Chapter 3

The MINDS - Minnesota Intrusion Detection System

Levent Ertöz[*], *Eric Eilertson*[*], *Aleksandar Lazarevic*[†], *Pang-Ning Tan*[*], *Vipin Kumar*[*†], *Jaideep Srivastava*[*†], *Paul Dokas*[*]
[*]Department of Computer Science, University of Minnesota
[†]Army High Performance Computing and Research Center (AHPCRC), Minnesota

Abstract:
This paper introduces the Minnesota Intrusion Detection System (MINDS), which uses a suite of data mining techniques to automatically detect attacks against computer networks and systems. While the long-term objective of MINDS is to address all aspects of intrusion detection, in this paper we focus on two specific contributions. First, we show how the behavior-based anomaly detection approach of MINDS is suitable for detecting new and previously unknown types of intrusions, which often indicate emerging threats. Specifically, we present an anomaly detection algorithm that assigns a score to each connection based on its probability of being an intrusion. Experimental results on live network traffic at the University of Minnesota show that our anomaly detection techniques are very promising and are successful in automatically detecting several novel intrusions that could not be identified using popular signature-based tools such as SNORT. Many of these have been reported on the CERT/CC list of recent advisories and incident notes. Second, we show how association pattern analysis can be used to summarize and characterize anomalous network connections. Given the very high vol-

ume of connections observed per unit time, such characterization of novel attacks is essential in enabling a security analyst to understand emerging threats. Experimental evaluation shows that the MINDS approach is useful in creating accurate summaries of attacks.
Keywords: network intrusion detection, rare class models, novel attacks, anomaly / outlier detection, association pattern analysis

3.1 Introduction

Traditional methods for intrusion detection are based on extensive knowledge of attack signatures that are provided by human experts. The signature database has to be manually revised for each new type of intrusion that is discovered. A significant limitation of signature-based methods is that they cannot detect novel attacks. In addition, once a new attack is discovered and its signature developed, often there is a substantial latency in its deployment. These limitations have led to an increasing interest in intrusion detection techniques based upon data mining [3, 4, 20, 23, 25], which generally fall into one of two categories: misuse detection and anomaly detection.

In misuse detection, each instance in a data set is labeled as 'normal' or 'intrusive' and a learning algorithm is trained over the labeled data. Research in misuse detection has focused mainly on detecting network intrusions using various classification algorithms [3, 7, 11, 20, 22, 23], rare class predictive models [13–16, 18], association rules [3, 20, 25] and cost sensitive modeling [10, 14]. Unlike signature-based intrusion detection systems, models of misuse are created automatically, and can be more sophisticated and precise than manually created signatures. In spite of the fact that misuse detection models have high degree of accuracy in detecting known attacks and their variations, their obvious drawback is the inability to detect attacks whose instances have not yet been observed. In addition, labeling dat ainstances as normal or intrusive may require enormous time for many human experts.

Anomaly detection algorithms build models of normal behavior and automatically detect any deviation from it [8, 12]. The major benefit of anomaly detection algorithms is their ability to potentially detect unforeseen attacks. In addition, they may be able to detect new or unusual, but non-intrusive, network behavior that is of interest to a network manager, that needs to be added to the normal profile. A major limitation of anomaly detection systems is a possible high false alarm rate. There are two major categories of anomaly detection techniques, namely supervised and unsupervised. In supervised anomaly detection, given a set of normal data to train on, and given a new set of test data, the goal is to determine whether the test data is 'normal' or anomalous. recently, there have been several efforts in designing supervised network-based anomaly detection algorithms, such as ADAM [3], PHAD [24], NIDES [2], and other techniques that use neural networks [27], information theoretic measures [21], network activity models [6], etc. Unlike supervised anomaly detection where the models are built only according to the normal behavior on the network, unsupervised anomaly detection attempts to detect anomalous behavior without using any knowledge about the training data. Unsupervised anomaly detection approaches are based on statistical approaches [?, 30, 31], clustering [9], outlier detection schemes [1, 5, 17, 26], state machines [28], etc.

This paper introduces the Minnesota Intrusion Detection System (MINDS), which uses a suite of data mining techniques to automatically detect attacks against computer networks and systems. While the long-term objective of MINDS is to address all aspects of intrusion detection, in this paper we present details of two specific contributions: (i) an anomaly detection technique that assigns a score to each network connection that reflects how anomalous the connection is, and (ii) an association pattern analysis based module that summarizes those network connections that are ranked highly anomalous by the anomaly detection module.

We also provide an evaluation of our anomaly detection and association summarization schemes in the context of real life network data at the University of Minnesota. In the absence of labels of network connections (normal vs. intrusive), we are unable to provide any estimate of detection rate, but nearly all connections that are ranked highly by our anomaly detection algorithms are found to be interesting and anomalous by the network security analyst on our team. In particular, during the past few months our techniques have been successful in automatically detecting several novel intrusions. In fact, many of these attacks have been reported on the CERT/CC (Computer Emergency Response Team/Coordination Center) list of recent advisories and incident notes. Finally, experiments on real network data demonstrate that association pattern analysis is successful in creating useful summaries of many novel attacks detected by our anomaly detection algorithms.

3.2 The MINDS project

The Minnesota Intrusion Detection System (MINDS) is a data mining based system for detecting network intrusions. Figure 3.1 illustrates the process of analyzing real network traffic data using the system. Input to MINDS is Netflow version 5 data collected using flow-tools [29]. Flow-tools only capture packet header information (i.e., it does not capture message content), and build one way sessions (flows). We are working with Netflow data instead of tcpdump because we curently do not have the capacity to collect and store the tcpdump. Netflow data for 10 minute windows, which typically results in 1–2 million flows, are stored in flat files. The analyst uses MINDS to analyze these 10-minute data files in a batch mode. The reason the system is running in a batch mode is not due to the time it takes to analyze these files, but it is convenient for the analyst to do so. Before data is fed into the anomaly detection module, a data filtering step is performed by the analyst to remove network traffic that the analyst is not interested in analyzing. For example, data filtered may include traffic from trusted sources or unusual/anomalous network behavior that is known to be intrusion free.

The first step in MINDS is extracting features that are used in the data mining analysis. Basic features include source and destination IP adresses, source and destination ports, protocol, flags, number of bytes and number of packets. Derived features include time-window and connection-windows based features. Tiem-window based features are constructed to capture conections with similar characteristics in the last T seconds. A similar approach was used for constructing features in KDD Cup'99 data [20]. Table

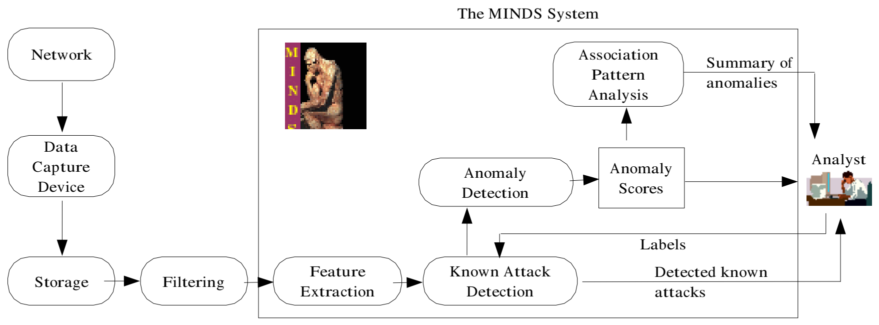


Figure 3.1: MINDS System

Table 3.1: Time-window based features

Feature name	**Feature description**
count-dest	Number of flows to unique destination IP addresses inside the network in the last T seconds from the same source
count-src	Number of flows from unique source IP addresses inside the network in the last T seconds to the same destination
count-serv-src	Number of flows from the source IP to the same destination port in the last T seconds
count-serv-dest	Number of flows to the destination IP address using same source port in the last T seconds

3.1 summarizes the time-windows based features.

"Slow" scanning activities, i.e., those that scan the hosts (or ports) and use a much larger time interval than a few seconds, e.g. one touch per minute or even one touch per hour, cannot be separated from the rest of the traffic using time-window based features. To do so, we also derive connection-window based features that capture similar characteristics of connections as time-window based features, but are computed using the last N connections originating from (arriving at) distinct sources (destinations). The connection-window based features are shown in Table 3.2.

After the feature construction step, the known attack detection module is used to detect network connections that correspond to attacks for which signatures are available, and then to remove them from further analysis. For results reported in this paper, this step is not performed.

Next, the data is fed into the MINDS anomaly detection module that uses an outlier detection algorithm to assign an anomaly score to each network connection. A human analyst then has to look at only the most anomalous connections to determine if they are actual attacks or other interesting behavior.

MINDS association pattern analysis module summarizes network connections that are ranked highly anomalous by the anomaly detection module. The analyst provides a feedback after analyzing the summaries created and decides whether these summaries are helpful in creating new rules that may be used in the known attack detection module.

Table 3.2: Connection-window based features

Feature name	**Feature description**
count-dest-conn	Number of flows to unique destination IP addresses inside the network in the last N flows from the same source
count-src-conn	Number of flows from unique source IP addresses inside the network in the last N flows to the same destination
count-serv-src-conn	Number of flows from the source IP to the same destination port in the last N flows
count-serv-dest-conn	Number of flows to the destination IP address using same source port in the last N flows

3.3 MINDS Anomaly Detection Module

In this section, we only present the density based outlier detection scheme used in our anomaly detection module. For more detailed overview of our research in anomaly detection, the reader is referred to [19].

MINDS anomaly detection module assigns a degree of being an outlier to each data point, which is called the local outlier factor (LOF) [5]. The outlier factor of a data point is local in the sense that it measures the degree of being an outlier with respect to its neighborhood. For each data example, the density of the neighborhood is first computed. The LOF of a specific data example p represents the average of the ratios of the density of the example p and the density of its neighbors. To illustrate the advantages of the LOF approach, consider a simple two-dimensional data set given in Figure 3.2. It is apparent that the density of cluster C_2 is significantly higher than the density of cluster C_1. Due to the low density of cluster C_1, for most examples q inside cluster C_1, the distance between the example q and its nearest neighbor is greater than the distance between the example p_2 and its nearest neighbor, which is from cluster C_2, and therefore example p_2 will not be considered as outlier.

Hence, the simple nearest neighbor approach based on computing the distances fail in these scenarios. However, the example p_1 may be detected as an outlier using the distances to the nearest neighbors. On the other hand, LOF is able to capture both outliers due to the fact that it considers the density around examples.

LOF requires the neighborhood around all data points be constructed. This involves calculating pairwise distances between all data points, which is an $O(n^2)$ process, which makes it computationally infeasible for millions of data points. To address this problem, we sample a training set from the data and compare all data points to this small set, which reduces the complexity to $O(n * m)$ where n is the size of the data and m is the size of the sample. Apart from achieving computational efficiency by sampling, anomalous network behavior will not be able to match enough examples in the sample to be called normal. This is because rare behavior will not be represented in the sample.

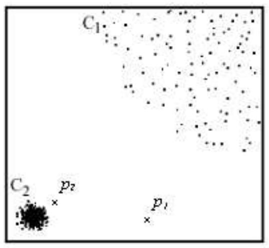


Figure 3.2: 2-D Outlier Example

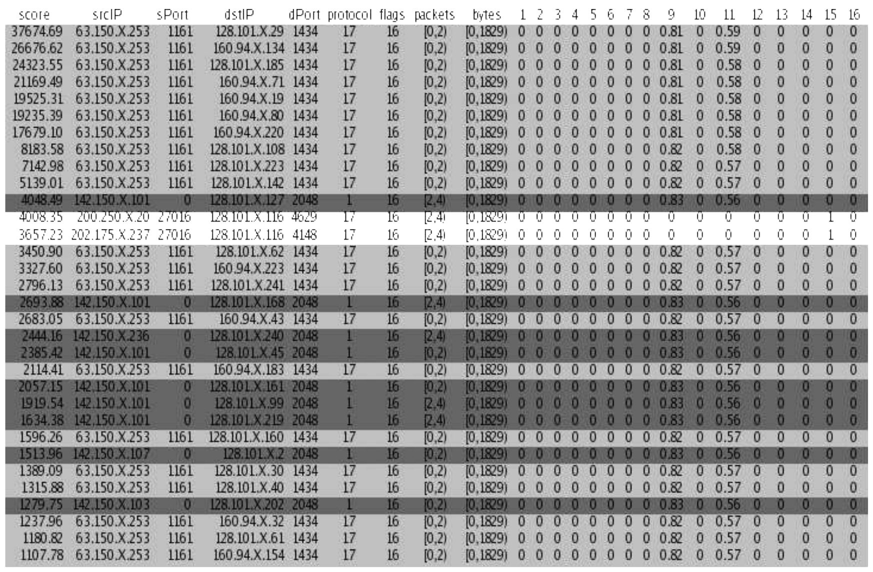
3.4 Evaluation of MINDS Anomaly Detection Results on Real Network Data

This section reports results of applying MINDS anomaly detection module on live network traffic at the University of Minnesota. When describing results on real network data, we are not able to report the detection rate and false alarm rate due to difficulty in obtaining the complete labelling of network connections.

Since a human analyst needs to manually evaluate outliers, it was not practical to investigate all of the outlier detection algorithms on the real network data. For this purpose we have selected the LOF approach, since it achieved the most successful results on publicly available DARPA '98 data set and it is more robust than other anomaly detection schemes that we used. The LOF technique also showed great promise in detecting novel intrusions on real network data.

Figure 3.3 shows the output of the system on January 27^{th} for a 10-minute window. Most of the top ranked connections shown belong to the SQL Slammer / Sapphire worm. This is despite the fact that for this period (which was 2 days after the worm started) network connections due to the worm were only about 2% of the total traffic. This shows the effectiveness of the MINDS anomaly detection scheme in identifying connections due to worms. The connections due to the worm are highlighted in light gray. It can be observed that the highest contributions to the anomaly score for these connections were due to features 9 and 11. This was due to the fact that the infected machines outside our network were still tryingto communicate with many machines inside our network. In Figure 3.3, it can also be observed that during this time interval, there is another scanning activity (ping scan, highlighted in dark gray) that was detected mostly due to features 9 and 11. The two non-shaded flows are replies from "half-life game servers", which were flagged anomalous since those machines were talking to only port 27016/udp. For web connections, it is common to talk only on port 80, and it is well represented in the normal sample. However, since half-life connections did not match any normal samples with high counts on feature 15, they became anomalous.

The University of Minnesota network security analyst has been using MINDS to analyze the network traffic since August 2002. During this period, MINDS has been successful in detecting many novel network attacks and emerging network behavior that could not be detected using siganture based systems such as SNORT. In general, MINDs is able to routinely detect various suspicious behavior (e.g. policy violations),



score	srcIP	sPort	dstIP	dPort	protocol	flags	packets	bytes	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
37674.69	63.150.X.253	1161	128.101.X.29	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.81	0	0.59	0	0	0	0	0
26676.62	63.150.X.253	1161	160.94.X.134	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.81	0	0.59	0	0	0	0	0
24323.55	63.150.X.253	1161	128.101.X.185	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.81	0	0.58	0	0	0	0	0
21169.49	63.150.X.253	1161	160.94.X.71	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.81	0	0.58	0	0	0	0	0
19525.31	63.150.X.253	1161	160.94.X.19	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.81	0	0.58	0	0	0	0	0
19235.39	63.150.X.253	1161	160.94.X.80	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.81	0	0.58	0	0	0	0	0
17679.10	63.150.X.253	1161	160.94.X.220	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.81	0	0.58	0	0	0	0	0
8183.58	63.150.X.253	1161	128.101.X.108	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.82	0	0.58	0	0	0	0	0
7142.98	63.150.X.253	1161	128.101.X.223	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.82	0	0.57	0	0	0	0	0
5139.01	63.150.X.253	1161	128.101.X.142	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.82	0	0.57	0	0	0	0	0
4048.49	142.150.X.101	0	128.101.X.127	2048	1	16	[2,4)	[0,1829)	0	0	0	0	0	0	0	0	0.83	0	0.56	0	0	0	0	0
4008.35	200.250.X.20	27016	128.101.X.116	4629	17	16	[2,4)	[0,1829)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
3657.23	202.175.X.237	27016	128.101.X.116	4148	17	16	[2,4)	[0,1829)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
3450.90	63.150.X.253	1161	128.101.X.62	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.82	0	0.57	0	0	0	0	0
3327.60	63.150.X.253	1161	160.94.X.223	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.82	0	0.57	0	0	0	0	0
2796.13	63.150.X.253	1161	128.101.X.241	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.82	0	0.57	0	0	0	0	0
2693.88	142.150.X.101	0	128.101.X.168	2048	1	16	[2,4)	[0,1829)	0	0	0	0	0	0	0	0	0.83	0	0.56	0	0	0	0	0
2683.05	63.150.X.253	1161	160.94.X.43	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.82	0	0.57	0	0	0	0	0
2444.16	142.150.X.236	0	128.101.X.240	2048	1	16	[2,4)	[0,1829)	0	0	0	0	0	0	0	0	0.83	0	0.56	0	0	0	0	0
2385.42	142.150.X.101	0	128.101.X.45	2048	1	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.83	0	0.56	0	0	0	0	0
2114.41	63.150.X.253	1161	160.94.X.183	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.82	0	0.57	0	0	0	0	0
2057.15	142.150.X.101	0	128.101.X.161	2048	1	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.83	0	0.56	0	0	0	0	0
1919.54	142.150.X.101	0	128.101.X.99	2048	1	16	[2,4)	[0,1829)	0	0	0	0	0	0	0	0	0.83	0	0.56	0	0	0	0	0
1634.38	142.150.X.101	0	128.101.X.219	2048	1	16	[2,4)	[0,1829)	0	0	0	0	0	0	0	0	0.83	0	0.56	0	0	0	0	0
1596.26	63.150.X.253	1161	128.101.X.160	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.82	0	0.57	0	0	0	0	0
1513.96	142.150.X.107	0	128.101.X.2	2048	1	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.83	0	0.56	0	0	0	0	0
1389.09	63.150.X.253	1161	128.101.X.30	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.82	0	0.57	0	0	0	0	0
1315.88	63.150.X.253	1161	128.101.X.40	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.82	0	0.57	0	0	0	0	0
1279.75	142.150.X.103	0	128.101.X.202	2048	1	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.83	0	0.56	0	0	0	0	0
1237.96	63.150.X.253	1161	160.94.X.32	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.82	0	0.57	0	0	0	0	0
1180.82	63.150.X.253	1161	128.101.X.61	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.82	0	0.57	0	0	0	0	0
1107.78	63.150.X.253	1161	160.94.X.154	1434	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	0.82	0	0.57	0	0	0	0	0

Figure 3.3: Most anomalous connections found by the MINDS anomaly detection module in a 10-minute window, 2 days after the "slammer worm" started (January 27th, 2003)

worms, as well as various scanning activities. In the following, we present a few examples from each of these groups that demonstrate the effectiveness of the MINDS anomaly detection algorithm, in addition to the example shown before.

Worm detection

- On October 10th 2002, our anomaly detection module detected two activities of the slapper worm that were not identified by SNORT since they were variations of an existing worm code. Once a machine is infected with the worm, it communicates with other machines that are also infected and attempts to infect other machines. The most common version of the worm uses port 2002 for communication, but some variations use other ports. Our anomaly detector flagged these connections as anomalous for two reasons. First, the source or destination ports used in the connection may not have been rare individually but the source-destination port pairs were very rare (the anomaly detector does not keep track of the frequency of pairs of attributes; however, while building the neighborhoods of such connections, most of their neighbors will not have the same source-destination port pairs, which will contribute to the distance). Second, the communication pattern of the worm looks like a slow scan causing the value of the variable that corresponds to the number of connections from the source IP to the same destination port in the last N connections to become large. SNORT has a rule for detecting worm that uses port 2002 (and a few other ports), but not for all possible variations. A single general SNORT rule can be written to detect the variations of the worm at the expense of a higher false positive rate.

Scanning and DoS activities

- On August 9^{th} 2002, CERT/CC issued an alert for “widespread scanning and possible denial of service activity targeted at the Microsoft-DS service on port 445/TCP” as a novel Denial of Service (DoS) attack. In addition, CERT/CC also expressed “interest in receiving reports of this activity from sites with detailed logs and evidence of an attack.” Network connections due to this type of scanning were found to be the top ranked outliers on August 13^{th}, 2002, by our anomaly detection module in its regular analysis of University of Minnesota traffic. The port scan module of SNORT could not detect this attack, since the port scanning was slow. A rule to catch this type of attack was added later in Septermber 2002.
- On August 13^{th}, 2002, our anomaly detection module detected “scanning for an Oracle server” by ranking connections associated with this attack as the second highest ranked block of connections (top ranked block of connections belonged to the DoS activity targeted at the Microsoft-DS service on port 445/TCP). This type of attack is difficult to detect using other techniques, since the Oracle scan was embedded within a much larger Web scan, and the alerts generated by Web scan could potentially overwhelm the analysts. On June 13^{th}, CERT/CC had issued an alert for the attack.

Policy Violations

- On August 8^{th} and 10^{th} 2002, our anomaly detection techniques detected a machine running a Microsoft PPTP VPN server, and another one running a FTP server on non-standard ports, which are policy violations. Both policy violations were the top ranked outliers. Our anomaly detector module flagged these servers as anomalous since they are not allowed, and therefore very rare.
- On February 6^{th}, 2003, unsolicited ICMP echo reply messages to a computer previously infected with Stacheldract worm (a DDoS agent) were detected by our anomaly detection techniques. Although the infected machine has been removed from the network, other infected machines outside our network were still trying to talk to the previously infected machine from our network.

3.4.1 SNORT versus MINDS anomaly detection module

In addition to attacks detected only by our anomaly detection module, there are attacks that were detected equally well by SNORT, and some attacks for which SNORT was more successful. In order to compare general capabilities of SNORT and MINDS in detecting novel anomalous behavior, several categories of anomalous network behavior are considered:

- Content-based attacks
- Scanning activities
- Policy violations

3.4.1.1 Content-based attacks

These attacks are out of scope for our anomaly detection module since it does not consider content of the packets, and therefore SNORT is superior in identifying those attacks. However, SNORT is able to detect only those content-based attacks that have known signatures/rules. Despite the fact that SNORT is more successful in detecting the content based attacks, it is important to note that once a computer has been attacked successfully, its behavior could become anomalous and therefore detected by our anomaly detection module, as seen in previous examples). This type of anomalous behavior will be further discussed in "policy violations" section.

3.4.1.2 Scanning activities

When detecting various scanning activities SNORT and MINDS anomaly detection module may have similar performance for certain types of scans, but they have very different detection capabilities for other types. In general there are two types of scans: inbound scans when an attacker outside the network is scanning for vulnerabilities within the monitored network, and an outbound scan, when someone within the monitored network is scanning outside. There are two categories of inbound scanning activities, where SNORT and our anomaly detection module might have different detection performance:

- Fast (regular) scans
- Slow scans

When detecting regular inbound scans from an outside source, SNORT portscan module keeps track of the number of destination IP addresses accessed by each source IP address in a given time window (default value is 3 seconds). Let's denote this variable count-dest, already defined in Table 3.1. Whenever the value of count-dest is above a specified threshold (SNORT default value is 4), SNORT raises an alarm, thus indicating a scan by the source IP address. Our anomaly detection module is also able to assign high anomaly score to such network connections, since for most normal connections the value of count-dest is low. In addition, connections from many types of scanning activities tend to have other features that are unusual (such as very small payload), which make additional contributions to the anomaly score.

An inbound scan can be detected by SNORT provided the scan is fast enough for chosen time window (default value is 3 seconds) and count threshold (default value is 4). If a scanning activity is not fast enough (outside specified parameters), it will not be detected by SNORT. However, SNORT can still detect such activities by increasing the time window and/or decreasing the number of events counted within the time window, but this will tend to increase false alarm rate. On the other side, our anomaly detection module is more suitable for detecting slow scans since it considers both time-window based and connection-window based features (as opposed to SNORT that uses only time-window based features), as well as other features of the connections such as number of packets, number of bytes per packet, etc.

SNORT is unable to detect outbound scans simply because it does not examine them. Contrary, our anomaly detection module is able to detect both inbound and outbound scans. Reversing the inputs to the portscan module will allow SNORT to detect outbound scans as well. However, this will increase the memory requirements, and SNORT will still have the same problem with slow outbound scans as it has with slow inbound scans.

3.4.1.3 Policy violations

MINDS anomaly detection module is much more successful than SNORT in detecting policy violations (e.g. rogue and unauthorized services), since it looks for unusual network behavior. SNORT may detect these policy violations only if it has a rule for each of these specific activities. Since the number and variety of these activities can be very large and unknown, it is not practical to incorporate them into SNORT for the following reasons. First, processing of all these rules will require more processing time thus causing the degradation in SNORT performance. It is important to note that it is desirable for SNORT to keep the amount of analyzed network traffic small by incorporating rules as specific as possible. On the other hand, very specific rules limit the generalization capabilities of a typical rule based system, i.e., minor changes in the characteristics of an attack might cause the attack to be undetected.

Second, SNORT's static knowledge has to be manually updated by human analysts each time a new suspicious behavior is detected. In contrast, MINDS anomaly detection module is adaptive in nature, and it is particularly successful in detecting anomalous behavior originating from a compromised machine (e.g. attacker breaks into a machine, installs unauthorized software and uses it to launch attacks on other machines). Such behavior is often undetected by SNORT's signatures.

3.5 MINDS Module for Summarizing Anomalous Connections Using Association Rules

In the past decade, mining association rules has been the subject of extensive research in data mining. Techniques for mining association rules were originally developed to analyze sales transaction data, where analysts are interested to know what items are frequently bought together in the same transaction. In general, an association rule is an implication expression of the form $X \Rightarrow Y$, where X and Y are sets of binary features. An association rule can be used to predict the occurrence of certain features in a record given the presence of other features. For example, the rule $\{Bread, Butter\} \Rightarrow \{Milk\}$ indicates that most of the transactions that contain bread and butter also involve the purchase of milk. The sets of items or binary features are known as item sets in association rule terminology.

Given a set of records, the objective of mining association rules is to extract all rules of the form $X \Rightarrow Y$ that satisfy a user-specified minimum support and minimum confidence thresholds. Support measures the fraction of transactions that obey the rule while confidence is an estimate of the conditional probability $P(Y|X)$. For example, suppose 10% of all transactions contain bread and butter, and 6% of the transactions contain bread, butter, and milk. For this example, the support of the rule $\{Bread, Butter\} \Rightarrow \{Milk\}$ is 6% and its confidence is $6\%/10\% = 60\%$. If the minimum support threshold is chosen to be 1% and the minimum confidence threshold is 50%, then this rule would be extracted by the association rule mining algorithm. In this example, the set $\{Bread, Butter, Milk\}$ is also referred to as a frequent item set.

Association patterns, often expressed in the form of frequent item sets or association rules, have been found to be valuable for analyzing network traffic data [3,20,25]. These patterns can be used for the following purposes:

- To construct a summary of anomalous connections detected by the IDS. Often times, the number of anomalous connections flagged by an IDS can be very large, thus requiring analysts to spend a large amount of time interpreting and analyzing each connection that has a high anomaly score. By applying association pattern discovery techniques, analysts can obtain a high-level summary of anomalous connections. For example, scanning activity for a particular service can be summarized by a frequent set:
  srcIP=X, dstPort=Y
  If most of the connections in the frequent set are ranked high by the anomaly detection algorithm, then the frequent set may be a candidate signature for addition to a signature-based system.

- To construct a profile of the normal network traffic behavior in anomaly detection systems [3,25]. As previously noted, an anomaly detection system requires some information about how the normal network traffic behaves in order to ascertain the anomalous connections. Association patterns can provide the necessary information by identifying sets of features that are commonly found in the normal network traffic data. For example, a Web browsing activity, (almost always on port 80 and uses the TCP protocol with a small number of packets) could generate the following frequent set:
  protocol=TCP, dstPort=80, NumPackets=3...6
  In addition, association patterns generated at different time frames can be used to study the significant changes in the network traffic at various periods of time [20]

- Recurrent patterns in normal or anomalous connections can serve as secondary features to be augmented to the original data in order to build better predictive models of the network traffic data.

Mining association patterns in network traffic data is a challenging task due to the following reasons:

- *Imbalanced class distribution.* Standard association pattern discovery techniques rely on a user-specified minimum support threshold to eliminate patterns that occur infrequently in the data. For network intrusion data, the proportion of network traffic that corresponds to an attack is considerably smaller than the proportion of normal traffic. As a result, one has to apply a very low minimum support threshold to detect patterns involving the attack class. This will degrade the performance of association pattern discovery algorithms considerably and produces an overwhelmingly large number of patterns for the normal class.

  Connections that have high anomaly scores are mostly likely to be attacks and those with low anomaly scores are most likely to be normal traffic. For association pattern analysis, we choose connections that appear in the top few percentage of anomaly scores to be the attack class and the bottom few percentage of anomaly scores to be the normal class. Connections with intermediate anomaly scores will be ignored. We then mine the frequent patterns for each class separately using different minimum support thresholds, depending on the number of connections that belong to each class. If the class is small, then a low minimum support threshold is chosen. Finally, a vertical association rule mining algorithm [20, 32] is applied to efficiently discover frequent patterns of each class.

- *Binarization and grouping of attribute values.* The network intrusion data contains several continuous attributes such as number of packets, number of bytes, and duration of each connection. These attributes must be transformed into binary features first before applying standard association pattern algorithms. The transformation can be performed using a variety of supervised and unsupervised discretization techniques. Using the output scores of the anomaly detector as its ground truth, MINDS employs a supervised binning strategy to discretize the attributes. Initially, all distinct values of a continuous attribute is put into one bin. Worst bin in terms of purity is selected for partitioning until the desired number of bins is reached. Gini index is used to determine the best split. Binning for a continuous attribute is illustrated in Table 3.3.

Table 3.3: Discretization of a continuous attribute

Class	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
Anomalous	0	0	20	10	20	0	0	0	0
Normal	150	100	0	0	0	100	100	150	100
	bin1		bin2			bin3			

In addition, the source and destination IP addresses can be grouped together by applying varying sizes of net-masks. For example, the group 160.94.*.* represents the class B address for all IP addresses whose first two octets are 160 and 94. However, by doing so, an IP address will now belong to multiple groups, which may give rise to multiple patterns describing similar types of connections. For example, if the pattern (SourceIP = $IP1$, Protocol=TCP) is frequent, then the pattern (SourceIP = $IP1'$, Protocol=TCP) where $IP1^{'} = IP1\&mask$ given a net-mask size, must also be frequent.

- *Pruning the redundant patterns.* Although association patterns can detect sets of features that occur frequently in the network traffic data, the number of patterns extracted can be quite large, depending on the choice of minimum support threshold. Some of the patterns are redundant because they correspond to the subsets of other patterns. For example, given two frequent sets:

Protocol=TCP, DstPort=8888,TCPflags=SYN
DstPort=8888,TCPflags=SYN

the first one is more descriptive than the second. If the support of these two item sets is very close, then the second rule is redundant. MINDS applies a flexible pruning scheme to eliminate redundant patterns by comparing the support and confidence of patterns that share similar features. If the support and confidence of such patterns are almost identical, the more descriptive pattern is retained.

- *Finding discriminating patterns.* Eventually, the goal of mining association patterns is to discover patterns that occur regularly in the normal class or anomaly class, but not both. To do this, we need a measure that can rank the patterns according to their discriminating power. MINDS allows the users to rank the discovered patterns according to various measures, as illustrated in Figure 3.4. Consider a set of features X that occur $c1$ times in the anomalous class and $c2$

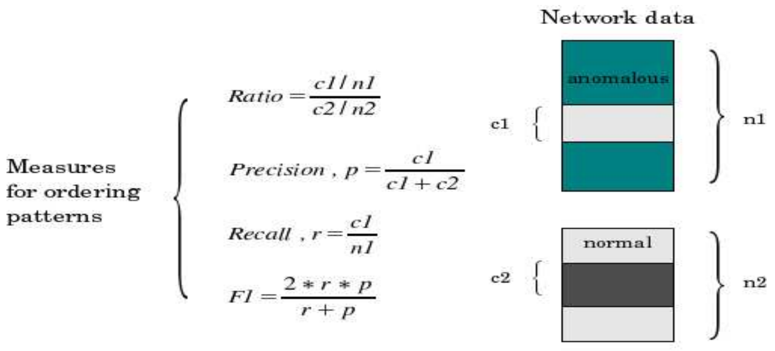

Network data
anomalous
normal
c1
c2
n1
n2
Measures for ordering patterns
Ratio = c1 / n1 over c2 / n2
Precision , p = c1 over c1 + c2
Recall , r = c1 over n1
F1 = 2 * r * p over r + p

Figure 3.4: Measures for ordering patterns

times in the normal class. Also, let $n1$ and $n2$ be the number of anomalous and normal connections in the data set. Assuming that we are only interested in finding profiles of the anomalous class, the ratio $c1/n1$ to $c2/n2$ would indicate how well the pattern could discern anomalous connections from normal connections. If the proportion of samples in each class is the same, i.e., $n1 = n2$, then the ratio measure is a monotone function of precision. Ratio or precision alone is insufficient because they often characterize only a small number of anomalous connections. In the extreme case, a rare pattern that is observed only once in the anomalous class and does not appear in the normal class will have a maximum value of ratio and precision, and yet, may not be significant. To account for the significance of a pattern, the recall measure can be used as an alternative. Unfortunately, a pattern that has high recall may not necessarily be discriminating. The F1-measure, which is the harmonic mean of precision and recall, provides a good trade-off between the two measures.

- *Grouping the discovered patterns.* It is worth noting that some of the extracted patterns can describe a similar set of anomalous connections. For example, a probe or scan may give rise to multiple patterns that are very similar to each other (e.g., these patterns may involve the same source IP address and port number, but different destination IP addresses). Thus, it is useful to group together the related patterns before presenting them to the analysts.

The overall architecture of our association analysis module is shown in Figure 3.5.

As previously noted, MINDS would use the anomaly scores of the connections to determine whether a connection belongs to the normal or attack class. In our experiments, we choose connections that have the top 10% anomaly score to be the anomaly class and the bottom 30% anomaly score to be the normal class. Connections with intermediate anomaly scores are ignored. Next, the association pattern generator is applied to each class and the patterns are ranked according to the various measures described above. The extracted patterns can be used to create summaries and profiles for normal and anomaly connections. Once the profile for the attack class is created, a follow-up

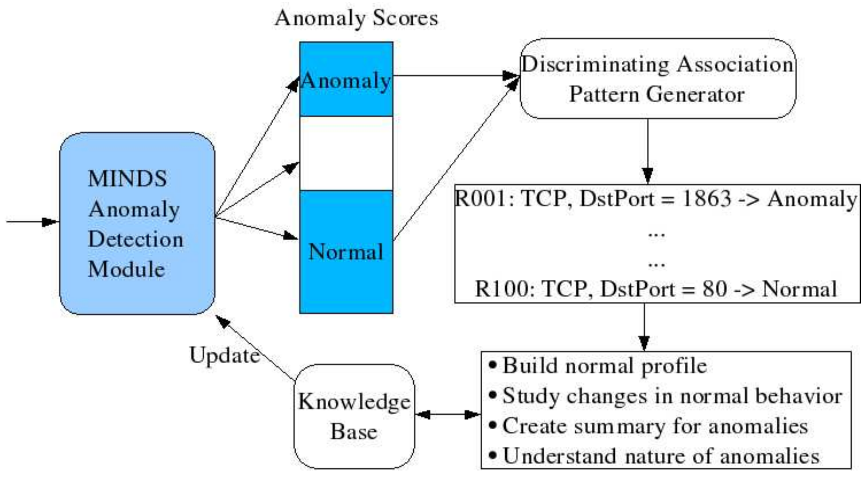

Anomaly Scores
Anomaly
Normal
MINDS Anomaly Detection Module
Discriminating Association Pattern Generator
R001: TCP, DstPort = 1863 -> Anomaly
...
...
R100: TCP, DstPort = 80 -> Normal
Update
Knowledge Base
• Build normal profile
• Study changes in normal behavior
• Create summary for anomalies
• Understand nature of anomalies

Figure 3.5: MINDS association analysis module

analysis is often performed to study the nature of the anomalous connections. A typical follow-up analysis involves connecting via telnet to the suspected computer at the specific port and examining the returned information. Another possibility of analyzing the suspected computer is to start capturing packets on that machine at the particular port and to investigate the contents of the packets.

3.5.1 Evaluation of attack summaries on real network data

In this section, we report some of the highest ranked (most discriminative) patterns generated by the our association pattern analysis module. These patterns represent a summary of the most frequently occurring and discriminating anomalous traffic flagged by MINDS anomaly detection module. A typical output of the summarization module for a 10-minute window on May 21^{st} is shown in Figure 3.4. In addition to the information reported by the anomaly detection module, the summarizer has two additional columns c_1 and c_2, which are used to evaluate the quality of rules discovered. Given a rule, c_1 denotes how many times this rule occurred among anomalous connections, while c_2 denotes how many times this rule occurred in normal connections. Single network connections that are not part of a summary have dashes in these columns. In Figure 3.4, light gray colored connections correspond to a University of Minnesota computer connecting to a remote FTP server, which happens to be running on port 5002. Further investigation of the local machine showed that it is also running multiple peer-to-peer file sharing applications. These connections were not summarized due to the large number of similar connections had low scores. The second line is a summary of TCP reset packets received from 64.156.X.74. This computer is believed to have been the victim of a DoS attack, and we were observing backscatter, i.e. replies to spoofed packets. The dark gray lines are summaries of connections involved in an FTP scan from a computer in Columbia (200.75.X.2). The summary involving destination port 113, is a summary of IDENT lookups, where a remote computer is trying to get the username of a user. The summary with destination port 119, is a summary ofa USENET server transferring a large amount of data.

Table 3.4: Output of MINDS summarization module

score	c1	c2	src IP	sPort	dst IP	dPort	protocol	flags	packets	bytes	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
31.17	-	-	218.19.X.168	5002	134.84.X.129	4182	6	27	[5,6)	[0,2045)	0	0.01	0.01	0.03	0	0	0	0	0	0	0	0	0	0	1	0
3.04	138	12	64.156.X.74	——	xx.xx.xx.xx	——	xxx	4	[0,2)	[0,2045)	0.12	0.48	0.26	0.58	0	0	0	0	0.07	0.27	0	0	0	0	0	0
15.41	-	-	218.19.X.168	5002	134.84.X.129	4896	6	27	[5,6)	[0,2045)	0.01	0.01	0.01	0.06	0	0	0	0	0	0	0	0	0	0	1	0
14.44	-	-	134.84.X.129	4770	218.19.X.168	5002	6	27	[5,6)	[0,2045)	0.01	0.01	0.05	0.01	0	0	0	0	0	0	1	0	0	0	0	0
7.81	-	-	134.84.X.129	3890	218.19.X.168	5002	6	27	[5,6)	[0,2045)	0.01	0.02	0.09	0.02	0	0	0	0	0	0	1	0	0	0	0	0
3.09	4	1	xx.xx.xx.xx	4729	xx.xx.xx.xx	——	6	——	——	——	0.14	0.33	0.17	0.47	0	0	0	0	0	0	0.2	0	0	0	0	0
2.41	64	8	xx.xx.xx.xx	——	200.75.X.2	——	xxx	——	——	[0,2045)	0.33	0.27	0.21	0.49	0	0	0	0	0	0	0	0	0.28	0.25	0.01	0
6.64	-	-	218.19.X.168	5002	134.84.X.129	3676	6	27	[5,6)	[0,2045)	0.03	0.03	0.03	0.15	0	0	0	0	0	0	0	0	0	0	0.99	0
5.6	-	-	218.19.X.168	5002	134.84.X.129	4626	6	27	[5,6)	[0,2045)	0.03	0.03	0.03	0.17	0	0	0	0	0	0	0	0	0	0	0.98	0
2.7	12	0	xx.xx.xx.xx	——	xx.xx.xx.xx	113	6	2	[0,2)	[0,2045)	0.25	0.09	0.15	0.15	0	0	0	0	0	0	0.08	0	0.79	0.15	0.01	0
4.39	-	-	218.19.X.168	5002	134.84.X.129	4571	6	27	[5,6)	[0,2045)	0.04	0.05	0.05	0.26	0	0	0	0	0	0	0	0	0	0	0.96	0
4.34	-	-	218.19.X.168	5002	134.84.X.129	4572	6	27	[5,6)	[0,2045)	0.04	0.05	0.05	0.23	0	0	0	0	0	0	0	0	0	0	0.97	0
4.07	8	0	160.94.X.114	51827	64.8.X.60	119	6	24	[483,-)	[8424,-)	0.09	0.26	0.16	0.24	0	0	0.01	0.91	0	0	0	0	0	0	0	0
3.49	-	-	218.19.X.168	5002	134.84.X.129	4525	6	27	[5,6)	[0,2045)	0.06	0.06	0.06	0.35	0	0	0	0	0	0	0	0	0	0	0.93	0
3.48	-	-	218.19.X.168	5002	134.84.X.129	4524	6	27	[5,6)	[0,2045)	0.06	0.06	0.07	0.35	0	0	0	0	0	0	0	0	0	0	0.93	0
3.34	-	-	218.19.X.168	5002	134.84.X.129	4159	6	27	[5,6)	[0,2045)	0.06	0.07	0.07	0.37	0	0	0	0	0	0	0	0	0	0	0.92	0
2.46	51	0	200.75.X.2	——	xx.xx.xx.xx	21	6	2	——	[0,2045)	0.19	0.64	0.35	0.32	0	0	0	0	0.18	0.44	0	0	0	0	0	0
2.37	42	5	xx.xx.xx.xx	21	200.75.X.2	——	6	20	——	[0,2045)	0.35	0.31	0.22	0.57	0	0	0	0	0	0	0	0	0.18	0.28	0.01	0
2.45	58	0	200.75.X.2	——	xx.xx.xx.xx	21	6	——	——	[0,2045)	0.19	0.63	0.35	0.32	0	0	0	0	0.18	0.44	0	0	0	0	0	0

In the following, we present several examples of summarization output.

- Example 1
  srcIP=IP1, dstPort=80, Protocol=TCP, Flag=SYN, NumPackets=3, NumBytes=120. . . 180 (c1=256, c2=1)
  srcIP=IP1, dstIP=IP2, dstPort=80, Protocol=TCP, Flag=SYN, NumPackets=3, NumBytes=120. . . 180 (c1=177, c2=0)

  The first rule indicates that the source of the anomalous connections originates from IP1, the destination port is 80, the protocol used is TCP with tcpflags set to SYN, the number of packets is 3, and the total number of bytes is between 120 and 180. Furthermore, this pattern is observed 256 times (c1 = 256) among the anomalous connections and only once (c2=1) in the normal connections. Therefore, it has a high ratio and precision, which is why it is ranked among the top few patterns found by the system.

  At first glance, the first rule indicates a Web scan since it appears mostly in the anomaly class with a fixed source IP address but not with a fixed destination IP address. However, the second rule suggests that an attack was later launched to one of the specific machines since the pattern originates from the same source IP address but has a specific destination IP address and covers only anomalous connections. Further analysis confirms that a scan has been performed from the source IP address IP1, followed by an attack on a specific machine that was previously identified by the attacker to be vulnerable.

- Example 2
  dstIP= IP3, dstPort=8888, Protocol=TCP (c1=369, c2=0)
  dstIP= IP3, dstPort=8888, Protocol=TCP, Flag=SYN (c1=291, c2=0)

  This pattern indicates a high number of anomalous TCP connections on port 8888 to a specific machine. Follow-up analysis of the connections covered by the pattern indicates possible existence of a machine that is running a variation of the KaZaA file-sharing protocol. KaZaA file sharing software is typically used for sharing audio, video, and software files, which are very often illegal copies.

- Example 3
  srcIP=IP4, dstPort=27374, Protocol=TCP, Flag=SYN, NumPackets=4, NumBytes=189200 (c1=582, c2=2)
  srcIP= IP4, dstPort=12345, NumPackets=4, NumBytes=189200 (c1=580, c2=3)
  srcIP= IP5, dstPort=27374, Protocol=TCP, Flag=SYN, NumPackets=3, NumBytes=144 (c1=694, c2=3)

  The patterns above indicate a number of scans on port 27374 (which is a signature for the SubSeven worm) and on port 12345 (which is a signature for the NetBus worm). Further analysis has shown that there are no fewer than five machines scanning for one or both of these ports within an arbitrary time window.

3.6 Conclusions and Future Work

Our overall goal is to develop MINDS into an overall framework for defending against attacks and threats to computer systems. Data generated from network traffic monitoring tends to have very high volume, dimensionality and heterogeneity, making the performance of serial data mining algorithms unacceptable for on-line analysis. In addition, cyber attacks may be launched from several different locations and targeted to many different destinations, thus creating a need to analyze network data from several networks in order to detect these distributed attacks. Therefore, development of new classification and anomaly detection algorithms that can take advantage of high performance computers and be computationally tractable for on-line and distributed intrusion detection is a key component of this project. To detect known attacks, our approach will use the public-domain signature-based techniques, while unknown and novel attacks will be detected using our anomaly detection schemes. According to our initial analysis, the intrusions detected by MINDS are complementary to those of SNORT, which implies that they could be combined to increase overall attack coverage. In addition, MINDS will have a visualization tool to aid the analyst in better understanding anomalous/suspicious behavior detected by the anomaly detection engine.

The anomaly detection approach used by MINDS is suitable for detecting many types of threats. Figure 3.6 shows three such types. First type of threats corresponds to outsider attacks that represent deviations from normal connection behavior. Second threat type is insider attack, where an authorized user logs into a system with malicious intent. However, the malicious behavior shown by such a user is often at variance with normal procedures, and can be picked up as anomalous behavior by our scheme. Since no security mechanism is fool proof, an undetected successful outsider attack creates is equivalent to an insider attack, and the same ideas apply. Third threat type corresponds to a situation where a virus/worm has entered an environment - either undetected by a perimeter protection mechanism such as virus scan of attachments, or through bringing in of an infected portable hardware device, e.g. a laptop. The unusual behavior shown by such a machine can potentially be detected by our approach of analyzing anomalous behavior.

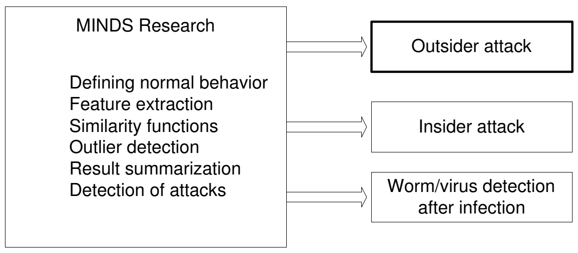


Figure 3.6: Three types of threats that can be detected by MINDS anomaly detection module

A number of applications outside of intrusion detection have similar characteristics, e.g. detecting credit card and insurance frauds, early signs of potential disasters in industrial process control, early detection of unusual medical conditions - e.g. cardiac arrhythmia, etc. We plan to explore the use of our techniques to such problems.

3.7 Acknowledgements

The authors are grateful to Richard Lippmann and Daniel Barbara for providing data sets. This work was supported by Army High Performance Computing Research Center contract number DAAD19-01-2-0014. The content of the work does not necessarily reflect the position or policy of the government and no official endorsement should be inferred. Access to computing facilities was provided by the AHPCRC and the Minnesota Supercomputing Institute.

Bibliography

[1] Charu C. Aggarwal and Philip S. Yu. Outlier detection for high dimensional data. In *SIGMOD Conference*, 2001.

[2] D. Anderson, T. F. Lunt, H. Javitz, A. Tamaru, and A. Valdes. Detecting unusual program behavior using the statistical component of the next-generation intrusion detection expert system NIDES. Technical Report SRI-CSL-95-06, Computer Science Laboratory, SRI International, 1995.

[3] Daniel Barbara, Ningning Wu, and Sushil Jajodia. Detecting novel network intrusions using bayes estimators. In *Proceedings of First SIAM Conference on Data Mining*, Chicago, IL, 2001.

[4] Eric Bloedorn, Alan D. Christiansen, William Hill, Clement Skorupka, Lisa M. Talbot, and Jonathan Tivel. Data mining for network intrusion detection: How to get started. Technical report, The MITRE Corporation, 2001.

[5] Markus Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying densitybased local outliers. In *Proceedings of the ACM SIGMOD Conference*, Dallas, TX, 2000.

[6] J. B. D. Cabrera, B. Ravichandran, and R. K. Mehra. Statistical traffic modeling for network intrusion detection. In *Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, San Francisco, CA, 2000.

[7] Sara Matzner Chris Sinclair, Lyn Pierce. An application of machine learning to network intrusion detection. In *Proceedings of the 15th Annual Computer Security Applications Conference*, pages 371–377, Phoenix, AZ, 1999.

[8] Dorothy E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, SE-13:222–232, 1987.

[9] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. *Data Mining for Security Applications*, 2002.

[10] Wei Fan, Salvatore J. Stolfo, Junxin Zhang, and Philip K. Chan. AdaCost: misclassification cost-sensitive boosting. In *Proc. 16th International Conf. on Machine Learning*, pages 97–105, Bled, Slovenia, 1999. Morgan Kaufmann, San Francisco, CA.

[11] Anup K. Ghosh and Aaron Schwartzbard. A study in using neural networks for anomaly and misuse detection. In *Proceedings of the Eighth USENIX Security Symposium*, pages 141–151, Washington, DC, 1999.

[12] Harold S. Javitz and Alfonso Valdes. The nides statistical component: Description and justification. Technical report, Computer Science Laboratory, SRI International, 1993.

[13] M. Joshi and V. Kumar. Credos: Classification using ripple down structure (a case for rare classes). In *Proceedings of 19th International Conference on Data Engineering*, Bangalore, India, 2003.

[14] Mahesh V. Joshi, Ramesh Agarwal, and Vipin Kumar. Mining needles in a haystack: Classifying rare classes via two-phase rule induction. In *Proceedings of ACM SIGMOD Conference on Management of Data*, Santa Barbara, CA, 2001.

[15] Mahesh V. Joshi, Ramesh Agarwal, and Vipin Kumar. Predicting rare classes: Can boosting make any weak learner strong? In *KDD*, Edmonton, Canada, 2002.

[16] Mahesh V. Joshi, Vipin Kumar, and Ramesh C. Agarwal. Evaluating boosting algorithms to classify rare classes: Comparison and improvements. In *ICDM*, pages 257–264, San Jose, CA, 2001.

[17] Edwin M. Knorr and Raymond T. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proc. 24th Int. Conf. Very Large Data Bases, VLDB*, pages 392–403, New York City, NY, 1998. Morgan Kaufmann.

[18] Aleksander Lazarevic, Nitesh V. Chawla, Lawrence O. Hall, and Kevin W. Bowyer. Smoteboost: Improving the prediction of minority class in boosting. Technical Report 2002-136, AHPCRC, 2002.

[19] Aleksander Lazarevic, Aysel Özgur, Levent Ertöz, Jaideep Srivastava, and Vipin Kumar. A comparative study of anomaly detection schemes in network intrusion detection. in review.

[20] Wenke Lee and Salvatore Stolfo. Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, 1998.

[21] Wenke Lee and Dong Xiang. Information-theoretic measures for anomaly detection. In *IEEE Symposium on Security and Privacy*, 2001.

[22] Richard P. Lippmann and Robert K. Cunningham. Improving intrusion detection performance using keyword selection and neural networks. *Computer Networks (Amsterdam, Netherlands: 1999)*, 34:597–603, 2000.

[23] Jianxiong Luo. Integrating fuzzy logic with data mining methods for intrusion detection. Master's thesis, Department of Computer Science, Mississippi State University, 1999.

[24] Matthew V. Mahoney and Philip K. Chan. PHAD: Packet header anomaly detection for identifying hostile network traffic. Technical report, Florida Tech., 2001.

[25] Stefanos Manganaris, Marvin Christensen, Dan Zerkle, and Keith Hermiz. A data mining analysis of rtid alarms. In *Proceedings of the 2nd International Workshop on Recent Advances in Intrusion Detection RAID*, West Lafayette, IN, 1999.

[26] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the ACM SIGMOD Conference*, pages 427–438, Dallas, TX, 2000.

[27] Jake Ryan, Meng-Jang Lin, and Risto Miikkulainen. Intrusion detection with neural networks. In *Proceedings of AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management*, pages 72–77. AAAI Press, 1997.

[28] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou. Specification based anomaly detection: A new approach for detecting network intrusions. In *ACM Conference on Computer and Communications Security*, 2002.

[29] Stuart Staniford, James A. Hoagland, and Joseph M. McAlerney. Practical automated detection of stealthy portscans. *Journal of Computer Security*, 10:105–136, 2002.

[30] www.splintered.net/sw/flow tools. Flow-tools.

[31] K. Yamanishi, J. Takeuchi, G. Williams, , and P. Milne. On-line unsupervised oultlier detection using finite mixtures with discounting learning algorithms. In *KDD*, pages 320–324, Boston, MA, 2000.

[32] Nong Ye and Qiang Chen. An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. *Quality and Reliability Engineering International*, 17:105–112, 2001.

[33] Mohammed J. Zaki and Karam Gouda. Fast vertical mining using diffsets. Technical Report 01-1, 2001 11, Rensselaer Polytechnic Institute, 2001.