UNIVERSITY OF COLORADO AT COLORADO SPRINGS
COLLEGE OF ENGINEERING AND APPLIED SCIENCE
COMPUTER SCIENCE DEPARTMENT

Ph. D. Thesis Proposal

Automated Network Anomaly Detection with Learning and QoS Mitigation

May 2011

Dennis Ippoliti

Respectfully submitted by:
Dennis Ippoliti
Ph.D. Candidate
5509 Langley Way SW
Washington DC 20032
dippoliti@aol.com
home 202-450-5856
mobile 402-238-8208

Date:
11 May 2011

Advisor:
Dr. Xiaobo Zhou

Committee members:
Dr. Xiaobo Zhou (chair)

Dr. Terry Boult

Dr. Edward Chow

Dr. Song Fu

Dr. Chuan Yue

Abstract

Anomaly detection is a challenging problem that has been researched within a variety of application domains such as image processing, fault isolation, fraud detection, and network intrusion detection. In network intrusion detection, anomaly based techniques are particularly attractive because of their ability to identify previously unknown attacks without the need to be programmed with the specific signatures of every possible attack. There is a significant body of work in anomaly based intrusion detection applying statistical analysis, data-mining, and machine learning disciplines. However despite more than two decades of active research, there is a striking lack of anomaly based systems in commercial use today. Many of the currently proposed anomaly based systems do not adequately address a series of challenges making them unsuitable for operational deployment. In existing approaches, every step of the anomaly detection process requires expert manual intervention. This dependence makes developing practical systems extremely challenging.

In this thesis, we propose to integrate the strengths of machine learning and quality-of-service mitigation techniques for network anomaly detection, and build an operationally practical framework for anomaly based network intrusion detection. We will devise methods for self-adaptive, self-tuning, self-optimizing, and automatically responsive network anomaly detection. In specific, we will propose methods for adaptive input normalization that will adjust scaling parameters online based on evolving values in observed traffic patterns. We will propose algorithms for dynamic threshold adaptation that will identify both small aggregate anomalies and single point anomalies while adaptively adjusting to account for concept drift. We will propose a model for dictating optimal performance in an anomaly detection system and propose responsive reinforcement learning algorithms for automated tuning and optimization. We will develop algorithms for dynamic anomaly classification into an evolving attack taxonomy and propose a model for confidence forwarding to feed an automated response engine. We will develop a fair bandwidth sharing and delay differentiation mechanism for scalable automated response to a variety of network attacks that will insulate network resources from malicious traffic while minimizing collateral damage. We will model a prototype network anomaly detection system that integrates the proposed and developed techniques. Extensive experiments will be conducted by using the 1999 Knowledge Discovery and Data-mining Cup datasets, but also we will create a new dataset based on a combination of live network traces and controlled simulated data injects (TD-SIM). As we formulate these ideas, we aim to advance the understanding of the practical capabilities and limitations of anomaly based intrusion detection. By successfully completing this research, we will advance the field by demonstrating an effective and efficient framework that can be applied to real world operational networks.

Table of Contents

1. Introduction:

1.1 Network Anomaly Detection

Revolutions in communications and information technology have given birth to a virtual world. With the growth of cyberspace and its potential, there has been a subsequent change in every facet of daily life. In today's information age, everything we do relies on access to information networks. The internet is used in the home to shop, pay bills, and stay connected via social networks. National infrastructure relies on information networks to deliver oil and gas, power and water. They support hospitals and schools, public transportation and air traffic control. Businesses are relying more and more on e-commerce. As the use of cyberspace grows, the need to protect it also grows.

Attacks against computer networks are increasing at an alarming rate. In the last three years many major institutions have experienced disruptions in service due to cyber-attacks. Facebook, Twitter, Visa, MasterCard, and Google have all been victims. Akamai, the world's largest cloud computing company reported that they experienced more denial of service attempts in the fourth quarter of 2010 than in the previous three quarters combined. One of the most significant trends in cyber-attacks is the prevalence of zero-day attacks[1]. A zero day attack occurs when a vulnerability is exploited before the software developer or administrator is aware the vulnerability exists. Therefore, detection systems are not programed to identify the attack and software patches have not been developed or deployed.

Not only are attacks increasing in number, there effect is becoming potentially devastating. In 2003 worms such as SoBig and Klez caused $80 billion in cleanup costs[2] and in 2008, cyber criminals stole over 1 trillion dollars in intellectual property[3]. In 2009 the *conficker* worm infected the armed forces network in France forcing aircraft at several airbases to be grounded because their flight plans could not be downloaded[4]. In 2007, researchers at the department of energy demonstrated the potential of cyber-attacks by hacking into a power generator and making it destroy itself[5]. According to Senator Carl Levin the Chairman of the Senate Committee on Armed Services "cyber weapons are approaching weapons of mass destructions in their effect."[6]

Existing detection methods, based primarily on identifying and mitigating specific attack signatures are by themselves are inadequate. Signature based methods are ineffective against zero-day attacks. Furthermore, even when similar attack signatures are known, attackers use techniques that disguise the threat. In 2010 Google, Adobe and as many as 33 other companies were attacked using tactics that combined encryption, stealth programming and a vulnerability in Internet Explorer. Despite exploiting vulnerabilities similar to those discovered months earlier, the extremely sophisticated use of encryption prevented discovery by traditional means.

Even if attacks are identified easily, existing response methods are often too slow or impractical causing collateral damage to legitimate customers. In March 2011, CODERO, a cloud and server hosting service for small and medium business, was assaulted by a denial of service attack. Although the attack was specifically targeting a single customer, service to all their customers was disrupted. Despite quickly identifying the /24 network of the target victim, no automated means to analyze and defend affected users

[1] SANS institute (2010) Top Security Risks, http://www.sans.org/top-cyber-security-risks/
[2] Dave Zwieback 2005 *The Gathering Storm: The Future of Cyber Attacks http://www.counterstorm.com*
[3] President Obama, 2009, http://www.whitehouse.gov/the_press_office/Remarks-by-the-President-on-Securing-Our-Nations-Cyber-Infrastructure/
[4] Willsher, Kim (2009-02-07), *French fighter planes grounded by computer worm*, London: The Daily Telegraph
[5] Kroft, Steve. "Cyber War: Sabotaging the System - 60 Minutes - CBS News." CBS News. 8 Nov. 2009.
[6] Singel, Ryan. "Cyberwar Commander Survives Senate Hearing." *Wired News*. 15 Apr. 2010.

was available. Engineers manually redesigned routing tables attempting to isolate malicious traffic from benign, gradually restoring service to all users except those on the same /24 as the targeted system and then eventually to all users.

With the elevated damage caused by intrusions, the increase in zero-day attacks, and the growing sophistication used by attackers, anomaly based methods are well suited to the current threat environment. For Network Intrusion Detection, anomaly based techniques are particularly attractive. In a traditional signature based system, the signature for each new attack must be identified and programmed before a detection system can be sensitive to it. Anomaly based systems do not rely on signatures. They maintain models of normal behavior and heuristically flag abnormal conditions as potential attacks. In 1987 Denning proposed the hypothesis that security violations could be detected by monitoring a system's audit records for abnormal patterns of system usage [Denning 1987]. Anomaly detection schemes are an important step forward in network defense. They excel at identifying zero-day attacks, previously unknown, or well-disguised attack methods. However despite the longevity of research, there are very few anomaly detectors in operational use today.

1.2. Technical Issues and Challenges

A detection system is only effective if the alerts it generates are timely, accurate, and provide useful actionable information to the security team. Unfortunately, anomaly based schemes detect anomalies, not attacks. Having detected an anomaly rather than a well understood attack signature, they are generally unable to provide security teams with information that can be used to guide response actions. Maintaining accurate systems, interpreting alerts, and responding to potential threats requires extensive expert manual intervention. The ability to capitalizing on the advantages of anomaly detection while at the same time providing a useful tool to security teams is hindered by a number of technical challenges.

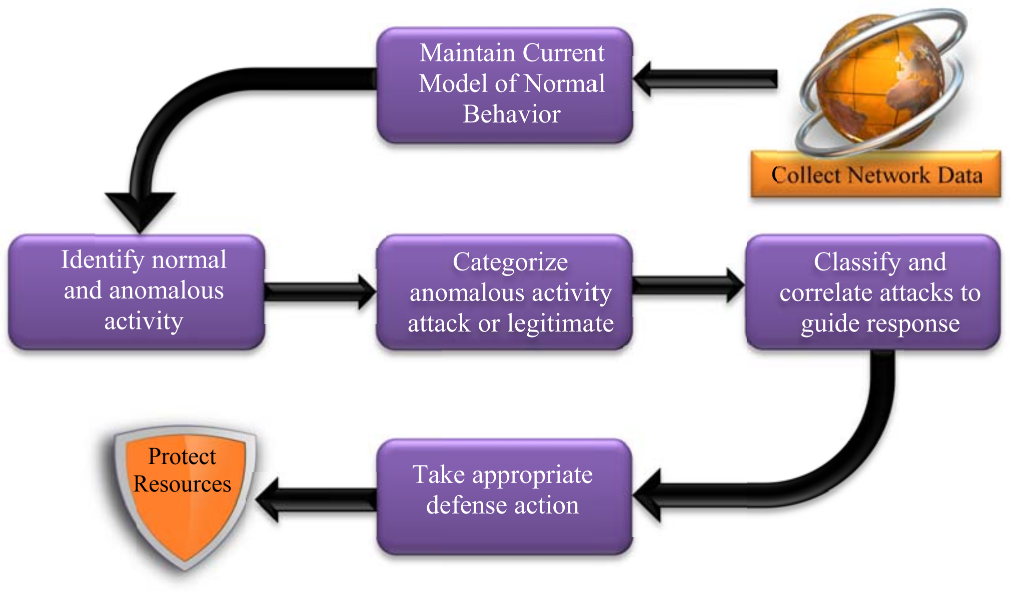

Maintain Current Model of Normal Behavior
Collect Network Data
Identify normal and anomalous activity
Categorize anomalous activity attack or legitimate
Classify and correlate attacks to guide response
Protect Resources
Take appropriate defense action

Figure 1 Basic Elements of the Anomaly Detection Process

Figure 1 identifies the basic steps that must be accomplished in order to use an anomaly detection system for network defense. Existing research in anomaly based network intrusion detection is not evenly distributed among each of the steps. The majority of existing work focuses on identifying attacks by detecting anomalies [Smaha 1988], [Forrest et el 1996], [Vlades et al 2000], [Shon et al 2007] ,[Wang et al 2010]. Very little work is dedicated to other elements of the process. While a body of work has been done on distinguishing between legitimate anomalies and attack anomalies, most of this work involves reducing false positive rates [Li et al 2007], [Wang et al 2010]. [Ciocarlie et al 2009] examines the act of sanitizing training data used to develop base traffic models. [Bolzoni et al 2009] and [Robertson et al 2006] discuss approaches to classifying attacks in anomaly based detection systems. There are also proposals for active systems that automatically respond to attacks. However these systems primarily involve disallowing anomalous requests [Spaffod and Zamboni 2000] [Roesch 1999]. They lack specific response methods tailored to the identified threat level. Combining existing proposals into a comprehensive anomaly detection solution is hindered by the level of operator involvement required. In current systems, each of these elements requires significant human intervention delaying adequate response.

Maintain current model of normal behavior: As the normal behavior of a monitored system evolves, the underlying base model used by an anomaly detection system must be updated. This is traditionally accomplished periodically off line using training data that has been either sanitized by an expert, or certified to be either clean or at least noisy (mostly clean with very few attack instances). If the data used to update the model is not sanitized, sophisticated attacks can fool anomaly detectors into accepting attack traffic as normal. We will work toward an automated system that will incrementally update its base models on-line while at the same time not fall prey to attacker manipulation.

Identify normal and anomalous activity: Numerous methods have been proposed to accomplish the task of anomaly detection. In these approaches exists a variety of control parameters and settings that must be tuned by an expert familiar with both the specific approach and the operating environment of the monitored systems. Failure to properly tune these parameters can impact the effectiveness of the approach. Because these settings are often environment specific, they must be tuned for each potential application. Additionally, as the operating environment evolves, systems can become out of tune. There is an unmet need for systems to be able to dynamically tune based on their current operating and threat environment.

Categorize anomalous activity as attack or legitimate: Anomaly detection systems classify events as either normal or anomalous. This is not the same thing as classifying them as malicious or benign. In regards to network anomaly detection there are four possible conditions for a network event

1. Benign and not anomalous
2. Malicious and anomalous.
3. Malicious but not anomalous
4. Benign but anomalous

Existing systems excel at identifying events that belong to categories 1 and 2. They assume a direct relationship between anomalous and malicious. However, effectively managing events that belong to categories 3 and 4 is an ongoing challenge faced by anomaly detectors. Stealthy attacks that operate over time can gradually force anomaly detectors to accept malicious behavior as normal. On the other hand, network upgrades, legitimate use of new software, and normal evolution of user behavior are all examples of circumstances that while anomalous compared to recorded models, are nonetheless benign. By flagging legitimate traffic that appears to be anomalous as attack traffic, the security operator is forced to address excessive false positive events. Anomaly based schemes need to be able to learn how to relate anomalies with intrusion attempts.

Classify and correlate attack events to guide response: Performing attack classification is an important step to attack mitigation and response. In order for an operator to effectively respond to a potential threat, they must have more than a generic anomaly alert. Modern networks are constantly under attack. While many of these attacks are not relevant (i.e. an old attack targeting an already patched system), others must be addressed immediately by a variety of attack specific means (patching vulnerable systems, terminating sessions, updating routing tables, rate limiting traffic from specific sources, etc…). Performing attack classification in signature based systems is trivial. It is merely a matter of matching current behavior with a set of clearly analyzed and defined patterns. Existing anomaly based approaches generally lack sufficient information and methods to properly classify attack events. By not providing attack classification, security operators must manually research each alert to identify potential threat and take appropriate action. We will propose approaches for intelligent automated attack classification.

Take appropriate action: The last step in the process is for the operator to take action. Most intrusion detection systems are passive in that they raise alerts only. Others are active and take predetermined steps depending on the particular attack. However these approaches generally rely on trained professionals to determine appropriate steps and in many cases are limited in the actions that they are able to take, usually resorting to blocking access to resources or terminating sessions. Due to the inherent uncertainty between an anomaly and malicious intent, blocking, terminating, or disallowing all anomalous events is too excessive. However, by waiting until operators have been able to research each alert, it may already be too late by the time responses are initiated.

1.3. Objectives in Research

The objective of this research is to advance the field of intrusion detection by contributing algorithms and methodologies for automating the anomaly detection process. We will study the applicability of machine learning methodologies to each of the elements of the detection cycle in Figure 1 and provide a novel framework that demonstrates four important features: self-adaptive, self-tuning, self-optimizing, and automated response there by eliminating or significantly reducing the need for operator intervention.

Self-Adaptive: We will propose methods for anomaly detection models to dynamically adapt themselves on-line with little or no expert operator input. Our proposed methods will adapt to changes in normal operations, changes in threat environment, and changes to appropriate response actions.

Self-Tuning: There are numerous approaches to identifying anomalies in network traffic. Within each of these approaches exists a platform dependent set of control parameters. Tuning these parameters require extensive knowledge of both the detection model and environment that the model will be operating in. We will propose algorithms for automatically tuning detection systems in any environment without requiring expert understanding of the underlying detection scheme.

Self-Optimizing: Optimizing a detection system involves tuning the system to achieve an *optimal* objective. In intrusion detection systems, the *optimal* objective is constantly changing in response to changes in operating environment and threat conditions. We will propose a model for defining *optimal* performance in an anomaly detection system and then devise algorithms for self-optimizing capability.

Automated Response: Existing, anomaly detection systems excel at detecting anomalies not attacks. Because of this condition, relative to signature based systems, there is inherent error in attack predictions made by anomaly detection systems. We will propose algorithms for calculating confidence measures in attack predictions. We will research methods for automated response actions that will combine novel Quality of Service algorithms with automated threat assessment approaches, protecting resources while limiting collateral damage to legitimate traffic.

We will demonstrate the effectiveness of our approaches by proposing a prototype integrated detection and response framework based on an Adaptive Growing Hierarchical Self Organized Map. The integrated framework will apply broad identification of network events into categories of normal, attack, and confidence filtering. Attack predictions will automatically be further classified into a dynamic and evolving attack taxonomy. The prototype model will utilize novel Quality of Service algorithms to apply appropriate defenses to the attack taxonomy and scalable response actions to normal and attack predictions with low confidence ratings. The integrated automated response will protect resources from potential attack but will also allow for fair differentiation and bandwidth sharing to confidence filtered events reducing collateral damage to legitimate traffic. We will evaluate the prototypes ability to demonstrate self-adaptive, self-tuning, self-optimizing and automated response capability by conducting extensive experiments on both KDD99 Intrusion Detection Data set and a new data set we will develop combining actual network trace data and simulated network attacks.

To explain the proposal in detail, this document is organized as follows. In Section 2 we discuss related work. The thesis proposed work is presented in Section 3. Section 4 summarizes work completed to date. The research plan and summary are provided in Sections 5 and 6 respectively.

2 Related Work

2.1 Classification of Anomaly Detection Systems

[Denning 1987] proposed a model for intrusion detection based on the assertion that exploitation of a system's vulnerabilities involves abnormal use of the system. Therefore, intrusions could be detected by identifying abnormal patterns of system usage. Since then, a significant amount of research has been accomplished in the area of anomaly based intrusion detection. A number of surveys have been published that characterize the features of intrusion detection systems. Some surveys have categorized a broad range of intrusion detection approaches while others have focused on anomaly based techniques. [Debar et al 1999] classified systems according to Detection method, behavior on detection, audit source location, usage frequency, and whether systems were knowledge based or behavior based. Building on the idea of knowledge versus behavior based, [Axelsson et al 2000] identified two broad categories of intrusion detection systems: anomaly based and signature based. Anomaly based systems were further divided into Self-learning systems and Programmed Systems. [Tsai et al 2009] and [Kumar et al 2010] provide comprehensive reviews of Machine Learning Techniques and Artificial Intelligence techniques to intrusion detection respectively. [Patcha and Park 2007] surveyed Anomaly based Intrusion Detection classifying models based on approach. [Chandola 2009] provides a survey on anomaly detection applied to a variety of disciplines including intrusion detection. Inspired by these surveys, we discuss related work according to the following distinctions: The input data examined for detection, and the method of identifying anomalies. We first characterize anomaly based methods based on the input data. Host based systems monitor data such as system call stacks, registry access and application logs, network based systems typically monitor packet header, packet content, and connection information. We further classify approaches based on the detection method. The surveys identified previously classify approaches in a variety of disciplines including statistical analysis, machine learning, data mining, and association rule discovery. Data mining and machine learning are closely related disciplines and there is significant overlap in classifying methods in these disciplines. Within the context of network anomaly detection we use three broad classifications: statistical analysis, rule based approaches, and machine learning based approaches. The basic assumption of statistical anomaly detection techniques is that normal data instances occur in high probability regions of a stochastic model, while anomalies occur in the low probability regions of the stochastic model [Chandola et al 2009]. Rule based approaches are methods that heuristically identify a set of rules that a system can use to identify anomalous network events. Machine learning methods involve approaches that use input data to learn patterns of normal and abnormal traffic and evolve identification and or classification capability of suspicious traffic according to the algorithm used. The work in this thesis is network anomaly detection with machine learning and quality of service mitigation. As such we do not address signature based approaches. In our discussion of related work, we first briefly discuss host based approaches. We then discuss various network based approaches including statistical, rule based, and machine learning methodologies with the most emphasis on machine learning approaches. Figure 2 illustrates the relationship between surveyed detection methods.

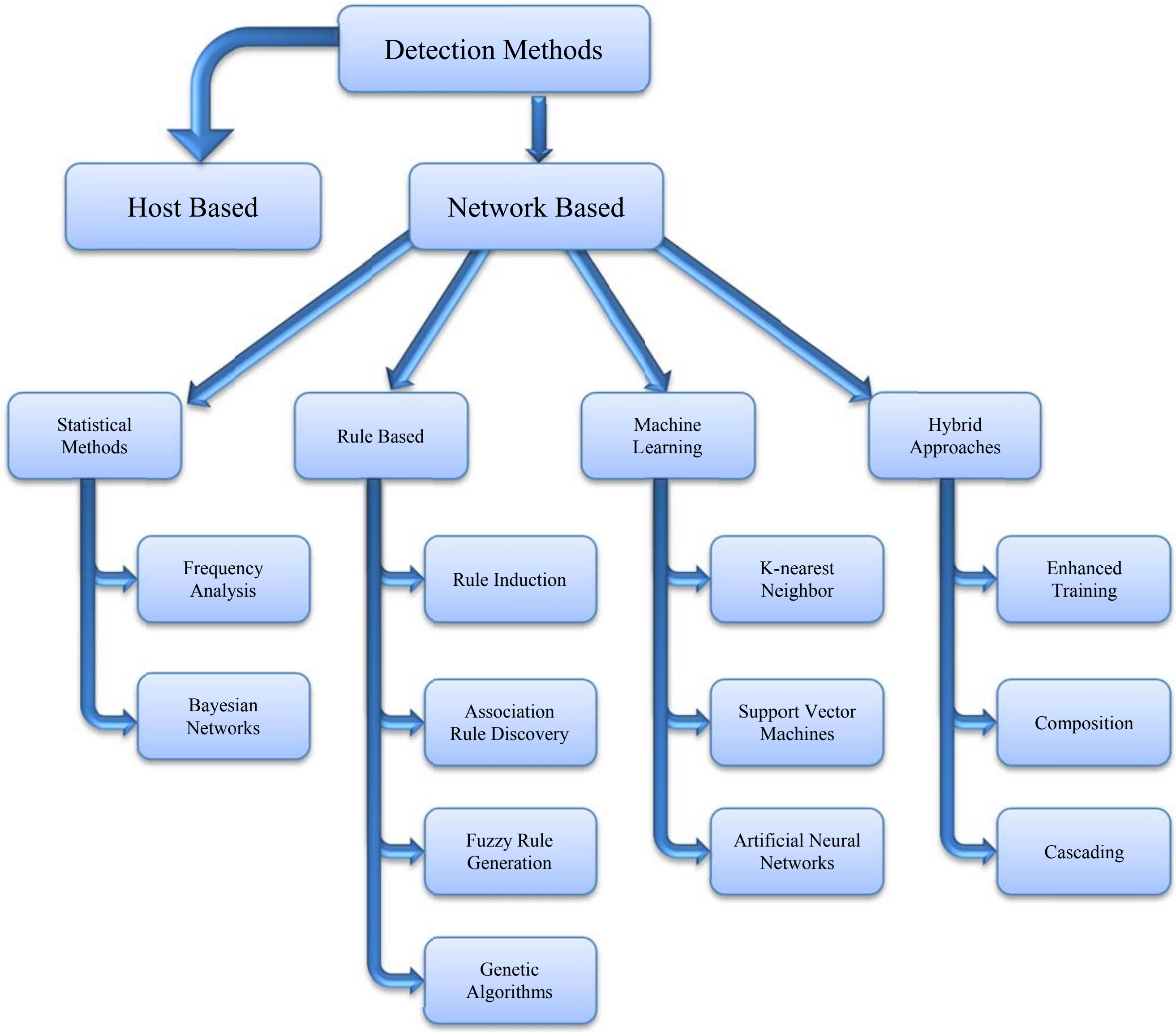


Figure 2: Relationship between Detection Methods

2.2 Host Based Anomaly Detection

Host based systems are primarily concerned with modeling the behavior of individual users or nodes on the network. Examples of input data include system call stack, computer registry entries and executable file configurations. One of the earliest examples of a host based system was Haystack [Smaha 1988]. Haystack monitored user behavior by extracting event information from user session audit trails. Haystack considered a variety of session features and defined ranges that were considered normal for those features. Ranges were selected for both individual users and user groups. Probability distributions were used to determine if ranges were exceeded by too much or too often and alarms were raised to the System Security Officer for investigation when anomalous behavior was suspected. In [Ryan et al 1998] the Neural Network Intrusion Detector was proposed that used back propagation neural networks to monitor user behavior by learning what commands they typically use during a day. Unusual patters would be flagged as in intrusion. These approaches were designed to be utilized off line. On-line systems that attempted to monitor user behavior were proposed in [Debar et al 1992][Fox et al 1990]. These approaches developed models of user command history and used these models to predict future user commands. This information was used to determine if issued commands were "predictable" or anomalous. Other host based models monitor program behavior. [Forrest et el 1996] proposed a method that analyzed individual programs sequences of system calls and developed profiles of normal behavior. They proposed an analogy between the human immune system and intrusion detection arguing that within programs, the short range ordering of system calls will be consistent provide a definition of "self" for that program. Deviation from this sense of "self" or normal behavior would indicate intrusion. Additional host based systems analyzing sequences of system calls were proposed by [Lee et al 1997] who applied the RIPPER algorithm [Cohen 1995] to Unix process traces, [Warrender et al 1999] who modeled call traces using Hidden Markov Models, and [Ghosh et al 1998][Liao and Vermuri 2002] who applied machine learning techniques. Host based techniques designed to monitor Windows Registry access have also been proposed. [Apap et al 2002] used a probabilistic anomaly detector and [Heller et al 2003] examined registry access using Support Vector Machines to identify malicious behavior. Other recent approaches utilize an ensemble of techniques to monitor program behavior and identify malicious software or malware [Chen et al 2005] [Menahem et al 2009].

2.3 Statistical Approaches to Network Anomaly Detection

In statistical methods for anomaly detection, the system observes the activity of subjects and generates profiles to represent their behavior. Typically, two profiles are maintained for each subject: the current profile and the stored profile. As events are processed, the system updates the current profile and compares it to the stored profile using some stochastic model. The basic assumption of statistical anomaly detection techniques is that Normal data instances occur in high probability regions of a stochastic model, while anomalies occur in the low probability regions of the stochastic model [Chandola et al 2009].

Frequency Analysis: A basic approach to stochastic anomaly detection is the use of frequency histograms. Statistical Packet Anomaly Detection Engine (SPADE) [Staniford et al 2002] is a statistical anomaly detection system used for automatic detection of stealthy port scans. SPADE uses a simple frequency based approach, to calculate the ''anomaly score'' of a packet. The fewer times a given packet was seen, the higher was its anomaly score. Once the anomaly score crossed a threshold, the packets were forwarded to a subsystem designed to detect port scans. In [Anderson et al. 1995] the Network Intrusion Detection System (NIDES) included a subsystem that maintained long-term statistical profiles to monitor normal behavior of a computer system [Javitz and Valdes 1991]. Two statistics are maintained, a Q statistic and an S statistic. Q represents the long term behavior of the monitored value, S is a transformation of Q and represents the variance between short term behavior and long term behavior. Large values for S indicate abnormal behavior. The original approach monitored individual system behavior. In [Porras and Neumann 1997] the EMERALD system expanded the capability to include monitoring of the entire enterprise and allowed for detection of Internet worms and DDoS attacks. [Sargor 1998] also extended this subsystem to perform anomaly detection in link-state routing protocols. [Kruegel et al 2002] used the basic histogram approach to perform service specific anomaly detection. The ASCII characters in network payloads are sorted by frequency. Anomaly scores were calculated by monitoring request type, request length, and payload distribution. [Mahoney et al 2001] also used a simple technique to learn normal ranges of packet header fields. It then compared the observed frequency of header values with the expected probability distribution of header values to identify anomalous behavior. In [Yeung and Chow 2002] a Parzen Window probability density frequency classifier was used to not only identify anomalies, but also perform basic classification. [Himura et al 2009] propose a dynamic model based on Sketch [Dewaele et al 2007] where parameters are tuned on-line. [Callegari et al 2008] proposed a model based on high-order Markov chains.

Bayesian Networks: A Bayesian network is an approach that models statistical dependencies and causal relationships between system variables. The model is typically represented by a directed acyclic graph where each node represents a variable and each link indicates a Bayesian relationship of one variable on another. A naïve Bayes network is a restricted network that assumes independence between nodes. This restriction results in a tree structure. In Bayesian approaches conditional probabilities are used to predict the traffic class of network events. [Vlades et al 2000] proposed a Naïve Bayes approach that evaluated network traffic bursts for anomalous behavior. This method was capable of detecting distributed attacks where each individual attack session was not anomalous enough to raise alerts on its own. [Scott 2004] Proposed a general model for network intrusion detection based on Bayesian reasoning. It proposed general methods applicable to many different types of networks arguing that Bayesian methods lead to coherent systems that can handle the complex distributions associated with network traffic. [Wang et al 2006] proposed a Bayesian latent class modeling approach that utilized unsupervised learning and did not require labeled training data. [Cha and Lee 2007] used Bayesian Networks to detect anomalous attacks in FTP traffic.

2.4 Rule Based Approaches to Network Anomaly Detection:

In rule based approaches, an anomaly detector learns rules that describe normal behavior of the system. A network event that is not covered by the defined rules is considered anomalous. The first stage of rule based systems is to apply a rule learning algorithm.

Rule Induction: A common approach to induced rule generation is RIPPER [Cohen 1995] In the RIPPER algorithm, a large rule set is first induced, it is then iteratively revised to increase its accuracy. [Lee and Stolfo 2000] used RIPPER to characterize network trace data and identify anomalous network connections. [Fan et al 2004] used artificially generated anomalous events to better train RIPPER and increases its effectiveness.

Association rule mining: The basic elements of association rule learning include a database D of transactions T, where each transaction T in D consists of a set of items in the database. An association rule is a rule in the form of X -> Y, where X subset of T, Y subset of T and X intersection Y = null. The rule X -> Y is said to have confidence c if c% of the transactions containing X also contain Y. The rule X -> is said to have support s if s% of the transactions in D contain X union Y. [Lee et al 1999] applied an association rule mining algorithm to finding anomalous network connections. In [Barbara et al 2001] the ADAM system used a sliding window approach to finding frequent associations in TCP connection data and identify malicious events.

Fuzzy Rule Generation: In [Hosmer 1993] it was first argued that Fuzzy Logic was particularly suited to network security problems. [Bridges et al 2000] point out that several parameters that are used in the intrusion detection methodology can be viewed as Fuzzy values. They also asserted that fuzzy approaches are well suited for network anomaly detection because they help smooth the boundary between normal and abnormal conditions. Many approaches have been based on Fuzzy Logic principles. [Dickerson et al 2000] proposed the Fire Intrusion Recognition Engine (FIRE). FIRE mined TCP header information and created an aggregate data key from the mined information. The data keys were then combined with frequency information and sorted into fuzzy sets. Fuzzy rules were then manually generated. The rules were then used to identify anomalous connections. While FIRE was effective against probe events, the manual rule generation process was intensive. [Idris and Shanmugam 2005] proposed a method for automating rule generation for the FIRE system. In [Gomez and Dasgupta 2002] An Evolving Fuzzy classifier was proposed that used genetic algorithms to find simple Fuzzy rules to identify abnormal connections. In [Chimphlee et al 2006], A Fuzzy Rough classifier was used to identify anomalous connections. The Rough classifier grouped the object space into three regions: lower, boundary, and negative. The Fuzzy classifier applied fuzziness to connections in the boundary region. The test connections were then grouped into five classes including normal and four abnormal cases.

Genetic Programming: Genetic algorithms are a computational way of mimicking natural selection and evolution. They are commonly used to find approximate solutions to optimization and search problems. They have also been applied to the intrusion detection problem. [Li 2004] used a genetic algorithm to identify a rule set for anomaly detection. This approach used a connection field weighted approach to determine level of suspicion. The weights used for each field are manually tuned by an operator. [Pillai et al 2004] also proposed a genetic approach to generate rule sets. In this approach manual intervention for operator input is required during a periodic retraining process.

2.5 Network Anomaly Detection with Machine Learning

Machine learning is the ability of computer programs to improve performance on a set of tasks over time. Machine learning techniques are a broad range of identification and classification techniques that evolve detection models based on the input of previous data. Methods in this class include K-NN clustering techniques, Support Vector machines, and Artificial Neural networks

K-nearest neighbor: K-NN is a clustering technique in which new events are classified by majority vote of their K nearest neighbors in the trained vector space. Where K is the number of neighbors that get a vote, and the concept of "nearest" is calculated by methods such as Euclidean distance or Mahalanobis distance. An advantage of the KNN approach is its flexibility. A disadvantage is the computational complexity of calculating the K nearest neighbors of each data point. In [Eskin et al 2002], the search space was first clustered into subsets to reduce the time required to identify the neighbor set. In [Bouzida et al 2004] Principle Component Analysis was first applied to the input set to reduce the dimensionality of the vector space. An enhancement to the basic KNN approach was proposed in [Li et al 2007] where transduction techniques [Gammerman and Vovk 2002] were applied to calculate a "strangeness" measure. This approach reduced false positives over other K-NN approaches and reduced the need for labeled training datasets.

Support Vector Machines: Support vector machines provide a supervised learning method for performing classification of high dimension description vectors [Vapnik 1998] Support vector machines use hyper-planes to divide samples into one of two classes. In the case if anomaly detection they identify normal and abnormal network events. In [Eskin et al 2002] an unsupervised learning approach to SVM was used that could be trained with unlabeled data. [Nguyen et al 2002] proposed a one-class SVM for intrusion detection that identified outliers among positive examples and treated them as negative examples. [Zang and Shen 2005] proposed an SVM that conducted training on-line to manage concept drift. Rather than periodically training the model off line, a learning algorithm was applied where training data was fed in sequence rather than in batch optimizing the training process.

Artificial Neural Networks: An artificial neural network (ANN) is a computational model that is inspired by the structure and functional aspects of biological neural networks. Feed forward Back propagation networks have been used primarily in host based anomaly detection [Ghosh et al 1999] [Ghosh et al 2000]. In [Zheng et al 2001] several types of neural networks were applied to network anomaly detection and tested individually. The authors experimented with Perceptron, Backpropagation (BP), Perceptron-backpropagation-hybrid , Fuzzy ARTMAP, and Radial-based Function neural networks. Their approach was particularly effective at detecting UDP flood attacks. An ANN approach that has been commonly used in network anomaly detection is based on the Self-organizing map. The Self organizing map converts non-linear relationships between data points in high-dimensional space into geometrical relationships between points in two dimensional space [Kohonen 1982]. [Rhodes et al 2000] used SOMs for network anomaly detection. In this approach input vectors were mapped to regions of the map and then categorized both on which region they were assigned, and how well they fit that region. Individual specialist maps were trained to recognize anomalies in specific protocols. [Ramadas et al 2003] proposed ANDSOM as an anomaly detection module for the INBOUNDS intrusion detection system [Tjaden et al

2000]. ANDSOM modeled traffic in six dimensions and was particularly effective against buffer overflow attacks. [Kayacik et al 2003] and [Sarasamma et al 2005] proposed using a fixed hierarchy of self organizing maps. In [Kayacik et al 2003] three layers were employed. The first layer was associated basic TCP features. The second layer was used to classify connections clustered by the first layer. The third layer was used to further process connections for those neurons, which win for both attack and normal behaviors in the second layer. [Sarasamma et al 2005] proposed a fixed hierarchal model where each layer in the map was trained to cluster different features of a connection. Test connections are only fed to subsequent layers if they are classified by low accuracy clusters in parent maps.

[Palomo et al 2008] proposed using a Growing Hierarchical Self Organizing Map [Rauber et al 2002] for anomaly detection. In this approach, the map size and dimensionality is not fixed before training. Rather, it is grown to fit the training space based on observed quantization error. Once training is complete, the map dimensions are fixed. Other variations on SOM approaches have been proposed in [Wang et al 2010][Jiang et al 2009].

2.6 Hybrid Techniques

It is becoming more common to combine several approaches into a single model for intrusion detection. The idea behind this approach is to combine the strengths of several different approaches to maximize accuracy and minimize false positives. One simple approach to a hybrid system is a composition where modules are deployed as part of a comprehensive system. Examples of this approach include SPADE [Staniford et al 2002] which was a plugin for SNORT and [Vlades et al 2000] that developed plugin for EMERALD [Porras and Neumann 1997]. Another approach to hybrid systems involves using one approach to enhance the training and configuration of another. In [Sinclair et al 1999][Gomez and Dasgupta 2002][Shon et al 2006] Genetic algorithm approaches were used to enhance training of other machine learning models. Multiple approaches can also be combined to produce a single prediction output. Cascading systems involve using the output of one approach as the input to another. In [Kayacik et al 2007] a cascading series of Self Organizing Maps was used. In [Depren et al 2005] a Self Organizing map was used as an anomaly detection preprocessor for a decision tree module that raised event alerts. [Peddabachigari et al 2007] proposed a cascading Decision Tree and Support Vector Machine approach. An integrated hybrid approach directly combines multiple methodologies. In [Farid et al 2010] an approach combining Naïve Bayes and Decision Tree methods into one classifier was proposed. [Shon and Moon 2007][Wang et al 2010] Combined Support Vector Machines with Self Organizing Maps.

2.7 Dynamic Anomaly Detection

The majority of surveyed approaches are static. That is that they consist of some sort of off line training or modeling phase and then an on-line testing or running phase. During the off-line training phase, a model of normal behavior is learned. During the on-line running phase, incoming events are compared to the learned model and tested for normality. Over time, the normal behavior of the system evolves and the model must be updated. Static systems do this through periodic off-line retraining. There is growing research in performing this re-training incrementally on-line. [Cannady et al 2000] proposed an adaptive neural network that autonomously learned new attacks on-line based on feedback from protected systems. The operating state of protected systems was fed back into the detection model. When incoming traffic was predicted normal, operating state should remain normal. If the operating state degraded after receiving “normal” traffic, the model could use this feedback to learn new attack types. [Rieck et al 2008] proposed a model where previously evaluated packets that were identified as normal were periodically used to retrain the detection system on-line thus evolving the model of normal behavior. To prevent poisoning, the feedback input was selected randomly and was additionally sanitized prior to being utilized for training. [Rehak et al 2009] proposed a method of dynamically updating an intrusion detection system by using “challenges”. A challenge is a previously evaluated normal or attack pattern

Author	Year	Host	Network	Statistical	Rule Based	Machine Learning	Hybrid	Dynamic
Farid et al 2010	2010		x				x	
Himura et al 2010	2010		x	x				x
Wang et al 2010	2010		x			x	x	
Ciocarlie et al 2009	2009		x	x				x
Himura et al 2009	2009		x	x				
Jiang et al 2009	2009		x			x		
Menahem et al 2009	2009	x						
Rehak et al 2009	2009		x					x
Callegari et al 2008	2008		x	x				
Rieck et al 2008	2008		x				x	x
Cha and Lee 2007	2007		x	x				
Dewaele et al 2007	2007		x	x				
Li et al 2007	2007		x			x		
Peddabachigari et al 2007	2007		x				x	
Shon and Moon 2007	2007		x				x	
Chimphlee et al 2006	2006		x		x			
Shon et al 2006	2006		x				x	
Chen et al 2005	2005	x						
Depren et al 2005	2005		x				x	
Idris and Shanmugam 2005	2005		x		x			
Sarasamma et al 2005	2005		x			x		
Zang and Shen 2005	2005		x			x		
Bouzida et al 2004	2004		x			x		
Fan et al 2004	2004		x		x			
Li 2004	2004		x		x			
Pillai et al 2004	2004		x		x			
Scott 2004	2004		x	x				
Apap et al 2002	2003	x						
Heller et al 2003	2003	x						
Kayacik et al 2003	2003		x			x		
Ramadas et al 2003	2003		x			x		
Esket al 2002	2002		x			x		
Gammerman and Vovk 2002	2002		x			x		
Gomez and Dasgupta 2002	2002		x		x		x	
Kruegel et al 2002	2002		x	x				
Liao and Vermuri 2002	2002	x						
Nguyen et al 2002	2002		x			x		
Staniford et al 2002	2002		x	x			x	
Yeung and Chow 2002	2002		x	x				
Zheng et al 2001	2001		x			x		
Mahoney et al 2001	2001		x	x				
Bridges et al 2000	2000		x		x			
Cannady et al 2000	2000		x			x		x
Dickerson et al 2000	2000		x		x			
Ghosh et al 2000	2000		x			x		
Lee and Stolfo 2000	2000		x		x			
Rhodes et al 2000	2000		x			x		
Tjaden et al 2000	2000		x			x		
Vlades et al 2000	2000		x	x				
Ghosh et al 1999	1999		x			x		
Lee et al 1999	1999		x		x			
Sinclair et al 1999	1999		x				x	
Warrender et al 1999	1999	x						
Ghosh et al 1998	1998	x						
Sargor 1998	1998		x	x				
Lee et al 1997	1997	x						
Porras and Neumann 1997	1997		x	x				
Forrest et el 1996	1996	x						
Anderson et al 1995	1995		x	x				
Hosmer 1993	1993		x		x			
Debar et al 1992	1992	x						
Javitz and Valdes 1991	1991		x	x				
Fox et al 1990	1990	x						
Ryan et al 1998	1988	x						
Smaha 1988	1988	x						

Table 1: Classification of Techniques

that is fed back into the system on-line. Based on the systems response to this challenge, the detection module can be evaluated and updated as necessary. [Ciocarlie et al 2009] proposed a method of updating models of normal behavior by using time-delimited slices of data to represent normal models and re-calibrate training data for a method independent detection system. They argue that specific attacks will be concentrated around certain time periods affecting only a small fraction of the micro models reducing the probability of data poisoning by stealthy attacks. In [Himura et al 2010], rather than adapting the base model of normal behavior, the system dynamically updates the parameters used for detection. They argue that at different times of day and under different traffic conditions, a different set of parameters will provide optimal detection capability. Table 1 summarizes the classification of surveyed approaches.

2.8 Open Issues in Network Anomaly Detection

Despite extensive research and a large solution set in the realm of network anomaly detection, the majority of operational systems in use today are misuse detectors, most commonly signature systems that scan traffic for byte sequences [Sommer and Paxson 2010]. This is due to a number of open issues that remain in the field of anomaly detection.

False Alarm Rates: Proposed anomaly detection solutions predominantly suffer from high false alarm rates. Many approaches report false positive rates between 10% and 1% [Eskin et al 2002], [Kayacik et al 2007], [Yu et al 2008]. [Tavallaee et al 2010] surveyed 276 anomaly detection approaches proposed in the last 8 years and reported that most achieve 98% accuracy with false positive rates at 1%. While this seems promising, [Axelsson 1999] points out that in operational scenarios 1% false positive rate may be unacceptably high. When the overall attack rate is very low, even with very high accuracy false positive alerts will outnumber true positive alerts in the alert log.

Relationship between anomaly and intrusion: Network anomaly detection is based on the premise that anomalous behavior is malicious behavior [Denning 1987]. This core assumption leads to two possible categories of behavior in a system.

- Not intrusive and not anomalous
- Intrusive and anomalous.

However, [Kumar and Stafford 1994] propose that anomalous behavior does not always mean intrusive activity. They argue that there are four possibilities:

- Intrusive but not anomalous
- Not Intrusive but anomalous
- Not intrusive and not anomalous
- Intrusive and anomalous.

The assumption that attacks are anomalous is challenged in [Gates and Taylor 2006], [Handley et al 2001], and [Tan et al 2002]. It is argued that stealthy attacks present as normal traffic despite being malicious. [Lakhina et al 2005], [Mahoney and Chan 2003b], and [Barford and Plonka 2001] identify several anomalous circumstances that although unusual, are nevertheless legal operations of the system.

Classification of attacks: Performing attack classification is an important step to mitigation and response [Ning et al 2002], [Valeur et al 2003]. Determining the attack class in a signature based system is trivial. The attack class is manually assigned during the signature development phase [Bolzoni et al 2009], [Roesch 1999]. However, in an anomaly detection system, the detection model is identifying an anomaly, not a clearly defined attack pattern. Frequently there is not enough information to accurately classify the attack. Most approaches only report "normal" or "anomalous" events [Bivins et al 2002], [Chimphlee et al 2006], [Chen et al 2005]. Few approaches actively study the ability of an anomaly detection system to accurately classify activity [Bolzoni et al 2009], [Robertson et al 2006]

Tuning and optimization: Network operations are constantly evolving. This constant change is directly related to the core effectiveness of an anomaly detection system. As normal network usage evolves, the model of "normality" maintained by the detection system must be updated. While there is some recent research in on-line adaptation of anomaly based systems [Rieck et al 2008], [Rehak et al 2009], [Ciocarlie et al 2009] effective schemes remains an open research issue. Further, there are very few examples of dynamic tuning and optimization of anomaly based systems [Himura et al 2010].

Evaluation of approaches: The KDD and DARPA datasets have been criticized [McHugh 2000], [Mahoney and Chan 2003a], [Brugger 2007] However in [Tavallaee et al 2010] of the 276 approaches surveyed, 70% were evaluated using publically available datasets with the majority of these using either the KDD data set or DARPA set. The other 30% of the studies utilized datasets that they generated themselves with either simulated or recorded trace data. [Ringberg et al 2008] argue for the need for carefully constructed evaluation datasets consisting of a combination of simulated and true trace data.

2.9 Attack mitigation though quality of service allocation.

Many systems have been proposed that attempts to prevent or mitigate cyber attacks by applying quality of service principles. Techniques involving applying varying delay and filtering mechanisms to both packet and connection level communications have recently been studied. [Mahajan et al 2002] proposed a generic system for aggregate based congestion control. During periods of sustained congestion such as those caused by DoS attacks, the model identified congestion signatures using dropped packet histories. It used a destination based detection method in which high bandwidth destinations are selected, then suspected aggregate source streams are filtered according to the detected signature. The model also proposed "pushback" where downstream routers requested upstream routers implement QoS mechanisms against individual streams in the congestion cluster. While this approach is effective it is inherently unfair to legitimate traffic destined for congested areas. [Garg and Reddy 2002] Proposed a QoS mitigation strategy that combined rate based regulation strategies with a resource windowing strategy that controlled resource consumption of aggregated flows. [Yau et al 2005] Proposed a method of installing rate throttles in upstream routers. Congested servers could then request incoming traffic be limited according to a min-max fair feedback control algorithm. [Williamson 2002] proposed a system for controlling the spread of viruses in enterprise networks by limiting connection rates to new hosts at the network layer. Suspected malicious connections were delayed rather than dropped gaining the network operator time to perform more serious mitigation strategies while limiting collateral damage to legitimate traffic. [Sellke et al 2005][Song et al 2010] proposed rate limiting connection requests from suspected infected clients or subnets. [Chen and Tang 2007] proposed a method of restricting work propagation by dynamically dropping connection requests from sources based on their connection failure rate. As the connection failure rate of a source increased beyond a threshold, random connections from that source were dropped in order to keep the failure rate below a predefined limit.

3 Thesis Proposed Work

3.1 Overview

As discussed in Section 1, the need to protect cyber resources is ever increasing. Existing detection methods, based primarily on identifying and mitigating specific attack signatures are by themselves inadequate. Signature based methods are ineffective against zero-day attacks, and many new attack delivery methods apply sophisticated use of encryption thus disguising themselves from signature based systems. Anomaly based systems excel at identifying zero-day attacks, previously unknown, or well-disguised attack methods. Because anomaly detection systems do not rely on pre-programmed attack signatures, they are able to identify a much broader range of network threats. However this strength is also the source of the greatest deficiency. Existing anomaly detection systems lack the ability to interpret anomalous events. While anomalies are readily identified, every step in the process of applying anomaly detection to network defense requires extensive expert intervention. This requirement significantly delays the implementation of appropriate response actions thereby reducing overall effectiveness.

We propose to integrate learning techniques with quality of service approaches to move away from existing models that rely almost entirely on operator intervention. We will explore automated approaches that are able to quickly, accurately, and dynamically evolve with their environments with little or no expert input. We seek to develop an approach that can automatically identify, classify, assert confidence, and respond to anomalous events. We intend to advance the state of the art of intrusion detection by proposing fully integrated detection and response framework that is self-adaptive, self-tuning, self-optimizing and automatically responsive. We intend to specifically achieve the following contributions:

- Automate the gathering and maintenance of base models of network behavior on-line
- Automate the process of classifying anomaly alerts into an evolving attack taxonomy.
- Automate tuning of control parameters eliminating the need for expert knowledge of the operating environment and the underlying detection algorithm
- Propose a new model for optimal performance of an anomaly detection system
- Automate the optimization process on line to achieve the goals established in the proposed performance model.
- Develop algorithms for determining prediction confidence.
- Develop novel Quality of Service algorithms providing for delay differentiation with fair bandwidth sharing and buffer management.
- Automate initial defense response actions incorporating proposed QoS algorithms to provide a range of responses beyond traditional allow/disallow methodology.

Figure 3 illustrates the relationship between the theoretical and technical concepts proposed in this thesis. In the following sections, we describe our proposed ideas for achieving the desired contributions. In section 3.2 we discuss the need for confidence filtering. In sections 3.3 through 3.6 we discuss self-adaptive, self-tuning, self-optimizing, and automated response approaches. In section 3.7 we describe the elements of the proposed foundation framework. We conclude this section with the proposed evaluation approach, expected contributions and success criteria in Sections 3.8, 3.9 and 3.10 respectively.

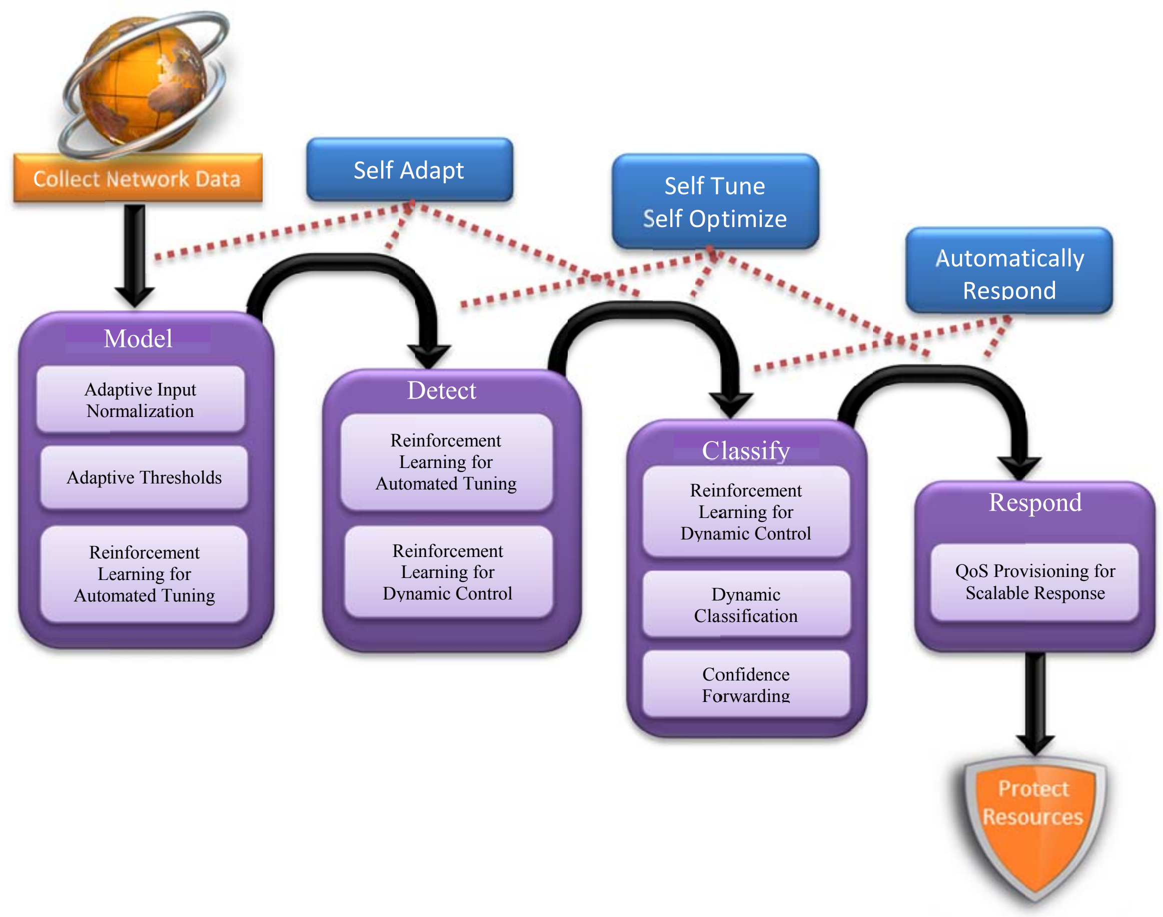

Collect Network Data
Self Adapt
Self Tune
Self Optimize
Automatically Respond
Model
Adaptive Input Normalization
Adaptive Thresholds
Reinforcement Learning for Automated Tuning
Detect
Reinforcement Learning for Automated Tuning
Reinforcement Learning for Dynamic Control
Classify
Reinforcement Learning for Dynamic Control
Dynamic Classification
Confidence Forwarding
Respond
QoS Provisioning for Scalable Response
Protect Resources

Figure 3: Relationship between Theoretical and Technical Principles

3.2. Prediction Confidence Forwarding

The need for confidence filtering in an anomaly detection system comes from the fact that compared to signature based systems, while anomaly based systems are able to identify a much larger number of attacks, they are also known for a high false alarm rate. Additionally, sophisticated attack methods are capable to training anomaly detectors to accept malicious activity as normal.

3.2.1 A Generic Detection Model

We begin the discussion with a brief look at a general model of intrusion detection. The model considers network events that are observed and represented by description vectors, and attempts to sort them into a set of event classes according to a well-defined policy.

Let X = the set of all possible network events $\{x_1, x_2, \dots x_n\}$. A network event is the action, state or behavior that we are evaluating to determine if it is malicious or benign. In practice, network events could represent individual packets, packet streams, connections, sessions, etc… Here we use the generic term event. Every x_i is a member of a single event class. Let C^M be the set of all events constituting malicious intent and let C^B be the set of all events constituting benign intent. $C^M \cup C^B = X$ and $C^M \cap C^B = \{\emptyset\}$.

Let V = the set of all possible description vectors $\{v_1, v_2, \dots v_k\}$. Where v_i is a vector of observable values related to x_i and represents the subset of the global state that the detection system can observe and is aware of. Vector v_i is model dependent and is the input to whatever detection algorithm is being used. In practice this typically includes information collected and aggregated from network logs, packet headers, system logs, statistical analysis, etc… It can be raw data or processed data. Any information which the classifier has access to and can use to make predictions about x_i is contained in v_i . In the case of anomaly detectors v_i also includes the learned model of normal behavior.

Let Q = a generic classifier. Classifier Q accepts v_i describing event x_i as input and produces a prediction of class membership p_i as output. When a network event occurs, Q is presented with v_i and based on the detection approach it is using, makes a prediction as to the class that the event belongs to. Due to generalization during vector construction $|X| > |V|$. As a simple example, a system that considers network level events, but is unable to consider packet payload during vector construction due to channel encryption, may construct identical vectors for individual events with identical network data but distinct content. It is then possible for a unique vector to describe events belonging to multiple distinct classes. That is $0 < P(x_i \in C^M|v_i) < 1$ and $0 < P(x_i \in C^B|v_i) < 1$. When multiple distinct events described by identical description vectors belong to different classes, prediction errors can occur.

Figure 4 is a Venn diagram where *M* represents the set of all possible events that are described by description vectors where given that description vector the probability the event is malicious is greater than 0. *B* represents the set of all possible events that are described by description vectors where given that description vector the probability that the event is benign is greater than 0. *E* represents the set of all events that are described by description vectors where given that description vector, sometimes the described event is malicious and sometimes it is benign. Assuming a perfectly trained and tuned system, all vectors describing events in *M*-*B* will be correctly predicted as attacks and all vectors describing events in *B*-*M* will be correctly predicted as benign. However vectors describing events in *E* can produce errors.

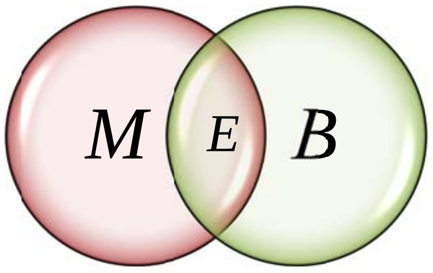


Figure 3: Classification of Events Based on Description Vectors

Consider a vector v_i describing an event x_i in *E* where $P(x_i \in C^M|v_i) = 0.99$ and $P(x_i \in C^B|v_i) = 0.01$. In this example, a reasonably trained or programmed classifier will make prediction $p_i \rightarrow x_i \in C^M$. However, 1% of the events described by vectors identical to v_i indicate $x_i \in C^B$ and will generate false positive errors.

3.2.2 Relationship between anomalies and intent

Specific to anomaly detectors there are two more overlapping sets we are concerned with. Anomaly detectors detect anomalies not necessarily attacks. In Figures 5a and5b: *A* = set of all anomalous events, *N* = set of all normal events, *B* = set of all benign events, *M* = set of all malicious events, and *FP*, *FN* = set of all events that can potentially cause false positive and false negative errors respectively. Figures 5a and 5b illustrate that there is overlap between anomalous behavior and benign intent. There is also overlap between normal behavior and malicious intent. When anomaly detectors raise alerts on every anomaly, false positive errors may occur. When attackers train anomaly detectors to recognize malicious activity as normal, false negatives may occur.

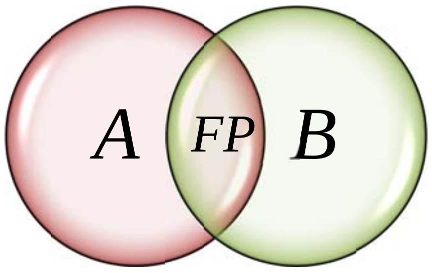


Figure 4a: Anomalous and Benign Events

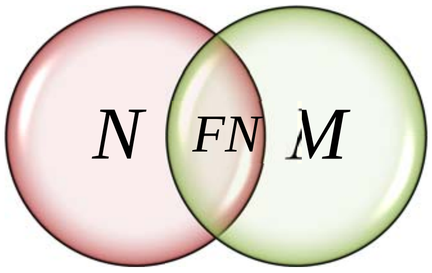


Figure 5b: Normal and Malicious Events

Due to the events that potentially belong to sets *E, FP*, and *FN*, we know in advance that there will be prediction errors. The need for confidence filtering comes from this certainty. In order to properly tune and optimize system performance we need to be able to not only make predictions, but also be able to report what the systems confidence in those predictions is. We will develop algorithms that will monitor prediction accuracy and consistency and provide a confidence rating for each subsequent prediction. The proposed model will identify three possible broad classifications for events, attack, benign, or confidence forwarding. Events in the confidence forwarding category are events that we suspect are either malicious or benign but the model has low confidence in that prediction. These events include normal predictions that are anomalous enough that it would be imprudent to let them go unmolested and anomalous events that are normal enough that it would be counterproductive to treat as full attacks. These events will be flagged and receive special handling by downstream processes. We will use the prediction rating as input into the self-tuning and self-optimizing and automated response processes which we will discuss in the following sections.

3.3. Self-Adaptive anomaly detection

We will propose several methods for a self-adapting framework. We will propose methods for on-line learning using dynamic input normalization and feedback-based thresholds to adapt over time to changes in the input data. Responding to monitored system performance and operator behavior we will adaptively classify attack predictions.

3.3.1 Adaptive Input Normalization

We will propose an adaptive input normalization approach that will bring attention to the true difference between individual input vectors. In many approaches, data is normalized to account for differences in scale between two different data points. As an example consider that the two data points are height and weight, height being 5~6 feet and weight being 100~200 pounds. Weight would always take precedence over height if the data is not normalized.   To capture relative distance between two different values of the same feature, we adjust the input pattern so that all values are in the range of 0 to 1 using simple linear scaling for the normalization. We consider the "distance" to be how different these two values are from one another. The smaller the distance, the more similar the values are considered. The scale used to normalize input data has an effect on relative distance.

In a simple example, Table 2, the reference range used to normalize input values varies from 1~16 to 1~256. For each reference range, two values 8 and 14 are normalized based on the range and their normalized distance is calculated. When the reference range is 1~16, two values are considered very different from each other as the normalized distance is 0.4. When the reference range is 1~256, they are considered very similar as the normalized distance is 0.023.

Minimum	Maximum	Value-1	Value-2	Normalized Distance
1	16	8	14	0.4
1	32	8	14	0.194
1	64	8	14	0.095
1	128	8	14	0.047
1	256	8	14	0.023

Table 2: Effect of Scale on Normalized Distance

In a network intrusion detection problem, there are several data points that the scale will be known ahead of time. Some points are in the scale of 0~1, while some others are in 0~255. However, as an example, in the KDD'99 dataset which is commonly used to evaluate anomaly detection approaches, 17 of the 42 data points have an undefined scale. Furthermore, the scale in the training data is not the same as the scale in the live data. Therefore, scaling done during the training process will not be accurate when applied to the live data. We will propose an adaptive input normalization approach that will automatically tune scaling parameters online based on the observed traffic patterns.

3.3.2 Adaptive Thresholds

We will propose methods for adaptive thresholds to enable and adjust anomaly detection capability. In addition we will propose methods for identifying anomalies in individual data points within a vector. Existing machine learning techniques use parameters such as the mean quantization error of the vector, or Euclidean distance between vectors [Rauber et al 2002], [Palomo et al 2008]. However, these approaches do not always identify anomaly conditions because they distribute the anomaly across the vector.

For example, consider three vectors representing two connections and one weight vector. Connection-1 and Connection-2 are each compared to the weight vector. Figure 6 shows the individual quantization errors for each data point in the vectors. If we were to use mean quantization error or Euclidian distance to evaluate Connection-1 and Connection-2, we would find that the mean quantization error and Euclidian distance are higher for Connection-1. However, it is obvious that Connection-2 has an anomalous value in the vector. For effective anomaly detection, we are interested in not only the aggregate small discrepancies, but also identifying large single discrepancies. We will develop threshold formulas that are able to identify both small aggregate anomalies and single point anomalies. We will then adaptively adjust these thresholds to account for concept drift.

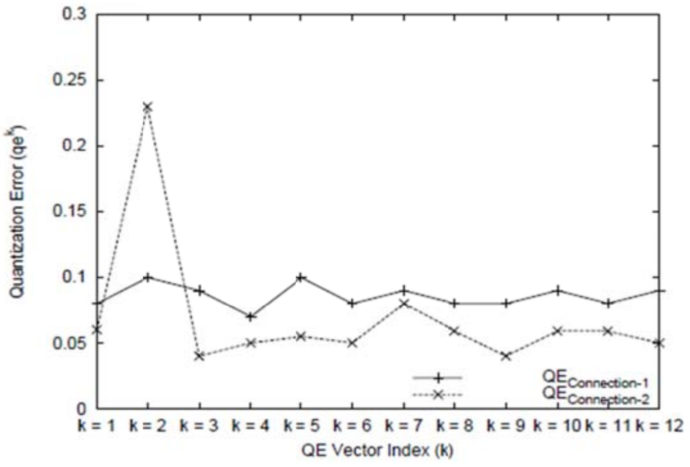


Figure 5: Example Quantization Error of two Input Vectors

3.3.3 Dynamic Attack Classification
Most anomaly detectors are only able to make binary predictions. Because anomaly based systems identify anomalies and not attacks events are either categorized as normal or anomalous. There is generally a direct assumption from there that events are malicious or benign. No additional classification is attempted. It is left to the security operator to research the anomaly and determine its threat level. Performing attack classification is an important step to mitigation and response [Ning et al 2002], [Valeur et al 2003].

Determining the attack class in a signature based system is trivial. The attack class is manually assigned during the signature development phase [Bolzoni et al 2009], [Roesch 1999]. However, in an anomaly detection system, the detection model is identifying an anomaly, not a clearly defined attack pattern. Frequently there is not enough information to accurately classify the attack. Most approaches only report "normal" or "anomalous" events [Bivins et al 2002], [Chimphlee et al 2006], [Chen et al 2005]. Few approaches actively study the ability of an anomaly detection system to accurately classify activity. [Robertson et al 2006] proposed a method for classifying anomaly detection alerts that relied on human programmed heuristics. While this method was effective, it relied entirely on off-line human intervention. It required extensive expert knowledge of the underlying system, and could not be adapted in a responsive manner. [Bolzoni et al 2009] proposed a semi-automatic approach that could infer anomaly classification by using machine learning approaches. The system used a combination of input from two primary sources: a signature based IDS, and feedback from anomalies that had been manually classified. While this approach eliminated the need for operators to program specific class signatures, it was only able to learn off-line. There was no incremental training. Thus, new previously unseen classes could not be classified until off-line training could be again accomplished. Further it relied on sanitized training data.

In this work we will propose methods that will go beyond binary classification and include adaptive attack classification capability. [Robertson et al 2006] observed that similar attacks have similar descriptive characteristics. We will propose machine learning techniques that respond to feedback and will use knowledge about similar known attack categories to derive appropriate classification methods and response actions for novel and zero-day attacks. When unclassified anomalies are discovered where no existing similar attack category is known, we will use feedback to add appropriate categories to the learned attack taxonomy.

3.4 Self Tuning Anomaly Detection

3.4.1 Parameter Tuning for Anomaly Detection
Parameter tuning is a difficult task and little attention has been paid to it [Ringberg et al 2007]. There is a need to obtain appropriate parameter tuning dynamically and automatically in order to accomplish real-time anomaly intrusion detection [Himura et al 2010]. In the tuning process we have a set of control parameters *CP* appropriate for a specific operating environment *OE*, a set of performance metrics *PM*, and a set of performance goals *PG*. Where *CP* is detection approach specific and includes all variables in the approach that are independent of the data being evaluated and can be set by a controller (i.e. size and dimensionality of a SOM, thresholds in statistical approaches, K in K-NN approaches, etc…). *OE* is the operating environment the approach is being tuned for (i.e. traffic statistics, attack rate, attack types, micro variance, macro variance, etc…). *PM* is the set of performance metrics (i.e. detection accuracy, false positive rate , classification accuracy, etc…). *PG* is the established acceptable values for the performance metrics. In a perfect system, accuracy would always be 100%, false positive rate would always be 0%. In practice this is not the case. *PG* represents the values that the researcher or operator considers acceptable. Typical values include 95% accuracy with 1% false positive. The primary difference between *tuning* and *optimizing* is that *tuning* focuses on establishing optimal values for the

control parameters to meet *PG* while *optimizing* focuses on identifying optimal values for *PG* to reach an overall performance objective. We discuss automated optimization in section 3.4.

In the tuning process, when given *OE*, we attempt to identify *CP* such that all objectives of *PG* are satisfied. In most approaches today, *CP* is established manually utilizing expert knowledge of both *OE* and the underlying detection approach. There are two main problems with this tuning method. First, by requiring expert knowledge of the detection method a burden is placed on network operators to learn the detection algorithm for every system they deploy. This is impractical for operational deployment. Second, presetting parameters in *CP* assumes that *OE* is stable. This is simply not the case. Network traffic is not stable in time or space. Macro operating conditions are constantly evolving. Workloads increase, attacks grow more sophisticated, and traffic patterns change. As these changes occur, changes to values for *CP* required to meet *PG* also occur. Micro changes must also be accounted for. Anomalies that would easily be detected during off hours when background traffic is light may be missed during peak hours when background traffic is very high. Tuning must account for both micro and macro environment changes. In this thesis we will develop algorithms for self-tuning based on reinforcement learning approaches. The proposed tuning process will not require manual input from an expert familiar with underlying detection approaches and will account for both micro and macro changes to the operating environment.

3.4.2 Reinforcement Learning for Automated Tuning

We will propose a reinforcement learning approach to automated tuning. The reinforcement learning process adapts how a controller makes decisions in an environment in order to maximize cumulative reward. The process is modeled as a Markov decision process (MDP) which consists of a set of environment states and a set of control actions for each state. After each state transition the controller receives a reward and observes the resulting state. The objective of the learning process is to develop a policy π that maximizes the cumulative rewards based on trial and error iterations. The state space consists of all possible operating environments. The action space consists of all possible control actions. A policy is a mapping from states to actions that optimizes the long term reward. The main advantage of using reinforcement learning for control is that the trial and error approach eliminates the need for expert intervention. There is no need to specifically tell the model which parameter to change and how to change it. However, when the state space and or the action space are very large, trial and error approaches can be slow to converge [Sutton and Barto 1998].

In intrusion detection, both the state space and the action space can be extremely large. In traditional approaches, initial expected rewards for each state-action pair are set to some arbitrary value and adjusted as the model iterates. This implies that until a particular state-action pair has been iterated over several times, its estimated value is inaccurate.

With a large state space, if every possible state-action pair were set to arbitrary initial values, convergence toward true expected reward would be slow and in efficient. Additionally, maintaining values for every possible state-action pair requires excessive system resources. One possible approach would be to reduce the size of the state space by further generalizing the environment. This method would increase the responsiveness of the learning approach but would decrease the ability to fine tune control policies to a wide range of environments. We will propose a learning algorithm for automated tuning using a combination of arbitrary initial value and nearest neighbor inheritance to set initial values for state-action pairs. Inheritance will come from a stored sub-set of possible environment states. Our proposed approach will balance responsiveness, control resolution, and resource consumption.

3.5 Self Optimizing Anomaly Detection

A self-optimizing control problem is one which involves both the explicit determination of the optimal objective and the control actions necessary to achieve that objective [Li and Horowitz 1997]. In anomaly detection the traditional optimal objective is maximizing accuracy while minimizing false positive rates. Control actions involve tuning parameters to adjust the sensitivity of the anomaly detector to achieve this objective. We will develop three dynamic performance metrics evaluating the effectiveness of false positive, false negative and confidence forwarding. We will propose self-optimizing algorithms that achieve an optimal balance between these three functions. Our proposed self-optimizing approach will allow operators to set performance goals for any two of the three metrics and will then achieve these goals while attempting to optimize performance in the third.

3.5.1 A Model for Optimal Objective for Intrusion Detection

In many works it is common place to attempt to achieve static and arbitrary performance goals for both false positive rates and false negative rates [Tavallaee et al 2010]. A common goal is < 1% false positive rate while maintaining 95% accuracy. In this thesis we will argue that arbitrary base rates for false positive and false negative are not optimal.

False Positive: In many cases, false positive rates well below 1% are still too high to be an effective tool to network operators [Axelsson 1999]. The reason is based on Bayesian inference. Even with high accuracy and low false alarm rates, in time of low overall attack rates, false alarms will still dominate the alert log. Rather than attempting to achieve a static false positive rate, we aim to achieve an efficiency rating *Eff* where *Eff* is the percentage of alerts in the alert log that are true positives . *Eff* is a function of traffic load, accuracy, false positive and attack rate

False Negative: Base rate false negative metrics are also less effective. Individual instances of malicious activity are not equally harmful. Today's networks are under constant attack. Accurately identifying a large number of harmless probes and old attacks that have already been mitigated but yet missing a tiny number of new or destructive events could be catastrophic for a network despite maintain a near perfect accuracy rating. In our proposed framework we will propose weighted accuracy metrics based on potential threat to the network.

Confidence Forwarding: With the use of confidence forwarding some percentage of evaluated events will be flagged for special handling by downstream processors. This special processing will incur some cost. The cost may include the additional processing, the delay to benign traffic during the additional processing, or the added threat to delaying response to actual attack events. We will propose a forwarding metric as a function of the amount of traffic forwarded and the cost caused by the additional processing and response delay.

3.5.2 Reinforcement learning for Dynamic Control

The proposed model will balance three performance metrics in real time. The *False Positive* metric will define the performance goal for false positive rate as a function of attack rate, target efficiency, and system accuracy. The *False Negative* metric will define the performance goal for false negative as a function of accuracy and attack threat. The *Confidence Forwarding* metric will define the performance goal for confidence forwarding rates as a function of accuracy, false positive and processing cost. This system of equations is severely affected by the dynamic system state. As the system operates, the specific micro goals for these three functions are constantly changing as a result of variations in system state and in relationship to each-other. Maintaining steady control of this system of equations will require constant tuning of control parameters.

Achieving optimized control is made difficult by two challenges. First, there is a large number of potential control parameters that have an effect on more than one performance metric. Achieving the desired response to a change in system performance or environment may require adjustment from one to all possible control parameters. The second challenge is that making a change to a particular control parameter has a varying effect on that parameters target metric depending on the current environment state. That is, if parameter 1 is intended to control metric A, and we make an adjustment to parameter 1, the magnitude and direction of the effect that this change has on metric A depends on the environment state. With one environment state, increasing 1 may increase A, while in a different environment state, increasing 1 may decrease A.

An effective optimization mechanism will need to know both the current operating environment, and the current control objectives. We will propose methods for communicating this information to the controller though the state space. We will capture the current environment state by using a Multi-Layer Perceptron (MLP) with hidden layers calculating the control sub state and the operation sub state illustrated in Figure 7. The control sub-state will help inform the controller of which metric or metrics are in most need of correction. The operation sub-state will give the controller information about the particular operating environment so that it may anticipate the impact that its control decisions are going to have on the reward signal.

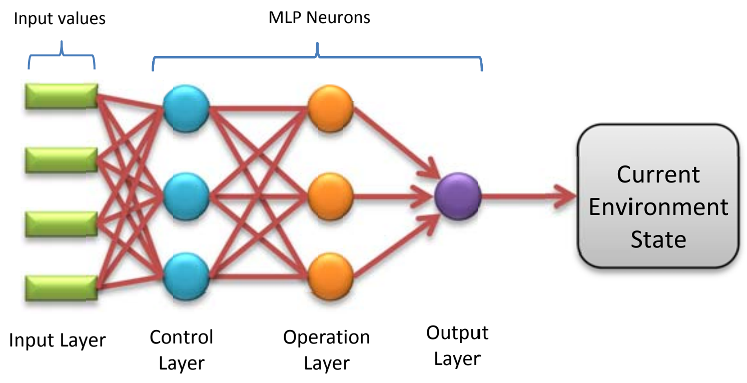


Figure 7: MLP to Capture Environment State

3.6 Automated response

Most existing detection methods are passive methods that raise alerts but do not initiate response actions. Our proposed approaches will identify and classify attacks and will also assert prediction confidence. Based on the attack class and prediction confidence we will propose methods for both acute and scalable automated response.

A key element of our proposed approach is the idea of confidence filtering. Anomaly detection systems, when compared to signature based systems, are inherently over sensitive. Not all anomalies are attacks. We therefore propose methods for not only making attack predictions, but also learning to assess confidence in those predictions. When we have low confidence in the predictions that we make, we find that there are one of two possible situations.

- We have predicted that an event is benign. However, we reasonably believe that it may be malicious.
- We have predicted that an event is malicious. However, we reasonably believe that it may be benign.

Traditional methods for responding to malicious activity involve various ways of completely denying access to the suspected attack. In our model, in cases where our prediction confidence is low, we propose to develop quality of service algorithms and use these algorithms to provide a set of scalable response actions to the response engine.

3.6.1 Learned Response to Attack Taxonomy

We will propose methods for automated response based on an evolving attack taxonomy as proposed in section 3.3.4. We will develop an evolving two dimensional taxonomy. One dimension will identify the type of attack (DoS, U2R, R2L, etc…). The second dimension will help identify appropriate action. This evolving taxonomy will be used to drive automated response actions. Consider two worm attacks. Attack *A* is an older attack that has already been mitigated. Attack *B* is a new worm attack spreading across the enterprise. While both are of the type worm, they do not both warrant the same response. Attack *A* may simply warrant logging while attack *B* requires actions to prevent spreading and sanitize already affected network resources. We will develop algorithms for monitoring operator behavior and further separating attack taxonomy categories into response groups. We will propose a set of acute response actions appropriate to the attack taxonomy and will develop methods for automatically implementing responses to high confidence attack predictions.

3.6.2 QoS Provisioning for Scalable Response

In cases where events have been flagged for confidence filtering, we will develop a set of scalable response actions. By using scalable responses against low confidence predictions, we will be able to partially mitigate potential attacks while limiting collateral damage. We will develop and integrate novel QoS algorithms to provide delay differentiation, bandwidth sharing, and dynamic buffer management capabilities. While traditionally, delay differentiation, bandwidth sharing and buffer management are seen as orthogonal issues, our packet scheduling with buffer management approach will provide delay and bandwidth differentiation in an integrated way. Our proposed algorithms will enhance the flexibility of network resource management and multi-dimensional QoS provisioning providing a basis for scalable attack response.

Consider a low confidence attack prediction against a DoS, worm, or information theft attack. Using our proposed approach we will be able to implement QoS partial mitigation to the suspected activity. In the case of a DoS attack the partial mitigation will build a firewall against aggressive behavior and hence protect network resources from saturation. In the case of a worm, the partial mitigation will delay spreading of the infection and limit potential damage. In the case of information theft, the partial mitigation will limit the amount of information that could be retrieved. In all these cases, if the attack prediction turns out to be accurate, the delay implemented by the partial mitigation offers security operators time to respond before critical damage has been achieved. If the attack prediction turns out to be incorrect, while some legitimate traffic has been interrupted, overall quality of service has been reduced, but not terminated thus limiting collateral damage.

3.7 A Framework for Automated Anomaly Detection

We will propose an integrated framework that includes automated methods for maintaining a base model of network behavior, anomaly detection, confidence forwarding and automated response. We will demonstrate our ideas by developing a prototype framework based on a Growing Hierarchical Self Organizing Map. The Self Organizing Map is a popular anomaly detection approach [Rhodes et al 2000] [Ramadas et al 2003] [Kayacik et al 2003] [Sarasamma et al 2005] [Palomo et al 2008] [Jiang et al 2009] [Wang et al 2010]. One variant of the SOM is the GHSOM [Palomo et al 2008]. We chose the SOM because of its popularity as a basic approach and because many of the specific algorithms we propose to modify it with can easily be ported to other machine learning methodologies.

3.7.1 Growing Hierarchical Self Organizing Map

GHSOMs have been proposed and used effectively to classify high dimensional data [Rauber et al 2002]. We will use an Automated GHSOM as a prototype to demonstrate our methods. In a GHSOM, the size and dimensionality of the map architecture are determined during the training phase. The initial map size is very small, usually a single layer 2x2. During the training process, the map is grown both vertically and horizontally until the training process is complete. After each training iteration, the deviation of the input data (quantization error) is computed. The map is grown horizontally by adding rows and columns to the map to reduce quantization error. The map is grown vertically by adding child layers to parent layers that exceed the pre specified maximum dimensionality of each map. This process continues until the quantization error of the map is below an established threshold. The purpose of growing the map in this manner is to ensure it is well suited to the input set. After training, regions of the map are labeled as either normal or attack regions. It is important to note that at the end of the training process, each layer and sub-layer can have a different number of maps and sub-maps of varying dimensionality as illustrated in Figure 8.

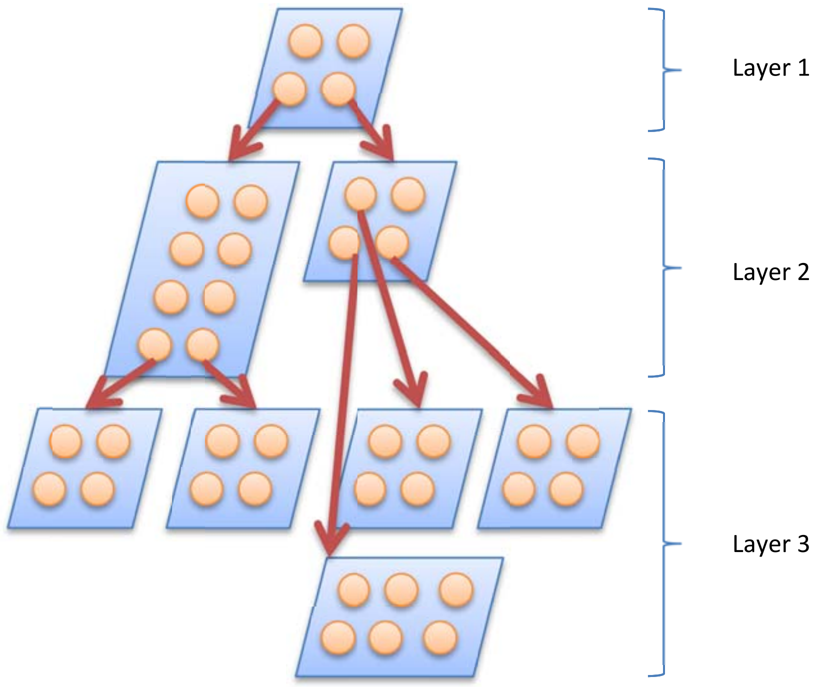


Figure 8: GHSOM After Off-Line Training

During live operation input vectors representing network events are presented for evaluation and classification. Using some vector distance formula, they are matched to the best matching unit (BMU) in the map. The quantization error (QE) between the input vector and the BMU is observed. If the BMU is in an attack region the connection is considered attack. If the BMU is in a normal region and the QE is below some threshold, the connection is considered normal. If the QE exceeds some threshold, the connection is considered anomalous and treated as an attack.

Dynamic Size and Dimensionality: After offline training, the GHSOM will have a finite structure consisting of a hierarchy of fixed size SOMs. Each region in each map will be labeled to either predict benign or malicious network activity. This structure will be well suited to the training set and subsequent connections that match patterns in the training set. Existing GHSOM approaches to anomaly detection to not adapt the map dimensionality after off line training. However, over time, the normal behavior of the network will evolve. As new patterns are consistently observed that the model is not well suited to predict by other adaptations, there will be a need to adapt the size and or dimensionality of the hierarchy itself. For example when there is new legitimate behavior that is consistently flagged as anomalous or new attacks that are matched to normal regions of the map. We will propose methods to grow the underlying structure on-line to account for these new patterns. The larger the map, the higher the resolution and greater the number of distinct patterns it can identify. However, do to resource constraints we cannot simply grow the model indefinitely. We must strike a balance between performance and accuracy. We will propose methods for identifying under-utilized or under-performing regions of the map and devise self-pruning algorithms. We will develop algorithms that maximize accuracy and confidence while minimizing resource utilization.

We will use the GHSOM as the foundation for our prototype detection engine. We will apply the algorithms we develop for Dynamic Input Normalization, Adaptive Thresholds, Confidence Forwarding, Reinforcement Learning for Automated Tuning and Optimization, and Dynamic Attack Classification. Detection predictions will then be processed for automated response.

3.7.2 QoS Based Mitigation for automated response

Most existing detection methods are passive methods. That is, they raise alerts, but they do not actively respond to attacks [Eskin et al 2002], [Kayacik et al 2007], [Yu et al 2008]. While there are some active response anomaly detection systems, they do not provide automatic scalable response actions[Spaffod and Zamboni 2000] [Roesch 1999]. Instead, they provide mechanisms for terminating sessions. Our proposed model will use quality of service algorithms to provide scalable automated response based in part on confidence filtering.

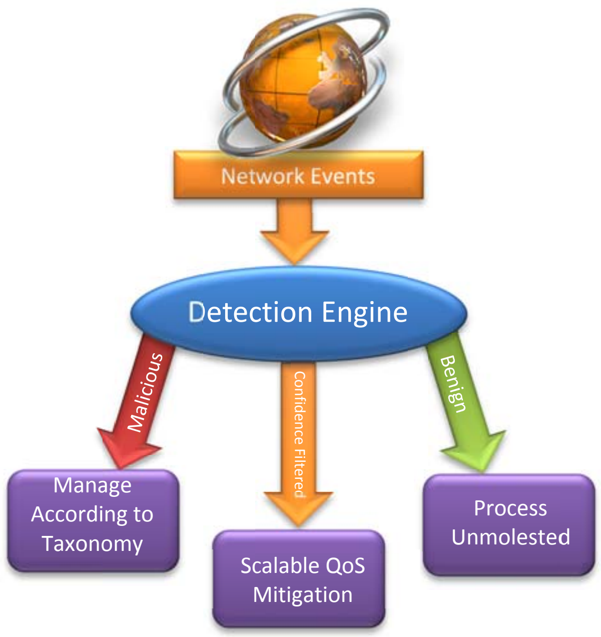


Figure 9: Proposed Detection and Response Model

Figure 9 illustrates this concept. When events are processed by the detection engine they will be broadly categorized in one of three ways. Benign predictions will be processed unmolested. Malicious predictions will be managed according to the appropriate response based on evolving attack taxonomy. Events identified for confidence forwarding will neither be unmolested nor will they be handled solely according to an attack taxonomy. Scalable QoS measures will be applied based on a combination of the prediction and the confidence in that prediction. We will develop novel algorithms to allow QoS provisioning in delay, bandwidth utilization and packet loss. We will combine attack class prediction with confidence forwarding to apply mitigation actions to confidence forwarded events. Events with suspected high threat and higher confidence will receive more severe mitigation. Events with limited threat or very low confidence will receive limited mitigation. We propose learning algorithms to adapt the scalable mitigation to balance network defense with collateral damage to legitimate traffic.

3.8 Evaluation Approaches

This section presents our initial ideas for evaluating the effectiveness of our proposed methodologies. A common method for testing new intrusion detection systems is the KDD and DARPA datasets [Hettich and Bay 1999] [Tavallaee et al 2010]. The KDD'99 dataset contains records describing TCP connections. Each connection is represented by 41 features. The dataset includes normal connections as well as 23 different types of attacks belonging to four categories: Denial of Service (DOS), Probe attacks, User to Root (U2R), and Remote to Local (R2L). The dataset includes a training set, and two test sets: the "whole" set and the "corrected" set. The "corrected" set includes 17 attack types that are not sampled in the training set and thus are "unknown" to a trained model. Due to the popularity of the KDD set we will use this set as a way to baseline performance and compare our work against other anomaly detection schemes.

There have been arguments made against the use of the KDD dataset [Brugger 2007], [McHugh 2000]. Indeed that set alone will not be adequate to evaluate our objectives. In order to address these concerns, we will build an alternate data set consisting of a mixture of live trace data and simulated network traffic. The database will be constructed in three steps. We will start with a publicly available trace dataset and perform minor pre-processing to correct header/payload discrepancies created during packet anonymization. We will then process that dataset through a commercial intrusion detection system to categorize attack vs normal traffic. Finally, we will insert simulated traffic to ensure adequate coverage of a variety of network attacks. We will evaluate the capability of our QoS algorithms to achieve desired differentiation under various network conditions using delay, and loss rate as performance metrics. We will test the capability of our detection framework using false positive, accuracy and forward rates as base performance metrics. We will design tests to evaluate the ability of our proposed methods to achieve desired goals under a variety of circumstances and evaluate the ability of the model to adapt itself to changing conditions. We will design tests to evaluate the effectiveness of our proposed automated responses compared to a network requiring manual response. We will chart analyze and document the ability of the proposed methods to achieve intended goals.

3.9 Expected Contributions

We intend to advance the state of the art of intrusion detection by developing algorithms for self-adaptive, self-tuning, self-optimizing and automatically responsive anomaly detection systems. We intend to specifically achieve the following contributions:

- Automate the gathering and maintenance of base models of network behavior on-line
- Automate the process of classifying anomaly alerts into an evolving attack taxonomy.
- Automate tuning of control parameters eliminating the need for expert knowledge of the operating environment and the underlying detection algorithm
- Propose a new model for optimal performance of an anomaly detection system
- Automate the optimization process on line to achieve the goals established on the proposed performance model.
- Develop algorithms for determining prediction confidence.

- Develop novel Quality of Service algorithms providing for delay differentiation with fair bandwidth sharing and buffer management.
- Automate initial defense response actions incorporating proposed QoS algorithms to provide a range of responses beyond traditional allow/disallow methodology.

We seek to advance network anomaly detection from pure theory to a useable model by significantly reducing the need for operator intervention and proposing methodologies for fully integrated automated detection and response systems.

3.10 Success Criteria

In this section we define specific criteria for success. Four key areas are essential to the success of this thesis. These include the following.

1. Contribution of new algorithms to automatically adapt anomaly detection methods without the need for expert reprogramming or extensive operator knowledge of the underlying detection approach.

2. Contribution of new algorithms to automatically tune control parameters in a variety of environments and circumstances without the need for expert knowledge of the operating environment or the underlying detection and response algorithms.

3. Contribution of new algorithms to automatically optimize anomaly detection systems without the need for expert knowledge of the operating environment or the underlying detection and response algorithms.

4. Contribution of new algorithms and demonstration of the effectiveness of scalable response actions in uncertain detection environments over traditional all or nothing response approaches.

Specifically, we will construct a prototype system using the proposed approaches. We will demonstrate the ability to update the base model accounting for concept drift and will demonstrate resistance to base model poisoning. We will demonstrate the ability to accurately classify attacks into the correct category at least 90% of the time. For previously unseen attack categories we will demonstrate the ability to learn the appropriate threat level and take correct mitigation actions 90% of the time. We will demonstrate the ability to self-tune modes under a variety of conditions to achieve base performance rates of 95% accuracy and 1% false positive rates. We will demonstrate the ability to set performance goals in any 2 of three detection metrics (false positive, accuracy, and confidence forwarding). Our proposed methods will self-optimize to consistently achieve the 2 set goals and through learning approach optimal on the third performance metric under a variety of operating circumstances. We will demonstrate the ability of our QoS algorithms to defend against DoS attacks, and delay worm propagation. Additionally, we will demonstrate the effectiveness of scalable automated response compared to response requiring operator intervention.

4. Preliminary Studies

This section will discuss some initial work we have competed. The following sections discuss preliminary studies on the proposed framework including an Adaptive Growing Hierarchical Self Organizing Map, Quality of Service algorithms for packet scheduling with buffer management, and a TD-SIM data set consisting of a mixture of trace data from an operational network and simulated traffic.

4.1 Adaptive Growing Hierarchical Self Organizing Map

Our A-GHSOM approach develops an online learning process using dynamic input normalization and feedback-based quantization error thresholds to adapt the system over time to changes in the input data. It also uses a confidence filtering and forwarding mechanism to identify traffic with low prediction confidence. The following subsection give the design details for important components of the A-GHSOM detection framework.

Dynamic Input Normalization Process: We normalize input vectors to account for differences in scale. Input data contains both numeric and symbolic values. In the dynamic input normalization process, numeric inputs are normalized to values between 0 and 1. Flag values are normalized to integer values greater than 1 so that the possible outcomes of comparing two flag values are only differences equal to zero or greater than or equal to one. All other values are normalized according to the normalization function *F(x)*. When compared to the weight vector, differences close to 0 are considered extremely similar with 0 being identical, differences close to 1 are considered very different (differences ≥ 1 are considered infinitely different).

The normalization function *F(x)* is given as follows:

$$F(x) = \begin{cases} \dfrac{x - \min(t)}{\max(t) - \min(t)} & max(t) \neq \min(t) \\ 0 & \max(t) = \min(t) \end{cases}$$

where *x* is a data point in the input pattern, *min(t)* and *max(t)* are the expected minimum and maximum values for that data point at time *t* respectively. Prior to training, the minimum and maximum values in the training set are calculated for each data point. These values are used to normalize the input patterns at time *t*.

As the effect scale can have significant impact on identifying the variance, the ideal situation is to have the narrowest range possible to normalize the data and therefore amplify the true difference between vectors. Over time, if no input data is as low as the minimum, raise the minimum. If no data input is as high as the maximum, lower the maximum. As live traffic is processed, at periodic intervals from time *t* to time $t{+}n$ where n is the number of records processed, the observed minimum and maximum values are recorded as $obs_{min}(t)$ and $obs_{max}(t)$ respectively. At time *t*, the minimum and maximum values are adjusted according to the following equations.

$$\begin{aligned} \max(t+n) &= \min(t) + R(t) \cdot e^{-\alpha} + \beta max\Delta(t) \\ \min(t+n) &= \max(t) - R(t) \cdot e^{-\alpha} - \beta min\Delta(t) \end{aligned} \qquad (1), (2)$$

In equations (1) and (2), we have

$$max\Delta(t) = \begin{cases} obs_{max}(t) - \max(t) & obs_{max}(t) > \max(t) \\ 0 & obs_{max}(t) \leq \max(t) \end{cases}$$

$$min\Delta(t) = \begin{cases} \min(t) - obs_{min}(t) & obs_{min}(t) \leq \max(t) \\ 0 & obs_{min}(t) > \min(t) \end{cases}$$

$$R(t) = \max(t) - \min(t)$$

Control parameters α and β have values between 0 and 1. α controls the exponential growth/decay rate. β controls how differences between the current used min/max and the observed min/max is used.

Figures 10a and 10b demonstrate the relationship between the observed min/max values and the adapted min/max values. They represent a subset of the ``test-bytes" feature of the KDD dataset. The x-axis represents the time interval that the measurement was recorded.

The y-axis represents the min or max value in bytes. The plotted series represent the relationship between the observed values and the adapted values with α = 0.05 and β = 1.0. The adapted values are used for scaling during the normalization process.

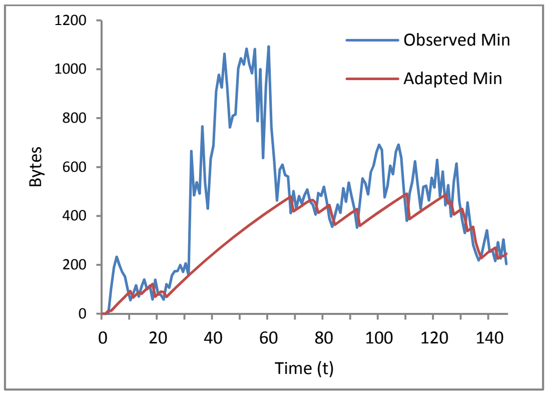


Figure 10a: Examples of Adapted Minimum Values

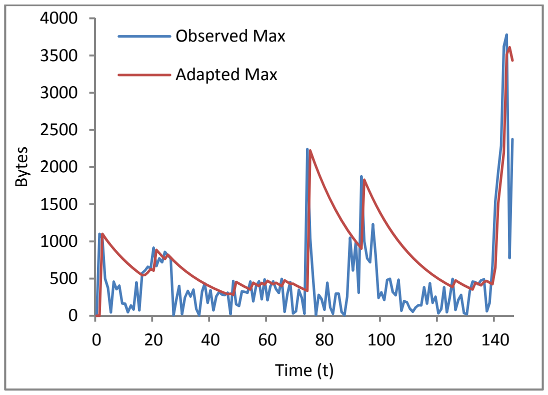


Figure 10b Examples of Adapted Maximum Values

Feedback Based Threshold Adaptation: A-GHSOM is further enhanced by use of feedback-based quantization error threshold adaptation. It adaptively adjusts thresholds for each node as input patterns are applied and add new nodes when appropriate. Each node is assigned two initial threshold parameters. $\tau_1$is used to calculate the threshold error value. The threshold error value is calculated using the quantization error vector as follows:

$$TEV_j = \sum_{k=0}^{n} f(k)$$

$$f(k) = \begin{cases} qe_j^k & if\ k > \tau_1 \\ 0 & if\ k \leq \tau_1 \end{cases}$$

Here, τ is a threshold that controls how closely an element of an input vector must match an element of a weight vector before it is no longer factored into the threshold error value. $\tau_2$is used as an upper limit on the acceptable total quantization error and used to calculate the quantization error boundary (QEB) for a selected node as follows.

$$QEB_j = \begin{cases} 0 & tqe_j < \tau_2 \\ 1 & tqe_j \geq \tau_2 \end{cases}$$

Where $tqe_j = \sum_{k=1}^{n} qe_j^k$

During live intrusion detection, input patterns are applied to the A-GHSOM and the best matching unit is selected. However, because of the size of the problem space, there is no guarantee that the best matching unit is a good match to the input pattern. The only guarantee is that it is a better match than all the other nodes in the neural network. The threshold error value and quantization error boundary are used to determine if the connection being processed is within thresholds. If and only if they are identically equal to 0, is the pattern considered within thresholds.

Each node in the final A-GHSOM is marked either ``normal", ``unmarked", or with an ``attack class". As patterns are examined, they are mapped to one of the three node types and considered within thresholds or not. Results are then used to make predictions to identify the suspected connection type, either ``normal" or ``attack class". Any pattern that is not within thresholds is identified as a suspected attack.

Even though its best matching unit is labeled ``normal", the fact that its threshold error value or its quantization error boundary is greater than 0 means this connection is anomalous in nature and it is assumed to be an attack. Likewise, any patterns mapped to a best matching unit of type ``unmarked" are patterns that have not been previously identified and are suspected attack.

The actual connection type is then compared to the suspected connection type and a result of ``correct" or ``incorrect" is identified. The distinction of correct or incorrect is made based on observing operator behavior. It is assumed that to some degree, that by monitoring operator response to alerts or lack of alerts on a small percentage of predictions we will be able to identify that an attack prediction was actually a false positive or that an attack was missed. In the absence of feedback, the system assumes that its prediction was correct. Table 3 details the adaptation rules based on perceived prediction result.

Best Matching Unit Type	Within Thresholds	Prediction	Result	Adaptation Rule
Attack	Yes	Attack	Correct	No Action
Attack	No	Attack	Correct	No Action
Attack	Yes	Attack	Incorrect	Grow Network
Attack	No	Attack	Incorrect	Grow Network
Normal	Yes	Normal	Correct	No Action
Normal	No	Attack	Correct	No Action
Normal	Yes	Normal	Incorrect	Lower τ_1, τ_2
Normal	No	Attack	Incorrect	Raise τ_1, τ_2
Unmarked	Yes	Attack	Correct	No Action
Unmarked	No	Attack	Correct	Label Node
Unmarked	Yes	Attack	Incorrect	Label Node
Unmarked	No	Attack	Incorrect	Label Node Raise τ_1, τ_2

Table 3: Feedback Based Adaptation Rules

Dynamic Pruning: In the adaptation rules detailed in Table 3 make the A-GHSOM dynamic in one direction only. That is over time, the network grows to accommodate concept drift. However, the system does not self-prune. Given an infinite amount of time, the model will grow continuously until resource utilization is out of control. The size of the A-GHSOM directly affects its performance. The larger the network, the more diverse the event set it is able to accurately classify. However, the larger the network becomes, the more resources it consumes. In order to limit resource consumption we establish a static maximum size Φ and a controllable dynamic maximum size ϱ. We define the current size of the A-GHSOM as the total number of nodes in the hierarchy designated σ. Every time an event is processed, the parent SOM of the best matching node and all of that SOM's ancestors are marked with a freshness value equal to the current time stamp. As the system processes events, the controller dynamically adapts ϱ. When the A-GHSOM size exceeds ϱ, stale branches of the hierarchy are pruned so that at all times, size $\sigma <= \varrho <= \Phi$. Algorithm 1 summarizes the process

Algorithm 1

```
1:  evaluate $\sigma$
2:  **if** ($\sigma > \varrho$)
3:     **repeat**
4:         using breadth first search, identify map $\Gamma$ with oldest time stamp.
5:         **if** $\Gamma$ has a child
6:              **repeat**
7:                  $\Gamma \leftarrow$ stalest child of $\Gamma$
8:              **until** $\Gamma$ has no children
9:         **end if**
10:        **delete** $\Gamma$
12:    **until** ($\sigma <= \varrho$)
13: **end if**
```

By pruning the network, we increase efficiency however, we sacrifice classification resolution. The existing nodes are forced to generalize more and more, and prediction errors may occur.

Confidence Filtering and Forwarding: Here we discuss initial ideas for a prediction confidence filtering and forwarding mechanism. It monitors the neuron consistency and accuracy and uses those measures to develop a neuron confidence rating. It then uses the rating to identify an appropriate action for the system to take regarding predictions made by nodes with which the system has low confidence.

In a traditional GHSOM, individual neurons and regions of the map are organized to classify connections. All similar connections matched to identical regions of the map will be classified as the same. However, in A-GHSOM, due to the addition of dynamic input normalization and feedback based threshold adaptation, this is not the case. It is possible for two connections to be matched in an identical region on the map, but be placed into different classification categories depending on their threshold error value and quantization error boundary value. Thus, under varying conditions, agitation of an individual neuron will prompt varying predictions. Neurons that are essentially trained to agitate on normal connections can indicate attack and vice versa.

We monitor the frequency that each neuron predicts each connection class and use this condition to calculate a consistency rating. After each connection is processed at time *t* and a prediction made by neuron *j*, consistency for the predicting node is calculated according to

$$Attack_t^j = \frac{\lambda(Attack_{t-1}^j + \ PREDICT_{Attack}}{\lambda + 1}$$
$$Normal_t^j = \frac{\lambda(Normal_{t-1}^j + \ PREDICT_{Normal}}{\lambda + 1}$$
$$Consistancy_t^j = \frac{max(Attack_t, Normal_t)}{Attack_t + Normal_t}$$

where $\lambda$is the size of the history, $PREDICT_{attack}$ = 1 if an attack is predicted and 0 otherwise, and $PREDICT_{normal}$ = 1-$PREDICT_{attack}$.

Because A-GHSOM also expects limited feedback from a network operator, we are able to estimate an accuracy rating for each node. As the amount of feedback received by the system is limited, the system assumes that its predictions are correct in the absence of feedback. $PREDICT_{accurate}$ is equal to 0 if corrective feedback is received and 1 otherwise. Accuracy for predicting node j is calculated according to.

$$Acc_t^j = \frac{\lambda\left(Acc_{t-1}^j\right) + \ PREDICT_{Accurate}}{\lambda + 1}$$

We use a combination of consistency and accuracy to calculate a confidence rating as follows.

$$Confidence_t^j = \frac{\lambda\left(Confidence_{t-1}^j\right) + \ Acc_t^j + Consistancy_t^j}{\lambda + 1}$$

As the number of predictions by a node increases, three possible consistency scenarios can occur.

1) A node trained ``attack" consistently makes ``attack" predictions.
2) The node trained ``normal" consistently makes ``normal" predictions.
3) A node trained ``normal" makes a significant number of ``attack" predictions.

Situations 1 and 2 are not a concern if the accuracy is also high. If an ``attack" neuron consistently makes inaccurate ``attack" predictions, new neurons will be added to that region trained to predict ``normal" and correct the deficiency. Likewise, if a ``normal" neuron consistently makes inaccurate ``normal" predictions, new neurons will be added to that region trained to predict ``attack" and again, the deficiency is corrected. The third situation, however, is a major concern.

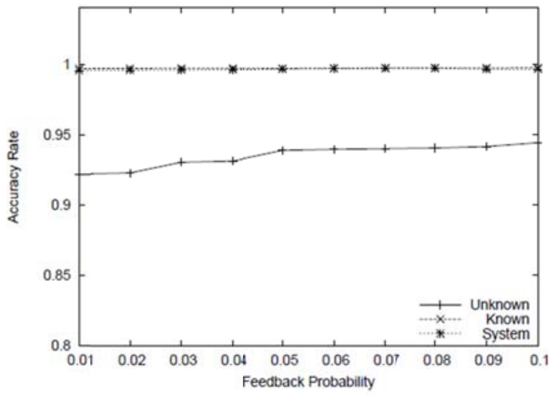
According to the adaptation rules of A-GHSOM model, all neurons should eventually stabilize to a consistent state. However, it assumes that normal connections and connections of attack types are differentiable based on the available data. If there exist normal connections and attack connections with identical or extremely similar connection vectors, the adaptation process will be stalled and inconsistent predictions will be made. We identify nodes that have low confidence and flag them for special handling by subsequent processors.

Initial Simulation Results: In this section we discuss preliminary results of the initial adaptation principles applied to the A-GHSOM. We test our approach using the KDD dataset. For each connection, a decision is made to process the connection or to forward to the operator based on system confidence. For each connection processed, a classification of "attack" or "normal" is made. The performance metrics are the accuracy rate and the false positive rate. The accuracy rate is the ratio of the total number of correct attack predictions over the total number of attacks processed. The false positive rate is the ratio of the total number of false positive predictions over the total number of normal connections processed. To establish a baseline for performance comparison, we processed the corrected dataset without applying any of the four enhancements. Without using thresholds and adaptations, connections are simply classified based on the node they are mapped to. Table 4 gives the overall accuracy rate, individual rates for each attack category, and unknown and known attacks. The overall false positive rate is 0.58%. Next, we study the impact of enhancements of the A-GHSOM approach on the performance of intrusion detection.

Category	Accuracy
Overall	92.21
Known	92.81
Unknown	10.95
DoS	98.48
R2L	7.66
U2R	51.43
Probe	78.83

Table 4: Baseline Accuracy of A-GHSOM

Integrating the proposed enhancements in the A-GHSOM approach we see a significant increase in accuracy with only minor increase in false positive rates. Figure 11 and 12 demonstrate the effect of the integrated A-GHSOM approach on intrusion detection. The system is adapted every 20000 connections using different parameters. Input normalization parameters α and β are set to .03, and the confidence forwarding probability is 80%.



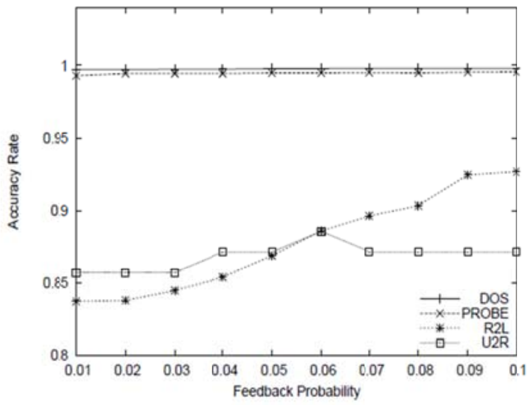


Figure 11: Detection Accuracy by Attack Category **Figure 12: Detection Accuracy for Known and Unknown attacks**

In the experiment, 91.9% of the connections are classified by the integrated approach. 8.1% of the connections are selected for forwarding. The integrated approach achieves an overall accuracy rate of 99.63% while maintaining the false positive rate of 1.8%. All four categories of attacks are predicted with above 87% accuracy. We see the sharpest increase in the accuracy of R2L detection. This is due to the fact that as the system receives more feedback, it becomes less confident in the nodes processing R2L connections and thus is able to more accurately identify potential vectors for forwarding.

The integrated approach is exceptional at identifying previously "unknown" attacks. It is intuitive that with excessive feedback, the accuracy would increase with "unknown" attacks. However, the approach does not require excessive feedback. With the system being corrected on just 282 predictions, it is able to achieve 94.04% accuracy on "unknown" attacks.

Approach	Accuracy	False Positive
A-GHSOM	99.63%	1.8%
GHSOM [Palomo et a 2008]	90.87%	2.69%
Bouzida et al 2004	91.89%	0.48%
Sarasamma et al 2005	93.46%	3.99%
Kayacik et al 2007	90.4%	1.38%
Eskin et al 2002 (Data Mining)	90.00%	2.0%
Eskin et al 2002 (Clustering)	93.00%	10.0%
Eskin et al 2002 (SVM)	98.00%	10.0%
Yu et al 2008 (Adaptive)	96.02%	10.0%

Table 5: Summary of performance comparisons

As the summary, Table 5 compares the performance of the integrated A-GHSOM approach with eight major existing approaches based on machine learning techniques. A-GHSOM is consistently able to produce higher accuracy rates while maintaining low false positive rates. Its false positive rate is lower than all approaches except three approaches proposed in [Bouzida et al 2004] and [Kayacik et al 2007]. Note that the false positive rate is just slightly higher (1.32% and 0.42%), but the improvement in the major performance metric accuracy is almost 10% higher. A-GHSOM is able to make the efficient false positive and accuracy tradeoff while being adaptive to a changing problem domain of network intrusion.

4.2 FBS-DD: A Fair Bandwidth Sharing and Delay Differentiation Mechanism

In this section we discuss proposed algorithms for packet forwarding with buffer management Fair Bandwidth Sharing and Delay Differentiation (FBS-DD). The concepts discussed here are for generic QoS algorithms. The results show that they are effective at achieving desired QoS goals. Part of the remaining work in this thesis is to adapt these algorithms for use in scalable automated response.

We consider a lossless and work-conserving packet scheduler that serves N queues, one for each class. The FBS-DD model is to maintain the multi-dimensional QoS spacing of two classes with respect to both their delay ratio (D_i/D_j) and bandwidth sharing ratio (B_i/B_j) be proportional to their pre-specified differentiation parameters δ_i and δ_j.That is,

$$\frac{D_i(T,T+t)}{D_j(T,T+t)} \cdot \frac{B_j(T,T+t)}{B_i(T,T+t)} = \frac{\delta_i}{\delta_j}, 1 \leq i,j \leq N \qquad (3)$$

for time intervals *(T, T+t)* where *t* is the monitoring timescale. Note that the lower average delay or higher bandwidth sharing represents higher QoS. For implementation as part of a scalable automated response capability the learning engine combined with the suspected attack class and confidence level will establish QoS categories for network defense.

VPS-TWP: Throughput normalized waiting time priority scheduling

First, we consider the general case, that is, packets from a class have various sizes and different classes may have different packet size distributions. We revisit the time dependent priority scheduling discipline and propose the VPS-TWP scheme, which focuses on the instantaneous behavior. The time dependent priority scheduling was first studied in queuing foundations. It was later studied in WTP [Dovrolis et al

2002] for PDD provisioning. We describe VPS-TWP as the throughput normalized waiting time priority scheduling algorithm for FBS-DD provisioning.

At the beginning of scheduling, $TWP_i = \infty \; for \; 1 \le i \le N$. Suppose that class *i* is backlogged at time *t*, $s_i(t)$ is the size of the packet at the head of the class *i* at *t*, and that $w_i(t)$ is the head waiting time of class *i* at *t*, i.e., the waiting time of the packet at the head of the class *i* at *t*. We define the throughput normalized head waiting time of class *i* at *t* as

$$TWP_i(t) = \frac{w_i(t)}{\delta_i s_i(t)}$$

Every time a packet is to be transmitted, the VPS-TWP scheduler selects the backlogged class *j* with the maximum throughput normalized head waiting time,

$$j = \arg max_{i \in G(t)} TWP_i(t)$$

where *G(t)* is the set of backlogged classes at time *t* Tie breaks by the use of priority. The throughput of class *j* is increased by the size of the transmitted packet. Its throughput normalized head waiting time will be minimized as its packet delay will not increase any more. VPS-TWP attempts to minimize the differences between the bandwidth normalized waiting times of successively departing packets. It essentially aims to achieve instantaneous FBS-DD.

Next, we consider a special case, that is, all packets have the uniform size. The experienced bandwidth ratio of classes *i* and *j* is given as $B_i(T, T+t)/\, B_j(T, T+t) = b_i/b_j$ where b_i and b_j are the number of packets departed in the interval *(T, T+t)*, essentially the throughput of the classes in the interval. When all packets have the uniform packet size, VPS-TWP is reduced to UPS-TWP. The throughput normalized head waiting time of class *i* at *t* is calculated as

$$TWP_i(t) = \frac{w_i(t)}{b_i \delta_i}$$

VPS-TAD: Throughput normalized average delay scheduling

While VPS-TWP focuses at the instantaneous behavior, we propose the VPS-TAD scheme which focuses on the long-term behavior. (3) can be rewritten as

$$\frac{D_i(T,T+t)}{\delta_i B_i(T,T+t)} = \frac{D_j(T,T+t)}{\delta_j B_j(T,T+t)} \tag{4}$$

The way to interpret it is that the throughput normalized average delay (TAD) must be equal in all classes. That is $TAD_i = TAD_j$. Note that given a same interval, bandwidth sharing ratio of two classes is the same as the throughput ratio. The VPS-TAD scheme, tailored from PAD [Dovrolis et al 2002], aims to equalize the throughput normalized average delays among all classes so as to achieve the FBS-DD goal.

Let *G(t)* is the set of backlogged classes at time *t*, $L_i(t)$ be the sequence of class *i* packets that were transmitted during the interval *(T, T+t)*, d_i^m be the delay of the *m*th packet in $L_i(t)$, and s_i^m be the size of the *m*th packet in $L_i(t)$. Assuming that there was at least one packet transmitted from class *i* during interval *(T, T+t)*, the throughput normalized average delay of class *i* at *t* is

$$TAD_i(t) = \frac{D_i(T, T = t)}{\delta_i B_i(T, T + t)} = \frac{1}{\delta_i \sum_{m=1}^{|L_i(t)|} s_i^m} \frac{\sum_{m=1}^{|L_i(t)|} d_i^m}{|L_i(t)|}$$

where $|L_i(t)|$ is the number of packets in $L_i(t)$.

At the beginning of scheduling, $TAD_i = \infty \; for \; 1, \leq i \leq N$ Suppose that a packet is to be transmitted at time *t*. VPS-TAD selects the backlogged class *j* with the maximum bandwidth normalized average delay,

$$j = \arg max_{i \in G(t)} TAD_i(t)$$

Tie is broken by the priority. The rationale of VPS-TAD is that each time a packet from class *j* is transmitted, its throughput normalized average delay decreases. This is because its throughput increases by the size of the transmitted packet. The delay of that transmitted packet will not increases any more, and thus the increase to the average packet delay will be minimized. VPS-TAD therefore attempts to minimize the differences between the throughput normalized average delay of classes. It essentially aims to achieve FBS-DD in the long term. It, however, needs to maintain the state information about the current throughput and average delay per each class.

When all packets have the uniform size, VPS-TAD is reduced to UPS-TAD. The throughput normalized average delay of class *i* is calculated as

$$TAD_i(t) = \frac{D_i(T, T + t)}{\delta_i B_i(T, T + t)} = \frac{1}{b_i \delta_i} \frac{\sum_{m=1}^{|L_i(t)|} d_i^m}{|L_i(t)|}$$

PID Control-theoretic Buffer Management
When the overall workload is greater than the link capacity, packet loss is inevitable and loss rate becomes the dominant QoS metric. The proposed packet scheduling schemes, however, have no control over the loss rate differentiation between classes. We propose a control-theoretic buffer management scheme, to be integrated with the packet scheduling schemes, for the FBS-DD provisioning and proportional loss rate differentiation at the same time. One nice feature of the buffer management based approach is that the packets will be dropped from the tail due to the buffer overflow. This avoids the packet pushout issue and facilitates the packet ordering.

The buffer management scheme is to dynamically allocate the buffer space into a number of virtual mini-buffers, one mini-buffer for one class. The size of a mini-buffer directly affects a class's loss rate. Feedback control theory has been applied to adjust the resource allocation for service differentiation provisioning [Le et al 2006], [Lu et al 2004]. We propose to use a proportional integral derivative (PID) controller to adjust the buffer allocation. Let l_i be the loss rate of class *i*. The goal is to ensure that the observed relative loss rate l_i be proportional to the pre-specified QoS parameter δ_i, that is, $l_i / l_j = \delta_i / \delta_j$. Let L_i be the relative loss rate ratio of class *i*, that is, . $L_i = \frac{l_i}{l_1 + l_2 + \cdots + l_n}$ Let L_i^d be the desired relative loss rate ratio of class *i*, that is, . $= L_i^D \frac{\delta_i}{\delta_1 + \delta_2 + \cdots + \delta_n}$ During the *k*th sampling period, the relative error is calculated as difference between the desired value and the observed value, that is,

$$e_i(k) = L_i^d(k) - L_i(k)$$

One property of the model is the sum of the relative errors is always zero since

$$\sum_{i=1}^{n} e_i(k) = \sum_{i=1}^{n} \left(L_i^d(k) - L_i(k)\right) = 0$$

. This important property makes it feasible for us to adaptively adjust the buffer allocation for a class independent of the adjustments of other classes while maintaining a constant overall buffer size.

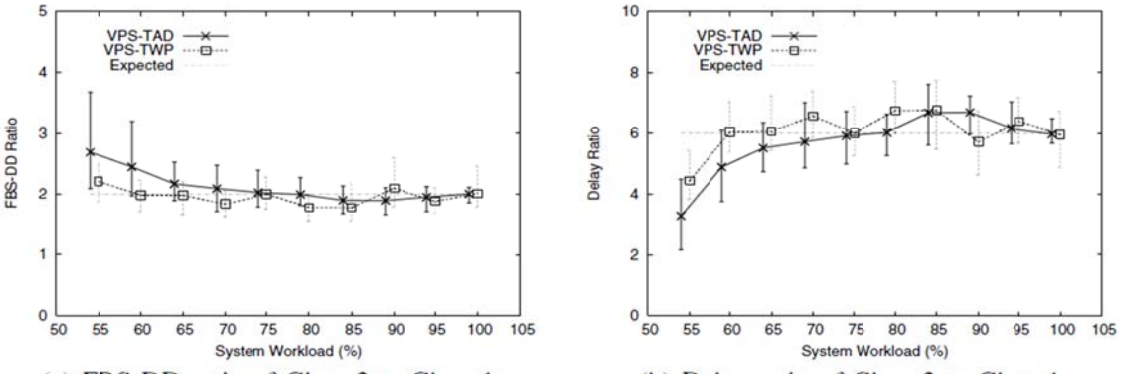
The buffer size allocated to a class is adjusted in proportion to the error between the desired relative loss rate ratio and the observed one. Specifically, the operation of the PID controller is described as follows:

$$s_i(k+1) = s_i(0) + G_p e_i(k) + G_I \sum_{j=0}^{k-1} e_i(j) + G_D \Delta e_i(k)$$

$s_i(k+1)$ denotes the buffer size allocated to class *i* in the new sampling period. $s_i(0)$ denotes the initial buffer size allocated. The three terms added to $s_i(0)$ denote proportional, integral, and derivative components, respectively. Setting a large proportional feedback gain (G_P) typically leads to faster response at the cost of increasing system instability. The integral controller (G_I) can eliminate the steady-state error and avoid over-reactions to measurement noises. The derivative control (G_D) considers the change of errors in adjusting the buffer size allocation and hence responds fast to errors. The derivative error with class *i* is calculated as $\Delta e_i(k) = e_i(k) - e_i(k-\mathbf{1})$.

Initial Performance Evaluation: We developed a simulator to study the performance of the packet scheduling schemes and the feedback control based buffer management. Due to the space limitation, we report the two-class experimental results. For the packet size distribution of two classes, we used two Bell Labs-I trace files adopted from the National Laboratory for Applied Network Research[7]. Without loss of generality, let Class-1 be the high priority class and Class -2 be the low priority class.

Packet Scheduling Algorithms: The first set of experiments is to study the impact of the packet scheduling schemes on FBS-DD provisioning when the overall workload is within the link capacity. We considered a lossless model. Figure 13 shows the performance of the packet scheduling schemes. The differentiation weight ratio of two classes (δ1 : δ2) is set to 1:2 and their workload ratio is set to 1:3. Figure 13(a) shows the achieved FBS-DD ratio with its 95th and 5th percentiles when the overall workload changes from 55% to 100% of the link capacity. Figure 13(b) shows the achieved delay ratio with its 95th and 5th percentiles.

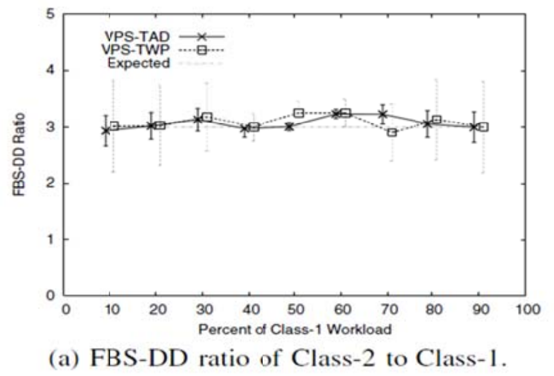


(a) FBS-DD ratio of Class-2 to Class-1. (b) Delay ratio of Class-2 to Class-1.

Figure 13: The Performance of VPS when overall workload changes from 55% to 100%.

[7] NLANR. PMA: Special traces archive. http://pma.nlanr.net/Special/.

The results show that the scheduling schemes can achieve the goal of providing fair bandwidth sharing with delay differentiation when the overall workload is greater than 60%. But the variance, as demonstrated by the 95th and 5th percentiles, is a nontrivial issue. It is due to the variance of the packet size distributions and the inter-arrivals. When the workload is light, there is a feasibility issue with the packet scheduling for service differentiation provisioning [Dovrolis et al 2002].



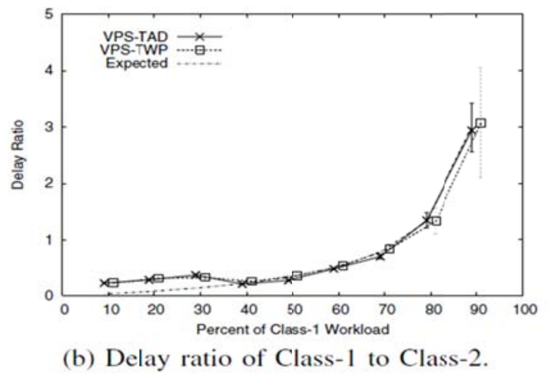
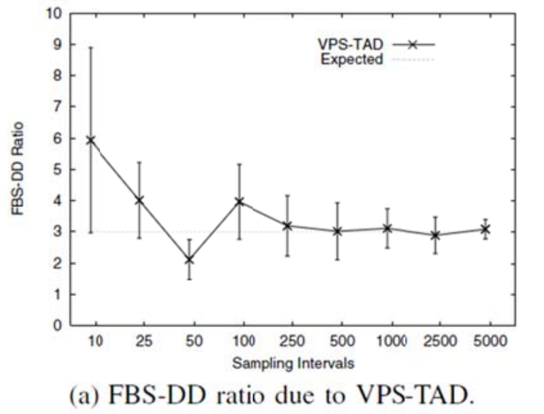


Figure 14: The performance of VPS when Class-1's workload changes from 10% to 90% of the overall workload

We next change the ratio of δ1 : δ2 to 1:3. We fix the overall workload to 80% and vary the Class-1's workload from 10% to 90% of the overall workload. Figure 14(a) shows the achieved FBS-DD ratio with its 95th and 5th percentiles. It shows the FBS-DD ratio can be achieved as expected. But the variance is high when Class-1's workload deviates from the middle value 50%. This is due to the fact that there are too few or too many packets from Class-1, limiting the capability of the packet scheduling schemes. Figure 14(b) shows the achieved delay ratio with its 95th and 5th percentiles. We can see that the proposed VPS scheduling schemes can achieve the fair bandwidth sharing with delay differentiation when the workload percentage of the classes changes dynamically. When the Class-1 contributes 75% of the overall workload, the delay ratio of two classes becomes 1. This demonstrates the benefit of the FBS-DD model that can make adaptive tradeoff between delay differentiation and fair bandwidth sharing. While both VPS-TAD and VPS-TWP schemes can achieve FBS-DD provisioning from the long-term perspective, they have different behaviors.



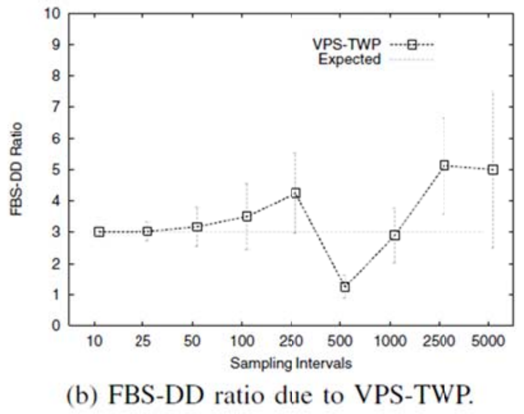


Figure 15: The behaviors of VPS scheduling schemes for FBS-DD provisioning in different sampling intervals

Figure 15 shows the FBS-DD ratios achieved by the scheduling schemes in different sampling intervals. Interestingly, in short sampling intervals, VPS-TAD does not perform well for FBS-DD provisioning. Figure 15(a) shows that its performance improves as the sampling interval increases. This is explained by the fact that VPS-TAD takes into account the average of a number of packets in the interval. It aims to minimize the differences between the normalized average class delays and thus its performance improves as the sampling interval increases. On the other hand, Figure 15(b) shows that VPS-TWP achieves desirable FBS-DD ratios when the sampling interval is short and the performance deteriorates as the interval increases. This is due to the fact that VPS-TWP attempts to minimize the differences between the normalized head waiting times. Essentially, it aims to achieve the instantaneous FBS-DD provisioning.

Performance of PID Control Buffer Management: Previous experimental results have shown that the packet scheduling schemes can achieve the fair bandwidth sharing and delay differentiation at the same time. But when the overall workload is beyond the link capacity, there will be packet loss and the packet scheduling schemes have no control over the loss rate differentiation between classes. Figure 16 depicts the impact of the PID control-theoretic buffer management on loss rate differentiation and the controllability of FBSDD provisioning. The overall workload is set to 150% of the link capacity. The differentiation weight ratio of two classes (δ1 : δ2) is set to 1 : 3. Figure 16(a) shows the impact of the PID control-theoretic buffer management on the proportional loss rate differentiation. It shows that with the buffer management, the loss rate ratio of two classes is fairly proportional to the differentiation weight ratio as the percentage of the Class-1's workload changes from 10% to 90% of the overall workload. On the other hand, without the buffer management, both classes experience almost the same loss rate.

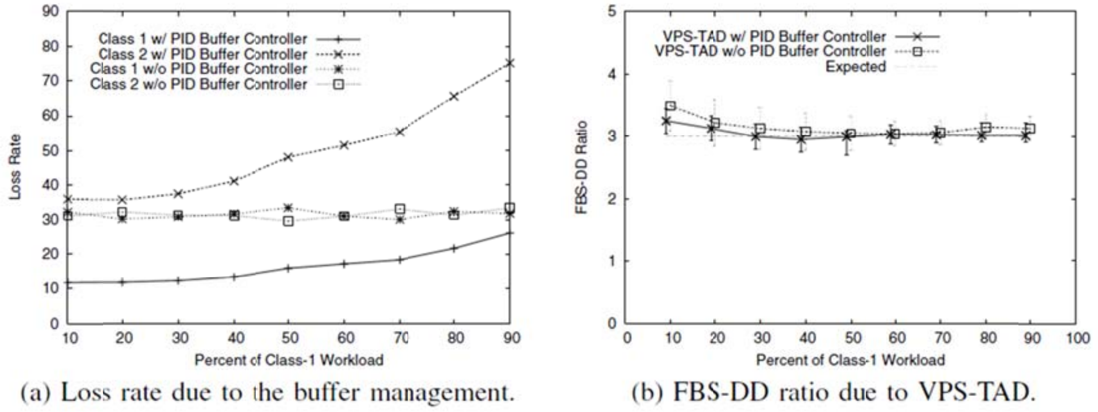


Figure 16: The impact of the control-theoretic buffer management on FBS-DD provisioning

Figures 16(b) shows the achieved FBS-DD ratio by the use of VPS-TAD packet scheduling scheme with and without the PID controller based buffer management. The results show that with the PID controller based buffer management, the VPS-TAD scheme is able to achieve more consistent and desirable FBS-DD ratios with respect to both the mean and the variance. Without the feedback control based buffer management, the variance of the FBS-DD ratio is higher. One reason is that if there is no feedback based buffer management, a class with some bursty traffic can saturate the buffer easily, leaving little buffer space for another class. The VPS-TAD scheduling schemes aims to minimize the normalized average delays. Its capability is limited by the availability of packets from certain classes for scheduling. This benefits the low-priority but high-workload class. Therefore, the buffer management should be integrated with packet scheduling for controllable FBS-DD provisioning. The integrated approach is capable of self-adapting to varying workloads from different classes, which automatically builds a firewall around aggressive clients and hence protects network resources from saturation.

4.3 TD-SIM: an Integrated Dataset of Live Traces and Simulated Data

There have been many critiques or arguments made against the use of the KDD dataset [Brugger 2007], [McHugh 2000]. One major issue is that no validation has been done to ensure the KDD dataset is in fact similar to real network traffic [Brugger 2007]. Additionally, the distribution of attacks in the KDD dataset is said to be unrealistic. Due to the way the traffic in the dataset was generated, some of the differences between attack traffic and normal traffic are statistically obvious.

To avoid drawing research results and conclusions solely based on experiments with the KDD dataset, we have built a dataset (TD-Sim), which consists of a mixture of live trace data and simulated network traffic. TD-Sim was constructed in three steps. We first addressed the issue of similarity between the dataset and live data by using live trace data as the foundation. We started with a publicly available trace dataset and performed preprocessing to correct header/payload discrepancies created during packet anonymization. We then processed the dataset through a commercial intrusion detection system to categorize attack vs. normal traffic. Finally, we inserted simulated traffic to ensure adequate coverage of a variety of network attacks.

The packet trace dataset we used as a base for TD-Sim is from the Lawrence Berkeley National Laboratory (LBNL)[8]. The data was collected from Oct 2004 through Jan 2005. The database includes 11GB of packet header traces in 131 files. Each day trace covers a range of ten minutes to one hour. In this dataset, the payload field is deleted and the IP address field is anonymized. Because the payload field was deleted, calculations in the packet headers were not consistent with the actual packet sizes. We used the commercial tool WireShark[9] to pad the payload fields to match packet header calculations. We chose this method over recalculation in order to preserve the original packet size and data transfer information. As the baseline for identifying malicious activities, we processed the trace dataset through a commercial intrusion detection system (IDS). We selected Snort [Roesch 1999] IDS because it is publicly available and widely used. We used the official "Sourcefire VRT Certified Rules (registered user release)" for version 2.8.5. We processed the dataset with all available rules engaged. The alerts generated by SNORT were separated into the same categories used by the KDD dataset (DOS, PROBE, U2R, and R2L). Additionally, we used a "Miscellaneous Activity" category to capture behaviors that SNORT identified as violating security rules but did not identify a specific attack. Connections in this category are behaviors that are potential warning signs of malicious behaviors. Examples include protocol mismatch issues, destination unreachable alerts, overlapping TCP packet thresholds, etc. After processing the trace data, we discovered that the SNORT alerts consisted of primarily of Misc, DoS, and Probe alerts with no R2L alerts and less than 1% of the alerts suspected U2R.

In order to ensure adequate coverage of attack types, we generated and inserted simulated traffic into TD-Sim. We did it by setting up a testbed network consisting of clients and servers. Clients included a mix of Windows XP, Windows 2000, and UBUNTU operating systems. Servers included a mix of Solaris, Windows 2000 server, and UBUNTU operating systems. Servers also included dedicated database, web, file, and mail servers as well as windows Domain and DNS services. We generated normal traffic by making HTTP requests, file transfers, normal logons, etc. We then generated attack traffic using the MetaSploit[10] and NMAP toolkits[11]. We repeatedly performed network scans and brute force compromise attempts. We also repeatedly performed all exploits available for the target operating systems. Levels of aggression available in the toolkits include "paranoid", "sneaky", "polite", "normal", "aggressive", and "insane". The generated connections were randomly inserted into the trace data. Note that the majority of the records in the dataset are from the trace data. Over 70% of the connections used in the TD-Sim dataset are from the trace data obtained from LBNL.

TD-Sim dataset also addresses the distribution of attack connections. In the KDD "corrected" dataset, 74% of all records are DoS attacks and only 19.48% of the records are normal connections. Further, R2L

[8] LBNL/ICSI Enterprise Tracing Project. http://www.icir.org/enterprisetracing/.
[9] WireShark network protocol analyzer. http://www.wireshark.org/.
[10] The MetaSploit Project. http://www.metasploit.com/.
[11] NMAP Security Scanner. http://nmap.org/.

attacks make up less than 5% and U2R account for less than 1%. It has become common practice using the KDD dataset to test an intrusion detection system to include an overall success rate. However, when run against the KDD dataset, an intrusion detection system that is tuned to identify DoS and Probe can fail to identify all occurrences of U2R or R2L and still maintain a high overall success rate. To address this issue, in the TD-SIM dataset, attack traffic accounts for less than 30% of all connections and the distribution of attacks is more equitable. Table 6 shows the mix of attack types in the TD-Sim dataset.

Probe	49%
DoS	14.12%
U2R	14.91%
R2L	14.93%
Misc	7.04%

Table 6: Composition of Attack Traffic In The TD-SIM Dataset

In this thesis we will continue to develop the TD-SIM dataset to update its currency and to include specific capability to evaluate not only detection success, but also evaluate response effectiveness.

5. Research Plan

As discussed throughout, there are four key features of the proposed research: self-adaptive, self-tuning, self-optimizing and automated response. To that end the following represent the major phases of this research

Design of the System Framework

- Finalize Self-adaptive algorithms
- Develop Self-tuning algorithms
- Formulate Model for Optimal Performance
- Develop Self-Optimizing Algorithms
- Adapt QoS Algorithms for Scalable automated response.
- Finalize integrated framework

Test and evaluation

- Develop TD-SIM Database
- Develop Comprehensive Evaluation Plan
- Conduct Experiments

Publication of results

- Work already published
- Planed publications

5.1 Design of the System framework

Finalize self-adaptive algorithms: The majority of the work already accomplished in automation has been in self-adaptive anomaly detection. There are still adjustments to be made to the final proposed approaches. This step will be accomplished first and will be complete by 1 July 2011.

Develop Self-tuning algorithms: We will propose reinforcement learning methodologies for self-tuning and self-optimizing. Self-tuning will be accomplished against static performance metrics. There are a number of challenges to overcome relating to the size of the state space, size of the action space and the ability of a self-tuning mechanism to be agile in its response. This step is also ongoing and will be complete by 1 Aug 2011.

Formulate Model for Optimal Performance: The model for optimal performance is a combination of a system of performance metrics. Impact of false positives is well known. Impact of false negatives in an evolving attack taxonomy and impact of confidence forwarding is not well known. These areas will be researched and a finalized model for optimal performance will be proposed. This will be completed by 1 Oct 2011.

Develop Self-Optimizing Algorithms: This is an extension of the previous two steps. The main challenge here is to be able to adapt tuning to the extremely dynamic performance requirements in a responsive manner while at the same time utilizing realistic resource consumption. This will be completed by 1 Dec 2011

Adapt QoS Algorithms for Scalable automated response: As stated previously, the proposed algorithms have been shown effective at achieving QoS performance goals. This phase involves developing a model for scalable response based on a combination of attack taxonomy, prediction confidence. This phase will also consist of developing algorithms for learning appropriate automated methods. This will be completed by 1 Feb 2012.

Finalize integrated framework: While some work has been accomplished in each of the individual areas of automation these areas must be combined into an integrated framework. This will be completed by 1 Mar 2012.

5.2 Test and evaluation

Develop TD-SIM Database: Work in this area is ongoing. An initial TD-SIM dataset has been completed. However, it is limited in its usefulness to testing overall performance of an adaptive detection model. It must be extended to include capability to evaluate a response mechanism. This work will be completed by 1 Mar 2012.

Develop Comprehensive Evaluation Plan: The specific objectives of this research and proposed criteria for success are detailed in sections 3.8 and 3.9. We must develop a comprehensive evaluation plan that will effectively evaluate the proposed model effectiveness at achieving all established objectives. This will be completed by 1 Apr 2012.

Conduct Experiments: We will conduct experiments in accordance with the evaluation plan and adjust final model as required. This will be accomplished by 1 Aug 2012.

5.3 Publication of results

Work already published: The following papers related to this work have been published.

Dennis Ippoliti and Xiaobo Zhou, **An Adaptive Growing Hierarchical Self Organizing Map for Network Intrusion Detection,** Proc. of the 19th IEEE International Conference on Computer Communications and Networks (ICCCN), IEEE Communications Society, Switzerland, August 2010.

Dennis Ippoliti, Xiaobo Zhou, and Liqiang Zhang, **Packet Scheduling with Buffer Management for Fair Bandwidth Sharing and Delay Differentiation,** Proc. of the 16th IEEE International Conference on Computer Communications and Networks (ICCCN), IEEE Communications Society, Honululu, August 2007.

Xiaobo Zhou, Dennis Ippoliti, and Liqiang Zhang, **Packet Scheduling with Buffer Management for Fair Bandwidth Sharing and Delay Differentiation,** Computer Communications, Elsevier, Vol. 31, No. 17, pp. 4072 - 4080, Nov 2008.

Xiaobo Zhou, Dennis Ippoliti, and Terrance Boult, **Hop-count based probabilistic packet dropping: congestion mitigation with loss differentiation**, Computer Communications, Elsevier, Vol. 30, No. 18, pp. 3859 - 3869, December 2007.

Planed publications: At present, there is one additional paper submitted for peer review and one publication in draft that is currently in work. As we complete this research we will submit other works for publication as appropriate.

Document Final Results and Defend Dissertation: We will complete all work and document results for final defense in Dec 2012.

6 Summary

In this paper, we present our proposal to advance the state of the art in intrusion detection. We will contribute algorithms and methodologies for self-adaptive, self-tuning, self-optimizing, and automatically responsive network anomaly detection systems. We will demonstrate the effectiveness of our approaches by proposing a prototype integrated detection and response framework based on an Adaptive Growing Hierarchical Self Organized Map. The integrated framework will apply broad identification of network events into categories of normal, attack, and confidence filtering. Attack predictions will automatically be further classified into a dynamic and evolving attack taxonomy. The prototype model will utilize novel Quality of Service algorithms to apply appropriate defenses to the attack taxonomy and scalable response actions to normal and attack predictions with low confidence ratings. The integrated automated response will protect resources from potential attack but will also allow for fair differentiation and bandwidth sharing to confidence filtered events reducing collateral damage to legitimate traffic. We will evaluate the prototypes ability to demonstrate self-adaptive, self-tuning, self-optimizing and automated response capability by conducting extensive experiments on both KDD99 Intrusion Detection Data set and a new data set we will develop combining actual network trace data and simulated network attacks. We expect that our work will continue through December 2012 , at which time we will defend the results.

Acknowledgement
This research was supported in part by U.S. National Science Foundation CAREER Award CNS-0844983 and research grant CNS-0720524.

References

[Anderson et al 1994] D. Anderson, T. Frivold, A. Tamaru, A. Valdes, Next generation intrusion detection expert system (NIDES), Software Users Manual, Beta-Update release, Computer Science Laboratory, SRI International, Menlo Park, CA, USA, Technical Report SRI-CSL-95-0, May 1994.

[Anderson et al 1995] D. Anderson, T.F. Lunt, H. Javitz, A. Tamaru, A. Valdes, Detecting Unusual Program Behavior Using the Statistical Component of the Next-generation Intrusion Detection Expert System (NIDES), Computer Science Laboratory, SRI International, Menlo Park, CA, USA SRI-CSL-95-06, May 1995.

[Apap et al 2002] F. Apap, A. Honig, S. Hershkop, E. Eskin, and S.J. Stolfo. Detecting Malicious Software by Monitoring Anomalous Windows Registry Accesses. In *Proceedings of the Fifth International Symposium on Recent Advances in Intrusion Detection* (RAID-2002). Zurich, Switzerland, October 2002

[Axelsson 2000] S. Axelsson, Intrusion Detection Systems: A Survey and Taxonomy, Chalmers University, Technical Report 99-15, March 2000.

[Axelsson et al 1999] S. Axelsson. The base-rate fallacy and its implications for the difficulty of intrusion detection. In *6th ACM Conference on computer and communications security*, pages 1–7, Kent Ridge Digital Labs, Singapore, 1–4 November 1999.

[Barbara et al 200] D. Barbara, J. Couto, S. Jajodia, and N. Wu. (2001). Adam: a testbed for exploring the use of data mining in intrusion detection. *SIGMOD Rec. 30,* 4, 15

[Bivins et al 2002] Bivens A, Chandrika P, Smith R, Szymanski B (2002) Network-based intrusion detection using neural networks. In: *Proceeding of ANNIE 2002 conference*, ASME Press, pp 10–13

[Bolzoni et al 2009] D. Bolzoni, S. Etalle, and P. Hartel. (2009). Panacea: Automating Attack Classification for Anomaly-Based Network Intrusion Detection Systems. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection* (RAID '09)

[Bouzida et al 2004] Y. Bouzida, F. Cuppens, N. Cuppens-Boulahia, & S. Gombault. (2004). Efficient intrusion detection using principal component analysis. *In Paper presented at the proceedings of the 3eme conference surla securite et architectures reseaux (SAR).* Orlando, FL, USA.

[Barford and Plonka 2001] P. Barford and D. Plonka. Characteristics of network traffic flow anomalies. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, San Francisco, USA, 2001.

[Bridges et al 2000] S. M. Bridges, & R.B. Vaughn. (2000). Intrusion detection via fuzzy data mining. In *Paper presented at the twelfth annual Canadian information technology security symposium*. Ottawa, USA.

[Brugger 2007] S.-T. Brugger, J. Chow. An assessment of the DARPA IDS Evaluation Dataset using Snort. *Technical Report CSE-2007-1*, University of California, Davis, Department of Computer Science, Davis, CA, 2007.

[Callegari et al 2008] C. Callegari, S. Vaton, and M. Pagano, A new statistical approach to network anomaly detection, in *Proc. of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, 2008.

[Cannady et al 2000] J. Cannnady. Next generation intrusion detection: autonomous reinforcement learning of network attacks. *In 23th. National Information Systems Security Conference,* 2000.

[Cha and Lee 2010] B. R. Cha and D.S. Lee, 2007. Network-based anomaly intrusion detection improvement by Bayesian network and indirect relation. *Lecture Notes Comput. Sci*., 4693: 141-148. DOI: 10.1007/978-3-540-74827-4_18

[Chandola et al 2009] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. ACM Computing Survey, 41(3), 2009.

[Chen and Tang 2007] S. Chen, & Y. Tang. (2007). DAW: A distributed antiworm system. *IEEE Transactions on Parallel and Distributed Systems, 18*(7), 893–906.

[Chen et al 2005] Chen, W.-H., Hsu, S.-H., & Shen, H.-P. (2005). Application of SVM and ANN for intrusion detection. *Computer and Operations Research*, 32, 2617–2634.

[Chimphlee et al 2006] W. Chimphlee, A. H. Addullah, M. N. M. Sap, S. Srinoy, & S. Chimphlee (2006). Anomaly-based intrusion detection using fuzzy rough clustering. *In Paper presented at the international conference on hybrid information technology* (ICHIT'06).

[Ciocarlie et al 2009] G. F. Cretu-Ciocarlie, A. Stavrou, M.E. Locasto, S. Stolfo, Adaptive anomaly detection via self-calibration and dynamic updating. *In: RAID '09: Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, Berlin, Heidelberg, Springer-Verlag (2009) 41-60

[Cohen 1995] W.W. Cohen, Fast effective rule induction, *in: Proceedings of the 12th International Conference on Machine Learning*, Tahoe City, CA, 1995, pp. 115–123.

[Debar et al 1992] H. Debar, M. Becker, and D. Siboni. A neural network component for an intrusion detection system. In *Proceedings of the 1992 IEEE Computer Sociecty Symposium on Research in Security and Privacy*, pages 240.250, Oakland, CA, USA, May 1992. IEEE, IEEE Computer Society Press, Los Alamitos, CA, USA.

[Debar et al 1999] H. Debar, M. Dacier, and A. Wespi. Towards a taxonomy of intrusion detection systems. *Computer Networks*, 31(8):805.822, April 1999.

[Denning 1987] D.E. Denning, An intrusion-detection model, *IEEE Transactions in Software Engineering* 13 (1987) 222–232.

[Depren et al 2005] O. Depren, M. Topallar, E. Anarim, & M. K. Ciliz (2005). An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. *Expert Systems with Applications*, 29, 713–722.

[Dewaele et al 2007] G. Dewaele, K. Fukuda, P. Borgnat, P. Abry, and K. Cho, Extracting hidden anomalies using sketch and non gaussian multiresolution statistical detection procedure, in *SIGCOMM LSAD'07*, 2007, pp. 145–152.

[Dickerson et al 2000] J.E. Dickerson, J.A. Dickerson, Fuzzy network profiling for intrusion detection, *in: Proceedings of the 19th International Conference of the North American Fuzzy Information Processing* Society (NAFIPS), Atlanta, GA, 2000, pp. 301–306.

[Dovrolis et al 2002] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. *IEEE/ACM Trans. on Networking*, 10(1):12–26, 2002.

[Eskin et al 2002] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: detecting intrusions in unlabeled data. *Applications of Data Mining in Cumputer Security*, Chapter 4, 2002.

[Fan et al 2004] W. Fan, W. Lee, M. Miller, S. Stolfo, & P. K. Chan, (2004). Using artificial anomalies to detect unknown and known network intrusions. *Knowledge and Information Systems*, 507–527.

[Farid et al 2010] D. M. Farid, N. Harbi, and M. Z. Rahman, Combining naïve bayes and decision tree for adaptive intrusion detection, *International Journal of Network Security & Its Applications* (2010) 12-25

[Forrest et el 1996] S. Forrest, S.A. Hofmeyr, A. Somayaji, T.A. Longstaff, A sense of self for unix processes, *in: Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, USA, 1996, pp. 120–128.

[Fox et al 1990] K. Fox, R. Henning, J. Reed, R. Sitnonian, *A Neural Network Approach Towards Intrusion Detection,* Harris Corporation, Government Information Systems Division, P.O. Box 98000, Melbourne, FL 32902, July 1990.

[Gammerman and Vovk 2002] A. Gammerman, V. Vovk. Prediction algorithms and confidence measure based on algorithmic randomness theory. *Theoretical Computer Science* 2002;287(1):209–17.

[Garg and Reddy 2002] A. Garg and A. L. Reddy, Mitigation of DoS attacks through QoS regulation, *In Proc. of the IEEE Intl. Workshop on Quality of Service*, Miami: IEEE Press, 2002, pp. 45-53.

[Gates and Taylor 2006] C. Gates and C. Taylor, "Challenging the Anomaly Detection Paradigm: A Provocative Discussion," in *Proc. Workshop on New Security Paradigms*, 2007.

[Gomez and Dasgupta 2002] Gomez J,DasguptaD(2001) Evolving fuzzy classifiers for intrusion detection. *IEEEworkshop on information assurance*, United States Military Academy, NY

[Gosh et al 1998] A.K. Ghosh, J. Wanken, F. Charron (1998) Detecting anomalous and unknown intrusions against programs. *In: Proceedings of the 14th annual computer security applications conference*, IEEE, pp 259–267

[Gosh et al 1999] A.K. Ghosh, A. Schwartzbard, A study in using neural networks for anomaly and misuse detection, *in: Proceedings of the Eighth USENIX Security Symposium*, Washington, DC, 1999, pp. 141–151.

[Gosh et al 2000] A.K. Ghosh, C. Michael, M. Schatz, A real-time intrusion detection system based on learning program behavior, *in: Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection* Toulouse, France, 2000, pp. 93–109.

[Handley et al 200] C. M.Handley and V.Paxon. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. *In Proceedings of the 10th USENIX Security Symposium*, Washington, DC, August 2001.

[Heller et al 2003] K.A. Heller, K.M. Svore, A. D. Keromytis, & S. Stolfo, (2003). One class support vector machines for detecting anomalous window registry accesses. *In Paper presented at the 3rd IEEE conference data mining workshop on data mining for computer security*. Florida.

[Hettich and Bay 1999] S. Hettich, S. Bay. The UCI KDD archive. University of California, Irvine. http://kdd.ics.uci.edu. 1999.

[Himura et al 2009] Y. Himura, K. Fukuda, K. Cho, H. Esaki. An automatic and dynamic parameter tuning of a statistics-based anomaly detection algorithm. In *IEEE ICC 2009*, June 2009; 6.

[Hosmer 1993] H.H. Hosmer, Security is fuzzy!: applying the fuzzy logic paradigm to the multipolicy paradigm, *Proceedings of the 1992-1993 workshop on New security paradigms*, ACM New York, NY, USA, 1993, pp. 175–184.

[Idris and Shanmugam 2005] N.B. Idris, B. Shanmugam (2005) Artificial intelligence techniques applied to intrusion detection. *In: IEEE Indicon 2005 conference*, Chennai, India, pp 52–55

[Internet Security Systems (ISS) 2010] Internet Security Systems (ISS) (2010) Real Secure http://www.iss.net. Accessed 4 August 2010

[Javitz and Valdes 1991] H. S. Javitz, and A. Valdes. 1991. The sri ides statistical anomaly detector. In *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy*. IEEE Computer Society.

[Jiang et al 2009] D. Jiang, Y. Yang, M. Xia. Research on intrusion detection based on an improved SOM neural network. In *Proc. of the Fifth Int'l Conference on Information Assurance and Security*, 2009.

[Kayacik et al 2003] G. Kayacik, N. Zincir-Heywood, M. Heywood (2003) On the capability of an SOM based intrusion detection system. *In: Proceedings of the 2003 IEEE IJCNN*, Portland, USA

[Kayacik et al 2007] G. Kayacik, N. Zincir-Heywood, & M. Heywood. (2007). A hierarchical SOM-based intrusion detection system. *Engineering Applications of Artificial Intelligence*, 20, 439–451.

[Kohonen 1982] T. Kohonen (1982). Self-organized formation of topologically correct feature maps. Biological Cybernetics, 43, 59–69.

[Kruegel et al 2002] C. Kruegel, T. Toth, E. Kirda, Service specific anomaly detection for network intrusion detection, *in: Proceedings of the 2002 ACM symposium on Applied computing* Madrid, Spain 2002, pp. 201–208.

[Kumar and Stafford 1994] S. Kumar, E.H. Spafford, An application of pattern matching in intrusion detection, The COAST Project, Department of Computer Sciences, Purdue University, West Lafayette, IN, USA, *Technical Report CSD-TR-94- 013*, June 17, 1994.

[Kumar et al 2010] G. Kumar, K. Kumar, and M. Sachdeva. 2010. The use of artificial intelligence based techniques for intrusion detection: a review. *Artif. Intell. Rev.* 34, 4 (December 2010), 369-387.

[Lakhina et al 2005] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. *In SIGCOMM '05*: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications, pages 217–228, New York, NY, USA, 2005.

[Le et al 2006] L. Le, K. Jeffay, F.D. Smith. A loss and queueing-delay controller for router buffer management. In Proc. IEEE Int'l Conf. on Distributed Computing Systems (ICDCS), 2006.

[Lee and Stolfo 2000] W. Lee, & S. Stolfo. (2000). A framework for constructing features and models for intrusion detection systems. ACM Transactions on Information and System Security (TISSEC), 3(4), 227–261.

[Lee et al 1997] Lee, W., Stolfo, S., and Chan, P. 1997. Learning patterns from unix process execution traces for intrusion detection. In *Proceedings of the AAAI 97 workshop on AI methods in Fraud and risk management*.

[Lee et al 1999] W. Lee, S.J. Stolfo, K.W. Mok, A data mining framework for building intrusion detection models, in: Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, 1999, pp. 120–132.

[Li 2004] Li W (2004) Using genetic algorithm for network intrusion detection. C. S. G. Department of Energy, Ed, pp 1–8

[Li et al 2007] Y. Li, B. Fang, L. Guo and Y. Chen. Network anomaly detection based on TCM-KNN algorithm. In *Proc. of ACM Symposium on Information, Computer and Communications Security*, 2007.

[Liao and Vermuri 2002] Liao, Y., & Vemuri, V. R. (2002). Use of K-nearest neighbor classifier for intrusion detection. Computer and Security, 21(5), 439–448.

[Lu et al 2004] Y. Lu, T. F. Abdelzaher, and A. Saxena. Design, implementation, and evaluation of differentiated caching services. IEEE Trans. on Parallel and Distributed Systems, 15(5):440–452, 2004.

[Mahajan et al 2002] R. Mahajan, S. Bellovin, S. Floyd, V. Paxson, and S. Shenker. "Controlling high bandwidth aggregates in the network." *ACM Computer Communications Review*, 32(3), July 2002.

[Mahoney and Chan 2003a] M. V. Mahoney and P. K. Chan. An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection. *In Proceedings of the Sixth International Symposium on Recent Advances in Intrusion Detection*, 220 – 237, PA, USA, September 2003.

[Mahoney and Chan 2003b] M. V. Mahoney and P. K. Chan. Learning rules for anomaly detection of hostile network traffic. *In Proceedings of Third IEEE International Conference on Data Mining*, pages 601 – 604, 2003.

[Mahoney et al 2001] M.V. Mahoney, P.K. Chan, PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic Department of Computer Sciences, Florida Institute of Technology, Melbourne, FL, USA, *Technical Report CS- 2001-4*, April 2001.

[McHugh 2000] J. McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. In *ACM Transactions on Information and System Security*, 3(4): 262 - 294, 2000.

[Menahem et al 2009] Menahem E, Shabtai A, Rokach L, Elovici Y (2009) Improving malware detection by applying multi-inducer ensemble. Comput Stat Data Anal 53(4):1483–1494

NLANR. PMA: Special traces archive. http://pma.nlanr.net/Special/.

[Nguyen et al 2002] B. Viet Nguyen, An Application of support vector machines to anomaly detection, Research in Computer Science-Support Vector Machine, Report, Fall 2002.

[Ning et al 2002] Ning, P., Cui, Y., Reeves, D.: Constructing attack scenarios trough correlation of intrusion alerts. In: CCS '02: Proc. 9th ACM Conference on Computer and Communication Security, ACM Press (2002) 245{254

[Palomo et al 2008] E. Palomo, E. Dominguez, R. Luque, J. Munoz. A new GHSOM model applied to network security. In *Proc. of the 18th Int'l conference on Artificial Neural Networks*, 2008.

[Patcha and Park 2007] Patcha, A. and Park, J.-M. 2007. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Comput. Networks 51,* 12, 3448.

[Peddabachigari et al 2007] Peddabachigari, S., Abraham, A., Grosan, C., & Thomas, J. (2007). Modeling intrusion detection system using hybrid intelligent systems. Journal of Network and Computer Applications, 30, 114–132.

[Pillai et al 2004] M.M. Pillai, J.H.P. Eloff, H.S. Venter, An approach to implement a network intrusion detection system using genetic algorithms, *in: Proceedings of the 2004 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries*, Stellenbosch, Western Cape, South Africa, 2004, pp. 221–228.

[Porras and Neumann 1997] P.A. Porras, P.G. Neumann, EMERALD: event monitoring enabling responses to anomalous live disturbances, *in: Proceedings of the 20th NIST-NCSC National Information Systems Security Conference*, Baltimore, MD, USA, 1997, pp. 353–365.

[Ramadas et al 2003] M. Ramadas, S.O.B. Tjaden, Detecting anomalous network traffic with self-organizing maps*, in: Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection*, Pittsburgh, PA, USA, 2003, pp. 36–54.

[Rauber et al 2002] Rauber, A., Merkl, D. and Dittenbach, M., 2002. The Growing Hierarchical Self-Organizing Map:exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks*, 13(6): 1331-1341.

[Rehak et al 2009] Rehak, M., Staab, E., Fusenig, V., Pěchouček, M., Grill, M., Stiborek, J., Bartoš, K., Engel, T.: Runtime Monitoring and Dynamic Reconfiguration for Intrusion Detection Systems. *In: Proceedings of RAID '09*. Volume 5758 of LNCS., Springer-Verlag (2009) 61–80

[Rhodes et al 2000] B. Rhodes, J. Mahaffey, and J. Cannady. Multiple self-organizing maps for intrusion detection. In *Proc. of the 23rd national information systems security conference*, 2000.

[Rieck et al 2008] K. Rieck, S. Wahl, P. Laskov, P. Domschitz, and K.-R. M¨uller. A Self-learning System for Detection of Anomalous SIPMessages. In *Proceedings of the* 2nd *Internation Conference on Principles, Systems and Applications of IP Telecommunications. Services and Security forNext Generation Networks: Second International Conference, (IPTComm)*, pages 90–106,July 2008.

[Ringberg et al 2007] Haakon Ringberg, Augustin Soule, Jennifer Rexford, and Christophe Diot. 2007. Sensitivity of PCA for traffic anomaly detection. *In Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems* (SIGMETRICS '07). ACM, New York, NY, USA, 109-120.

[Ringberg et al 2008] Haakon Ringberg, Matthew Roughan, and Jennifer Rexford. 2008. The need for simulation in evaluating anomaly detectors. *SIGCOMM Comput. Commun. Rev.* 38, 1 (January 2008), 55-59.

[Robertson et al 2006] Robertson, W., Vigna, G., Kruegel, C., Kemmerer, R.: Using generalization and characterization techniques in the anomaly-based detection of web attacks. In:NDSS '06: Proc. 13th ISOC Symposium on Network and Distributed Systems Security. (2006)

[Roesch 1999] M. Roesch. Snort – lightweight intrusion detection for networks. In *13th USENIX Systems Administration Conference (LISA '99)*, Seattle, WA, Nov. 1999.

[Ryan et al 1998] Ryan J, Lin M-J, Risto M (1997) Intrusion detection with neural networks. Adv Neural Inf Process Syst MIT 943–949

[Sarasamma et al 2005] S.T. Sarasamma, Q.A. Zhu, J. Huff, Hierarchical Kohonenen net for anomaly detection in network security, IEEE Transactions on Systems, Man and Cybernetics—PART B: Cybernetics 35 (2005) 302–312.

[Sargor 1998] Sargor, C. 1998. Statistical anomaly detection for link-state routing protocols. In *Proceedings of the Sixth International Conference on Network Protocols*. IEEE Computer Society,Washington, DC, USA, 62.

[Scott 2004] Scott, S.L. A Bayesian paradigm for designing intrusion detection systems, Computational Statistics Data Analysis, Volume 45, Issue 1, pp. 69-83, 2004.

[Sellke et al 2005] S. Sellke, N. B. Shroff, and S. Bagchi, Modeling and Automated Containment of Worms, in *DSN '05*, 2005, pp. 528-537.

[Shon et al 2006] Shon, T., Kovah, X., & Moon, J. (2006). Applying genetic algorithm for classifying anomalous TCP/IP packets. Neurocomputing, 69, 2429–2433.

[Shon et al 2007] Shon, T., & Moon, J. (2007). A hybrid machine learning approach to network anomaly detection. Information Sciences, 177, 3799–3821.

[Sinclair et al 1999] Chris Sinclair, Lyn Pierce, and Sara Matzner. 1999. An Application of Machine Learning to Network Intrusion Detection. In *Proceedings of the 15th Annual Computer Security Applications Conference* (ACSAC '99). IEEE Computer Society, Washington, DC, USA, 371-.

[Smaha 1988] S.E. Smaha, Haystack: An intrusion detection system, *in: Proceedings of the IEEE Fourth Aerospace Computer Security Applications Conference*, Orlando, FL, 1988, pp. 37–44.

[Spafford and Zamboni 2000] Spafford EH, Zamboni D (2000) Intrusion detection using autonomous agents. Comput Netw 34(4):547–570

[Sommer and Paxson 2010] Robin Sommer and Vern Paxson. 2010. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy* (SP '10). IEEE Computer Society, Washington, DC, USA, 305-316.

[Staniford et al 2002] S. Staniford, J.A. Hoagland, J.M. McAlerney, Practical automated detection of stealthy portscans, Journal of Computer Security 10 (2002) 105–136.

[Sutton and Barto 1998] Richard S. Sutton and Andrew G. Barto. 1998. Introduction to Reinforcement Learning (1st ed.). MIT Press, Cambridge, MA, USA.

[Tan et al 2002] K. Tan, K. Killourhy, and R. Maxion. Undermining an anomaly-based intrusion detection system using common exploits. *In International Symposium on Recent Advances in Intrusion Detection (RAID) 2002*, pages 54 – 73, Zurich, Switzerland, 2002.

[Tavallaee et al 2010] Mahbod Tavallaee, Natalia Stakhanova, and Ali Akbar Ghorbani. Toward credible evaluation of anomaly-based intrusion-detection methods. *Trans. Sys. Man Cyber Part C* 40, 5 (2010), 516-524.

[Tjaden et al 2000] B. Tjaden, L. Welch, S. Ostermann, D. Chelberg, R. Balupari, M. Bykova, M. Delaney, A. Mitchell, S. Li, D. Lissitsyn, and L. Tong, "INBOUNDS: The Integrated NetworkBased Ohio University Network Detective Service", *4th World Multiconference on Systemics, Cybernetics, and Informatics* (SCI'2000), Jul. 2000.

[Tsai et al 2007] Tsai C-F, Hsu Y-F, Lin C-Y, Lin W-Y (2009) Intrusion detection by machine learning: a review. Expert Syst Appl 36(10):11994–12000

[Valeur et al 2003] Valeur, F., Vigna, G., Kruegel, C., Kremmerer, R.: A comprehensive approach to intrusion detection alert correlation. IEEE Trans. Dependable Secur. Comput. 1(3) (2004) 146{169

[Vapnik 1998] V. Vapnik, "Statistical Learning Theory", ISBN 0-4761-03003-1, John Wiley & Sons, 1998

[Valdes et al 2000] A. Valdes, K. Skinner, Adaptive model-based monitoring for cyber attack detection, in: Recent Advances in Intrusion Detection Toulouse, France, 2000, pp. 80–92.

[Wang et al 2006] Wang, Y., Kim, I., Mbateng, G., & Ho, S.-Y. (2006). A latent class modeling approach to detect network intrusion. Computer Communications, 30, 93–100.

[Wang et al 2010] Wang F, Qian Y, Dai Y, Wang Z (2010) A model based on hybrid support vector machine and self-organizing map for anomaly detection. *In: International conference on communications and mobile computing*, cmc 2010, vol 1. Shenzhen, China, pp 97–101

[Song et al 2010] Lipeng Song and Zhen Jin. 2010. An automated worm containment scheme. In *Proceedings of the 2010 international conference on Web information systems and mining* (WISM'10), Fu Lee Wang, Zhiguo Gong, Xiangfeng Luo, and Jingsheng Lei (Eds.). Springer-Verlag, Berlin, Heidelberg, 187-193.

[Warrender et al 1999] C. Warrender, S. Forrest, B. Pearlmutter, Detecting intrusions using system calls: alternative data models, *in: Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 1999, pp. 133–145.

[Williamson 2002] M. M.Williamson, “Throttling viruses: Restricting propagation to defeat mobile malicious code,” presented at the 18th Annu. Computer Security Applications Conf., Las Vegas, NV, Dec. 2002.

[Yau et al 2005] D. Yau, J. Lui, and F. Liang, 2005. Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. In *Proceedings of the IEEE Transactions on Networking* , Vol 13-1 29-42.

[Yeung and Chow 2002] D.-Y. Yeung and C. Chow. Parzen-window network intrusion detectors. In *Proc. of the 16th Int'l Conf. on Pattern Recognition*, volume 4, pages 385{388, aug 2002.

[Yu et al 2008] Z. Yu, J. Tsai, T. Weigert. An adaptive automatically tuning intrusion detection system. In *ACM Transactions on Autonomous and Adaptive Systems*, 3(3), 2008.

[Zang and Shen 2005] Zhang, Z., & Shen, H. (2005). Application of online-training SVMs for real-time intrusion detection with different considerations. Computer Communications, 28, 1428–1442.

[Zhang et al 2001] Zhang, Z., Li, J., Manikopoulos, C., Jorgenson, J., and Ucles, J. 2001. Hide: a hierarchical network intrusion detection system using statistical preprocessing and neural network classi¯-cation. In *Proceedings of IEEE Workshop on Information Assurance and Security*. West Point,