Data Mining Methods for Network Intrusion Detection

S TERRY BRUGGER
University of California, Davis

June 9, 2004

Abstract

Network intrusion detection systems have become a standard component in security infrastructures. Unfortunately, current systems are poor at detecting novel attacks without an unacceptable level of false alarms. We propose that the solution to this problem is the application of an ensemble of data mining techniques which can be applied to network connection data in an offline environment, augmenting existing real-time sensors. In this paper, we expand on our motivation, particularly with regard to running in an offline environment, and our interest in multisensor and multimethod correlation. We then review existing systems, from commercial systems, to research based intrusion detection systems. Next we survey the state of the art in the area. Standard datasets and feature extraction turned out to be more important than we had initially anticipated, so each can be found under its own heading. Next, we review the actual data mining methods that have been proposed or implemented. We conclude by summarizing the open problems in this area and proposing a new research project to answer some of these open problems.

Chapter 1

Overview and previous work

1.1 Introduction

Network Intrusion Detection Systems (NIDS) have become a standard component in security infrastructures as they allow network administrators to detect policy violations. These policy violations range the gamut from external attackers trying to gain unauthorized access (which can usually be protected against through the rest of the security infrastructure) to insiders abusing their access (which often times is not easy to protect against). Detecting such violations is a necessary step in taking corrective action, such as blocking the offender (by blocking their machine at the parameter, or freezing their account), by reporting them (to their ISP or supervisor), or taking legal action against them. Alternatively, detecting policy violations allows administrators to identify areas where their defenses need improvement, such as by identifying a previously unknown vulnerability, a system that wasn't properly patched, or a user that needs further education against social engineering attacks.

The problem is that current NIDS are tuned specifically to detect known service level network attacks. Attempts to expand beyond this limited realm typically results in an unacceptable level of false positives. At the same time, enough data exists or could be collected to allow network administrators to detect these policy violations. Unfortunately, the data is so volumous, and the analysis process so time consuming, that the administrators don't have the resources to go through it all and find the relevant knowledge, save for the most exceptional situations, such as after the organization has taken a large loss and the analysis is done as part of a legal investigation. In other words, network administrators don't have the resources to proactively analyze the data for policy violations, especially in the presence of a high number of false positives that cause them to waste their limited resources.

Given the nature of this problem, the natural solution is data mining in

an offline environment. Such an approach would add additional depth to the administrators defenses, and allows them to more accurately determine what the threats against their network are though the use of multiple methods on data from multiple sources. Hence, activity that it is not efficient to detect in near real-time in an online NID, either due to the amount of state that would need to be retained, or the amount of computational resources that would need to be expended in a limited time window, can be more easily identified. Some examples of what such a system could detect, that online NIDS can not detect effectively, include certain types of malicious activity, such as low and slow scans, a slowly propagating worm, unusual activity of a user based on some new pattern of activity (rather than a single connection or small number of connections, which are bound to produce a number of false positives), or even new forms of attacks that online sensors are not tuned for. Additionally, such a system could more easily allow for the introduction of new metrics, that can use the historical data as a baseline for comparison with current activity. It also serves to aid network administrators, security officers, and analysts in the performance of their duties by allowing them to ask questions that would not have occurred to them a priori. Ideally, such a system should be able to derive a threat level for the network activity that it analyzes, and predict future attacks based on past activity.

In this paper, we concentrate on the mining of network connection data as a first step. Network connection data is easy to collect from most firewalls and online network intrusion sensors, or it can be constructed based on packet logs. It presents less legal hassle than other forms of data that could be collected in many environments since it does not identify users (only machines), it does not contain details of what was done (just what service was contacted, and perhaps the duration and number of bytes transfered), and it is easily anonymisable. While we concentrate on mining connection information, the methods presented here should be applicable to other data sources, for instance the alert logs from online NIDS, which would not be replaced, but augmented by this approach. Certainly, we expect that the incorporation of numerous forms of data will serve to increase the accuracy of such a system.

We begin by looking at the general motivation for doing data mining in an offline environment, with an emphasis on the advantages of an offline system versus online systems, using an ensemble of classifiers, and multisensor correlation. We'll then look at existing systems, from IDSs to services that incorporate some aspects of the methods discussed. Next, we'll focus on current research in this area, particularly datasets, feature selection, and methods, with a brief look at visualization and the potential for predictive analysis. The second part of this work presents the open problems in this area and presents a proposal for a project to answer some of those questions.

1.2 Goal

In order to figure out how data mining can be applied to find relevant computer security information, we must first define what data mining is. Generally, data mining is the process of extracting useful and previously unnoticed models or patterns from large data stores (Bass 2000; Lee and Stolfo 1998; Lee and Stolfo 2000; Lee et al. 1999a; Mannila 2002; Fayyad et al. 1996). Data mining is a component of the Knowledge Discovery in Databases (KDD) process (Carbone 1997; Fayyad et al. 1996). While this paper will address some other components of that process, such as feature selection, we are primarily concerned with the data mining techniques themselves.

Data mining techniques can be differentiated by their different model functions and representation, preference criterion, and algorithms (Fayyad et al. 1996). The main function of the model that we are interested in is classification, as normal, or malicious, or as a particular type of attack (Ghosh et al. 1999; Kumar 1995). We are also interested in link and sequence analysis (Lee and Stolfo 1998; Lee and Stolfo 2000; Lee et al. 1999a). Additionally, data mining systems provide the means to easily perform data summarization and visualization, aiding the security analyst in identifying areas of concern (Bloedorn et al. 2001). The models must be represented in some form. Common representations for data mining techniques include rules, decision trees, linear and non-linear functions (including neural nets), instance-based examples, and probability models (Fayyad et al. 1996). We see all of these representations used when mining for security knowledge. While some of the work we survey uses various preference criterion (such as processing cost), we are primarily concerned with accuracy. Like the model representation, mining for security knowledge employs a number of search algorithms, such as statistical analysis, deviation analysis, rule induction, neural abduction, making associations, correlations, and clustering (Bass 2000; Lee 1999; Lee and Stolfo 2000; Lee et al. 1999a; Thuraisingham and Ceruti 2000).

Given all these techniques, we hope to to find "hidden patterns based on previously undetected intrusions to help develop new detection templates," (Bass 2000). This allows us to transcend the limitations of many current IDS which rely on a static set of intrusion signatures (misuse detection systems) and "evolve from memorization to generalization," (Ghosh et al. 1999). Such a system would be for the most part a type of anomaly detection system. "Anomaly detection attempts to quantify the usual or acceptable behavior and flags other irregular behavior as potentially intrusive," (Kumar 1995). The first example of such a system was IDES, as described in Denning's seminal paper (1987), which focused on mining statistical measures to use for comparison when searching for anomalies.

Unfortunately, as Lee et al. (2001) note, anomaly detection approaches typically "have higher false positive rates... making them unusable in real environments". The high false alarm rate is easy to understand, as Marchette (1999) and Porras and Valdes (1998) observe, given the high data rates on current networks, even events that happen with a low probability occur at a non-trivial

rate. Axelsson formalizes this notion as the base-rate fallacy in (2000a). We need to employ more advanced techniques such that additional attacks may be detected while decreasing the false alarm rate. This information can be integrated with misuse detection techniques in order to develop a comprehensive threat assessment. Bass (2000) notes that the output of an ideal system "would be estimates of the identity (and possibly the location) of an intruder, the intruder's activity, the observed threats, the attack rates, and an assessment of the severity of the cyberattack." Such a system would allow the human analyst to more easily assess the situation and respond accordingly.

Such a system may advance well beyond current ID systems. Bass (2000) observes, "At the highest level the inference could be an analysis of the threat and the vulnerability." He goes on to propose that ID systems will advance to the point that they can "identify and track multiple hostile information flows for targets, attack rate, and severity in cyberspace." It will accomplish such a task though the use of self-training, applying "knowledge acquired in one learning task to another learning task in order to speed learning and possibly increase final accuracy," (Lane 2000). In the end, it will likely bear a good resemblance to a biological system. Some research, pioneered by Forrest's group at UNM, has likened an IDS to the human immune system (Forrest et al. 1996; Forrest et al. 1997; Tan and Maxion 2002; Hofmeyr and Forrest 1999; Warrender et al. 1999; Dasgupta and González 2002), suggesting that IDSs may eventually resemble complex, biological systems.

1.2.1 Offline processing

The act of detecting intrusions is, intuitively, a real-time task by necessity. As Bishop (2001) once noted, "Timely analysis is the difference between intrusion detection and post-mortem analysis." There are, however, distinct advantages in performing intrusion detection in an offline environment, in addition to the real-time detection tasks typically employed. For example, "in off-line analysis, it is implicitly assumed that all connections have already finished, therefore, we have the luxury to compute all the features and check the detection rules one by one," (Lee and Stolfo 2000). A larger factor is that "data-mining algorithms that generate profiles from data-sets are usually of $\mathcal{O}(n^3)$ or higher" (Singh and Kandula 2001). Similarly, Bass (2000) notes, "the estimation and detection process is highly mathematical and processor intensive," hence, the problem should be more tractable in an offline environment where the pseudo real-time constraints don't exist. Bloedorn et al. (2001) sum it up best when explaining that data mining is, "better suited to batch processing of a number of collected records," and that a daily processing regime provides a good trade-off between timeliness and processing efficiency.

While offline processing would seem to be solely a compromise between efficiency and timeliness, it provides for some unique functionality. For instance, periodic batch processing allows the related results (such as activity from the same source) to be grouped together, and all of the activity can be ranked in the report by the relative threat levels. Another feature unique to the offline

environment is the ability to transfer logs from remote sites to a central site for correlation during off-peak times.

Offline processing also allows us to more easily overcome shortcomings in real-time IDSs. For example, many IDSs will start to drop packets when flooded with data faster than they can process it. Other forms of denial of service can also be launched against an active (real-time) IDS system, such as flooding it with fragmented IP packets in order to cause it to spend an inordinate amount of time and memory attempting to reconstruct bogus traffic. Meanwhile, the attacker can break into the real target system without fear of detection. The off-line environment is significantly less vulnerable to such a threat, especially if given a high degree of assurance that any traffic admitted to the local network is logged (such as by the firewall responsible for the admittance of such traffic).

Not much prior work has been done specifically comparing and contrasting off-line and on-line (real-time) environments. Most researchers seem to assume whichever environment suits their technique best. Lee et al. challenged this approach in (1999b) by examining what was necessary to apply intrusion detection models to a real-time (what they call "data flow") environment. Specifically, they found that the induction rules generated were too inefficient to apply directly in a real-time intrusion detection environment, hence they did some preliminary work in optimizing the generated rule set. They expanded on this work in (Lee et al. 2001) where they added a formal costing model. While many of their optimizations can reduce processing time without loss of accuracy, there remained some high cost rules that would be better suited for off-line analysis.

Given the high computational requirements of data mining for network intrusion detection and the unique benefits to be gained, we believe that an offline processing element will become a standard part of future defense in depth security architectures.

1.2.2 Multisensor correlation

Multisensor correlation is necessary to detect some types of malicious activity, especially scan activity only looking for a small number of services across a large number of, potentially highly distributed, hosts. As such, analyzing the data from multiple sensors should increase the accuracy of the IDS (Lee et al. 2001). Kumar (1995) observes that, "Correlation of information from different sources has allowed additional information to be inferred that may be difficult to obtain directly." Such correlation is also useful in assessing the severity of other threats, be it severe because an attacker is making a concerted effort to break in to a particular host, or severe because the source of the activity is a worm with the potential to infect a large number of hosts in a short amount of time.

Lee et al. (1999a) note that, "an IDS should consist of multiple cooperative lightweight subsystems that each monitor a separate part (such as an access point) of the entire environment." Their argument is primarily one of robustness: if an attack should take out one of the sensors, others will still be running to detect it (Lee and Stolfo 2000). A prototypical architecture for

such an approach is outlined by Dickerson and Dickerson. They present a simple three tiered architecture to do data mining (specifically fuzzy analysis) in (2000). The first tier is made up of numerous Network Data Collectors that only capture packets off the wire and feed them into the second tier. This second tier is made up of numerous Network Data Processors, which process the packets for the information that's needed at the third tier. The third tier is a centralized Fuzzy Threat Analyzer, which applies their fuzzy analysis to ascertain the risk presented by a given connection. Helmer et al. (1999) note that a data warehouse makes this data fusion process even easier, as it allows that data from heterogeneous sources to be handled in a consistent manner (aiding data mining), it allows distributed attacks to be more readily detected, and it provides administrators with a convenient means for doing auditing and forensics. Honig et al. (2002) present a complete architecture for intrusion detection based around a data warehouse. Their model, like Dickerson and Dickerson's, separates the sensors and detectors, and also has components for feature extraction, model generation and distribution, data labeling, visualization, and forensic analysis. Bass also has high level architectures for both the data fusion and mining processes, which he presents in (2000).

Multisensor correlation has long been a theme in Intrusion Detection, especially as most of the early IDS work took place in the wake of the Morris Worm, as well as by the need to centrally manage the alerts from a network of host based IDSs. For more on these early IDSs, see section 1.3.1. Recently, a great deal of work has been done to standardize the protocols that IDS components use to communicate with each other. The first solid protocol to do this is the Common Intrusion Detection Format (CIDF) (Tung 1999). CIDF spurred additional work in protocols for multisensor correlation, for example, Ning et al. (2000) extended CIDF with a query mechanism to allow IDSs to query their peers to obtain more information on currently observed suspicious activity. The importance of heterogeneous IDS communication has risen to the point that the Internet Engineering Task Force (IETF) has put forth a proposed protocol called the Intrusion Detection Exchange Protocol (IDXP), that looks like an expanded form of CIDF, only based on XML rather than CIDF's S-expressions (Intrusion Detection Working Group (idwg) 2003).

These protocols and most other work on multisensor correlation focuses on cooperative real-time IDSs. We believe that for large scale multisensor correlation in an offline environment, an architecture such as Honig et al.'s should provide the most flexibility and scalability.

1.2.3 Ensemble Approaches

One way to improve certain properties, such as accuracy, of a data mining system is to use a multiplicity of techniques and correlate the results together. "It is well known in the machine learning literature that appropriate combination of a number of weak classifiers can yield a highly accurate global classifier," (Lane 2000). Likewise, Neri (2000a) notes the belief, "that combining classifiers learned by different learning methods, such as hill-climbing and genetic evo-

lution, can produce higher classification performances because of the different knowledge captured by complementary search methods." The use of numerous data mining methods is commonly known as an ensemble approach, and the process of learning the correlation between these ensemble techniques is known by names such as multistrategy learning, or meta-learning. Lee et al. call the actual application of this learned correlation meta-classification (Lee 1999; Lee and Stolfo 2000; Lee et al. 1999a; Lee et al. 2002). Chan and Stolfo (1993) note that the use of meta-learning techniques can be easily parallelized for efficiency. Additional efficiencies can be gained by pruning less accurate classifiers (Prodromidis and Stolfo 2001). Carbone (1997) notes that these multistrategy learning techniques have been growing in popularity due to the varying performance of different data mining techniques. She describes multistrategy learning as a high level controller choosing which outputs to accept from lower level learners given the data, what lower level learners are employed, and what the current goals are.

The generalizations made concerning ensemble techniques are particularly apt in intrusion detection. As Axelsson (2000b) notes, "In reality there are many different types of intrusions, and different detectors are needed to detect them." As such, the same argument that Lee et al. make in (Lee 1999; Lee and Stolfo 2000; Lee et al. 1999a) for the use of multiple sensors applies to the use of multiple methods as well: if one method or technique fails to detect an attack, then another should detect it. They note in (Lee and Stolfo 1998) that, "Combining evidence from multiple base classifiers ... is likely to improve the effectiveness in detecting intrusions." They went on to find, "that by combining [signature and anomaly detection] models, we can improve the overall detection rate of the system without compromising the benefits of either detection method," (Lee et al. 2001) and "that a well designed / updated misuse detection module should be used to detect the majority of the attacks, and anomaly detection is the only hope to fight against the innovative and "stealthy" attacks," (Lee 1999). Axelsson (2000b) concurred, noting, "If we wish to classify our source behavior correctly as either $H0$ or $H1$ [benign or malicious], knowledge of both distributions of behavior will help us greatly when making the intrusion detection decision." Mahoney and Chan (2002) suggest that, because their technique has significant non-overlap with other IDSs, combining their technique with others should increase detection coverage. The use of an ensemble approach for intrusion detection provides benefits beyond accuracy. Lee (1999) proposes that combining multiple models allows for a more easily adaptable and extendible framework, and that a hierarchical arrangement is a natural way to accomplish this. Fan (2001) expanded on this idea, noting that an ensemble approach allows one to quickly add new classifiers to detect previously unknown activity. His research in this area suggests that this can be done without any loss, and possibly a gain, in classification performance. Additionally, the use of many small classifiers allows one to drop classifiers when they are no longer valid, for instance if they are based on outdated data.

There are numerous ways that one can go about the meta-classification task. Popular methods include Bayesian statistics, belief networks, or covariance ma-

trices (Kumar 1995). The use of fuzzy logic may be useful in enhancing the accuracy of these approaches. IDES, and its successor, NIDES, used a covariance matrix to correlate multiple statistical measures in order to calculate a quantifiable measure of how much a given event differed from the user's profile. This number was determined by finding the norm of the vector obtained by multiplying the vector of individual measures times a correlation matrix times the transpose of itself. The correlation matrix allows the system to compensate for multiple metrics that are similar in what they measure, such that they don't dominate the calculation (Javitz and Valdes 1991). Unfortunately, it appears that this covariance matrix needed to be built manually using domain knowledge. Lee, Fan, and others at the Columbia University IDS lab have applied meta-classification both to improve accuracy and efficiency, as well as to make data mining based IDS systems more adaptable. In (Lee and Stolfo 2000; Honig et al. 2002) they used meta-classification to improve both accuracy and efficiency (by running high cost classifiers only when necessary), and combined the results using boolean logic. The classifiers are produced using cost factors that quantify the expense of examining any particular feature in terms of processing time, versus the cost of responding to an alert or missing an intrusion (Fan 2001; Fan et al. 2000). As noted above, this approach can naturally be extended to the offline environment where one is no longer concerned with the costs of processing time. Instead, classifiers can be combined based solely on weightings of their accuracy, still taking into account the costs of responding to alarms versus missing an intrusion. Similarly, in (Lee et al. 2000) they note that, "the best way to make intrusion detection models adaptive is by combining existing models with new models trained on new intrusion data or new normal data." In that work, they combined the rule sets that were inductively generated on separate days to produce a more accurate composite rule set.

In performing a manual post hoc analysis of the results of the 1998 DARPA Intrusion Detection Challenge (described below in section 1.4.1), the challenge coordinators found, "[t]he best combination of 1998 evaluation systems provides more than two orders of magnitude of reduction in false alarm rate with greatly improved detection accuracy," (Kendall 1999). Didaci et al. (2002) and Giacinto and Roli (2002) apply three different meta-classification methods – the majority voting rule, the average rule, and the "belief" function – to the outputs of three neural nets trained on different feature sets from the KDD tcpdump data (discussed below), based on the premise that "human experts use different feature sets to detect different kinds of attacks." They found that these multi-strategy techniques, particularly the belief function, performed better than all three neural nets individually. The overall performance was also comparable to or better than a single neural net trained on the entire feature set, however the single neural net did a better job identifying previously unseen attacks. Crosbie and Spafford (1995) uses an ensemble of "autonomous agents" to determine the threat level presented by network activity. A different ensemble approach was used in ADAM, which applies multiple methods in series to increase accuracy. First it performs anomaly detection to filter out most of the normal traffic, then it uses a classification technique to determine the exact nature of the remaining

activity (Barbará et al. 2001). Staniford et al. (2002) use a simple meta-classifier to combine the results of multiple heuristics in the evaluation function between two events, in order to cluster events together.

Despite the positive accolades and research results that ensemble approaches for intrusion detection have received, the only IDS that has been specifically designed for the use of multiple detection methods is EMERALD, itself a research IDS, which will be discussed in more detail in section 1.3.1. The lack of published research into applying multiple different data mining methods for network intrusion detection seems to be a significant oversight in the intrusion detection community. We believe that ensemble based IDSs are the only foreseeable way of achieving high accuracy with an acceptably low false positive rate. We are however concerned with the research that indicates that ensemble based approaches reduce generalization (Didaci et al. 2002; Giacinto and Roli 2002), however that result may have been an artifact of either their use of a single type of classifier (neural networks), or their meta-classifiers.

1.3 Existing systems

We will briefly review systems that have been have particularly pivotal or unique in the development of IDSs, and that provide insight to the sophistication of currently deployed systems. Some of these systems have provided inspiration for the features described in section 1.2, particularly multisensor correlation and the application of an ensemble of techniques. This section will not cover systems that will be described in section 1.4.

1.3.1 Research IDSs

The first couple of IDSs of record (CERIAS 2003) that performed data fusion and cross sensor correlation were the Information Security Officer's Assistant (ISOA) (Winkler and Page 1990; jtruitt@dw3f.ess.harris.com 1994) and the Distributed Intrusion Detection System (DIDS) (Snapp et al. 1991; Snapp et al. 1991). ISOA conglomerated the audit information for numerous hosts whereas DIDS conglomerated the audit information from numerous host and network based IDSs. Both used a rules based expert system to perform the centralized analysis. The primary difference between the two was that ISOA was more focused on anomaly detection and DIDS on misuse detection. Additional features of note were that ISOA provided a suite of statistical analysis tools that could be employed either by the expert system or a human analyst, and the DIDS expert system featured a limited learning capability.

EMERALD extended some of the seminal IDS work at SRI (Denning 1987; NIDES 2002) with a hierarchical analysis system: the various levels (host, network, enterprise, etc) would each perform some level of analysis and pass any interesting results up the chain for correlation (Porras and Neumann 1996; Neumann and Porras 1999; Porras and Valdes 1998). It provided a feedback system such that the higher levels could request more information for a given activity.

Of particular interest is the analysis done at the top level which monitored the system for "network-wide threats such as Internet worm-like attacks, attacks repeated against common network services across domains, or coordinated attacks from multiple domains against a single domain," (Porras and Neumann 1996). The EMERALD architects employed numerous approaches such as statistical analysis, an expert system and modular analysis engines as they believed, "no one paradigm can cover all types of threats. Therefore we endorse a pluralistic approach," (Neumann and Porras 1999). Unfortunately, the EMERALD papers published to date lack the technical depth of SRI's papers on their earlier systems, and no new work has been reported on the system in the past five years.

1.3.2 Commercial IDSs[1]

The data fusion and correlation capabilities of commercial intrusion detection systems spans over a wide range. A few products are specifically designed to do centralized alarm collection and correlation. For example RealSecure SiteProtector, which claims to do "advanced data correlation and analysis" by interoperating with the other products in ISS's RealSecure line (Internet Security Systems 2003b). Some products, such as Symantec ManHunt and nSecure nPatrol, integrate the means to collect alarms and the ability to apply multiple statistical measures to the data that they collect directly into the IDS itself (Symantec 2003b; nSecure Software 2002). Most IDSs, such as the Cisco IDS, or Network Flight Recorder (NFR) provide the means to do centralized sensor configuration and alarm collection (Cisco 2003; NFR Security 2003). NFR provides the notion of "central stations" for this task, although Singh and Kandula (2001) note that it was developed "as an after-thought," as each sensor was only designed to interoperate with a single Intrusion Detection Appliance, and that NFR doesn't support distributed pattern matching. The problem with all of these systems is that they are designed more for prioritizing what conventional intrusion (misuse) detection systems already detect, and not for finding new threats. Other products, such as Computer Associates' eTrust Intrusion Detection Log View, and NetSecure Log are more focused on capturing log information to a database, and doing basic analysis on it. Such an approach seems to be more oriented towards insuring the integrity of the audit trail (itself an important activity in an enterprise environment), than data correlation and analysis (Computer Associates International 2003; NetSecure Software 2003).

1.3.3 Services

One of the more promising projects taking place is the Internet Storm Center (ISC), formerly known as the Consensus Intrusion Database (CID), operated

[1]Of course, the usual caveats regarding the appearance of commercial products in an academic paper apply: We do not endorse any of the products discussed, and the reader should not make any purchase decision based on the information provided herein. All names and trademarks are the property of their respective owners.

by SANS (SANS 2004). The ISC collects, correlates and analyzes connection log information provided by numerous Storm Center Analysis and Coordination Centers (SACCs). These SACCs represent a diverse collection of network activity from places such as large corporations, ISPs, or independent SACCs which collect information from individuals and small companies. Currently, the only independent SACC is DShield (DShield 2004). This data allows the ISC and SACCs to determine what the most frequently attacked services are, who the worst offenders are (based on IP address), and if there are any trends indicative of such things as a new worm or exploit for a given service.

As the only independent SACC, DShield is the most open of the partner sites (DShield 2004). DShield accepts various types of firewall and IDS log files from any source, and allows anyone to view activity summaries, such as the IP addresses that are doing the most negative activity, or the ports that are being scanned most often. DShield also provides a general query that allows you to query their database for a given address (either source or destination), source and destination ports, IP protocol and date ranges. Given the wealth of data they have (currently averaging 13 million records a day), the information obtained from the queries may be very useful to an analyst. A service similar to DShield that does not funnel its data up to the ISC is offered by myNetWatchman (myNetWatchman 2003).

Another similar service is DeepSight Analyzer, formerly known as the Attack Registry and Intelligence Service (ARIS), from SecurityFocus (SecurityFocus 2003). DeepSight Analyzer is more of a commercial offering that allows network administrators to consult with SecurityFocus' staff, for instance to report offenders to the appropriate ISP. Administrators can also correlate the activity they're seeing on their network with what other network administrators are seeing. Unlike ISC, DeepSight Analyzer only offers its summary reports to its customers.

Numerous companies, such as Counterpane (Counterpane Internet Security, Inc. 2003), ISS (Internet Security Systems 2003a), and Symantec (Symantec 2003a), offer managed security services. These services monitor the customer's network and respond to threats. While Symantec has acquired SecurityFocus, they seem to be keeping DeepSight Analyzer a separate offering from their managed services. None of these companies publish the means by which they do analysis on the data they receive from their customers, presumably because they regard it as a trade secret.

Of all these services, we believe that the Internet Storm Center, in particular the DShield data, represents a great resource. While they don't currently perform any intricate data mining, the statistical reports they currently provide may serve as important input to a data mining NID system. This service could be additionally enhanced through the addition of a programmatic query interface (such as an XML based web-service) so an intrusion detection system could query it for further information, such as other malicious activity from a given IP address. Further, the DShield data represents an important resource for exploring some of the questions presented by this survey.

1.4 Research

This section examines the current state of the art in data mining for network intrusion detection. We'll begin by looking at the various datasets available to evaluate such systems. Then, we'll look at the critical step of selecting which network features should be used as input to the data mining methods. Finally, we'll look at the methods themselves, from basic statistical techniques through advanced machine learning techniques.

1.4.1 Evaluation Datasets

Most intrusion detection techniques beyond basic pattern matching require sets of data to train on. When work on advanced network intrusion detection systems began in earnest in the late 1990's, researchers quickly recognized the need for standardized datasets to perform this training. Such datasets allow different systems to be quantitatively compared. Further, they provide a welcome alternative to the prior method of dataset creation, which involved every researcher collecting data from a live network and using human analysts to throughly analyze and label the data.

The first such widely cited dataset was for the Information Exploration Shootout (IES)(IES 2001), which unfortunately, is no longer available. It was used in (Lee 1999; Lee and Stolfo 1998; Singh and Kandula 2001; Neri 2000a; Neri 2000b; Luo 1999) to test the anomaly-detection performance of those systems. It consisted of four collections of tcpdump data: one that contained purely normal data, and three consisting of normal data with injected attacks. The data was apparently captured from a real network, and consists of only the packet headers in order to protect the privacy of the users.

In one of the early papers from Lee and Stolfo (1998), they noted the anticipated arrival of a new dataset from the Air Force's Research Laboratory (AFRL) in Rome, NY. The AFRL, along with MIT's Lincoln Lab, collected network traffic from their network and used it as the basis for a simulated network. Using a simulated network allowed them to carefully control if and when attacks were injected into the dataset. Furthermore, it allowed them to collect the entire packet without needing to protect user privacy. Details on the simulated networks and injected attacks are available in (Kendall 1999) They used the simulated network to create a couple weeks of intrusion-free data, followed by a few weeks of data labeled with intrusions. This data was made available to researchers in 1998 as the DARPA Off-line Intrusion Detection Evaluation. Participants were then given two weeks of unlabeled data, including previously unseen attacks, and asked to label the attacks. Lippmann et al. (the group at Lincoln Lab who ran the challenge) present the results in (2000). Numerous researchers have used this data to test their systems, both as part of the DARPA evaluation, as well as independently (Lippmann and Cunningham 1999; Lee 1999; Lee et al. 1999a; Lee et al. 1999b; Neri 2000a; Ghosh et al. 1999; Li et al. 2002; Mahoney and Chan 2002).

In response to the 1998 challenge, McHugh wrote a rather scathing critique of

the evaluation (2000). While he presents many good points on how an evaluation of IDSs should be performed, he also criticizes numerous shortcomings in the challenge without acknowledging how difficult addressing some of the issues is. For example, he notes that the generated data was not validated against real traffic to ensure that it had the same rates of malicious traffic versus non-malicious anomalous traffic that caused false positives. Doing so would, however, require more insight into real traffic than we can possibly obtain (in particular, intent), further, modeling of traffic at that scale is still an area with much research left to be done. Some of his more directly applicable feedback was used for the IDS challenge the following year. In particular, Das (2000) outlines the improvements that were made in the testbed and injected attacks, and Korba (2000) provides details on the addition of Windows NT hosts and attacks in the 1999 evaluation. Lippmann et al. report the results of the 1999 challenge in (2000).

While McHugh's critique was based primarily on the procedures used to generate the DARPA data, Mahoney et al. provide a critique based on an analysis of the data compared to real world data captured off of their network. They note that many of the attributes that are well-behaved in the DARPA dataset are not in real world data. They found that by mixing their real-world data with the DARPA data, they were able to increase the number of legitimate detections (detections that were not an artifact of the data generation process), using five simple statistically-based anomaly detectors. While this approach is an excellent stop-gap measure to achieve a more realistic performance measure using the DARPA data, it is not suitable for all research for two reasons: 1. It requires the addition of attack-free (or at least accurately labeled) real world data, which no one is willing to share to use as a standard, and 2. It requires that the method not differentiate between the DARPA data and the real-world data, which might be controllable for some methods (particularly those that produce human readable rules), but not for others (such as artificial neural networks and hidden Markov models). To address the first point, Mahoney et al. analyzed their real world data with Snort, however they don't address the possibility of the data containing new or stealthy attacks that Snort is incapable of detecting (and which drive the development of more advanced intrusion detection techniques) (Mahoney and Chan 2003a).

Lee did a great deal of analysis using the DARPA data, and identified 41 features of interest to a data mining based network IDS (see the next section for details). He provided a copy of the DARPA data that was already preprocessed, by extracting these 41 features, for the 1999 KDD Cup contest, held at the Fifth ACM International Conference on Knowledge Discovery and Data Mining. Since this version of the dataset already has the tedious and time-consuming preprocessing step done, it has been used as the basis for most of the recent research on data mining IDSs (Didaci et al. 2002; Giacinto and Roli 2002; Neri 2000a; Neri 2000b; Chittur 2001; Portnoy et al. 2001; Eskin et al. 2002; Yeung and Chow 2002; Fan et al. 2000; Fan 2001; Mukkamala et al. 2002; Mukkamala and Sung 2003).

There are a couple of other datasets that are used occasionally. The first is

the Internet Traffic Archive from Lawrence Berkeley National Laboratory(ITA 2000). It consists of a collection of tcpdump data captures from a live network on the Internet. It has been used by (Lee 1999; Lee et al. 2000; Lee et al. 2002; Luo 1999), primarily to show that data mining methods are sensitive to traffic patterns, such as the difference in traffic between working hours and overnight. Another dataset is Security Suite 16, which was created by InfoWorld to test commercial network intrusion detection systems(McClure 1998). It was used by Lee et al. in (2000, 2002).

As an alternative to using datasets, such as those described above, Eskin et al. (2000a, 2000) present an intrusion detection approach that does not require training data. Rather, it separates the normal data and the noisy data (anomalies) into two separate sets using a mixture model. This model can then be applied for anomaly detection. The technique can also be applied to a dataset that has been manually labeled, in order to detect marking errors (Eskin 2000b).

Of all the datasets presented here, the DARPA/KDD dataset appears to be the most useful as a dataset that can be used without any further processing. Unfortunately, given the criticisms against this data, we recommend that any further research in this area use both the DARPA datasets and one of the DARPA datasets mixed with real-world data. Doing so and being able to compare and contrast the results should help alleviate most of the criticism against work based solely on the DARPA data, and still allow work to be directly compared. Ideally, someone using a mixed dataset will make their real-world data available for everyone to use. This approach will necessitate the regeneration of connection records as the KDD Cup data only processed the 1998 DARPA data and obviously doesn't include any new data that may be mixed in.

Finally, a couple observations on dataset utilization: First, the typical approach to using datasets is to have some normal (intrusion-free) data and / or data with labeled intrusions, which is used to train the data mining methods being applied. None of the literature, however, explicitly discusses the use of separate training sets for meta-classifiers and the classifiers they incorporate. It would probably be useful to train the meta-classifier using attacks the classifiers have not already seen, such that the meta-classifier can give proper weight to classifiers that do a good job of detecting previously unseen techniques. Second, we have noticed a disturbing trend in some published research to modify a standard dataset because the researchers do not believe it accurately models real Internet traffic, for instance they believe that it has too many or too few attacks. This is unfortunate as it precludes a qualitative comparison of their research to other work. Further, as a community we lack any solid statistics on traffic characteristics in different environments, hence the use of modified data implies that the given technique isn't robust enough to perform well on different or dynamic networks.

1.4.2 Feature selection

Feature selection from the available data is vital to the effectiveness of the methods employed. Lee et al. (2001) note, "having a set of features whose values

in normal audit records differ significantly from the values in intrusion records is essential for having good detection performance." As Carbone (1997) notes, "data mining algorithms work more effectively if they have some amount of domain information available containing information on attributes that have higher priority than others, attributes that are not important at all, or established relationships that are already known."

The most popular data format to do analysis on is the connection log. Besides being readily available and a much more reasonable size than other log formats (such as packet logs), the connection record format affords more power in the data analysis step, as it provides multiple fields that correlation can be done on (unlike a format such as command histories). Additionally, not examining data stream contents saves significant amounts of processing time and storage, and avoids privacy issues (Hofmeyr and Forrest 1999). While some have argued that not looking at the data stream will prevent the detection of user to root (U2R) attacks (Chan et al. 2003), we believe that some of these attacks will be detected as attackers will modify the network stream precisely to avoid IDS detection, as described in (Ptacek and Newsham 1998). Lee et al. and Neri participated in the 1998 DARPA ID shootout (Lippmann et al. 2000), which provided network data in raw packet form. Both found that converting the network data to connection logs aided performance with their data mining techniques (Lee 1999; Lee et al. 1999a; Lee et al. 1999b; Neri 2000a). In the event that connection logs are built based on packet information, certain features, such as the state of the connection establishment and tear down, overlapping fragments, and resend rate will need to be calculated (Lee and Stolfo 1998).

Connection records provide numerous features that are intrinsic to each connection. These features, along with what research projects used which features, are summarized in table 1.1. Lee et al. (1999b) note that the timestamp, source address and port, destination address and port, and protocol uniquely identify a connection, making them essential attributes. They go on to note that "association rules should describe patterns related to the essential attributes." Specifically, at least one of those attributes must be present in the antecedent of a rule in order for that rule to be useful. They call this the axis attribute for the rule. For example, a rule that is based solely on the number of bytes transfered really does not convey any useful information. Likewise, if the value of some feature must be kept constant through the processing of a set of records (for instance, the destination host), that feature is called a reference attribute (Lee 1999; Lee and Stolfo 2000; Lee et al. 1999a; Lee et al. 1999b; Lee et al. 2000; Lee et al. 2002). Other researchers also had success with this approach. Dickerson and Dickerson (2000) found that their best results were achieved when they limited their rules to only use a key consisting of the source IP, destination IP, and the destination port, which they call the sdp. Hofmeyr and Forrest (1999) used the same approach, although they assign all connections with unassigned privileged ports to one service group, and all connections with unassigned non-privileged ports to another group.

While essential attributes provide vital information about connections, most

Cite	Essential attributes						Secondary attributes													
	Timestamp	Source IP	Destination IP	Source port	Destination port	Protocol	Duration	Source bytes	Destination bytes	TCP Flags	Land packet	# wrong frag	# urgent	Resent rate	Wrong resent rate	Duplicate ACK rate	Hole rate	wrong data packet size rate	% data packets	% control packets
(Sinclair et al. 1999)		Y	Y	Y	Y	Y														
(Li et al. 2002)	Y	Y	Y	Y	Y															
(Staniford et al. 2002)		Y	Y	Y	Y	Y				Y										
(Neri 2000a)[a]		Y	Y	Y	Y	Y	Y	Y		Y										
(Barbará et al. 2001)	Y	Y	Y	Y	Y					Y										
(Marchette 1999)	Y	Y	Y	Y[b]	Y[b]	Y														
(Bloedorn et al. 2001)	Y	Y	Y	Y	Y	Y	Y													
(Chan et al. 2003)		Y[c]	Y[c]	Y[d]	Y[d]		Y	Y[e]	Y[e]	Y										
(Lee et al. 2000)[a]					Y		Y	Y	Y	Y										
(Lee et al. 2002)	Y	Y	Y		Y		Y	Y		Y										
(Lee et al. 1999b; Lee et al. 1999a)	Y	Y	Y	Y	Y		Y	Y	Y	Y										
(Lee et al. 2000)	Y	Y	Y	Y	Y		Y	Y	Y	Y										
(Lee and Stolfo 2000)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y							
(Lee et al. 2001)	Y	Y	Y	Y	Y		Y			Y										
(Lee and Stolfo 1998)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y				Y	Y	Y	Y	Y	Y	Y
(Lee 1999)	Y[f]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
(Chittur 2001)[g]					Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y					
(Portnoy et al. 2001)[a]						Y	Y	Y[e]	Y[e]	Y										
(Didaci et al. 2002)[a]					Y[h]	Y	Y			Y										
(Dickerson and Dickerson 2000)	Y[f]	Y	Y	Y[f]	Y			Y[e]	Y[e]	Y										
(Hofmeyr and Forrest 1999)		Y	Y	Y[i]	Y[ij]															
(Singh and Kandula 2001)								Y[e]	Y[e]	Y				Y	Y	Y	Y		Y	Y

Table 1.1: Intrinsic connection attributes cited as being used by different research systems

[a]The authors imply that they use additional attributes beyond the ones they listed.
[b]All unprivileged ports are grouped together.
[c]The separate bytes of the IP addresses are treated as separate attributes
[d]All infrequently seen ports (less than 1% of the data) are grouped together for their clustering method.
[e]The authors don't specify if the bytes transfered are handled for the source and destination separately, or as a total quantity.
[f]The authors don't treat this as an "Axis" attribute.
[g]The authors note that they used all the attributes in the KDD '99 dataset. This table only displays the features that the authors explicitly cited.
[h]Didaci et al. say they use the "type" of connection, which this author assumes means the service with respect to the destination port.
[i]The authors use a service feature based on the source and destination ports.
[j]See text for description of how unassigned ports are handled.

Cite	# total connections	# establishment errors	# SYN flags	# other errors	# RST flags	# FIN flags	# to privileged services	# to unprivileged services	# to certain services	# to the same service	# to the same host	# different services accessed	# packets to all services	# ICMP packets	# unique keys	# new keys	# keys with outside hosts	variance of packet count to keys	average duration all services	average duration current service	average duration current host	bytes transfered all services	bytes transfered current service	bytes transfered current host	% on same host to same service	% of same service to same host
(Singh and Kandula 2001)		Y		Y				Y	Y										Y	Y	Y	Y	Y	Y		
(Lee and Stolfo 2000)		Y[a]								Y	Y														Y	Y
(Lee et al. 2000)[b]		Y[c]		Y						Y									Y	Y		Y				
(Lee and Stolfo 1998)		Y		Y				Y	Y	Y									Y	Y		Y	Y			
(Lee 1999)		Y		Y			Y	Y		Y	Y								Y	Y		Y	Y		Y	Y
(Chittur 2001)[d]		Y		Y			Y	Y		Y	Y								Y	Y		Y	Y		Y	Y
(Lee et al. 1999b)[e]	Y	Y								Y	Y															
(Lee et al. 2002)	Y									Y[f]									Y			Y[g]				
(Portnoy et al. 2001)[h]		Y	Y																						Y	Y
(Didaci et al. 2002)[h]										Y	Y															
(Luo 1999)			Y		Y	Y						Y														
(Dasgupta and González 2002)													Y	Y								Y				
(Dickerson and Dickerson 2000)		Y[i]							Y				Y		Y	Y	Y	Y								

Table 1.2: Calculated connection attributes cited as being used by different research systems

[a] On both per host and per service basis, separated between SYN and REJ errors.

[b] The authors imply that they use additional attributes beyond the one listed. As Lee was one of the authors, the calculated features they used were presumably similar to those in (Lee 1999).

[c] The authors use both the number of successful connections and the total number of errors.

[d] The authors note that they used all the attributes in the KDD '99 dataset, so we assume that they are the same as in (Lee 1999).

[e] The authors treat most of the calculated features with respect to other calculated features, for example, "a count of connections to the same *dst_host* in the past 2 seconds, and among these connection, the percentage of those that have the same *service* as the current. . ."

[f] They counted number of connections to the same service with the same number of bytes from the source.

[g] They only looked at number of bytes from the destination.

[h] The authors imply that they use additional attributes beyond the one listed. As they used the KDD '99 dataset, they presumably used the same calculated features as (Lee 1999).

[i] The authors actually use the converse of this, number of successful connections.

research uses some of the secondary attributes, such as connection duration, TCP flags and the volume of data passed in each direction, as shown in table 1.1. Some researchers, such as Dickerson and Dickerson (2000), also treat the essential attributes that they do not key off of, such as timestamp and source port as secondary attributes. Perhaps the most interesting data point in table 1.1 is the work of Singh and Kandula (2001), who did not report the use of any essential attributes, despite their work being based heavily on that of Lee et al. This may account for their poor performance, which Singh and Kandula note "could be attributed to their [Lee et al.] greater care in choosing connection features," .

Unfortunately, the intrinsic attributes of a connection are insufficient to provide adequate detector performance against most attacks. Lee et al. found that including temporal information with each data point significantly increased accuracy, and most research since has corroborated this finding. Temporal information is captured in the form of calculated attributes. A calculated attribute provides the average value of an attribute, or the count or percentage of connections fulfilling some criteria over the last w seconds, or n connections. For example, in (Lee and Stolfo 1998; Lee and Stolfo 2000; Lee et al. 1999b; Lee et al. 2000) they included a count of how many packets in the last w seconds had the same value for an attribute as the current connection to the same service or destination host. Lee provides many more calculated features in his thesis (1999). Table 1.2 shows which features have been cited in the literature, and associates research projects with the features they used. In (Lee and Stolfo 2000; Lee et al. 2000) they formalized the notion of defining calculated features as functions of the other features, using a predefined set of operators such as count, percent, or average, as well as a set of data constraints such as same host, same service, different host, or time window. Honig et al. (2002) extended this approach by allowing the analyst to dynamically create new features using these functions. A new column is automatically added to the table to store the new feature in the database. In (Lee et al. 2002) they explain that the decision to count the occurrence of a given attribute's value is made when many frequent episode rules are generated that include the given feature with a constant value. Likewise, they generate an average value for an attribute if that attribute is seen repeated in many frequent episode rules with different values.

One question that remains regarding calculated attributes is the proper value for w or n (the time window). While Lee et al.'s work with the IES dataset indicated a 30-second time window was ideal (Lee and Stolfo 1998), the same algorithms applied to the IWSS16 data indicated a 140-second time window was ideal (Lee et al. 2000), indicating that the ideal value is dependent on the operational environment. A possible solution to this problem is the use of multiple time windows, as Barbará et al. (2001) did. Specifically, they used two time windows: a three second window to examine features that are highly correlated in time, and a one day window to examine features that are correlated over a longer time period.

The most interesting data point in table 1.2 is that of Luo's work. Luo (1999) appeared to use only calculated attributes, in particular the number of

connections with certain TCP flags in the past w seconds. Despite this, Luo had excellent success at identifying datasets containing anomalies, although no results are given as to identifying the anomalies themselves. This speaks well for either the use of calculated attributes, Luo's fuzzy-rule approach, or both.

There are numerous techniques to identify which of the secondary or calculated attributes provide the best feature set for a given method. Frank (1994) used backwards sequential search, beam search, and random generation plus sequential selection. Lee and Xiang (2001) do an excellent job of applying information-theoretic measurement techniques to feature sets in order to evaluate the relative utility of different sets (based on some earlier work by Lee in (1999)). The measures they use are entropy, conditional entropy, relative conditional entropy, information gain, and information cost.

The concept of time is particularly problematic for IDSs to handle, both in terms of correlating events over time, and behavior that changes over time. The ability to correlate events over time is useful, particularly for identifying regular activity, such as an automated process that transfers files at a specific time every night. Such behavior may either be expected and safely ignored, or it may indicate activity worth investigation as it may be from a Trojan horse or other form of malicious code. To address this problem, Li et al. (2002) developed the notion of a calendar schema. These calendar schemas build temporal profiles, which allow the mined induction rules to use multiple time granularities. The other problem that time presents is that the behavior of monitored networks will change over time. Because of this, the profiles that characterize the network will need to incorporate new behavior and age out old behavior. The manner in which this is accomplished is necessarily tied to the underlying data model. For instance, in IDES and NIDES they accomplished this by periodically updating their statistical models by multiplying in an exponential decay factor when adding in the currently observed values for an attribute (Javitz and Valdes 1991; Javitz and Valdes 1993). For the inductive rules used by Lee et al. a new rule set was created for each day's data, then merged with the existing rule set. By keeping track of how often a rule appeared in a daily rule set, and when a rule last appeared, they could ascertain the relevance of a rule and age out old rules.

Some of the techniques described below, particularly those that rely on a mapping between the network connection records and a geometric space (hyperplane), only produce optimal, or even usable, results if the features in the records are first normalized. This is typically accomplished by scaling continuous values to a given range, possibly scaling the values with a logarithmic scale to avoid having large values (typically seen in distributions of attributes of long-tailed network data) dominate smaller values (Chan et al. 2003). Discrete vales are typically mapped to their own features (Yeung and Chow 2002), coordinates that are equidistant from one another (Eskin et al. 2002), or represented based on their frequency (Chan et al. 2003). A similar problem is presented by zeros in the dataset, as features with an observed value of zero may either actually be zero, or they may be zero due to a lack of observations. To address that problem Barbará et al. (2001) applied pseudo-Bayes estimators to refine the zero

values in their training data. Chan et al. (2003) address the same problem in association rules by using a probability of novel events based on the frequency of rules supporting the antecedent in the training set. They also looked at Laplace smoothing, however found it was inappropriate as it required the alphabet sizes and distributions to be known at training time.

Another technique that can be applied to the dataset to improve accuracy is compression. Neri (2000a, 2000b) found that compressing features, by representing many discrete values with a single value is, "a valuable way of increasing classification performances without introducing complex features that may involve additional processing overhead."[2] Barbará et al. (2001) applied feature compression by grouping together connections that come from the same domain (subnet) in order to detect activity coming from a highly coordinated group of hosts. The information-theoretic work by Lee and Xiang (2001) explains that substituting a single record to represent a group of records (such as all those in the past w seconds for a given service), significantly increases the information gain (which should subsequently improve the accuracy of detection methods on that data).

Singh and Kandula (2001) note that the features they chose were based purely on heuristics and that, "It would be really useful if the choice of these features could be automated." Helmer et al. (1999) did exactly that with system call data using the "bag of words" technique, where every call was represented by a bit in a vector labeled as normal or intrusive. They then fed these vectors to a genetic algorithm and found that the set of necessary features was about half of the full set of available features. Using the pruned set resulted in comparable detection accuracy and reduced the false positive rate to zero.

In summary, the ideal set of features of connection data for network intrusion detection appears to be tied to the specific data mining methods employed. While the predominate method of feature selection has been heuristics, and trial and error, there is a growing body of literature on automated feature selection. For future work, the best approach will likely be to begin with a feature set comprised of all essential, intrinsic, and calculated attributes, as appear in tables 1.1 and 1.2, augmented by a calendar schema. The calculated attributes should be calculated for short (three second), medium (two minute), and long (one day) durations. The set can then be pared down for a particular method by successively eliminating features that decrease or provide no increase in classification performance. Once paired down, one should investigate if pseudo-Bayes estimation and compressing connections provides any increase in performance.

1.4.3 Applicable methods

In this section we will survey the wide field of methods that have been applied or proposed for data mining on network connection data. We will refrain from presenting any comparisons between methods in different papers, for two main reasons:

[2] Minor grammatical cleanup.

1. Very few of the methods used the same datasets as any other method here. In the few cases that did, all but one pair used different mixes of the available data in the set for training and testing (for example, one method may have reported results based on training and testing on the DARPA BSM data, whereas another may have used the DARPA tcpdump data, and yet a third may have used both).

2. Our assertion is that each method has strengths and weaknesses, and it is only through the application of numerous different methods that most attacks can be found while minimizing the false positive rate.

Statistical techniques

Statistical techniques, also known as "top-down" learning, are employed when we have some idea as to the relationship we're looking for and can employ mathematics to quantify our search. Three basic classes of statistical techniques are linear, non-linear (such as a regression-curve), and decision trees (Carbone 1997). Statistics also includes more complicated techniques, such as Markov models and Bayes estimators.

Denning (1987) described how to use statistical measures to detect anomalies, as well as some of the problems and their solutions in such an approach. The five statistical measures that she described were the operational model, the mean and standard deviation model, the multivariate model, the Markov process model, and the time series model. She noted that the time series model was similar to the mean and standard deviation model in terms of applicability, and that the time series model stood to provide more accurate results, however it was more costly than the standard deviation model. Since then, techniques based in immunology have shown the usefulness of the time series model (Tan and Maxion 2002), however the models themselves are built in an off-line environment due to the cost of their construction. These statistical measure models may be further assisted by the availability of a data warehouse to do mining from, as not all the metrics that are going to be used need to be known up front. For instance, if the system or analyst decides that the system should detect anomalies in the mean and standard deviation of duration of FTP sessions, the necessary mean and standard deviation can be constructed from the data warehouse on the fly, rather than having to wait for the collection of new data. Javitz and Valdes (1991, 1993) provide more details on the individual statistical measures used in (N)IDES. In particular, they note that there are 16 to 32 intervals for each statistic (called Q), ranked by probability of past activity. Each record is assigned a measure (called S), based on the probability of appearance in that interval. They also provide formulas for calculating the Q statistic from ordinal and categorical measures, and numerous variations for special situations. This work is extended into the network domain in (Porras and Valdes 1998). These statistical patterns can be calculated with respect to different time windows, such as day of the week, day of the month, month of the year, etc. (Frank 1994), or on a per-host, or per-service basis (Lee et al.

2000; Krügel et al. 2002).

Staniford et al. (2002) uses a similar approach by employing a Bayes (or belief) network to calculate the conditional probabilities of various connection[3] features with respect to other connection features. These probabilities are then used to determine how anomalous each connection is. Mahoney and Chan (2002) combined the output of five specific probability measures to determine how anomalous each connection observed was. They take a different approach in (Mahoney and Chan 2003b; Mahoney and Chan 2003a; Chan et al. 2003) by generating a set of rules for normal traffic where each rule retains the percentage of records in the training stream that support it. When a record is detected that violates a given rule, its anomaly score is the sum of each rule's support value times the time since that rule was last violated. While applying these types of approaches, studies such as (Bykova and Ostermann 2002; Bykova et al. 2001; Paxson et al. 1999) will prove useful in separating traffic that is anomalous due to malice, versus implementation and configuration problems.

Frank (1994) cites Decision Trees as a prime example of a classification method that is well suited for the intrusion detection domain, however he doesn't implement such a system. Sinclair et al. (1999) describes how they used Quinlan's ID3 algorithm to build a decision tree to classify network connection data. Bloedorn et al. (2001) and Barbará et al. (2001) also note that they are using decision tree based methods, although they provide no details on the construction.

Machine Learning

In contrast to statistical techniques, machine learning techniques are well suited to learning patterns with no a priori knowledge of what those patterns may be, hence they are sometimes referred to as "bottom-up" learning techniques (Carbone 1997). Here we will look at a variety of machine learning techniques that break down roughly into techniques that perform classification, clustering, and other techniques.

Classification techniques A classification based IDS attempts to classify all traffic as either normal or malicious in some manner. The primary difficulty in this approach is how accurately the system can learn what these patterns are. This ultimately affects the accuracy of the system both in terms of whether non-hostile activity is flagged (false positive) and whether malicious activity will be missed (false negative). Some classification techniques are binary (they classify data into one of two classes), while others are n-ary (they classify data into one of an arbitrary number of classes). We do not differentiate, as one can use multiple binary classifiers to emulate an n-ary classifier. Some of the techniques we look at have also been investigated as being used as a filtering mechanism to limit the amount of data that successive classification systems need to evaluate

[3]The authors actually look at individual packets, however their approach should perform identically on connection records.

(Frank 1994; Barbará et al. 2001). Five general categories of techniques have been tried to perform classification for intrusion detection purposes: inductive rule generation, genetic algorithms, fuzzy logic, neural nets, and immunological based techniques. We will look at each in turn.

Inductive rule generation, such as that done by RIPPER (Cohen 1995) has been shown to be a fairly effective and straightforward way to produce a system that classifies traffic into normal or various forms of intrusive patterns. The system is a set of association rules and frequent episode patterns than can be applied to the network traffic to classify it properly. One of the attractive features of this approach is that the rule set generated is easy to understand, hence a security analyst can verify it (Lee 1999; Lee and Stolfo 1998; Lee and Stolfo 2000; Lee et al. 1999a; Lee et al. 1999b; Lee et al. 2000; Lee et al. 2002). Another attractive property is that multiple rule sets may be generated and used with a meta-classifier (Lee 1999; Lee and Stolfo 2000; Lee et al. 1999a; Lee et al. 2002). Lee et al. found that RIPPER had to be slightly modified to generate rules for important, yet infrequent events. This was done using two techniques: "level-wise approximate mining", which does not reuse source data unless it provides information on a different axis attribute, and using "relative frequency support", where support for a rule is based on other data points with the same value for a particular attribute (Lee 1999; Lee et al. 2000; Lee et al. 2002). Helmer et al. (1999) duplicated Lee's work, and was able to achieve a comparable detection rate using a reduced feature set (see section 1.4.2 for more details on their work). Warrender et al. (1999) also used RIPPER to produce inductive rules, however in order to use the rules in an on-line environment, they raised an alarm on sequences which violated a number of high confidence value rules in a short time span.

Another classification approach is to produce a rule set via the application of a genetic algorithm using a system such as REGAL (Neri 2000a; Neri 2000b). The approach is essentially identical to the inductive approach, with the primary difference being the algorithm used to generate the rule set. Neri does not comment on the clarity or efficiency of the rules produced. Sinclair et al. (1999) turns the network connection features into multiple genes, where each gene takes on either a value or a wildcard. Each connection is then compared to the set of chromosomes, and if there is not a match, the connection is anomalous. Dasgupta and González (2001) also used a genetic algorithm, however they were examining host-based, not network-based IDSs. Nevertheless, their technique is easily applicable to network connection data. Instead of running the algorithm directly on the feature set, they used it only for the meta-learning step, on labeled vectors of statistical classifiers. Each of the statistical classifiers was a two bit binary encoding of the abnormality of a particular feature, ranging from normal to dangerous. Crosbie and Spafford (1995) used a similar approach where a genetic algorithm was used to determine the best population of "autonomous agents", where each agent is represented as a parse tree applied against network activity. Chittur (2001) applied a genetic algorithm to the vector composed of the weightings in a decision tree, where each node in the tree represents a single feature in the dataset, with symbolic features, or features

with values that span a large range, mapped to preset values. The sum of the feature values times their weightings determines the certainty of an attack.

An interesting combination of the statistical metrics and classification approaches has been done using fuzzy logic. In (Dickerson and Dickerson 2000) the authors classify portions of the data (where the portions are temporally related) based on various statistical metrics. They then create and apply fuzzy logic rules to these portions of data to classify them as normal or some type of intrusion. They found that the approach is particularly effective against scans and probes. They expand on this work in (Dickerson et al. 2001), where they primarily use the fuzzy rules to correlate the information coming from multiple sensors. The primary disadvantage to this approach is the labor intensive rule generation process. To that end, Luo (1999) expanded upon the work of Lee et al. by adding fuzzy logic to the association rules and frequency episodes constructed inductively from the data. Luo notes that the use of fuzzy logic allows "more abstract and flexible patterns for intrusion detection," additionally, he notes that intrusion detection is a natural application of fuzzy logic, as frequently we can not make an absolute determination if a given connection is good or bad. This is known as the "sharp boundary problem." Instead of having a threshold to determine if a connection is good or bad, we use fuzzy logic to quantify our confidence in the maliciousness of a given connection. Luo used the fuzzy association rules mining algorithm by Kuok et al., with an additional normalization step. Although Luo doesn't provide a comparison of the detection rates, he does show a significant reduction in the false positive rate over non-fuzzy methods. If fuzzy logic is used with an n-ary classifier, it may be useful to report the next closest classifier for anomalies, for instance, "x is an anomaly, however there is a low probability that it's actually normal." versus, "x is an anomaly, however there is a low probability that it's actually a denial of service attempt."

Numerous projects have used neural nets for intrusion detection using data from individual hosts, such as BSM data (Ghosh et al. 1999; Ryan et al. 1998; Endler 1998). Ghosh et al. (1999) found that a well trained, pure feed-forward, backpropagation neural network performed comparably to a basic signature matching system. They then replaced the pure feed-forward neural net with an Elman network, which includes hidden context nodes. "Despite being the least extensively tuned of the three methods employed, the Elman nets produced the best results overall." This was attributed to the temporal state nature of system traffic. Recurrent neural networks, and similar networks such as Kohonen, Hoppfield, and RBF have properties that make them attractive for intrusion detection, such as the ability to forget old behavior as they learn new behavior, and to perform sequence analysis (Frank 1994; Endler 1998). Less work has been done applying neural nets to network traffic, with the only published work being used as a demonstration of ensemble methods in (Didaci et al. 2002; Giacinto and Roli 2002), which used a fully connected, three layer perception configuration, with an appropriate number of input neurons, a variable number of hidden neurons, and five output neurons.

Hofmeyr and Forrest (1999) present an interesting technique based on im-

munological concepts. They define the set of connections from normal traffic as the "self", then generate a large number of "non-self" examples: connections that are not part of the normal traffic on a machine. These examples are generated using a byte oriented hash and permutation. They can then compare incoming connections using the r-contiguous bits match rule. If a connection matches one of the examples, it is assumed to be in non-self and marked as anomalous. Dasgupta and González (2002) used a similar approach, although they generated a set of fuzzy rules using a genetic algorithm. They found that while this non-self approach was not as accurate as a nearest neighbor match with the self set, it was significantly more efficient. Fan also used a similar approach in (2001), most likely unintentionally. He found that injecting artificial anomalies into the dataset significantly increased detection of malicious anomalies, including those that had never been seen before. In the work of both Fan, and Dasgupta and González, the best improvements were made detecting user-to-root and remote-to-local attacks.

Clustering techniques Clustering is a data mining technique where data points are clustered together based on their feature values and a similarity metric. Frank (1994) breaks clustering techniques into five areas: hierarchical, statistical, exemplar, distance, and conceptual clustering, each of which has different ways of determining cluster membership and representation. Berkhin presents an excellent survey of specific methods for techniques in most of these areas in (2002). Frank (1994) notes that clustering is an effective way to find hidden patterns in data that humans might otherwise miss. Clustering is useful in an intrusion detection as malicious activity should cluster together, separate from non-malicious activity.

Clustering provides some significant advantages over the classification techniques already discussed, in that it does not require the use of a labeled data set for training. For example, Portnoy et al. (2001), Eskin et al. (2002), and Chan et al. (2003) have applied fixed-width and k-nearest neighbor clustering techniques to connection logs looking for outliers, which represent anomalies in the network traffic. Bloedorn et al. (2001) use a similar approach utilizing k-means clustering. Marin et al. (2001) also use a similar approach utilizing learning vector quantization (LVQ), which is designed to find the Bayes Optimal boundary between classes, using k-means clustering to determine initial vector positioning. Unfortunately, this approach will overlook intensive malicious traffic, such as heavy probing and denial of service attacks, which should form their own clusters. Indeed, Marin et al. found that the majority of their false negatives ended up in the same cluster. Chan et al. (2003) accounted for this by looking at both the distance and density of clusters as they found that attacks were often in outlying clusters with statistically low or high densities. Staniford et al. (2002) use simulated annealing to cluster events (anomalous packets) together, such that connections from coordinated portscans should cluster together. By using simulated annealing they reduce the run time from polynomial to linear. An alternative approach, that we haven't seen explicitly used, however is sug-

gested by (Sequeira and Zaki 2002), is the use of clustering for data reduction or post hoc labeling. This would allow analysts to quickly review, and potentially label, a large corpus of data, by operating on a small number of clusters (on the order of 10^2 to 10^3, rather than than the huge number (starting on the order of 10^4, with no upper bound) of individual network connections.

Marchette (1999) used clustering to project high dimensionality data into a lower dimensional space where it could be more easily modeled using a mixture model. He did this using a method known as approximate distance clustering (ADC). The method was then refined using an algorithm called AKMDE which, in addition to clustering the data, determines the proper number of clusters for that data. Sequeira and Zaki (2002) also note the difficulty in determining the number of clusters beforehand, and hence created the "Dynamic Clustering" method to cluster similar user activity together, creating the proper number of clusters as it proceeds. This is a very useful feature, considering that few papers on applying clustering to intrusion detection present any findings as to the optimal number of clusters, although Marin et al. (2001) cite a rule of thumb that one should have 5 times the number of clusters as dimensions.

Yeung and Chow (2002) used an estimation for the density function, based on a Parzen-window, to detect anomalous network traffic. They note that this approach is similar to the k-nearest neighbor method. With this approach, they achieved an impressive detection rate for user-to-root and remote-to-local attacks in the KDD Cup dataset, although this came at the expense of a higher false positive rate.

Lane (2000) based his host-based IDS in part on an instance-based learner, a type of exemplar clustering approach. It performed comparably to his Hidden Markov Model on the same data.

While not all clustering techniques are applicable to the intrusion detection domain, the wealth of techniques that Berkhin presents easily leaves the impression that there is a great deal of potential for further research in the application of clustering techniques to network intrusion detection.

Support Vector Machines Another technique that, like clustering, relies on mapping the network connections to a hyper-plane, is Support Vector Machines (SVMs). SVMs attempt to separate data into multiple classes (two in the basic case) though the use of a hyper-plane. Eskin et al. (2002), and Honig et al. (2002) used an SVM in addition to their clustering methods for unsupervised learning. The SVM algorithm had to be slightly modified to operate in the unsupervised learning domain. Once it was, its performance was comparable to or better than both of their clustering methods. Mukkamala, Sung, et al. (2002, 2003) used a more conventional SVM approach. They used five SVMs, one to identify normal traffic, and one to identify each of the four types of malicious activity in the KDD Cup dataset. Every SVM performed with better than 99% accuracy, even using seven different variations of the feature set. As the best accuracy they could achieve with a neural network (with a much longer training time) was 87.07%, they concluded that SVMs are superior to neural nets in

both accuracy and speed.

Other techniques Numerous other techniques have been suggested that could be used to mine security information from network connection logs. Here we will briefly look at a couple techniques that have been successfully applied for intrusion detection, as well as the methods that have been proposed.

A technique that has been successfully applied to misuse detection systems is colored Petri nets (Kumar 1995). In IDIOT, the colored Petri nets were created by hand. It would be instructive to investigate if similar Petri nets could be constructed automatically and used for intrusion detection in an off-line environment.

Another approach that has been successfully applied for intrusion detection is the use of graphs. This approach was pioneered by GrIDS, the Graph based Intrusion Detection System. GrIDS creates graphs of network activity which reveal the causal structure of the network traffic, hence allowing coordinated attacks to be easily detected (Staniford-Chen et al. 1996). This concept was expanded on by Tölle and Niggermann (2000) who note, "We believe that graph clustering delivers patterns that make it possible to visualize and automatically detect anomalies in the network traffic." In their system, the traffic is used to construct a graph, where nodes representing similar traffic are clustered together, and a mapping function is used to classify the type of traffic the cluster contains, based on the properties of the cluster and the nodes in it. Interestingly, they noticed that, "the learning method relies mainly on average values of cluster properties when deciding whether an intrusion happens." The primary disadvantage that they found in their approach was "that this method is only able to detect intrusions producing a considerable amount of network traffic."

Much work has been done or proposed involving Markovian models. For instance, the generalized Markov chain may improve the accuracy of detecting statistical anomalies. Unfortunately, it has been noted that these are complex and time consuming to construct (Kumar 1995), however their use may be more feasible in a high-power off-line environment. Lane (2000) trained a Hidden Markov Model (HMM)[4] on the same data that he used to train his instance-based learner. He notes that the Hidden Markov Model "assumes that the state variables are hidden and correspond to phenomena that are, perhaps, fundamentally unobservable," and as such, should perform well in modeling user actions. He concluded that the HMM and the instance-based learner mentioned above, trained using the same data, performed comparably. Eskin (2000a) notes that the anomaly detection scheme, covered in section 1.4.1 can use any probabilistic model, including Hidden Markov Models, Markov chains, or sparse Markov transducers, as well as non-Markovian models such as naive Bayes, or maximum entropy. Warrender et al. applied a Hidden Markov Model to system call data. They noted that best performance was obtained by using a number of states corresponding to the number of system calls used by an application. While HMMs performed better than inductive rules or sequence matching, the

[4]See (Rabiner 1989) for a through tutorial on Hidden Markov Models.

authors questioned if the incremental improvement was worth the significantly longer (days versus minutes) training times (1999), and we argue that it does, especially given the ever increasing capabilities of CPUs and the need to analyze constantly increasing amounts of traffic.

A number of other methods have been proposed that may be useful. For instance, Bass (2000) (based on (Waltz and Llinas 1990)), suggested that the Dempster-Shafer Method, or Generalized EPT Method may be useful as combinatorial methods in a system employing multiple approaches, or in fusing the data from multiple sources. Lane (2000) identified numerous techniques from the signal processing and pattern recognition communities such as "spectral analysis, principle component analysis, linear regression, linear predictive coding, (γ, ϵ)-similarity, neural networks, and nearest-neighbor matching" that, while not well suited for data mining of command line histories as Lane was doing, may be better suited for network traffic analysis.

1.4.4 Visualization

Besides the potential for network visualization in GrIDS (above), a data-mining system for intrusion detection will ideally offer numerous ways to visualize the data to aid human analysts in identifying trends that may be missed by automated methods. Dickerson and Dickerson provide an example of this in (2000) where they plot the unique source and destination IPs and service versus time and data volume. Lee (1999) presents a method by which the RIPPER rules, which have been inductively mined from the data, can be visualized for comparison purposes. Some work has been done on explicitly using visualization for anomaly detection on networks. In (Teoh et al. 2003), the authors present a system to visually display changes in routing information. In doing so, large abnormal changes, which are difficult to detect in an automated fashion, can be easily identified by an analyst. Additionally, such an analyst should quickly learn the visual patterns of certain types of anomalous activity, aiding in its future detection.

1.4.5 Predictive analysis

Ideally, a data-mining based IDS will do more than just detect intrusions that have already happened: we would like it to provide some degree of predictive analysis. Frequent episode rules already give us some of this power on a short time scale. For example, if we find two rules, $A \rightarrow B$ and $B \rightarrow C$, then, when we see A, we should be able to predict the upcoming appearance of C (Carbone 1997). We would like to extend that power to look at activity on a higher level. Lee (1999) notes in that "a typical attack session can be split into three phases: a learning phase, a standard attack phase, and an innovative attack phase." Given that, we should be able to predict standard and innovative attacks to some degree based on prior activity. Complicating this, of course, is the fact that a sophisticated attacker will use different sources for each phase (making it difficult for us to correlate the actions back to that particular attacker). Even so,

any level of predictive analysis will be an improvement over currently available systems.

Another area that predictive analysis may be useful is in early detection of worms. Typically, retrospective analysis of worms such as Code Red have shown similar activity (presumably testing) of the worms a number of weeks before its widespread outbreak. Additionally, statistical trending should be able to detect the start of a worm's characteristic exponential curve before the infection rate begins increasing steeply, at least for traditional worms such as Code Red or Nimda. Unfortunately, fast infection rate worms, such as the SQL Slammer worm, will most likely have completed their exponential growth before the connection data can be fused and mined.

1.4.6 Summary

This section has provided an overview of the various data mining methods that have been employed or proposed for network intrusion detection. Table 2.5 summarizes these techniques using the components for data mining techniques proposed in (Fayyad et al. 1996). Most of the techniques presented here appeared to excel at detecting at least one type of malicious activity, however none of the techniques showed any exceptional ability to detect previously unseen intrusions. None of the techniques showed any promise of detecting all types of malicious activity. As such, we reiterate that a reasonable intrusion detection system can only be achieved though the application of many of these techniques tied together via a meta-classifier. Such an approach should be able to detect the union of different types of malicious activity detected by the individual techniques it employs.

Chapter 2

Future Work

While a great deal of research has been done in applying data mining techniques to network connection data for intrusion detection purposes, there remains much that needs to be researched in this arena. For example, there are a number of techniques that have been suggested by various authors for which we need more definitive data regarding their applicability and effectiveness. Table 2.1 summarizes these methods.

There are also a number of open problems in the area that require further research, for example,

1. For baseline purposes, what is the accuracy of a modern, signature based network intrusion detector on the standard evaluation datasets?
2. What is the ideal number of states for a Hidden Markov Model of network connection data, and what parameters influence this value?
3. What are the ideal feature sets for different data mining techniques?
4. Can the performance of some of the data mining techniques be improved by grouping related packets in connectionless protocols like UDP and ICMP, and treating them as a single connection (as in TCP)?
5. Should we use separate training sets for meta-classifiers and the classifiers they incorporate?
6. What is the accuracy of an offline network intrusion detection system employing numerous, different, data-mining techniques?
7. How much data is required in order to properly train a data-mining based IDS?
8. How true is the assumption that employing data mining techniques on a network other than the one on which they were trained renders them ineffective, or at least seriously degrades their performance?

Proposed by	Method
(Denning 1987)	operational model
(Denning 1987)	mean and standard deviation
(Denning 1987)	multivariate model
(Denning 1987)	Markov process model
(Kumar 1995)	generalized Markov chain
(Denning 1987)	time series model
(Frank 1994) and (Endler 1998)	Recurrent neural network[a]
(Chan and Stolfo 1993) and (Prodromidis and Stolfo 2001)[b]	C4.5, ID3, CART, WPEBLS
(Bass 2000) from (Waltz and Llinas 1990)	Dempster-Shafer method
(Bass 2000) from (Waltz and Llinas 1990)	Generalized EPT
(Lane 2000)	Spectral analysis
(Lane 2000)	Principle component analysis
(Lane 2000)	Linear regression
(Lane 2000)	Linear predictive coding
(Lane 2000)	(γ, ϵ)-similarity

[a] As well as similar networks such as Kohonen, Hoppfield, and RBF.
[b] Not proposed specifically for intrusion detection, however cited for use in similar domains.

Table 2.1: Proposed methods for intrusion detection that need further investigation.

9. Should normal usage profiles be based on individual hosts and services, or should hosts and / or services be grouped together?

10. What other forms of data compression (such as grouping connections by source network) improve performance?

11. What are the predictive capabilities of an offline network intrusion detection system?

12. How much will the incorporation of classifiers using different data sources, such as alerts from real-time IDSs, syslog, or system call data, improve the performance?

13. Do we improve accuracy by not looking at the connections themselves, but instead looking at the cumulative state of a host or group of hosts, where each connection acts as a state transition operator?

14. Does the ideal time window, w, depend on the current state of a host (For example, w decreases when the host is subjected to a DoS attack)?

There are also some open questions which have a wider scope:

15. What similarities or differences exist in the traffic characteristics between different types of networks (commercial, residential, educational, or gov-

ernment), such as the rate at which they are attacked, that impact the performance characteristics of a network intrusion detector?

16. What is the user acceptability level of false alarms?

17. How much can false alarms be reduced through the use of user feedback, and learning algorithms or classifier retraining?

With these questions in mind, we present the following proposal for further work in this area.

2.1 Goal

While all of the questions presented above are important, answering all of them would be far too time consuming for a single person in a reasonable time period. As such, the proposed goal of this research is to produce a dissertation that will provide the best possible answers to the first six questions listed above. These answers will be based on quantitative measurements of a system we will build. These six problems were picked because we believe they are tractable, and being able to answer these questions is a prerequisite to addressing most of the questions that follow. While answers to some of those other questions may affect specific details of our approach (such as the dataset or amount of data used), and hence our measurements, this can be addressed by iterative improvements to our system after the currently proposed research is finished.

For all of the questions, our primary performance metric is accuracy. We measure accuracy using Receiver Operating Characteristics (ROC) curves. Note that these are traditional ROC curves, not the reformulated Receiver Operating Curves used by (Lippmann et al. 2000) and criticized in (McHugh 2000) due to their arbitrary false-positive scaling. ROC curves give a succinct, human interpretable view of the accuracy of IDS systems. It is important to note that they only explain what the performance of a given system is – we rely on our own analysis to ascertain why that system performs as it does. We will be looking at six measures of accuracy: accuracy on each of the four categories of malicious activity (Probe, DoS, R2L, and U2R), accuracy on attacks not seen in the training data, and overall accuracy of the system. Each of these measurements will be based on test data not seen during training in accordance with standard practices in the data mining community.

To elaborate on the questions we will be addressing:

1. For baseline purposes, what is the accuracy of a modern, signature based network intrusion detector on the standard evaluation datasets?

   The impetus for research in data mining for network intrusion detection is the belief that signature based network IDSs have poor detection accuracy, however, as noted in (McHugh 2000), the performance of such systems on the DARPA dataset was never tested. The same is true of any other standard dataset used by the research community (such as the now

defunct IES). Such an evaluation is necessary in order to answer the most basic question: "Did a given technique perform better than contemporary systems?" We will address this question by generating the ROC curves described above for a given IDS on each dataset that we use.

2. What is the ideal number of states for a Hidden Markov Model of network connection data, and what parameters influence this value?

   While most of the techniques we propose to implement below automatically configure their number of states (such as decision trees), or have reasonable guidelines for their configuration (as is the case with neural networks), there has been very little research into the application of Hidden Markov Models (HMMs) for modeling network data, hence we will attempt to determine what an ideal number of states is, and what parameters influence that selection (for example, is it dependent on the number of hosts or services modeled).

3. What are the ideal feature sets for different data mining techniques?

   As shown in tables 1.1 and 1.2, numerous feature sets have been proposed for mining on connection data. Further, researchers have used a variety of data manipulation techniques (such as calendar schemas and pseudo-Bayes estimators) to improve classification. To date, no one has considered which of these (given the full set) allow a given technique to achieve the best performance. While such results may be dependent on the dataset and the ability to provide sufficient training data to the methods, they will remain instructive primarily to compare and contrast the different features used by different techniques and hopefully deduce a relation to classification performance. Additionally, the presence or absence of a given feature or technique should be indicative of its general utility for any future application.

4. Can the performance of some of the data mining techniques be improved by grouping related packets in connectionless protocols like UDP and ICMP, and treating them as a single connection (as in TCP)?

   While there are many ways that the data can be compressed (see question 10 above), in IP connection logs, the per-packet nature of UDP and ICMP can cause activity with these protocols to have a disproportionate number of records with respect to their actual traffic levels. We want to know if treating these protocols similar to TCP will improve performance. By treating them similarly, we mean that all packets between the same source and destination IPs using the same service or related ICMP types within the same time period (for example, not separated by more than five minutes) will be treated as a logical connection. Such a logical connection will have attributes such as number of packets, duration, and bytes transfered, which we believe will provide information gain and improve performance.

5. Should we use separate training sets for meta-classifiers and the classifiers they incorporate?

The goal of using an ensemble approach in data mining is to achieve better performance than can be achieved with an individual classifier. One of the drawbacks observed by Giacinto et al. was a loss of generalization. We suspect that this came either from a reinforcement effect of using the same type of classifier on different features, or from a failure to explicitly train the meta-classifier to generalize. The later should be possible by using one portion of the training data to train the classifiers and a second portion to train the meta-classifiers. Specifically, after all the classifiers are trained on the same portion of training data, they will be run on the second portion, and the meta-classifiers can be trained on their output. In addition to answering the basic question of whether such an approach improves accuracy, we will try different ratios of classifier to meta-classifier training data to see what ratio (if any) works best.

6. What is the accuracy of an offline network intrusion detection system employing numerous, different, data-mining techniques?

   This is probably the biggest question that we propose to answer. As with the individual techniques, we'll produce the ROC curves described above. These ROC curves can be compared to others, such as those generated in our baseline for question (1) above. This performance will then be qualitatively, and to a lesser extent quantitatively (using area under the ROC curves) with the individual techniques and our baseline to determine if this approach is, as we suspect, better than existing approaches, along with an analysis of what factors contributed to this performance difference.

2.2 Approach

In this section we'll outline the steps we'll follow in answering the questions presented in the previous section. In general, this will be accomplished by building a system that applies many different data mining techniques to network connection data and use a meta-classifier to combine the results. The end result will be a "threat score" for every connection representing the risk we believe that connection presents to the network. While different organizations may have different definitions of risk, we adopt the definition of risk as the likelihood that a connection is part of an attack as defined by a given labeled dataset.

Datasets We'll use multiple datasets to provide a higher degree of assurance that our measurements are correct. While we will adhere to current best practices, we retain the belief that these currently available data available for research purposes does not sufficiently model either real-world normal nor malicious traffic, and we encourage further work in this area. Should better datasets (or at least new datasets that correct some of the obvious flaws with existing ones) become available, we will incorporate them into this research.

For the first two datasets, we will use the 1998 and 1999 DARPA data. While they have obvious problems, they are a community standard that will allow a

qualitative comparison of our system with other work. Further, we believe that given the nature of the problems with that data, if a data mining approach can not perform well on that data, it doesn't stand a chance with much more chaotic real-world data. In other words, they provide a reasonable upper bound on system performance.

The third dataset will use the approach outlined in (Mahoney and Chan 2003a) by mixing the DARPA 1998 data with real world data collected from a personal webserver belonging to the author. The choice of this server will allow for publication of the data (after anonymisation of the client IP addresses), and eases analysis as it has a very clearly defined security policy and configuration.

Baseline As noted above, one of the problems in the DARPA IDS challenge cited by McHugh (2000) was the failure to baseline the dataset with performance of contemporary, deployed NID systems. Only once we have an objective baseline of existing systems can we evaluate how much better the advanced approaches are. Hence, our first task will be to create a detection baseline using Snort, a popular open-source, signature (misuse) based NID (Caswell and Roesch 2004).

We will do this by running the training portion of each dataset through Snort (utilizing its ability to directly analyze a tcpdump file). We will enable all rules designed to detect Probe, DoS, R2L, and U2R activity, and will not enable rules designed for purposes like detecting inappropriate use such as peer-to-peer file sharing. Each rule which generates an alarm will be scored as $1 - \frac{fp}{a}$ where fp is the number of false alarms (alarms not corresponding to a labeled attack), and a is the total number of alarms generated. We will begin by assuming that all rules which did not raise an alarm are good and have a score of 1. The testing portion of the data will then be run on the dataset with all rules enabled, and measurements of the the true and false positive count will be taken for each of the four types of malicious activity, previously unseen activity, and totals. Successively, the lowest scoring rule (or rules in the case of a tie) will be removed and the testing data rerun. Eventually we'll run the data using only rules with a score of 1. All of the true to false positive points can then be plotted out on a ROC curve for total performance, performance on new attacks, and performance on each of the four categories of malicious activity. The test can then be repeated eliminating all rules which did not generate alarms on the training data before the first run. This should more accurately model the reality of not being able to generate a signature for traffic that has not been seen before. At the same time, it ignores the probability that others have seen it and quickly make a signature available to the Snort community. We will defer this discussion until the baseline is complete and we have performance metrics to base it on.

Connection mining In order to apply our data mining methods to network connection logs, we need to derive connection records from our datasets. This is what the KDD dataset has already done for the 1998 DARPA data. Of course,

since we are using more than just the 1998 DARPA data for our analysis, this is insufficient, and since we need to process connection records from the other datasets, we should reprocess the 1998 DARPA data to ensure orthogonality. This task will be accomplished by extending an existing tool such as tcp-reduce (Paxson 1995), BRO (Paxson 2004), or IPSumDump (Kohler 2004).

Table creation In order to efficiently manage our data and results, we'll need a couple relational database tables. We'll begin with a table to store the connections that will contain the fields shown in table 2.2. The features that we've selected for this table represent the union of features cited as used by other researchers as summarized in tables 1.1 and 1.2, with the addition of (Li et al.)'s calendar schema and the standard deviation for calculated features using mean values. While we haven't seen the standard deviation used as a calculated feature before, we imagine that some types of malicious activity would retain the same mean value (perhaps deliberately to avoid detection), however it's true usage might be more tightly grouped or more spread than normal. We would however caution that the use of a standard deviation is most appropriate for features that follow a Poisson distribution, and many network features have been shown to be long-tailed, hence the use of a standard deviation here is only to provide a rough approximation of the diversity of activity.

We also modified wrong fragment flag and urgent packet counts to be percentages of a connection. This was done for two reasons: First, many data mining methods prefer normalized data and normalizing the count with respect to the total number of packets in each connection makes more sense than normalizing it with respect to other counts which may or may not be related. Second, a tiny percentage of packets that are urgent or poorly fragmented are understandable in a long running connection is easily attributable to corruption or implementation problems, however if the same small number of packets make up half the packets in a connection, most likely something suspicious (such as IDS evasion) is occurring.

Once the table is created, we can load in the connection summaries and manually add the training label and class information as provided with the dataset. Once the primary and secondary attributes have been loaded into the table, we'll create a program that fills in the calculated features. We should be able to leverage the features of the database system to ease this step. We will most likely only keep the table populated with one of the given datasets at any one time for space and simplicity reasons.

Next, we'll build a table to store our intermediate and final results, a connection threat score table with the fields shown in table 2.3. This table uses the same connection number as we do on the connection records table, such that the tables could be joined together on it. While we could have modeled this as a single table, for manageability, especially given the large number of fields in each, we chose to split it. The table then contains a assessment of the inverse probability $(1 - Pr(x))$ for each intrinsic attribute that will be assigned by a Bayesian network. The two other anomaly detection methods that we'll employ

Connection number	
Essential intrinsic attributes	Timestamp
	Month
	Day of week
	Day of month
	Hour of day
	Minute of hour
	Source IP
	Destination IP
	Source port
	Destination port
	Protocol
Secondary intrinsic attributes	Duration
	Source bytes
	Destination bytes
	TCP Flags
	Land packet
	% wrong frag
	% urgent
	% Resent
	% Wrong resent
	% Duplicate ACK
	% Hole
	% wrong data packet size
	% data packets
Calculated attributes (for 3 second, 120 second, and 24 hour windows)	# total connections (TCP and UDP)
	# SYN flags
	# establishment errors (SYN without connection establishment)
	# other errors (connection generates ICMP error or closes without FIN)
	# RST flags
	# FIN flags
	# to privileged services
	# to unprivileged services
	# to the same service
	# to the same host
	# different services accessed
	# packets to all services
	# ICMP packets
	# unique keys (source, destination, and destination port)
	# new keys
	# keys with outside hosts
	mean packet count to same key
	standard deviation of packet count to same key
	mean duration all services
	standard deviation duration all services
	mean duration current service
	standard deviation duration current service
	mean duration current host
	standard deviation duration current host
	mean bytes transfered all services
	standard deviation bytes transfered all services
	mean bytes transfered current service
	standard deviation bytes transfered current service
	mean bytes transfered current host
	standard deviation bytes transfered current host
	% on same host to same service
	% of same service to same host
Training label	
Training class	

Table 2.2: Attributes to be included in table of connections.

will each assign a single anomaly (threat) score between zero (normal) and one (highly anomalous) for each record. Each of our seven classification methods will assign a threat score (again between zero and one) for each category of malicious activity.

The "total anomaly score" field will be assigned a threat (or anomaly) value by the meta-classifiers based on the values of the inverse probabilities and two anomaly scores. Likewise, the meta-classifiers will assign a total threat score for each of the categories of malicious activity based on the seven threat scores for each of those categories assigned by the individual methods. Each meta-classifier will also assign a total threat score based on the 16 inverse probability, two anomaly, and 28 individual categorical scores. Finally, each meta-classifier will assign a total threat score based on the total anomaly and four total categorical scores. For comparison purposes, we will also calculate the max of those five scores to see if there are any instances where a categorical score reports that a connection is definitely threat (for example, the total R2L score is one), and the total threat score based on the categorical scores is less.

Base method creation The bulk of time on this project will be spent implementing the base classifiers, running them with our datasets, and analyzing the results. The use of good base classifiers is important because ensemble techniques allow us to leverage the strengths of individual classifiers, however if those base classifiers have no strengths, our meta-classifier becomes worthless. We'll prefer to use existing code and systems to ease implementation, and failing the existence of suitable pre-existing systems, we'll build one. Each method will run in one of two modes:

Training mode To read specified data (range of connection numbers) from the connection table and either learn normal behavior for the system (for anomaly detectors), or learn the appropriate classification of connections.

Classification mode To read specified data (range of connection numbers) from the connection table and fill in its columns in the threat score table. A method will have one anomaly column to fill out if it's an anomaly detector, where the value will be the certainty (real value between zero and one) that a connection is anomalous (many methods will just output zero or one). Classification based methods will have four columns to fill out: a probe threat score, denial of service threat score, remote to local threat score, and a user to root threat score. The method will assign a score between zero and one for the likelihood that a given connection belongs to one of these four categories of malicious activity. Most classification methods will only classify a connection to be in a single category, in which case all of the columns will be zero (for normal connections) or only one column will have a non-zero score (for malicious activity). Like the anomaly detection methods, many classification methods will only assign scores of zero or one.

Connection number
Inverse probability of Source IP
Inverse probability of Destination IP
Inverse probability of Source port
Inverse probability of Destination port
Inverse probability of Duration
Inverse probability of Source bytes
Inverse probability of Destination bytes
Inverse probability of TCP Flags
Inverse probability of % wrong frag
Inverse probability of % urgent
Inverse probability of % Resent
Inverse probability of % Wrong resent
Inverse probability of % Duplicate ACK
Inverse probability of % Hole
Inverse probability of % wrong data packet size
Inverse probability of % data packets
Anomaly score based on match with non-self bit-vector
Anomaly score based on HMM
Probe threat score from Decision tree
DoS threat score from Decision tree
R2L threat score from Decision tree
U2R threat score from Decision tree
Probe threat score from associative and frequency episode rules
DoS threat score from associative and frequency episode rules
R2L threat score from associative and frequency episode rules
U2R threat score from associative and frequency episode rules
Probe threat score from feed-forward ANN
DoS threat score from feed-forward ANN
R2L threat score from feed-forward ANN
U2R threat score from feed-forward ANN
Probe threat score from Elman network
DoS threat score from Elman network
R2L threat score from Elman network
U2R threat score from Elman network
Probe threat score from genetic algorithm trained rules
DoS threat score from genetic algorithm trained rules
R2L threat score from genetic algorithm trained rules
U2R threat score from genetic algorithm trained rules
Probe threat score from clustering algorithm
DoS threat score from clustering algorithm
R2L threat score from clustering algorithm
U2R threat score from clustering algorithm
Probe threat score from SVM
DoS threat score from SVM
R2L threat score from SVM
U2R threat score from SVM
Total anomaly score
Total Probe threat score
Total DoS threat score
Total R2L threat score
Total U2R threat score
Total threat score from all individual scores
Total threat score from categorical scores
Max of categorical scores

Table 2.3: Fields to be included in results table.

For each base classifier, we will build it, run it on each of our datasets, determine the base performance, and ensure proper operation. We will return to analysis of the individual classifiers when we're ready to determine the ideal feature set for each one.

The following methods will be implemented:

- Anomaly detection methods
  - Bayes network of probability of each intrinsic attribute
  - Matching against non-self bit-vectors
  - Hidden Markov Model (HMM)
- Classification methods
  - Decision tree
  - Associative rule with level-wise mining and frequent episode rule with relative frequency support
  - Pure feed-forward neural network
  - Elman network (or other neural network with context nodes)
  - Attribute matching rules trained with genetic algorithm
  - Clustering algorithm
  - Support Vector Machine (SVM)

For classification based methods which are binary classifiers, we will instantiate four copies of the classifier: one for each category of malicious data.

Ideal parameter determination For the Hidden Markov Model in particular, we need to determine the ideal number of states to use to model network connection data for anomaly detection purposes. This is something of a chicken and egg problem with the ideal feature set determination discussed next. The solution will likely require human intensive analysis starting with a simple (one state) HMM, examining its behavior and slowly growing the model attempting to ascertain which features are being leveraged and looking at performance as the number of states grows, attempting to ascertain where the global maximum (for performance) occurs. We want to monitor the features being used to ensure we have an understanding of why it's operating as it is, even if the details of exact probability values appear as a black box to us. Without this, the determination of the proper number of states may come from an artifact of our datasets, or something else equally invalid.

Ideal feature set determination We now wish to determine the ideal feature set for each of our base methods. We suspect that the different techniques, with their different strengths and weaknesses, will use different feature sets for optimal performance. Hence, we will not attempt to find an ideal feature set that works for all methods simultaneously. In a production environment, the

connections table in the database would include any feature in the ideal feature set of any method, and each method would be presented with a view to just its ideal feature set.

The ideal feature set determination presents a combinatorial problem that comes from the unknown interactions between different techniques that we wish to investigate. For example, a given method may perform best with pseudo-Bayes estimators if non-TCP connections are compressed, however performance with pseudo-Bayes estimators may be lower if non-TCP connections are not compressed, as such, all four combinations must be attempted. With all four combinations of pseudo-Bayes estimators and compressed connections, we should find the ideal feature set for each of the base classification methods.

The actual determination will be done by training each of the methods on 80% of the training data and testing their performance on the remaining 20% of the training data. This should be done on the full dataset, then repeated with each feature removed one at a time. The six ROC curves described above will be generated on this 20% validation set, and if the removal of any given feature improves performance for any of those six accuracy measures (determined by the area under the given ROC curve), then we remove the feature whose absence accounts for the best performance and repeat until removing additional features no longer improves performance. If the removal of a given feature results in better performance in one area, and worse performance in another, for example if it improves detection of previously unseen attacks, but decreases performance in detecting remote to local (R2L) attacks, then we'll copy the classifier and continue. If in the end we find that different feature sets result in significantly different performance in detecting different activity (by significant, we mean greater variations than can be attributed to noise), then we will maintain multiple instances of the classifier, and expand the threat score table to accept threat scores from all of them.

Base classifier analysis Once the ideal feature set for each method has been determined, we can perform a complete training and testing run on each base classifier using each of our datasets. For each classifier we'll generate the six ROC curves described above, attempt to ascertain why it performed as it did, and why it used the features that it did. We will also consider if its operation is based on metrics that we expect will hold on real-world data. In particular, we'll examine if the artificial nature of the DARPA datasets are skewing the results, or if the mixed dataset is being separated into its component parts. If this is the case, we'll consider what actions can be taken to correct the problem. If the solution is likely one that would be used in a real-world environment, we'll make the change and repeat the experiment. If the solution is only really applicable to our research environment, we'll likely continue with two iterations of the classifier and/or dataset: a pure one, and one that we believe to give more realistic results (for better or worse). Experiments with the meta-classifiers described below will then each be performed twice: once on the pure set and once on the modified set. If the previous step resulted in multiple instances of

any classifier to detect different types of activity, we will attempt to ascertain why the classifier performs differently on different feature sets, and why a single instance is not able to achieve ideal performance for all types of activity.

Training information population While the base classifiers can be trained entirely on the label information in the connections table, when we train the meta-classifiers, we'll want the threat score table populated with the expected results. To do this, we'll build a script that will fill in the total anomaly, probe, DoS, R2L, U2R, and total threat score for specified connections using the training class in the connection table. All malicious connections (regardless of attack class) will have the total anomaly, total threat score, and the appropriate threat class score to one and all other columns to zero. All normal connections will have all scores set to zero.

Meta-classifier creation Next we'll build the various meta-classifiers. Each method will be instantiated in seven different ways:

1. Total anomaly from individual anomaly scores
2. Total probe from individual probe scores
3. Total DoS from individual DoS scores
4. Total R2L from individual R2L scores
5. Total U2L from individual U2L scores
6. Total threat from all individual scores
7. Total threat from total anomaly, probe, DoS, R2L, and U2R scores

Each of the instantiations will run in one of two modes:

1. Training mode to read specified data (both range of connection numbers and proper columns for given instantiation) from the threat score table and either learn the proper settings for the appropriate threat score column for the given instantiation.
2. Classification mode to read specified data (range of connection numbers and proper columns for given instantiation) from the threat score table and fill in the appropriate threat score column for the given instantiation.

We will use the following meta-classifiers:

- Naive-Bayes
- Decision tree
- Associative rules
- Pure feed-forward neural network

- Rules trained with genetic algorithm
- Support Vector Machine (SVM)

Note that except for the Naive-Bayes meta-classifier, the rest can be built from methods that are already being built for the base classification.

As with the base classifiers, we'll perform a basic run with each instantiation of each meta-classifier to ensure that it works before moving on to the next step.

Ideal meta-classifier training Once we have the meta-classifiers, we want to determine what the best method to train them is. In particular, will they perform better, particularly for previously unseen attacks, if the meta-classifiers are trained on different data than the base classifiers? To determine this, we'll take the training portion of each dataset and perform an 80/20 split: the same 80% will be used for training in all experiments, and the same 20% will be used for testing in all experiments. We do this instead of using the testing data as we want to reserve the testing data for use in the next step, and do not want foreknowledge of the data to influence decisions as to which approach works best, as such foreknowledge is not, by definition, available in real environments.

We will run three experiments for the five categorical instances (including total anomaly score) of each meta-classifier for each dataset, for a total of 54 experiments. The first experiment will train the base classifiers on the full 80% set, then have the base classifiers populate threat scores for all the connections (the 80% they just trained on, plus the 20% validation set). The meta-classifiers will then be trained on the same 80% set, and then run on the 20% validation portion. The threat scores in the validation portion can then be used to create ROC curves for each of the four categories of malicious activity, attacks not seen in the 80% set, and three ROC curves for total malicious activity: one based on the meta-classifier instance that looks at all the individual threat scores, one based on the instance that looks at just the categorical scores, and one from the numeric maximum of the categorical scores. This experiment will then be repeated using just 64% of the data (80% of the 80%) to train the base classifiers. This time, only 16% of the data will be used to train the meta-classifiers, so only the base-classifiers only need to populate the threat score table for the remaining 36% of connections. The meta-classifiers will then be trained on the 16% of the data and validated on the same 20% of the data used above, generating the same ROC curves. Finally, we'll repeat the experiment using just 40% to train the base classifiers, 40% to train the meta-classifiers, and validate and generate ROC curves on the same 20% as before.

In analyzing the above data, we would like to find that the ideal performance on detecting each category of attacks was obtained by the various instances of a single meta-classifier on a particular ratio of base and meta-classifier training data. It is entirely likely however that instances of different meta-classifiers and training ratios will prove ideal for a particular detection goal. At this point, we will determine which instance of which meta-classifiers and training ratio is ideal for each of the five categories of attacks. Given this, we can repeat the test

above using the best performing meta-classifier instance for each category with its ideal training ratio, except after the categorical meta-classifier is trained on it's portion of training data, it will then be run on the same data (we believe that splitting this portion of the data further will result in too little data to be effective, however there is no research to support this assumption). Additionally, the categorical meta-classifiers will be run on the 20% validation set. We then train the two instances of each meta-classifier that assign a total threat score on the meta-classifier portion of training data, and run the two instances of each on the validation data.

We can now generate ROC curves for the overall performance of each meta-classifier based on the total threat scores, and a third ROC curve representing the max of the categorical scores. We will examine which of the three approaches works better in general, and which performs best overall.

Ideal performance testing By this point we should know the ideal feature sets and parameters for the base classifiers, as well as the best performing instances of meta-classifier and their base/meta-classifier training data mix. We will use all this information to train the system of base and meta-classifiers on the complete set of training data for each dataset. Then we can run the system on the respective test data and develop a final evaluation of the performance of an ensemble based data mining network intrusion detection system.

2.3 Completion

Once the above steps are completed, we should have gained the following knowledge:

1. We should have a baseline of what a contemporary signature based NID is capable of to compare our system against.

2. We should know what the ideal number of states is in a Hidden Markov Model used to model network connection data for anomaly detection purposes is.

3. We should know what the ideal feature set(s) are for each of our data mining techniques and how they influence different categories of detection. Additionally, we know for which (if any) methods pseudo-Bayes estimators improve performance.

4. We know for which (if any) methods compressing connectionless protocols into single connections improves performance.

5. We can ascertain if we achieved better overall performance from the meta-classifier trained on the same data as the classifiers, or the one trained on a different set of training data, and if so, an approximation of what this ratio should be.

6. Given our best meta-classification performance, we should now know what level of performance (detection rate versus false positive rate) we can expect from a data mining ensemble-based offline network intrusion detection system, as well as how this compares against our baseline.

The final deliverable for this project will be a dissertation which presents those answers, along with all the intermediate findings we made in our analysis during the research. This dissertation should be complete by 1 May 2006, as outlined in table 2.4.

Step	Finish date
Begin real traffic collection	1 July 2004
End real traffic collection	1 October 2004
Mixed dataset generated	7 October 2004
Baseline	1 November 2004
Connection mining	15 October 2004
Table creation	15 August 2004
Bayes network	15 November 2004
Non-self bit-vectors	1 December 2004
HMM	15 December 2004
Decision tree	1 January 2005
Associative rules	15 January 2005
Feed-forward neural network	1 February 2005
Elman network	15 February 2005
Genetic rules	1 March 2005
Clustering	15 March 2005
SVM	1 April 2005
HMM parameter estimation	15 April 2005
Ideal feature set determination	1 July 2005
Base classifier analysis	15 August 2005
Training information population	1 September 2005
Naive-Bayes meta-classifier	15 September 2005
Decision tree meta-classifier	22 September 2005
Associative rules meta-classifier	1 October 2005
Neural network meta-classifier	7 October 2005
Genetic rules meta-classifier	15 October 2005
SVM Meta-classifier	22 October 2005
Ideal meta-classifier training	15 December 2005
Ideal performance testing	1 February 2006
Dissertation	1 May 2006

Table 2.4: Timeline for dissertation completion.

Bibliography

Axelsson, S. (2000a, August). The base-rate fallacy and the difficulty of intrusion detection. *ACM Trans. Information and System Security 3*(3), 186–205.

Axelsson, S. (2000b). A preliminary attempt to apply detection and estimation theory to intrusion detection. Technical Report 00-4, Chalmers Univ. of Technology, Göteborg, Sweden.

Barbará, D., N. Wu, and S. Jajodia (2001). Detecting novel network intrusions using bayes estimators. In *Proc. of the First SIAM Int. Conf. on Data Mining (SDM 2001)*, Chicago. Society for Industrial and Applied Mathematics (SIAM).

Bass, T. (2000, April). Intrusion detection systems and multisensor data fusion. *Communications of the ACM 43*(4), 99–105.

Berkhin, P. (2002). Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA.

Bishop, M. (2001, 1 March). ECS-253: Data security and cryptography. Class Lecture.

Bloedorn, E., A. D. Christiansen, W. Hill, C. Skorupka, L. M. Talbot, and J. Tivel (2001, August). Data mining for network intrusion detection: How to get started. `http://citeseer.nj.nec.com/523955.html`.

Bykova, M. and S. Ostermann (2002). Statistical analysis of malformed packets and their origins in the modern Internet. In *Proc. of the 2nd ACM SIGCOMM Workshop on Internet measurment*, Marseille, France, pp. 83–88. ACM Press.

Bykova, M., S. Ostermann, and B. Tjaden (2001, 18–20March). Detecting network intrusions via a statistical analysis of network packet characteristics. In *Proc. of the 33rd Southeastern Symp. on System Theory (SSST 2001)*, Athens, OH. IEEE.

Carbone, P. L. (1997). Data mining or knowledge discovery in databases: An overview. In *Data Management Handbook*. New York: Auerbach Publications.

Caswell, B. and M. Roesch (2004, 16 May). Snort: The open source network intrusion detection system. `http://www.snort.org/`.

CERIAS (2003). Intrusion detection systems. http://www.cerias.purdue.edu/coast/intrusion-detection/ids.html.

Chan, P. K., M. V. Mahoney, and M. H. Arshad (2003). *Managing Cyber Threats: Issues, Approaches and Challenges*, Chapter Learning Rules and Clusters for Anomaly Detection in Network Traffic. Kluwer. To appear.

Chan, P. K. and S. J. Stolfo (1993). Toward parallel and distributed learning by meta-learning. In *Working Notes AAAI Work. Knowledge Discovery in Databases*, Portland, OR, pp. 227–240. AAAI Press.

Chittur, A. (2001). Model generation for an intrusion detection system using genetic algorithms. High School Honors Thesis, Ossining High School. In cooperation with Columbia Univ.

Cisco (2003). Cisco intrusion detection. http://www.cisco.com/warp/public/cc/pd/sqsw/sqidsz/index.shtml.

Cohen, W. W. (1995, 9–12 July). Fast effective rule induction. In A. Prieditis and S. Russell (Eds.), *Proc. of the 12th International Conference on Machine Learning*, Tahoe City, CA, pp. 115–123. Morgan Kaufmann.

Computer Associates International (2003). eTrust Intrusion Detection Log View. http://www.cai.com/solutions/enterprise/etrust/intrusion_detection/prod%uct_info/sw3_log_view.htm.

Counterpane Internet Security, Inc. (2003). Counterpane Internet Security homepage. http://www.counterpane.com.

Crosbie, M. and E. H. Spafford (1995, 15 February). Active defense of a computer system using autonomous agents. Technical Report CSD-TR-95-008, Purdue Univ., West Lafayette, IN.

Das, K. J. (2000, 22 May). Attack development for intrusion detection evaluation. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA.

Dasgupta, D. and F. González (2002, June). An immunity-based technique to characterize intrusions in computer networks. *IEEE Trans. Evol. Comput. 6*(3), 1081–1088.

Dasgupta, D. and F. A. González (2001, 21–23 May). An intelligent decision support system for intrusion detection and response. In *Proc. of International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security (MMM-ACNS)*, St.Petersburg. Springer-Verlag.

Denning, D. E. (1987, February). An intrusion-detection model. *IEEE Transactions on Software Engineering 13*(2), 222–232.

Dickerson, J. E. and J. A. Dickerson (2000, July). Fuzzy network profiling for intrusion detection. In *Proc. of NAFIPS 19th International Conference of the North American Fuzzy Information Processing Society*, Atlanta, pp. 301–306. North American Fuzzy Information Processing Society (NAFIPS).

Dickerson, J. E., J. Juslin, O. Koukousoula, and J. A. Dickerson (2001, July). Fuzzy intrusion detection. In *IFSA World Congress and 20th North American Fuzzy Information Processing Society (NAFIPS) International Conf.*, Volume 3, Vancouver, Canada, pp. 1506–1510. North American Fuzzy Information Processing Society (NAFIPS).

Didaci, L., G. Giacinto, and F. Roli (2002). Ensemble learning for intrusion detection in computer networks. `http://citeseer.nj.nec.com/533620.html`.

DShield (2004). DShield - Distributed Intrusion Detection System. `http://www.dshield.org/`.

Endler, D. (1998). Intrusion detection applying machine learning to Solaris audit data. In *Proc. of the 1998 Annual Computer Security Applications Conference (ACSAC'98)*, Scottsdale, AZ, pp. 268–279. IEEE Computer Society Press.

Eskin, E. (2000a). Anomaly detection over noisy data using learned probability distributions. In *Proc. 17th International Conf. on Machine Learning*, San Francisco, pp. 255–262. Morgan Kaufmann.

Eskin, E. (2000b, 29 April–4 May). Detecting errors within a corpus using anomaly detection. In *Proc. of 2000 North American Chapter of the Association of Computational Linguistics (NAACL-2000)*, Seattle. North American Chapter of the Association of Computational Linguistics (NAACL).

Eskin, E., A. Arnold, M. Preraua, L. Portnoy, and S. J. Stolfo (2002, May). A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In D. Barbar and S. Jajodia (Eds.), *Data Mining for Security Applications*. Boston: Kluwer Academic Publishers.

Eskin, E., M. Miller, Z.-D. Zhong, G. Yi, W.-A. Lee, and S. J. Stolfo (2000, November). Adaptive model generation for intrusion detection systems. In *Workshop on Intrusion Detection and Prevention, 7th ACM Conference on Computer Security*, Athens. ACM.

Fan, W. (2001). *Cost-Sensitive, Scalable and Adaptive Learning Using Ensemble-based Methods*. Ph. D. thesis, Columbia Univ.

Fan, W., W. Lee, S. J. Stolfo, and M. Miller (2000, 31 May–2 June). A multiple model cost-sensitive approach for intrusion detection. In R. L. de Mántaras and E. Plaza (Eds.), *Proc. of Machine Learning: ECML 2000, 11th European Conference on Machine Learning*, Volume 1810 of *Lecture Notes in Computer Science*, Barcelona, Spain, pp. 142–153. Springer.

Fayyad, U. M., G. Piatetsky-Shapiro, and P. Smyth (1996, November). The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM 39*(11), 27–34.

Forrest, S., S. A. Hofmeyr, and A. Somayaji (1997, October). Computer immunology. *Communications of the ACM 40*(10), 88–96.

Forrest, S., S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff (1996). A sense of self for Unix processes. In *Proc. of the 1996 IEEE Symp. on Research in Security and Privacy*, Oakland, CA, pp. 120–128. IEEE Computer Society Press.

Frank, J. (1994). Artificial intelligence and intrusion detection: Current and future directions. In *Proc. of the 17th National Computer Security Conference*, Baltimore, MD. National Institute of Standards and Technology (NIST).

Ghosh, A. K., A. Schwartzbard, and M. Schatz (1999, 9–12 April). Learning program behavior profiles for intrusion detection. In *Proc. 1st USENIX Workshop on Intrusion Detection and Network Monitoring*, Santa Clara, CA, pp. 51–62. USENIX.

Giacinto, G. and F. Roli (2002, 11–15 August). Intrusion detection in computer networks by multiple classifier systems. In *Proc. of the 16th International Conference on Pattern Recognition (ICPR)*, Volume 2, Quebec City, Canada, pp. 390–393. IEEE press.

Helmer, G., J. Wong, V. Honavar, and L. Miller (1999, January). Automated discovery of concise predictive rules for intrusion detection. Technical Report 99-01, Iowa State Univ., Ames, IA.

Hofmeyr, S. A. and S. Forrest (1999). Immunizing computer networks: Getting all the machines in your network to fight the hacker disease. In *Proc. of the 1999 IEEE Symp. on Security and Privacy*, Oakland, CA. IEEE Computer Society Press.

Honig, A., A. Howard, E. Eskin, and S. J. Stolfo (2002, May). Adaptive model generation: An architecture for the deployment of data mining-based intrusion detection systems. In D. Barbar and S. Jajodia (Eds.), *Data Mining for Security Applications*. Boston: Kluwer Academic Publishers.

IES (2001). Information exploration shootout. `http://iris.cs.uml.edu:8080`.

Internet Security Systems (2003a). Managed Intrusion Protection Service. `http://www.iss.net/products_services/managed_services/service_intrusion%.php`.

Internet Security Systems (2003b). RealSecure SiteProtector. `http://www.iss.net/products_services/enterprise_protection/rssite_prote%ctor/siteprotector.php`.

Intrusion Detection Working Group (idwg) (2003). Intrusion Detection Exchange Format (IDXF). `http://ietf.org/html.charters/idwg-charter.html`.

ITA (2000). The internet traffic archive. `http://ita.ee.lbl.gov/`.

Javitz, H. S. and A. Valdes (1991, 20–22 May). The SRI IDES Statistical Anomaly Detector. In *Proc. 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, Oakland, CA. IEEE Computer Society.

Javitz, H. S. and A. Valdes (1993, March). The NIDES statistical component: Description and justification. Technical report, SRI International.

jtruitt@dw3f.ess.harris.com (1994, August). PRC's Information Security Officer's Assistant (ISOA). Posting to the IDS mailing list. `http://www.geek-girl.com/ids/1994/0028.html`.

Kendall, K. (1999, 21 May). A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA.

Kohler, E. (2004, 26 January). Ip summary dump. `http://www.icir.org/kohler/ipsumdump/`.

Korba, J. (2000, 22 May). Windows NT attacks for the evaluation of intrusion detection systems. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA.

Krügel, C., T. Toth, and E. Kirda (2002). Service specific anomaly detection for network intrusion detection. In *Proc. of the 2002 ACM Symp. on Applied Computing (SAC2002)*, Madrid, Spain, pp. 201–208. ACM Press.

Kumar, S. (1995). *Classification and Detection of Computer Intrusions*. Ph. D. thesis, Purdue Univ., West Lafayette, IN.

Lane, T. D. (2000, August). *Machine Learning Techniques for the computer security domain of anomaly detection*. Ph. D. thesis, Purdue Univ., West Lafayette, IN.

Lee, W. (1999). *A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems*. Ph. D. thesis, Columbia Univ.

Lee, W., R. A. Nimbalkar, K. K. Yee, S. B. Patil, P. H. Desai, T. T. Tran, and S. J. Stolfo (2000). A data mining and CIDF based approach for detecting novel and distributed intrusions. In H. Debar, L. Mé, and S. F. Wu (Eds.), *Proc. of Third International Workshop on Recent Advances in Intrusion Detection (RAID 2000)*, Volume 1907 of *Lecture Notes in Computer Science*, Toulouse, France, pp. 49–?? Springer.

Lee, W. and S. J. Stolfo (1998). Data mining approaches for intrusion detection. In *Proc. of the 7th USENIX Security Symp.*, San Antonio, TX. USENIX.

Lee, W. and S. J. Stolfo (2000). A framework for constructing features and models for intrusion detection systems. *Information and System Security 3*(4), 227–261.

Lee, W., S. J. Stolfo, P. K. Chan, E. Eskin, W. Fan, M. Miller, S. Hershkop, and J. Zhang (2001, June). Real time data mining-based intrusion detection. In *Proc. Second DARPA Information Survivability Conference and Exposition*, Anaheim, CA, pp. 85–100. IEEE Computer Society.

Lee, W., S. J. Stolfo, and K. W. Mok (1999a, 9–12 May). A data mining framework for building intrusion detection models. In *Proc. of the 1999*

IEEE Symp. on Security and Privacy, Oakland, CA, pp. 120–132. IEEE Computer Society Press.

Lee, W., S. J. Stolfo, and K. W. Mok (1999b, 15–18 August). Mining in a data-flow environment: Experience in network intrusion detection. In S. Chaudhuri and D. Madigan (Eds.), *Proc. of the Fifth International Conference on Knowledge Discovery and Data Mining (KDD-99)*, San Diego, CA, pp. 114–124. ACM.

Lee, W., S. J. Stolfo, and K. W. Mok (2000). Adaptive intrusion detection: A data mining approach. *Artificial Intelligence Review 14*(6), 533–567.

Lee, W., S. J. Stolfo, and K. W. Mok (2002). Algorithms for mining system audit data. In T. Y. Lin, Y. Y. Yao, and L. A. Zadeh (Eds.), *Data Mining, Rough Sets and Granular Computing*, Volume 95 of *Studies in Fuzziness and Soft Computing*, pp. 166–189. Heidelberg, Germany: Physica-Verlag.

Lee, W. and D. Xiang (2001, May). Information-theoretic measures for anomaly detection. In *Proc. of the 2001 IEEE Symp. on Security and Privacy*, Oakland, CA, pp. 130–143. IEEE Computer Society Press.

Li, Y., N. Wu, S. Jajodia, and X. S. Wang (2002). Enhancing profiles for anomaly detection using time granularities. to appear in Journal of Computer Security.

Lippmann, R. P. and R. K. Cunningham (1999, 7–9 September). Improving intrusion detection performance using keyword selection and neural networks. In *Proc. of the Conf. on Recent Advances in Intrusion Detection (RAID) '99*, West Lafayette, IN.

Lippmann, R. P., D. J. Fried, I. Graf, J. W. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. K. Cunningham, and M. Zissman (2000, January). Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. In *Proc. of the DARPA Information Survivability Conference and Exposition*, Los Alamitos, CA. IEEE Computer Society Press.

Lippmann, R. P., J. W. Haines, D. J. Fried, J. Korba, and K. J. Das (2000, October). The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks 34*, 579–??

Luo, J. (1999). Integrating fuzzy logic with data mining methods for intrusion detection. Master's thesis, Mississippi State Univ.

Mahoney, M. V. and P. K. Chan (2002). Learning nonstationary models of normal network traffic for detecting novel attacks. In *Proc. of the 8th ACM SIGKDD International Conf. on Knowledge Discovery and Data mining*, Edmonton, Alberta, Canada, pp. 376–385. ACM Press.

Mahoney, M. V. and P. K. Chan (2003a, 8–10 September). An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection. In G. Vigna, E. Jonsson, and C. Krügel (Eds.), *Proc. 6th Intl. Symp. on Recent Advances in Intrusion Detection (RAID 2003)*, Volume

2820 of *Lecture Notes in Computer Science*, Pittsburgh, PA, pp. 220–237. Springer.

Mahoney, M. V. and P. K. Chan (2003b, 19-22 December). Learning rules for anomaly detection of hostile network traffic. In *Proc. Third IEEE Intl. Conf. on Data Mining (ICDM)*, Melbourne, FL, pp. 601–604. IEEE Computer Society: IEEE Computer Society Press.

Mannila, H. (2002, July). Local and global methods in data mining: Basic techniques and open problems. In *ICALP 2002, 29th International Colloquium on Automata, Languages, and Programming*, Malaga, Spain. Springer-Verlag.

Marchette, D. (1999, 9–12 April). A statistical method for profiling network traffic. In *First USENIX Workshop on Intrusion Detection and Network Monitoring*, Santa Clara, CA, pp. 119–128. USENIX.

Marin, J. A., D. Ragsdale, and J. Surdu (2001, 12–14 June). A hybrid approach to profile creation and intrusion detection. In *Proc. of DARPA Information Survivability Conference and Exposition*, Anaheim, CA. IEEE Computer Society.

McClure, S. (1998). InfoWorld Security Suite 16 debuts. `http://www.idg.net/crd_detection_16738.html`.

McHugh, J. (2000). Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Trans. Information System Security 3*(4), 262–294.

Mukkamala, S. and A. H. Sung (2003). Identifying significant features for network forensic analysis using artificial intelligent techniques. *International Journal of Digital Evidence 1*(4), 1–17.

Mukkamala, S., A. H. Sung, and A. Abraham (2002). Identifying key variables for intrusion detection using soft computing. `http://citeseer.nj.nec.com/544845.html`.

myNetWatchman (2003). myNetWatchman - Intrusion Reporting and Response. `http://www.mynetwatchman.com/`.

Neri, F. (2000a, 16–19 July). Comparing local search with respect to genetic evolution to detect intrusion in computer networks. In *Proc. of the 2000 Congress on Evolutionary Computation CEC00*, La Jolla, CA, pp. 238–243. IEEE Press.

Neri, F. (2000b, 31 May–2 June). Mining TCP/IP traffic for network intrusion detection. In R. L. de Mántaras and E. Plaza (Eds.), *Proc. of Machine Learning: ECML 2000, 11th European Conference on Machine Learning*, Volume 1810 of *Lecture Notes in Computer Science*, Barcelona, Spain, pp. 313–322. Springer.

NetSecure Software (2003). NetSecure Log. `http://www.netsecuresoftware.com/netsecurenew/Products/NetSecure_Log/ne%tsecure_log.html`.

Neumann, P. G. and P. A. Porras (1999, April). Experience with EMERALD to date. In *First USENIX Workshop on Intrusion Detection and Network Monitoring*, Santa Clara, CA, pp. 73–80. USENIX. `http://www.sdl.sri.com/papers/det99/`.

NFR Security (2003). NFR Intrusion Management System. `http://www.nfr.net/products/`.

NIDES (2002). Next-generation Intrusion Detection Expert System (NIDES). `http://www.sdl.sri.com/projects/nides/`.

Ning, P., X. S. Wang, and S. Jajodia (2000). Modeling requests among cooperating intrusion detection systems. *Computer Communications 23*(17), 1702–1716.

nSecure Software (2002). nSecure nPatrol. `http://www.nsecure.net/features.htm`.

Paxson, V. (1995, 20 July). TCP-Reduce: Scripts for summarizing TCP connections. `http://ita.ee.lbl.gov/html/contrib/tcp-reduce.html`.

Paxson, V. (2004, 10 March). Bro: A system for detecting network intruders in real-time. `http://www-nrg.ee.lbl.gov/bro.html`.

Paxson, V., M. Allman, S. Dawson, W. Fenner, J. Griner, I. Heavens, K. Lahey, J. Semke, and B. Volz (1999, March). RFC 2525: Known TCP implentation problems. Status: INFORMATIONAL.

Porras, P. A. and P. G. Neumann (1996, December). EMERALD: conceptual overview statement. `http://www.sdl.sri.com/papers/emerald-position1/`.

Porras, P. A. and A. Valdes (1998, 11–13 March). Live traffic analysis of TCP/IP gateways. In *Proc. of the 1998 ISOC Symp. on Network and Distributed Systems Security (NDSS'98)*, San Diego. Internet Society.

Portnoy, L., E. Eskin, and S. J. Stolfo (2001, 5–8 November). Intrusion detection with unlabeled data using clustering. In *Proc. of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001)*, Philadelphia. ACM.

Prodromidis, A. L. and S. J. Stolfo (2001). Cost complexity-based pruning of ensemble classifiers. *Knowledge and Information Systems 3*(4), 449–469.

Ptacek, T. H. and T. N. Newsham (1998, January). Insertion, evasion and denial of service: Eluding network intrusion detection. Technical report, Secure Networks, Inc.

Rabiner, L. R. (1989, February). A tutorial on hidden markov models and selected applications in speech recognition. *Proc. of the IEEE 77*(2), 257–286.

Ryan, J., M.-J. Lin, and R. Miikkulainen (1998). Intrusion detection with neural networks. In M. I. Jordan, M. J. Kearns, and S. A. Solla (Eds.), *Advances in Neural Information Processing Systems*, Volume 10, Cambridge, MA. The MIT Press.

SANS (2004). Internet Storm Center. http://isc.incidents.org/.

SecurityFocus (2003). DeepSight Analyzer. http://aris.securityfocus.com/.

Sequeira, K. and M. Zaki (2002). Admit: Anomaly-based data mining for intrusions. In *Proc. of the 8th ACM SIGKDD International conf. on Knowledge Discovery and Data mining*, Edmonton, Alberta, Canada, pp. 386–395. ACM Press.

Sinclair, C., L. Pierce, and S. Matzner (1999, 6–10 December). An application of machine learning to network intrusion detection. In *Proc. 15th Annual Computer Security Applications Conference (ACSAC '99)*, Phoenix, pp. 371–377. IEEE Computer Society.

Singh, S. and S. Kandula (2001, May). Argus - a distributed network-intrusion detection system. Undergraduate Thesis, Indian Institute of Technology.

Snapp, S. R., J. Brentano, G. V. Dias, T. L. Goan, T. Grance, L. T. Heberlein, C. lin Ho, K. N. Levitt, B. Mukherjee, D. Mansur, K. L. Pon, and S. E. Smaha (1991, 25 February–1 March). A system for distributed intrusion detection. In *COMPCON Spring '91, Digest of Papers*, San Francisco, pp. 170–176. IEEE Computer Society.

Snapp, S. R., J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C. lin Ho, K. N. Levitt, B. Mukherjee, S. E. Smaha, T. Grance, D. M. Teal, and D. Mansur (1991). DIDS (Distributed Intrusion Detection System) - motivation, architecture, and an early prototype. In *Proc. of the 14th National Computer Security Conference*, Washington, DC, pp. 167–176. National Institute of Standards and Technology (NIST).

Staniford, S., J. A. Hoagland, and J. M. McAlerney (2002). Practical automated detection of stealthy portscans. *Journal of Computer Security 10*(1-2), 105–136.

Staniford-Chen, S., S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. N. Levitt, C. Wee, R. Yip, and D. Zerkle (1996). GrIDS – A graph-based intrusion detection system for large networks. In *Proc. of the 19th National Information Systems Security Conference*, Baltimore, MD. National Institute of Standards and Technology (NIST).

Symantec (2003a). Symantec Enterprise Solutions: Managed Security Services. http://enterprisesecurity.symantec.com/SecurityServices/content.cfm?art%icleid=682.

Symantec (2003b). Symantec Enterprise Solutions: ManHunt. http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=%156.

Tan, K. M. C. and R. A. Maxion (2002, 12–15 May). Why 6? defining the operational limits of stide, an anomaly-based intrusion detector. In *Proc. of the 2002 IEEE Symp. on Security and Privacy*, Oakland, CA, pp. 188–201. IEEE Computer Society Press.

Teoh, S.-T., K.-L. Ma, S. F. Wu, D. Massey, X.-L. Zhao, D. Pei, L. Wang, L. Zhang, and R. Bush (2003, 20–22October). Visual-based anomaly detection for BGP Origin AS Change (OASC) events. In M. Brunner and A. Keller (Eds.), *14th IEEE/IFIP Workshop on Distributed Systems: Operations and Management (DSOM'2003)*, Volume 2867 of *Lecture Notes in Computer Science*, Heidelberg, Germany. Springer.

Thuraisingham, M. M. and M. G. Ceruti (2000, 25–27 October). Understanding data mining and applying to to command, control, communications and intelligence environments. In *The 24th Annual International Computer Software and Applications Conf. (COMPSAC)*, Taipei, pp. 171–175. IEEE Computer Society.

Tölle, J. and O. Niggermann (2000). Supporting intrusion detection by graph clustering and graph drawing. In H. Debar, L. Mé, and S. F. Wu (Eds.), *Proc. of Third International Workshop on Recent Advances in Intrusion Detection (RAID 2000)*, Volume 1907 of *Lecture Notes in Computer Science*, Toulouse, France. Springer.

Tung, B. (1999). Common intrusion detection framework. `http://www.isi.edu/gost/cidf/`.

Waltz, E. and J. Llinas (1990). *Multisensor Data Fusion*. Norwood, MA: Artech House.

Warrender, C., S. Forrest, and B. A. Pearlmutter (1999). Detecting intrusions using system calls: Alternative data models. In *Proc. of the 1999 IEEE Symp. on Security and Privacy*, Oakland, CA, pp. 133–145. IEEE Computer Society Press.

Winkler, J. R. and W. J. Page (1990). Intrusion and anomaly detection in trusted systems. In *Fifth Annual Computer Security Applications Conf., 1989*, Tucson, AZ, pp. 39–45. IEEE.

Yeung, D.-Y. and C. Chow (2002, 11–15 August). Parzen-window network intrusion detectors. In *Proc. of the Sixteenth International Conference on Pattern Recognition*, Volume 4, Quebec City, Canada, pp. 385–388. IEEE Computer Society.

Acknowledgments

This paper is a personal work of the authors. It was not developed under the auspices of the United States Department of Energy, the University of California or Lawrence Livermore National Lab. Any opinions expressed herein are solely those of the authors, and may or may not reflect the views of their respective employers or the University of California, Davis. An earlier version of this paper was developed under the auspices of the United States Department of Energy by Lawrence Livermore National Lab under contract number W-7405-ENG-48.

The author gratefully acknowledge the assistance of Matt Bishop, Marcey Kelley, Ken Sumikawa, and Shaun Wakimoto, without whose assistance this paper would not have been possible.

Appendix: Data Mining techniques

The following table lists all the data mining techniques that have been employed for intrusion detection that have been used or should be applicable (as described in the text) for network intrusion detection.

Table 2.5: Summary of data mining techniques for network intrusion detection.

Cite	Model Function	Model Representation	Preference Criterion	Search Algorithm
(Javitz and Valdes 1991; Javitz and Valdes 1993; Porras and Valdes 1998)	Anomaly detection	Intensity, audit record distribution, categorical, and counting measures	Recent user or host activity	Linear algorithms based on exponential decay over time or number of records for each measure
(Mahoney and Chan 2002)	Anomaly detection	Conditional probabilities	Recent network activity	Linear algorithms
(Mahoney and Chan 2003b; Mahoney and Chan 2003a; Chan et al. 2003)	Anomaly detection	Rules with probabilities of consequence given antecedent	Minimal rule set with maximal coverage of normal traffic and no false positives	Rules from random pairs of records then pruning
(Frank 1994; Bloedorn et al. 2001)	Classification	Decision Trees	Minimize size of the tree[1]	Expectation-Maximization (EM)[1]
(Sinclair et al. 1999)	Classification	Decision Trees	Accuracy in classifying connections	ID3 algorithm
(Barbará et al. 2001)	Anomaly detection	Contingency table for Naive Bayes classifier	Weigh zero values with pseudo-Bayes estimator	Naive Bayes classifier algorithm
(Lee 1999; Lee and Stolfo 1998; Lee and Stolfo 2000; Lee et al. 1999a; Lee et al. 1999b; Lee et al. 2000; Lee et al. 2001)	Classification	Associative rule and Frequent episode generation	Level-wise mining, relative frequency support, axis and reference attributes	FAST Rule Induction
(Fan et al. 2000)	Classification	Ordered associative rules and Frequent episodes	Operational Cost	FAST Rule Induction

Table 2.5 continues on next page.

[1]While none of the authors specified the exact methods used, this is the typical approach.

Table 2.5: continued

Cite	Model Function	Model Representation	Preference Criterion	Search Algorithm
(Warrender et al. 1999)	Anomaly detection	Associative rules	Complete coverage of the training data	FAST Rule Induction
(Helmer et al. 1999)	Classification	Associative rule and Frequent episode generation	Level-wise mining, relative frequency support, axis and reference attributes[2]	FAST Rule Induction on reduced feature set produced by genetic algorithm
(Singh and Kandula 2001)	Classification	Associative rule and Frequent episode generation	Maximize items definitively classified by rule set	FAST Rule Induction with iterative grow and prune
(Neri 2000a)	Classification	Associative rules represented in a Horn-clause language	Accuracy of ruleset in classifying connections	Genetic algorithm
(Li et al. 2002)	Classification	Association rules	Exact correlation in time	Precise Match (PMA) Algorithm
(Li et al. 2002)	Classification	Association rules	Close time correlation	Fuzzy Match (FMA) Algorithm
(Dasgupta and González 2001)	Classification	Classifier on statistics of attributes	Similarity to knowledge base, generality, and diversity	Genetic algorithm
(Chittur 2001)	Classification	Decision tree	Detection rate minus false positive rate	Genetic algorithm
(Crosbie and Spafford 1995)	Anomaly detection	Parse trees	Minimize false positives based on human input in feedback loop	Genetic algorithm designed to operate on parse trees
(Dickerson and Dickerson 2000; Dickerson et al. 2001)	Classification	Fuzzy logic rules	"common sense experiences by the security administrator"	Created manually [3]
(Luo 1999)	Classification	Fuzzy logic rules	Classify all traffic without bias for any part of that traffic	Fuzzy association rules mining algorithm with normalization

Table 2.5 continues on next page.

[2]Not explicitly stated, but assumed due to the authors' intent to duplicate Lee's work with the addition of a reduced feature set.

[3]The fact that the model is built by hand by definition means that this approach does not qualify as machine learning, however it is included here as it has been shown to be a useful technique for intrusion detection, and there is the possibility that it can be automated. Additionally, even if it remains a manually built model, it will still be useful as part of an ensemble of classifiers.

Table 2.5: continued

Cite	Model Function	Model Representation	Preference Criterion	Search Algorithm
(Didaci et al. 2002; Giacinto and Roli 2002)	Classification	Pure feed-forward neural network	Accuracy – different learning rates	Backpropagation
(Ghosh et al. 1999)	Classification	Pure feed-forward neural network with leaky bucket	Ratio of detection rate to false positive rate	Backpropagation
(Ghosh et al. 1999)	Classification	Elman network with leaky bucket	Similarity of output (predicted) value to next value	Backpropagation[4]
(Ryan et al. 1998)	Anomaly detection	Pure feed-forward neural network	Correct user identification	Backpropagation
(Hofmeyr and Forrest 1999)	Anomaly detection	Collection of non-self bit vectors	Must not match self	Byte oriented hash and permutation
(Dasgupta and González 2002)	Anomaly detection	Collection of rules specifying non-self area	Must not match self and covers large area	Genetic algorithm
(Sinclair et al. 1999)	Classification	Rules with wildcards	Accuracy in classifying connections	Genetic algorithm
(Fan 2001; Lee et al. 2001)	Classification[5]	Associative rule and Frequent episode generation	Level-wise mining, relative frequency support, axis and reference attributes[6]	FAST Rule Induction with anomaly injection
(Eskin et al. 2002; Portnoy et al. 2001)	Anomaly detection	Outliers from clusters	Nearest cluster within some given distance	Fixed width clustering
(Eskin et al. 2002)	Anomaly detection	Outliers from clusters	Normal traffic clusters together	K-nearest neighbor

Table 2.5 continues on next page.

[4]Presumably a modified backpropagation algorithm that accounts for the cycles.
[5]"Anomaly" was actually one of the categories, in that respect this work did anomaly detection.
[6]Fan does not specifically mention these preference criterion in his work, however given his close association with Lee, we assume that he was building off the work in (Lee 1999; Lee and Stolfo 1998; Lee and Stolfo 2000; Lee et al. 1999a; Lee et al. 1999b; Lee et al. 2000).

Table 2.5: continued

Cite	Model Function	Model Representation	Preference Criterion	Search Algorithm
(Bloedorn et al. 2001)	Anomaly detection	Outliers from clusters	Euclidean distance between hyper-points[1]	K-means clustering
(Chan et al. 2003)	Anomaly detection	Outlying clusters with high or low density	Record belongs to all clusters within the average distance of closest neighbors	Fixed width clustering
(Marin et al. 2001)	Classification	Learning vector quantization (LVQ)	Bayes Optimal boundary	Initialized by K-means clustering then LVQ algorithm
(Staniford et al. 2002)	Portscan detection	Bayes network and connections between detected anomalous events	Ensemble of heuristics	Simulated annealing
(Marchette 1999)	Anomaly detection	Mixture model	Tight Clusters	Approximate distance clustering (ADC) and AKMDE
(Sequeira and Zaki 2002)	Anomaly detection	Minimum number of representative samples	Similarity from LCS	Dynamic Clustering
(Yeung and Chow 2002)	Anomaly detection	Parzen-window	Minimize distance between data points	Parzen-window algorithm
(Lane 2000)	Anomaly detection	Set of instances representing normal activity	Applied numerous similarity functions representing distance between instances	Instance-based learner
(Eskin et al. 2002; Honig et al. 2002)	Anomaly detection	Support vector machine (SVM)	Normal elements maximize distance from origin	SVM Algorithm modified for unsupervised learning
(Mukkamala et al. 2002; Mukkamala and Sung 2003)	Classification	Support vector machine (SVM) for each class	Features used in SVM picked by 10 rules based on accuracy, training time, and testing time	SVM Algorithm
(Lane 2000)	Anomaly detection	Hidden Markov model (HMM)	Approximation of true sequence likelihood	Baum-Welch algorithm

Table 2.5 continues on next page.

Table 2.5: continued

Cite	Model Function	Model Representation	Preference Criterion	Search Algorithm
(Warrender et al. 1999)	Anomaly detection	Hidden Markov model (HMM)	Likelihood of matching training data without overfitting	Baum-Welch algorithm
(Eskin 2000a; Eskin 2000b; Eskin et al. 2000)	Anomaly detection	Probabilistic model [7]	Number of appearances of element in training data	Expectation-Maximization (EM)
(Kumar 1995)	Misuse detection	Colored Petri nets	Correctness	Created manually [3]
(Staniford-Chen et al. 1996)	Misuse detection	Graphs defined by rule set	Rules that the analyst sees fit to detect undesirable activity	Created manually [3]
(Tölle and Niggermann 2000)	Classification	Mapping function from graphs to classification	Accuracy[1]	Regression[8]

[7]Eskin used a Mixture, although he notes that hidden Markov models, Markov chains, sparse Markov transducers, naive Bayes, or maximum entropy could also be used.
[8]The authors note that any supervised-learning method can be used.