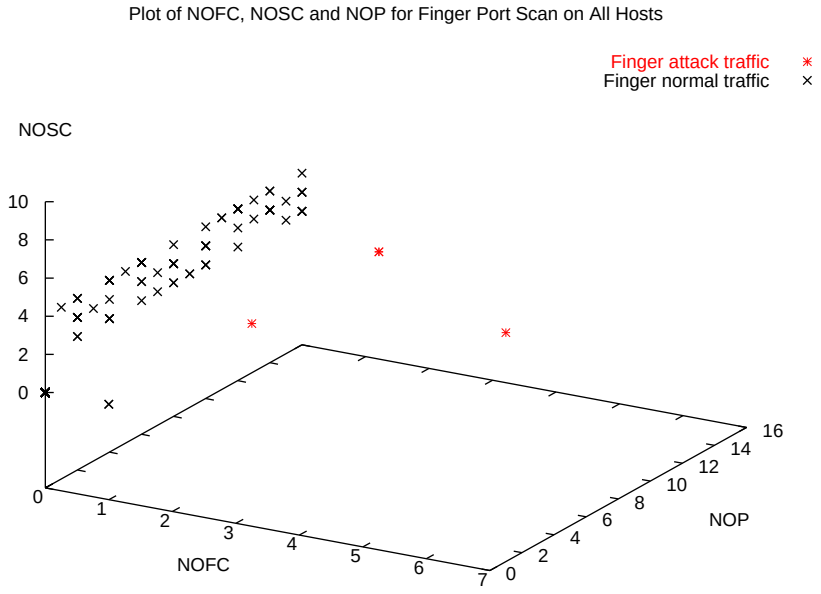

Figure 4.4. Finger Port Scan Parameter Values for All Hosts

1.0. However, version 1.1 introduced the concept of persistent connection, where a client can make multiple requests using the same connection. Though most of the times the server sends the information to the client, in some cases the client can send information by submitting a *form*.

4.3.1 Attack Description

The most popular implementation of a HTTP web server is the *Apache Web Server* [1]. The October 2003 Netcraft Web Server Survey [7] found that more than 64% of the web sites on the Internet are using Apache. The Apache web server provides a secure, and efficient open-source HTTP server for various platforms, implemented according to the current HTTP standards. Apache uses modules to handle various client requests, except for transfer of basic static files, since modules allow Apache to be customized. The *mod_mylo* [36] is one such module, which allows Apache to

log HTTP requests to a MySQL database. However, a security vulnerability was discovered in the logging section of the code [54]. An insufficient bounds checking on a buffer in the code could allow a remote attacker to overwrite saved data on the stack, and execute commands under the privileges of the Apache server daemon. The exploit is publicly available on the Web [53].

The exploit code constructs a HTTP GET request containing the shell code, and the return address to be placed on the stack. The return address causes the next instruction to point to the shell code on the stack when the vulnerability is exploited. As for most of the buffer overflow attacks, the return address needs to be guessed correctly in order to successfully execute the shell code. The exploit code performs a brute force attack, making connections to the victim HTTP server with various guessed return addresses.

4.3.2 CANDES Apache Buffer Overflow Detection

The CANDES module was trained on normal values for the connection, and port parameters as discussed in section 3.1.2.1 on port 80 (HTTP). At the end of training, expected mean, and allowable maximum values for the various parameters are obtained as shown in table 4.3. The table provides the normal values for the ASOQ, and ASOA connection parameters, and also the NOOC, NOFC, NOP, and NOSC parameters for port on a particular host for SMTP protocol. From the table, we can observe that the ASOA value is larger than ASOQ value. This is because the answers to HTTP requests are web pages, while the questions are typically GET requests. However, there might be data flow from client to server, in which case the ASOQ value is also high.

Apache HTTP server was installed along with the buggy mod_mylo module on the victim machine. The attack was then performed using the exploit code from the attacker machine onto the victim HTTP server. The data source (tcpdump), data processor (TCPtrace), and the CANDES intrusion detection module were executed while the attack was taking place. The 'A', 'H', and 'C' messages reported by the data

Table 4.3 HTTP Protocol Training Parameter Values

Parameter	Expected Mean	Allowable Maximum
ASOQ	1546.67	12885.35
ASOA	7021.88	85134.48
NOOC (per host)	1.38	9.48
NOFC (per host)	0.67	3.72
NOP (per host)	1.85	16.68
NOSC (per host)	2.51	15.67

processor were analyzed by the CANDES module. The CANDES module detected the attack in real-time and reported anomalies on port 80. Following are the messages displayed by CANDES intrusion detector:

- Intrusion on Connection: $< ServerIP >$ 80 $< ClientIP >$ 1560 Probability: 0.415735
- Intrusion on Connection: $< ServerIP >$ 80 $< ClientIP >$ 1562 Probability: 0.462045
- Intrusion on Port: 80 Host: $< ServerIP >$ Probability: 0.463100
- Intrusion on Connection: $< ServerIP >$ 80 $< ClientIP >$ 1564 Probability: 0.502222
- Intrusion on Connection: $< ServerIP >$ 80 $< ClientIP >$ 1566 Probability: 0.522793
- Intrusion on Port: 80 Host: $< ServerIP >$ Probability: 0.572000

- Intrusion on Connection: $< ServerIP > 80 < ClientIP > 1570$ Probability: 0.557946
- Intrusion on Connection: $< ServerIP > 80 < ClientIP > 1594$ Probability: 0.649495
- Intrusion on Connection: $< ServerIP > 80 < ClientIP > 1596$ Probability: 0.650846
- Intrusion on Port: 80 Host: $< ServerIP >$ Probability: 0.705980

The displayed messages do not include all the connection anomaly messages, since numerous connections were made for the brute force attack, all of which cannot be shown. Further analysis revealed that anomalies were found for ASOQ parameter on connections to port 80. This was because the shell code was sent in the HTTP GET requests present in the network packets to port 80. In addition, anomalies were found for NOOC, NOP, and NOSC parameters on port 80. This is due to the fact that a brute force attack was conducted by making numerous connections, and trying the exploit with various return addresses in the HTTP requests, which caused anomalies in NOOC, and NOP parameters. Since the attack was made from the same host, the NOSC parameter was also found anomalous. Since, most of the parameter values exhibited anomaly, the combined probability of anomaly was found to be high. This shows an advantage of Bayesian Networks, in that if multiple parameters display anomalies, the probability of attack is higher, aiding faster detection of an attack.

The graph illustrating the anomalous values for ASOA is shown in figure 4.5, and the anomalous values for NOOC, NOSC, and NOP on port 80 is shown in figure 4.6.

4.4 Process Table Attack

The process table attack description was first posted to the bugtraq mailing list in 1999 [35]. It has been described as a DOS attack that exploits network services, which create new processes to handle incoming client connections.

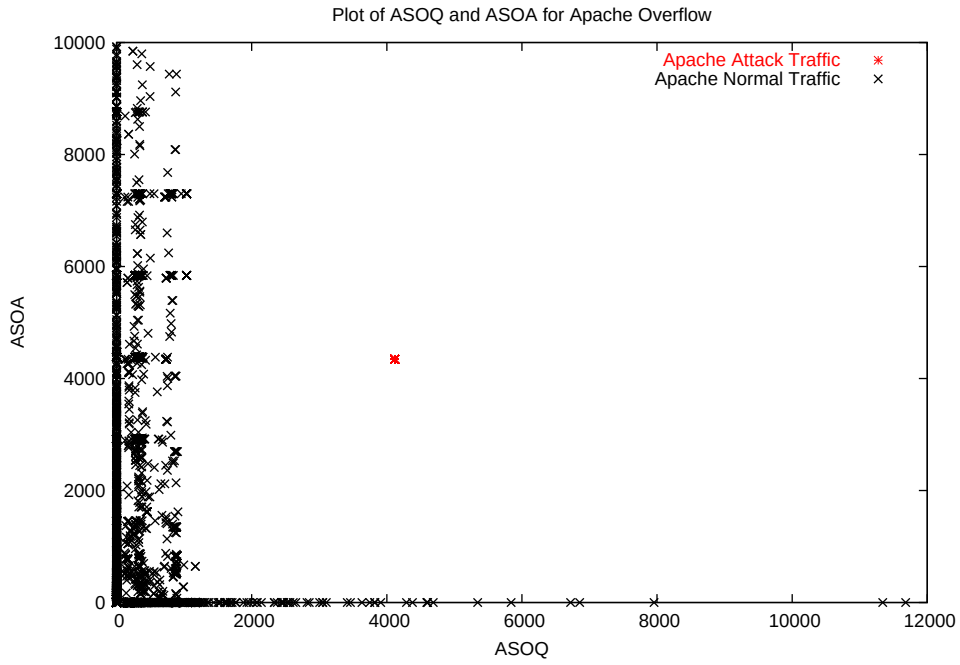

Figure 4.5. Apache Buffer Overflow Connection Parameter Values

4.4.1 Attack Description

Whenever a client connects to a network server, the server dedicates certain resources to the client, which include CPU, bandwidth, memory etc. Certain network server implementations include the creation of a new process, which is stored in the process table. Some of the network servers check the resource availability, or the maximum number of allowable connections, based on which a new connection is accepted. However, certain buggy implementations such as those of finger protocol (port 79), and sendmail (port 25) do not make such checks, and hence are prone to process table attacks.

A process table is a data structure that stores process information such as status of process, its priority etc, [27]. Information regarding processes is saved into this table during context switch. The table might have a finite size that imposes a restriction on the number of processes that a user can create. This can be exploited by an

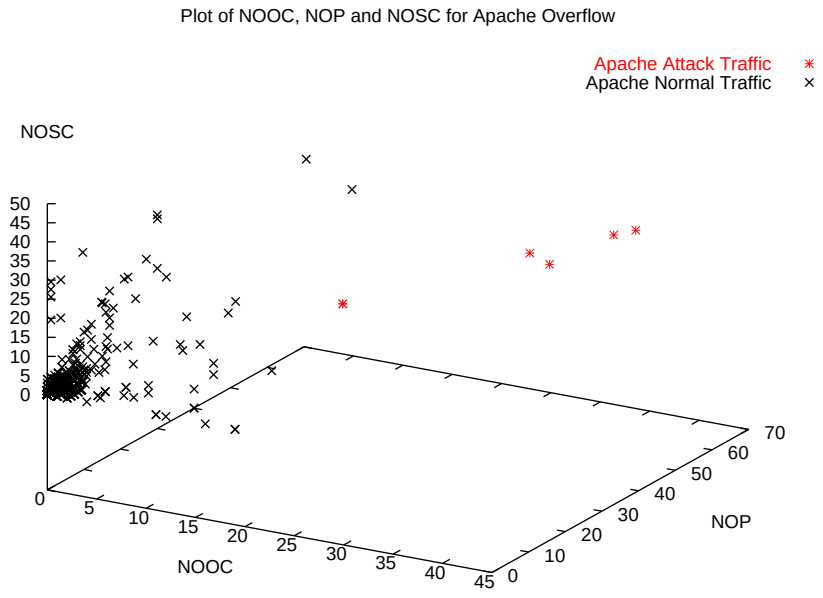

Figure 4.6. Apache Buffer Overflow Host Parameter Values

attacker to launch an attack against a network service that creates a new process for an incoming connection. An attacker can open numerous connections simultaneously so that the process table for that system is filled up, causing the system to accept no more new process requests. Here, we present a process table attack launched against sendmail.

4.4.2 CANDES Process Table Attack Detection

One of the network services that is vulnerable to a process table attack is sendmail. During the training phase of CANDES, the expected mean and allowable maximum parameter values on port 25 are calculated as described in section 3.3.1. The results of the training phase are shown in table 4.4.

The first four entries in the table are the NOOC, NOFC, NOP, and NOSC parameter values on all hosts in the network. The last four entries are the NOOC, NOFC,

Table 4.4 SMTP Training Parameter Values

Parameter	Expected Mean	Allowable Maximum
NOOC	5.53	12.94
NOFC	1.91	2.42
NOP	12.38	34.08
NOSC	1.46	6.82
NOOC (per host)	1.59	3.4
NOFC (per host)	0.36	1.2
NOP (per host)	1.15	6.33
NOSC (per host)	0.76	4.45

NOP, and NOSC parameter values for each host in the network. A simulation attack was carried out for about six minutes on sendmail server by opening numerous connections to fill the process table. The packet dumps were collected during the attack, and later analyzed as described in section 3.3.4. The analysis revealed an anomaly on port 25 for NOOC, and NOSC parameters. The alert messages generated by our module in the six time intervals are shown below:

- Intrusion on port: 25 Probability: 0.443750

  Intrusion on Port: 25 Host: $< HostIP >$ Probability: 0.417500

- Intrusion on port: 25 Probability: 0.506250

  Intrusion on Port: 25 Host: $< HostIP >$ Probability: 0.456500

- Intrusion on port: 25 Probability: 0.685000

  Intrusion on Port: 25 Host: $< HostIP >$ Probability: 0.579700

- Intrusion on port: 25 Probability: 0.793000

  Intrusion on Port: 25 Host: $< HostIP >$ Probability: 0.678260

- Intrusion on port: 25 Probability: 0.853400

  Intrusion on Port: 25 Host: $< HostIP >$ Probability: 0.757108

- Intrusion on port: 25 Probability: 0.917720

  Intrusion on Port: 25 Host: $< HostIP >$ Probability: 0.820186

CANDES analysis uses previous knowledge so that a previously seen attack strengthens the current evidence, leading to an increase in the probability of attack. Thus, the probability of process table attack increases from around 36% to 82% on the particular host. An implication of this Bayesian Network property is that a high probability of intrusion is reported only after certain time intervals of reporting anomalous parameter values. Thus, knowledge is acquired gradually about an attack before sending an alarm of high probability to the intrusion decision module. However, we can observe that the incremental updating of the anomaly probability is also not slow enough, which helps in taking the necessary action against the intrusion at the appropriate time. It had taken at most six minutes for our module to report a high probability of intrusion.

The graphs illustrating the anomalous values for NOOC, and NOSC parameters on all hosts and a particular host are shown in figures 4.7 and 4.8.

4.5 Neptune (SYN Flood Attack)

In the year 2000, attacks were made against web sites like Yahoo, eBay, and E Trade causing disruption of their services [34]. These attacks exploited the TCP connection establishment mechanism to launch DOS attacks that exposed the Internet to a whole new range of vulnerabilities. The significant DOS attack that was launched is the SYN flood attack, where spoofed TCP SYN packets were sent to a listening TCP port. A CERT advisory was published in 1996 [25] about TCP SYN flooding,

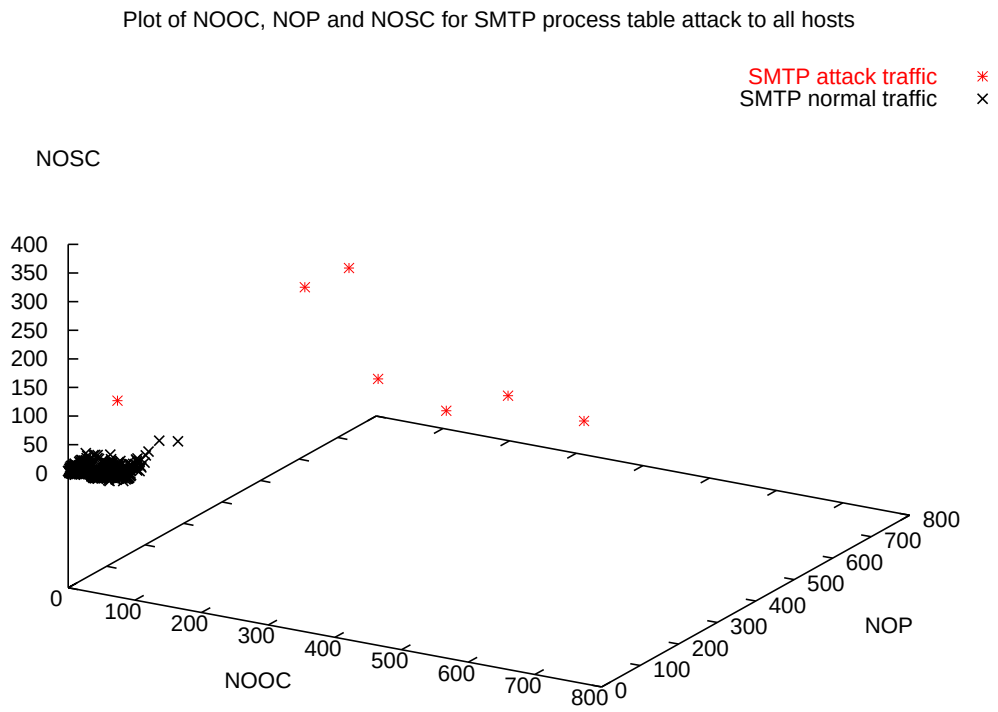

Figure 4.7. Process Table Attack Parameter Values on All Hosts

and IP spoofing attacks. Numerous remedies were proposed to encounter the attack, yet many TCP-based network services remain susceptible to the SYN flood attack.

4.5.1 Attack Description

- The Transport Control Protocol is a connection oriented, reliable, and in-sequence delivery transport protocol. This is the most widely used protocol for the Internet, used by various network services including mail (SMTP), secure shell (SSH) and Hyper Text Transfer Protocol (HTTP). RFC 793 [15] provides the header formats, and connection establishment paradigms for TCP. TCP is a connection oriented protocol and uses handshaking to establish and close a connection. When a system (client) attempts to establish, or close a

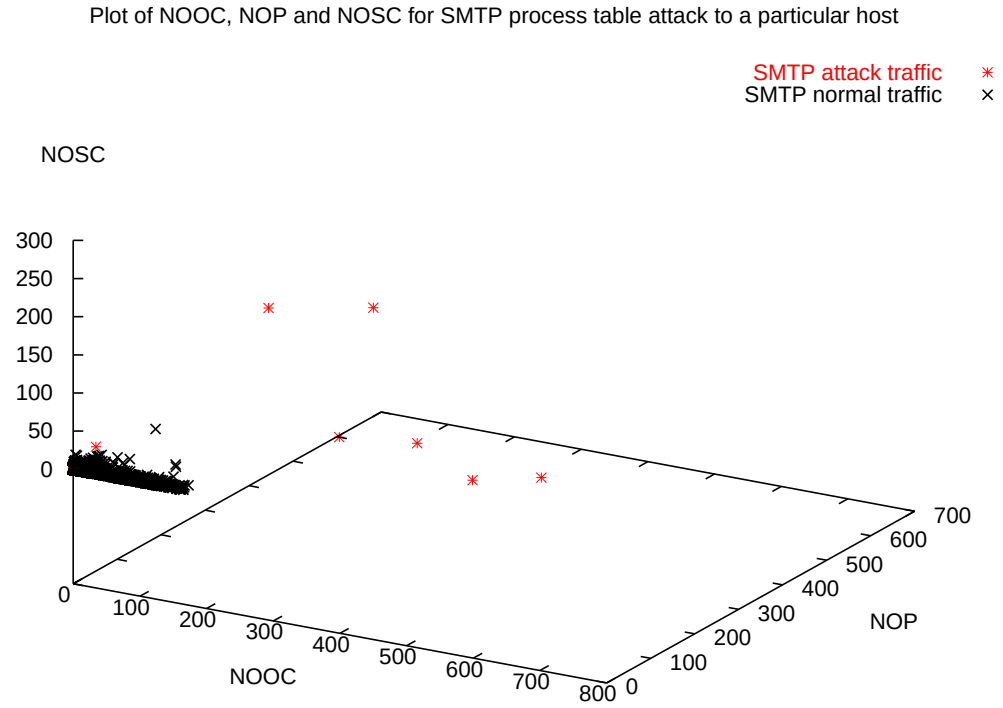

Figure 4.8. Process Table Attack Parameter Values on A Particular Host

connection with the system that provides a particular network service (server), the client and server exchange a set of messages. This handshaking for a normal connection setup and tear down is depicted in figure 4.9. In the figure, the SYN, ACK, and FIN represent the corresponding flags that are turned on in the TCP header while sending the packet. The SYN, and FIN packets delimit the beginning, and end of each connection.

- SYN Flood attack exploits the weakness in the connection setup process when the server sends an acknowledgment (SYN-ACK), and waits for an ACK from the client. Such a connection is termed *half-open*, and TCP servers typically have a data structure that keeps track of such pending connections. This data

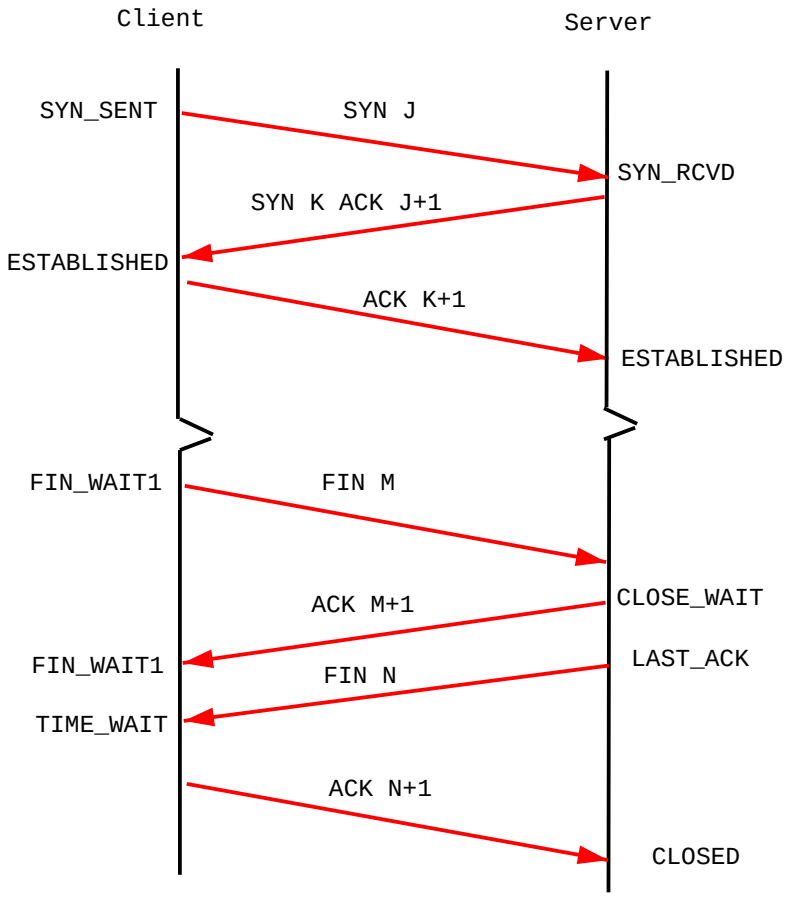

Figure 4.9. TCP Handshake for Connection Setup and Teardown

structure might be of finite size, causing the structure to fill up when numerous half-open connections are present. This causes the network service to be denied to other legitimate users. Most TCP service implementations impose a relatively long timeout period before incomplete connections are cleared out from the data structure, and hence are vulnerable to the attack.

During a SYN Flood attack, the attacker sends a large number of SYN packets without the corresponding ACK packet response to the victim's SYN/ACK packets. The easiest way to accomplish this is to spoof the IP address of the

Table 4.5 SSH Training Parameter Values

Parameter	Expected Mean	Allowable Maximum
NOOC	10.4	20.0
NOFC	0.25	1.54
NOP	39.69	483.24
NOSC	7.48	10.69
NOOC (per host)	1.53	5.9
NOFC (per host)	0.0	0.54
NOP (per host)	6.38	90.26
NOSC (per host)	0.94	3.62

machine from which the SYN packets appear to be sent, so that the SYN/ACKs are sent to an unreachable machine. During the attack, the victim's connection data structure fills up, thereby denying legitimate access to the service.

4.5.2 CANDES SYN Flood Attack Detection

One of the network services that is vulnerable to a syn flood attack is SSH. The CANDES module was trained on the normal values for the network parameters discussed in section 3.1.2.1 on SSH port (port 22). At the end of training, the expected mean, and allowable maximum values for the various parameters are obtained as shown in table 4.5. The first four entries are the NOOC, NOFC, NOP, and NOSC parameter values for all hosts in the network. The last four entries are the NOOC, NOFC, NOP, and NOSC parameter values for each host in the network.

A SYN flood attack was simulated on port 22, and the packet dumps collected by the data source, which were analyzed by the CANDES probability calculator module as described in section 3.3.4. An anomaly was reported on port 22 for NOFC,

and NOSC parameters by CANDES, which also calculated the probability of attack. Anomaly was reported for parameters per port on all hosts with the corresponding probability as well as parameters per port to a single host with the corresponding probability. The attack spanned for two time intervals (60 seconds) of our intrusion analysis module. The messages that our intrusion analysis module displayed in the three intervals are:

- Intrusion on Port: 22 Host: $< ServerIP >$ Probability: 0.450000
- Intrusion on port: 22 Probability: 0.507433
- Intrusion on Port: 22 Host: $< ServerIP >$ Probability: 0.547462
- Intrusion on port: 22 Probability: 0.606833
- Intrusion on Port: 22 Host: $< ServerIP >$ Probability: 0.579446
- Intrusion on port: 22 Probability: 0.699717

As we can see, the probabilities of intrusion are reported, and none of them is 100%. This behavior is due to the fact that we are combining four parameter values to obtain an intrusion decision. This implies that unless all parameters exhibit highly anomalous values, the probability is not a large value. Hence, fluctuation in one of the parameter values does not immediately raise an alarm of high probability. Also, the probability of attack increases as the attack continues in successive time intervals. This is inherent in the Bayesian Network algorithms as it uses previous knowledge to estimate current knowledge. These properties help reduce the number of false alarms reported to the intrusion decision module. However, this does not imply that probabilities increase at a slow pace. This fact was illustrated for the Process Table attack described in section 4.4.

The graphs illustrating the anomalous values for NOFC, and NOSC are shown in figures 4.10 and 4.11, for parameters on all hosts, and on each host respectively.

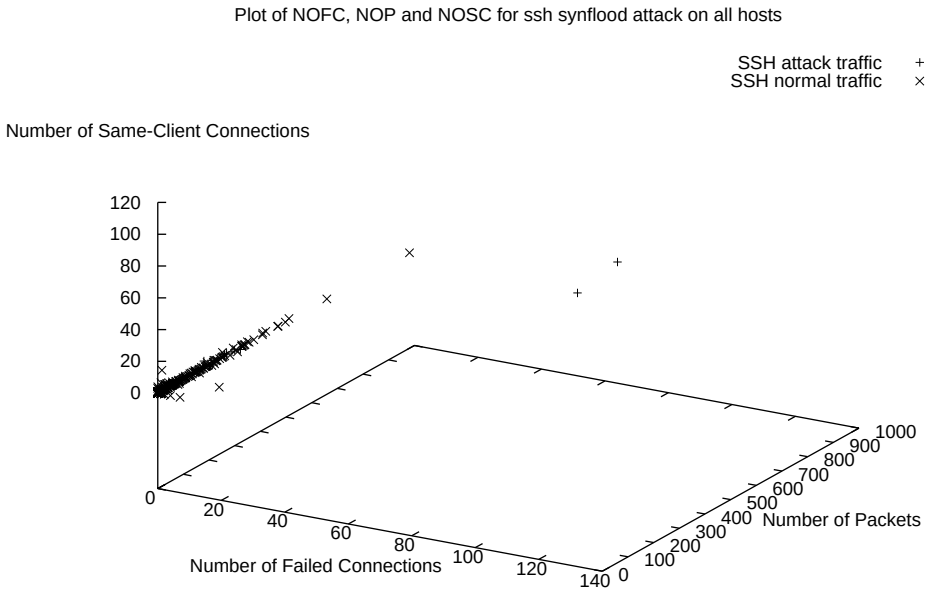

Figure 4.10. SYN Flood Attack Parameter Values on All Hosts

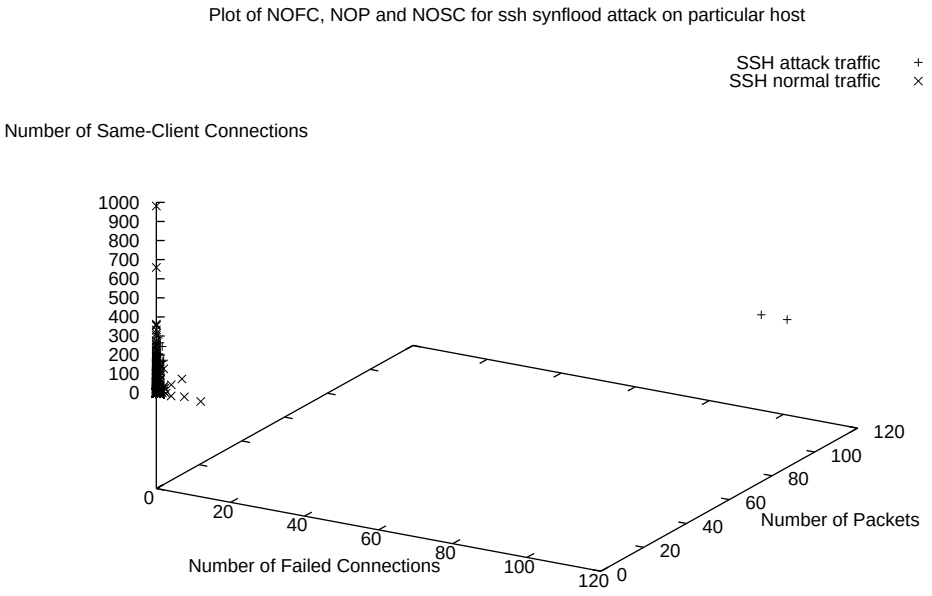

Figure 4.11. SYN Flood Attack Parameter Values on A Particular Host

5. Conclusions and Future Work

In this chapter, we present the conclusions, advantages and disadvantages of our approach. We also analyze the performance of CANDES intrusion detection system that uses Bayesian Networks, and compare the system with the previous intrusion detection system that used SOM approach. We then provide future directions to our research.

5.1 Conclusions

This section presents an overview of CANDES module approach to intrusion detection, and discusses the advantages and disadvantages of our approach.

5.1.1 Overview

The goal of our research is to effectively detect intrusions, for which we have developed a system called CANDES that uses Bayesian Network technique. A Bayesian Network is represented by a tree, the nodes being the parameters of the input domain, and links between the nodes showing the relationship between the parameters. Each node possesses discrete values, which are interpreted as probabilities of the node taking those values. The quantitative relationships between the nodes are represented by Conditional Probability Tables (CPT) that link a child to a parent. Each entry of the CPT gives the probability of the child node possessing a particular value, given its parent node value.

The data input streams containing the raw network data are captured and processed, which are reported as six parameters: Number Of Open Connections (NOOC), Number Of Failed Connections (NOFC), Number Of Packets (NOP), Number Of Same-client IP Connections (NOSC), Average Size Of Questions (ASOQ), and Aver-

age Size Of Answers (ASOA). These parameters are calculated either for connections to a network service, or for network services running on a monitored host or network.

The six reported parameters are represented as leaf nodes in the Bayesian network, and the values of these nodes are directly observable. The internal nodes are Network Service Behavior and Connection Behavior, which are hypothesis nodes, and whose values are used to determine the presence of an anomaly. When the leaf nodes obtain new values, the internal node values are calculated using Bayesian Network algorithms. If the internal node values are anomalous, an intrusion is flagged.

5.1.2 Advantages and Disadvantages

The Bayesian Network module for intrusion detection in INBOUNDS (CANDES) uses anomaly-based detection method, and provides the capability of detecting varied kinds of attacks such as Denial-Of-Service (DOS), Port Scan, and Buffer Overflow attacks. The module detects attacks, where the network parameter values on a network service, or on the connections to a network service for which the Bayesian Network was trained, are significantly different from the normal values. An advantage of using the Bayesian Network technique is the detection of attacks in real-time. Previous approaches for anomaly detection in INBOUNDS could detect intrusions only after the connection closed.

Bayesian Networks provide the ability to combine multiple network parameter values to detect intrusions. Since the relationships are taken into consideration while calculating the overall anomaly measure, the probability of intrusion depends on the number of parameters that exhibit anomaly. Another property of Bayesian Networks is that they combine previous knowledge with current evidence to calculate the belief in a hypothesis. Hence, a recurring attack triggers an attack faster than a novel attack since the previous knowledge provides strong evidence of the ongoing attack. Also, for a novel attack the probability of attack is gradually incremented until sufficient evidence is accumulated, in order to trigger an alert of high probability.

However, there are certain limitations to our approach. The Bayesian Network is

trained for certain network services, and hence can capture anomalies seen on these particular network services only. Intrusions can be detected only when the network parameters that we use, exhibit anomalous values. For example, certain attacks such as port sweeps affect all the network services to a particular host, without any anomaly in the individual network services. Such attacks are not captured by the parameters that we use and go unnoticed, leading to a false negative. Also, we considered simple relationships between parameters without analyzing their exact dependencies. This might trigger a false positive, or a false negative when unrelated parameter values are combined to obtain the anomaly measure.

The parameters that we use are categorized into four different anomaly levels. We defined ranges for these levels manually at the end of the CANDES training phase. An error in this range definitions might lead to either a false positive, or a false negative. Our approach considers the high activity on a network service as one of the criteria to classify anomalies. However, the time of the day, or day of the year are not considered. This might trigger false positives when there is a sudden spike in network activity, which might be normal during busy hours of the day, or busy days of the year.

5.2 Real-time Performance of CANDES

CANDES is a network-based intrusion detection system. A design issue while constructing such a system is its ability to handle large amounts of network traffic, without significant performance loss. Hence, we analyzed the performance of CANDES intrusion detection system in handling varying amounts of network data. The analysis was performed by capturing network data dumps of various sizes, and running the CANDES system on these dumps. The dumps were collected using TCPurify, which is a tool similar to tcpdump but with focus on privacy. A brief overview of TCPurify is given in section 2.3.2. The off-line analysis provided an estimate of the real-time performance of CANDES, and the results are presented in Appendix A.

5.3 Comparison With SOM Approach

We compare the performance of CANDES with the SOM-based approach described in section 2.3.4.2. The criterion for comparison is the capability of detecting attacks. We analyzed if the SOM approach could detect the attacks presented in this thesis. SOM approach uses the existing data processor, which reports network parameters only for each active network connection. Hence, the SMTP process table and SYN flood DOS attacks, and finger port scan attack could not be detected by the SOM approach. This is not a drawback of the SOM approach but is due to the limited processing done by the existing data processor.

However, we analyzed the SOM model for the two buffer overflow attacks on sendmail, and apache. The SOM model was trained on the four protocols used for our research: SSH, SMTP, FINGER, and HTTP, using the same training data sets that were used for CANDES approach. The dumps corresponding to sendmail and apache attacks were then fed to the SOM. SOM model uses a threshold of 2 units of standard deviation (SD) from the trained SOM i.e., if the attack value is greater than 2 units of SD from the trained SOM, an intrusion alert is reported. It was observed that SOM missed both the attacks. For the sendmail attack, the ASOA value was anomalous, but the level of anomaly was low. Hence, the value did not fall within the 2 units threshold, causing a false negative. In contrast, for the CANDES approach the low probability of anomaly was combined with the ASOA anomaly, causing the probability of intrusion to eventually increase to a value when an alert was reported. Similar explanation can be given for the missed HTTP attack, which exhibited a low anomaly on ASOQ parameter. CANDES module could detect the attack due to its ability to use previous knowledge to update its current evidence for an ongoing attack. Additionally, CANDES module has the capability of detecting attacks in real-time in contrast to SOM approach that could detect attacks only after the connection closed.

5.4 Future Work

The network parameters used by CANDES possess four discrete values: no anomaly, low anomaly, medium anomaly, and high anomaly. The ranges for these discrete values were defined manually, and hence might be error-prone. A scope for future work would involve adopting an automated, adaptive, and error-free technique to dynamically define the ranges. The simple relationships between the parameters need to be analyzed, and encoded into the structure of the Bayesian Network. This can reduce false positives by not combining anomaly measures of unrelated parameters.

The CANDES module uses the NOOC, NOFC, NOP, NOSC, ASOQ, and ASOA parameters to capture anomalies in network data. However, these parameters might not capture all kinds of attacks. Hence, the need for new parameters has to be analyzed, and the parameters included for intrusion analysis by CANDES. One way of achieving this is to apply feature selection algorithms such as genetic algorithms to search through the parameter input space, and find the optimal subset of parameters needed to successfully detect all kinds of intrusions. As stated previously, the parameters do not capture certain attacks such as port sweeps. However, CANDES module can be adapted for detecting such sweeps by introducing new parameters, which capture the behavior of multiple network services on a particular host.

The CANDES module uses Bayesian Network technique to store previous knowledge about attacks. This property helps detect recurring attacks more effectively. However, this runtime state information is lost for every new execution of the CANDES module. Hence, a technique to retain the state has to be proposed, and adapted.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Apache HTTP Server Project. URL: http://httpd.apache.org/.

[2] Back Orifice. URL: http://www.bo2k.com.

[3] CyberCop. URL: http://www.ngc.com/product_info/cybercop/ccdata/ccdata1.html.

[4] Ethereal. URL: http://www.ethereal.com.

[5] Finger Bomb. URL: http://securescannx.vigilante.com/tc/13109.

[6] History of the WWW. URL: http://public.web.cern.ch/public/about/achievements/www/history/-history.html.

[7] Netcraft Web Server Survey. URL: http://news.netcraft.com/.

[8] Nmap: Network Mapper. URL: http://www.insecure.org/nmap/.

[9] Packet Storm. URL: http://packetstormsecurity.nl.

[10] SAINT Vulnerability Scanner. URL: http://www.saintcorporation.com/products/saint_engine.html.

[11] SATAN: Security Administration Tool for Analyzing Networks. URL: http://www.porcupine.org/satan/.

[12] Sendmail Consortium. URL: http://www.sendmail.org.

[13] SNORT: The Open Source Intrusion Detection System. URL: http://www.snort.org.

[14] Tcpdump. URL: http://www.tcpdump.org.

[15] Transmission Control Protocol, September 1981. RFC 793.

[16] A. Valdes and K. Skinner. Probabilistic Alert Correlation. In *Recent Advances in Intrusion Detection* (Sept. 2001), Springer-verlag.

[17] Axelsson, S. Intrusion Detection Systems: A Survey and Taxonomy, March 2000.

[18] Balupari, R. Real-Time Network-Based Anomaly Intrusion Detection. Master's thesis, Ohio University, 2002.

[19] Berners-Lee, T., Fielding, R., and Frystyk, H. Hypertext Transfer Protocol – HTTP/1.0, May 1996. RFC 1945.

[20] Blanton, E. Tcpurify: TCP Packet Sniffer, Sept. 2002. URL: http://irg.cs.ohiou.edu/ eblanton/tcpurify.

[21] Bykova, M. INBOUNDS Tcptrace Module: Packet Analyzer, Sept. 2003. URL: http://cidds.cs.ohiou.edu/ inbounds/downloads.shtml.

[22] Carol Taylor and Jim Alves-Foss. NATE - Network Analysis of Anomalous Traffic Events, A Low-Cost Approach. New Security Paradigms Workshop.

[23] CERT. Blaster Worm Attack, CERT Advisory. URL: http://www.cert.org/advisories/CA-2003-20.html.

[24] CERT. Slammer Worm Attack, CERT Advisory. URL: http://www.cert.org/advisories/CA-2003-04.html.

[25] CERT. TCP SYN Flooding and IP Spoofing Attacks. URL: http://www.cert.org/advisories/CA-96.21.html.

[26] CERT. Sendmail prescan() Buffer Overflow Vulnerability, September 2003. URL: http://www.cert.org/advisories/CA-2003-25.html.

[27] Cesati, B. . *Understanding The Linux Kernel.* OŔeilly, 2001.

[28] Chen, K. *An Inductive Engine for the Acquisition of Temporal Knowledge.* PhD thesis, University of Illinois, Urbana Champagne, 1988.

[29] D. Anderson, T. Frivold and A. Valdes. Next-Generation Intrusion Detection Expert System. Tech. rep., SRI International, 1995.

[30] D. Zamboni, J. Balasubramaniyan, J. Garcia Fernandez, D. Isacoff and E. Spafford. An Architecture for Intrusion Detection Using Autonomous Agents. Tech. rep., Purdue University, COAST Laboratory, 1998.

[31] Deborah Russell and G.T. Gangemi Sr. *Computer Security Basics.* O'Reilly & Associates, 1991.

[32] G. Kim and E. Spafford. The Design and Implementation of Tripwire: A File System Integrity Checker. URL: http://www.ja.net/CERT/Kim_and_Spafford/Tripwire.txt.

[33] G. White and V. Pooch. Cooperating Security Managers: Distributed Intrusion Detection Systems. *Computer & Security.*

[34] Garber, L. Denial-of-Service Attack Rip The Internet, April 2000. Computer Magazine.

[35] Garfinkel, S. L. Process Table Attack. URL: http://cert.uni-stuttgart.de/archive/bugtraq/1999/02/msg00421.html.

[36] Gronnesby, O. Apache Module for Writing Logs to MySQL Database. URL: http://www.pvv.org/ oyving/code/mod_mylo/.

[37] H. Debar and A. Wespi. Aggregration and Correlation of Intrusion-Detection Alerts. In *Recent Advances in Intrusion Detection* (Sept. 2001), Springer-verlag.

[38] Herve Debar, Marc Dacier and Andreas Wespi. Towards a Taxonomy of Intrusion Detection Systems, April 1999.

[39] Ho, G. Intrusion Detection - Systems for Today and Tomorrow. URL: http://www.sans.org/rr/papers/index.php?id=341.

[40] Horvitz, E. Lumiere Project: Bayesian Reasoning for Automated Assistance. URL: http://research.microsoft.com/research/dtg/horvitz/lum.htm.

[41] Intel. Machine Learning Libraries. URL: http://www.newscientist.com/news/news.jsp?id=ns99993691.

[42] J. Stutz, W. Taylor and P. Cheeseman. AutoClass C - General Information, NASA, Ames Research Center: 1998. URL: http://ic-www.arc.nasea.gov/ic/projects/bayes-group/autoclass/autoclass-c-program.html#AutoClassC.

[43] Jensen, F. *An Introduction to Bayesian Networks.* Springer Verlag, 1996.

[44] J.Klensin. Simple Mail Transfer Protocol, April 2001. RFC 2821.

[45] K. Jackson, D. DuBois and C. Stallings. An Expert System Application for Network Intrusion Detection. *14th National Computer Security Conference.*

[46] Katie Hafner and John Markoff. *Outlaws and Hackers on The Computer Frontier*. Simon & Schuster, 1991.

[47] Kendall, K. A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems. Master's thesis, Massachusetts Institute of Technology, 1999.

[48] Kevin L.Fox, Ronda R.Henning, Jonathan H.Reed and Richard Simonian. A Neural Network Approach Towards Intrusion Detection. National Computer Security Conference.

[49] Kohonen, T. *Self-Organizing Maps*, 3rd ed. Springer, 2001.

[50] Koral Ilgun, Richard A. Kemmerer and Phillip A. Porras. State Transition Analysis: A Rule-Based Intrusion Detection Approach. *IEEE Transactions on Software Engineering*.

[51] Kumar, S. *Classification and Detection of Computer Intrusions*. PhD thesis, Purdue University, 1995.

[52] Lane, T. *Machine Learning Techniques for The Computer Security Domain of Anomaly Detection*. PhD thesis, Purdue University, 2000.

[53] Livitt, C. Exploit Code for mod_mylo Apache Exploit. URL: ftp://ftp.netsys.com/len/apache+mod_mylo.txt.

[54] Livitt, C. Remotely Exploitable Overflow in mod_mylo for Apache.

[55] Manikantan Ramadas, Shawn Ostermann and Brett Tjaden. Detecting Anomalous Network Traffic with Self-Organizing Maps. In *Recent Advances in Intrusion Detection* (Sept. 2003), Springer-verlag.

[56] Mehran Sahami, Susan Dumais, David Heckerman, Eric Horvitz. A Bayesian Approach to Filtering Junk E-Mail. URL: http://www.microsoft.com/~heckerman/spam98.ps.

[57] M.I.Jordan. *Learning in Graphical Models*. MIT Press, 1999.

[58] Murphy, K. The Bayes Net Toolbox. URL: http://www.cs.berkeley.edu/ murphyk/Bayes/usage.html.

[59] Murphy, K. P. An Introduction to Graphical Models, 2001.

[60] Nong Ye, Mingming Xu and Syed Masum Emran. Probabilistic Networks with Undirected Links for Anomaly Detection. URL: http://www.itoc.usma.edu/marin/Wshop/Papers2000/WA1_2.pdf.

[61] One, A. Smashing The Stack for Fun and Profit . URL: http://www.insecure.org/stf/smashstack.txt.

[62] Ostermann, S. Tcptrace. URL: http://www.tcptrace.org.

[63] P. Porras and P. Neumann. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. 20th National Information Systems Security Conference.

[64] Paxon, V. Bro: A System for Detecting Network Intruders in Real-Time. 7th USENIX Security Symposium.

[65] Pearl, J. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, 1988.

[66] R. Fielding and J. Gettys and J. Mogul and H. Frystyk and L. Masinter and P. Leach and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1, June 1999. RFC 2616.

[67] R. Sekar, A. Gupta, J. Frullo, T. Shanbhad, A. Tiwari, H. Yang and S. Zhou. Specification-Based Anomaly Detection: A New Approach for Detecting Network Intrusions. New Security Paradigms Workshop.

[68] R.Heady and G.Luger and A.Maccabe and M.Servilla. The Architecture of a Network Level Intrusion Detection System. Tech. rep., University of New Mexico, 1990.

[69] Rozenblum, D. Understanding Intrusion Detection Systems. URL: http://www.sans.org/rr/papers/index.php?id=337.

[70] S. Chen, S. Cheung, R. Crawford, M. Digler, J. Frank, J. Hoagland, K. Levitt, C. Lee, R. Yip and D. Zerkle. GrIDS: A Graph-Based Intrusion Detection System for Large Networks. 19th National Information Systems Security Conference.

[71] SANS. SANS Intrusion Detection FAQ. URL: http://www.sans.org/resources/idfaq/.

[72] Srinivas Mukkamala, Guadalupe Janoski, Andrew Sung. Intrusion Detection using Neural Networks and Support Vector Machines, 2002. URL: http://www.cs.nmt.edu/ĨT/papers/hawaii7.pdf.

[73] S.R.Snapp and S.E.Smaha. Signature Analysis Model Definition and Formalism. 4th workshop on Computer Security Incident Handling.

[74] S.Smaha. Haystack: An Intrusion Detection System. IEEE Computer Society Press.

[75] Stuart. Staniford, James A. Hoagland and Joseph M. McAlerney. Practical Automated Detection of Stealthy Portscans. IDS workshop of the 7th Computer and Communications Security Conference. URL: http://www.silicondefense.com/software/spice/.

[76] T.D. Garvey and T.F. Lunt. Model Based Intrusion Detection. 14th National Computer Society Conference.

[77] Todd Heberlein, Gihan Dias, Karl Levitt, Biswanath Mukherjee, Jeff Wood and David Wolber. A Network Security Monitor. IEEE Symposium on Research in Security and Privacy.

[78] Trivedi, K. *Probability and Statistics with Reliability, Queuing and Computer Science Applications.* Prentice Hall, 1982.

[79] W. Lee, S.J. Stolfo and K.W. Mok. Data Mining Approaches for Intrusion Detection. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[80] Werrett, J. Review of Anomaly-Based Network Intrusion Detection, May 2003.

[81] Zimmerman, D. The Finger User Information Protocol, December 1991. RFC 1288.

APPENDIX

A. Real-time Performance of CANDES

We analyzed the real-time performance of CANDES by capturing data dumps of varying sizes, and performing an off-line analysis. The results of the off-line analysis are presented in table A.1. The Bayesian Network for CANDES was trained for four different network services: SSH, SMTP, FINGER, and HTTP. The dump files included various kinds of traffic, and the data processor processed all the traffic. However, since the Bayesian Network was built for only four protocols, the intrusion detection module analyzed processed data belonging to only these four protocols. The communication between the two components was through pipes.

The various fields in table A.1 are:

- Duration: Duration for which dumpfile was captured
- Bytes: Total number of bytes captured in the dumpfile
- Packets: Total number of packets captured in the dumpfile
- Data Rate: Rate at which data was observed (in Mbps) during the capture
- Conns: Total number of connections captured in the dumpfile
- Proc Time: Total amount of time for processing the dumpfile, and analyzing the processed data using CANDES
- Proc Rate: Rate at which data was processed (in Mbps)

It can be observed that the processing rate varies from 69.64 Mbps to 334.85 Mbps, which varies with the number of packets and connections, which is directly related to the size of the dumpfile.

Table A.1 Analysis of CANDES Performance

Duration	Bytes	Packets	Data Rate (Mbps)	Conns.	Proc. Time (sec)	Proc. Rate (Mbps)
26 Feb 2004 (10 mins)	46,220,753	43,226	0.59	1,029	1.05	334.85
26 Feb 2004 (20 mins)	122,373,562	136,656	0.79	2,149	2.78	335.84
26 Feb 2004 (40 mins)	287,014,150	263,621	0.91	4,138	6.27	349.24
26 Feb 2004 (1 hour)	444,241,755	459,232	0.94	5,891	42.18	80.35
26 Feb 2004 (2 hours)	931,131,307	961,374	0.99	10,982	99.82	71.17
26 Feb 2004 (3 hours)	1,311,398,262	1,389,888	0.93	16,349	143.67	69.64

The components of CANDES are the data processor, which is the Bayes module for TCPtrace, and intrusion detection module, which uses Bayesian Network to detect intrusions. The data processor processes the network data, and reports certain network parameters as 'A', 'H', and 'C' messages. These messages are analyzed by the intrusion detection module for detecting intrusions. The runtimes for each of these components for the various dump files is presented in table A.2.

Table A.2 Analysis of CANDES Components

Duration	Pkts	Conns.	TCPtrace (sec) (pkts/sec) (conns/sec)	Candes (sec) (pkts/sec) (conns/sec)
26 Feb 2004 (10 mins)	43,226	1,029	0.82 52,714.63 1,254.87	0.15 288,173.33 6,860
26 Feb 2004 (20 mins)	136,656	2,149	2.16 63,266.67 994.9	0.4 341,640.0 5,372.5
26 Feb 2004 (40 mins)	263,621	4,138	4.76 55,382.56 869.33	0.94 280,447.87 4,402.13
26 Feb 2004 (1 hour)	459,232	5,891	36.7 12,513.13 160.52	1.57 292,504.46 3,752.23
26 Feb 2004 (2 hours)	961,374	10,982	93.78 10,251.38 117.1	3.56 270,048.87 3,084.83
26 Feb 2004 (3 hours)	1,389,888	16,349	130.65 10,638.25 125.14	7.5 185,318.4 2,179.87

The various fields in table A.2 are:

- TCPtrace: The module that processes the network dumpfile
- Candes: The module that used Bayesian Network for analyzing the processed data
- Pkts/sec: Rate at which packets are analyzed by a module
- Conns/sec: Rate at which connections are analyzed by a module

It can be observed that major part of the processing time is consumed by the data processor. The processing rates for both the modules decrement as the number of packets, and connections increase.