Detection of Novel Network Attacks Using Data Mining

Levent Ertoz, Eric Eilertson, Aleksandar Lazarevic,
Pang-Ning Tan, Paul Dokas, Vipin Kumar, Jaideep Srivastava

Computer Science Department, 200 Union Street SE, 4-192, EE/CS Building,
University of Minnesota, Minneapolis, MN 55455, USA

{ertoz,eric,aleks,ptan,dokas,kumar,srivasta}@cs.umn.edu

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 present two specific contributions. First, we present MINDS anomaly detection module that assigns a score to each connection that reflects how anomalous the connection is compared to the normal network traffic. Experimental results on live network traffic at the University of Minnesota show that our anomaly detection techniques have been successful in automatically detecting several novel intrusions that could not be identified using state-of-the-art signature-based tools such as SNORT. Many of these have been reported on the CERT/CC list of recent advisories and incident notes. We also present the results of comparing the MINDS anomaly detection module to SPADE (Statistical Packet Anomaly Detection Engine), which is designed to detect stealthy scans.

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, 21, 26, 28], 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, 10, 21, 24, 26, 33], rare class predictive models [14-17, 19], association rules [3, 21, 28] and cost sensitive modeling [9, 16]. 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 data instances 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 [7, 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, and 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 methods. In supervised anomaly detection, given a set of normal data to train from, and given a new piece of test data, the goal is to determine whether the test data belongs to "normal" or to an anomalous behavior. Recently, there have been several efforts in designing supervised network-based anomaly detection algorithms, such as ADAM [3], PHAD [27], NIDES [2], and other techniques that use neural networks [32], information theoretic measures [22], 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 [36, 37],

clustering [8], outlier detection schemes [1, 5, 18, 31], state machines [34], etc.

This paper introduces the Minnesota Intrusion Detection System (MINDS) that 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 only an anomaly detection technique that assigns a score to each network connection reflecting how anomalous the connection is compared to the normal network traffic. We also provide an evaluation of MINDS anomaly detection schemes in the context of real life network data at the University of Minnesota. During the last year, this evaluation has shown that anomaly detection algorithms have been successful in automatically detecting numerous novel intrusions that could not be identified using widely popular tools such as SNORT [35]. 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. We chose to present results on real life network data since publicly available data sets for evaluation of network intrusion detection systems (e.g. DARPA 1998, DARPA 1999 data sets [23, 25]) are known to have serious limitations [29]. In the absence of labels of network connections (normal vs. intrusive), we are unable to provide real 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.

2. The MINDS System

The Minnesota Intrusion Detection System (*MINDS)* is a data mining based system for detecting network intrusions. Figure 1 illustrates the process of analyzing real network traffic data using the *MINDS* system. Input to MINDS is Netflow version 5 data collected using Netflow tools. Netflow tools only capture packet header information (i.e., they do not capture message content), and build one way sessions (flows). We are working with Netflow data instead of tcpdump because we currently do not have the capacity to collect and store the tcpdump. Netflow data for each 10 minute window, which typically result in 1 to 2 million flows, is stored in a flat file. The analyst uses MINDS to analyze these 10-minute data files in a batch mode. Before applying MINDS to these data files, a data filtering step is performed by the system administrator to remove network traffic that the analyst is not interested in analyzing. For example, the removed attack-free network data in data filtering step may include the data coming from trusted sources, non-interesting network data (e.g. portions of *http* traffic) or unusual/anomalous network behavior for which it is known that it does not correspond to intrusive behavior.

The first step in MINDS includes constructing features that are used in the data mining analysis. Basic features include source IP address and port, destination IP address and port, protocol, flags, number of bytes, and number of packets. Derived features include time-window and connection-window based features. Time-window based features are constructed to capture connections with similar characteristics in the last *T* seconds, since typically of Denial of Service (DoS) and scanning attacks involve hundreds of connections. A similar approach was used for constructing features in KDDCup'99 data [39]. Table 1 summarizes the time-window based features.

"Slow" scanning attacks, i.e. those that scan the hosts (or ports) and use a much larger time interval than a few seconds, e.g. one scan per minute or even one scan per hour, cannot be detected using derived "time-window" based features. To capture these types of the attacks, we also derive "connection-window" features that capture the same characteristics of the connection records as time-window based features, but are computed in the last *N* connections. The connection-window based features are shown in Table 2.

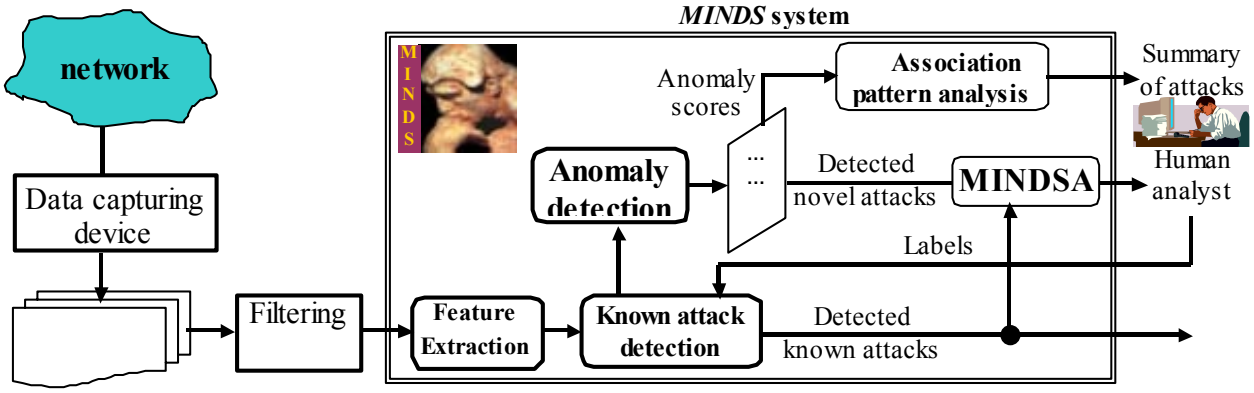

Figure 1 Architecture of MINDS system

Feature Name	Feature description
count_dest_conn	For the same source IP address, number of unique destination IP addresses inside the network in the last *N* connections
count_src_conn	For the same destination IP address, number of unique source IP addresses inside the network in the last *N* connections
count_serv_ src_conn	Number of connections from the source IP to the same destination port in the last *N* connections
count_serv_ dest_conn	Number of connections from the destination IP to the same source port in the last *N* connections

Table 1 The extracted "time-window" features

Feature Name	Feature description
count_dest	For the same source IP address, number of unique destination IP addresses inside the network in the last *T* seconds
count_src	For the same destination IP address, number of unique source IP addresses inside the network in the last *T* seconds
count_serv_src	Number of connections from the source IP to the same destination port in the last *T* seconds
count_serv_dest	Number of connections from the destination IP to the same source port in the last *T* seconds

Table 2 The extracted "connection-window" based features

After the feature construction step, the known attack detection module is used to detect network connections that correspond to attacks for which the signatures are available, and then to remove them from further analysis. For experiments 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.

3. MINDS Anomaly Detection

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 [20].

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 specific data example *p* represents the average of the ratios of the density of the example *p* and the density of its nearest neighbors. To illustrate advantages of the *LOF approach*, consider a simple two-dimensional data set given in Figure 2. It is apparent that the density of cluster C_2 is significantly higher that the density of cluster C_1. Due to the low density of cluster C_1 it is apparent that for every example *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 p1 may be detected as outlier using the distances to the nearest neighbor. On the other side, LOF is able to capture both outliers (p1 and p2) due to the fact that it considers the density around the points.

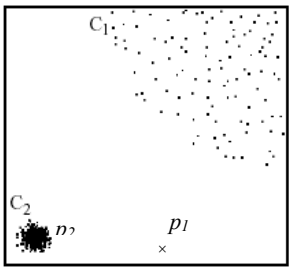

Figure 2 Outlier Examples

LOF requires the neighborhood around all data points be constructed. This involves calculating pair-wise distances between all data points which is an $O(n^2)$ process, which makes it computationally infeasible for millions of connections. To address this problem, we sample the data to use as a training set and compare all data points to this small set, which reduces the complexity to $O(n \cdot m)$ 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.

4. Experimental Evaluation of MINDS Anomaly Detection

The output of the MINDS anomaly detector contains the original Netflow data with the addition of the

anomaly score and relative contribution of each of the 16 attributes to the score. Table 3 shows the 16 attributes. The analyst typically looks at only the top few connections that have the highest anomaly scores. Figure 3 shows the most anomalous connections found by MINDS on January 26th in a 10-minute window, 48 hours after the slammer attack. The University of Minnesota network security analyst has been using MINDS and SNORT independently to analyze the university network traffic for the past seven months. During this period, MINDS has been successful in detecting many novel network attacks and emerging network behavior that could not be detected using SNORT.

In the following, we present a few examples that demonstrate the effectiveness of the MINDS anomaly detection algorithm. In addition, we present a comparison of MINDS performance on detecting scans to SPADE because it is already integrated into SNORT and thus available as open source. Note that other schemes exist that work in high bandwidth environments; e.g. the scheme presented in [38] identifies packets that are likely to be a probe and performs scan detection on only those packets. But most such schemes are not available as open source. Note that the comparison with SPADE is only restricted to detecting scans, as SPADE is not meant to find policy violations and worms, which can be detected by MINDS. We are unable to provide comparisons of MINDS with other anomaly detection systems that are potentially capable of finding intrusions other than scans, as either they are available only in commercial products [38], or require sanitized training data [27].

4.1 MINDS Anomaly Detection Results

Anomalies/attacks picked by MINDS include scanning activities, worms, and non-standard behavior such as policy violations and insider attacks. Many of these attacks detected by MINDS, have already been on the CERT/CC list of recent advisories and incident notes.

- On January 26, 2003, 48 hours after the "SQL Slammer/Sapphire" worm started, network connections related to the worm were only about 2% of the total traffic. Despite this, they were still ranked at the top by the anomaly detection algorithm (see Figure 3). The network connections that are part of the "slammer worm" are highlighted in light gray in Figure 3. It can be observed that the highest contributions to anomaly score for these connections were due to the features 9 and 11 (count_dest and count_serv_src from Table 1). This was due to the fact that the infected machines outside our network were still trying to communicate with many machines inside our network. Similarly, it can be observed from Figure 3 that during this time interval there is another scanning activity (ICMP ping scan, highlighted in dark gray) that was detected again mostly due to the features 9 and 11. The two non-shaded flows are replies from Half-Life game servers (running on port 27016/udp). They were flagged anomalous because 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.

score	srcIP	sPort	dstIP	dPort	protocc	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
17679.1	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
3450.9	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
3327.98	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
1237.97	63.150.X.253	**1161**	160.94.X.32	**1434**	17	16	[0,2)	[0,1829)	0	0	0	0	0	0	0	0	**0.83**	0	**0.56**	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	**0.83**	0	**0.56**	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	**0.83**	0	**0.56**	0	0	0	**0**	0

Figure 3 Most anomalous connections found by MINDS anomaly detection algorithm in a 10-minute window 48 hours after the "slammer worm" started.

- On October 10th, 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 count_serv_src_conn (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.

1 source IP	5 protocol
2 destination IP	6 duration
3 source Port	7 bytes/packet
4 destination Port	8 # packets
9 cnt_dest	13 cnt_src
10 cnt_dest_conn	14 cnt_src_conn
11 cnt_serv_src	15 cnt_serv_dest
12 cnt_serv_src_conn	16 cnt_serv_dest_conn

Table 3 List of features used in anomaly detection

- On August 9th, 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." This type of attack was the top ranked outlier on August 13th, 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.
- On August 13th, 2002, our anomaly detection module detected "scanning for an Oracle server" by ranking connections associated with this attack as the second highest ranked block set of connections (the top ranked block of connections belonged to the denial of service 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 much larger Web scan, and the alerts generated by Web scan could potentially overwhelm the human analysts. On June 13th, CERT/CC had issued an alert for the attack.
- On August 8th and 10th, 2002, our anomaly detection techniques detected a machine running a Microsoft PPTP VPN server, and another one running a FTP server, which are policy violations, on non-standard ports. 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. Since SNORT is not designed to look for rogue and unauthorized servers, it was not able to detect these activities. In addition, for the PPTP VPN server, the collected GRE traffic is part of the normal traffic, and not analyzed by tools such as SNORT.
- On January 27, 2003, our techniques detected odd, not routable RFC1918 traffic coming from the Internet. RFC1918 (Request for Comments) serves as Address Allocation for Private Internets, while RFC1918 blocks are segments of IP address space reserved by IANA (Internet Assigned Numbers Authority) for use within an organization. DNS records for RFC1918 addresses are legitimate only within the network on which a host with RFC1918 address resides. However, RFC1918 addresses are not globally routed and they should not appear on the public Internet.
- On February 6, 2003, our technique detected that the IP address 128.101.6.0, which does not correspond to a real computer, but to a network itself, has been targeted with IP Protocol 0 traffic from Korea (61.84.X.97). This type of network traffic is "exceedingly" bad as IP Protocol 0 is not legitimate.
- On February 6, 2003, our techniques detected a computer on the network apparently communicating with a computer in California over a VPN. This scenario in the worst case may correspond to a covert channel by which someone might be gaining access to the University network in an unauthorized way, and in the best case to someone at the University creating unauthorized tunnels between the University and some other network, which is not allowed. However, both types of behavior are extremely useful for security analysts.
- On February 7, 2003, a computer in the CS department talking on IPv6 was detected using our techniques. This type of communication is extremely rare and represents a possible covert tunnel to the outside world. The follow-up analysis diagnosed that a suspect who was doing this is on system staff and is in fact using this as a covert tunnel to his home computers.
- On February 6, 2003, our anomaly detection techniques detected unsolicited ICMP ECHOREPLY

messages to a computer previously infected with Stacheldract worm (a DDos agent). Although infected machine has been removed from the network, other infected machines outside our network were still trying to talk to infected machine from our network.

4.2 Comparison of MINDS and SPADE

SPADE: A brief overview. SPADE is a SNORT plug-in that automatically detects stealthy port scans [36]. Unlike traditional scan detectors that look for X events in Y seconds, SPADE takes a radically different approach and looks at the amount of information gained by probing. It has four different methods of calculating the likelihood of packets. However, the most successful method measures the direct joint probability P(dest IP, dest Port). SPADE examines TCP-SYN packets and maintains the count of packets observed on (destIP, destPort) tuples. When a new packet is observed, SPADE checks the probability of observing that packet on the (dest IP, dest Port) tuple. The lower the probability, the higher the anomaly score. Note that SPADE raises alarms on individual SYN packets regardless of how many other destination IP/ports have been scanned by the same source. In the case of an IPsweep, the scanner will eventually touch a machine that does not have the service being scanned for and therefore will raise a SPADE alarm. In the case of a portscan, an alarm will be raised when the scanner touches a port for which the service is not available on the target machine. In addition, SPADE will raise false alarms on legitimate traffic for which (destination IP, destination port) combinations are infrequent. This will be even more prevalent on outbound connections. The reason is that the number of IPs outside is much bigger than the number of IPs inside the network. As reported in [36], on DARPA99 data, as the number of variables in the direct joint probability is increased to include the source IP and source Port, the accuracy of SPADE decreases. This can be attributed to the fact that when extra attributes are used, the model essentially becomes sparser and therefore gives more false alarms.

MINDS and SPADE both assign a score to each connection that indicates its degree of being an outlier. They are both unsupervised anomaly detection schemes since neither one requires a labeled training set. The key difference is in the method for computing the anomaly score. In SPADE, the anomaly score is inversely related to the probability of observing the connection based upon the features used. If too many features are used (or if any feature has too many values), then the probability estimates are not reliable. The reason is that many of the legitimate combinations of features may be previously unseen or infrequent. In contrast, MINDS does not suffer from increased dimensionality as much as SPADE does, since MINDS constructs neighborhoods around data points which is a better estimate of actual probabilities when there is not enough data to adequately develop the model. This allows MINDS to use a large number of features as long as they are not spurious. Even if MINDS used only two features (destination IP and destination port), we argue that it can provide higher quality outlier scores. To illustrate this consider the following probability distribution (Table 4), where blank represents no occurrences. If we observe a packet P1 on IP2/Port3 and another packet P2 on IP4/Port1, SPADE will assign equal anomaly scores to both of the packets. However, one could argue that P2 should be more anomalous than P1 since in the case of P2 neither the port (Port1) nor the IP (IP4) by themselves are used frequently, whereas for P1, both the port (Port3) and the IP (IP2) are frequently used.

If we use MINDS anomaly detection module with only 2 attributes, namely the destination IP and destination port, packet P2 will be assigned a higher anomaly score than P1. The reason is that P1 will be closer to its neighbors than P2 will be to its neighbors. Both P1 and P2 will have neighbors that are in dense regions. If we compare the ratios of densities of P1 and P2 to the density of their respective neighbors, P2 will have a lower ratio, hence a higher anomaly score.

Frequency	**IP1**	**IP2**	**IP3**	**IP4**
Port1		Low		**P2**
Port2		High		
Port3	High	**P1**	High	
Port4	Low	High	Low	

Table 4. IP/Port frequency distribution

We ran the latest version of SPADE (v021031.1) on live network traffic at the University of Minnesota for a 10-minute period using a threshold of 8. It generated 296,921 alarms out of approximately one million TCP_SYN packets received during this 10-minute window. Nearly 26% (76,956) of the alarms were on inbound packets. Vast majority of outbound packets were false alarms. This is not surprising since the space of (IP/Port) combinations outside our network is very large and the data is very sparse. This is a serious limitation of SPADE, since outbound alarms tend to be very important, as they often indicate infected machines inside the network. In the rest of the discussion, we focus on alarms on inbound packets. 25% (19020) of inbound alarms were to web alarms, 6% (4608) to common services other than web (https, mail, ftp, ssl enabled imap, web proxies), 28.5% (21895) to peer-to-peer applications (kazaa, gnutella, edonkey) and the remaining 41% (31433) were hard to interpret. If we ignore repetitions of the same alarm (same source IP, destination IP, destination Port), we are left with 28973 alarms. There were a total of 21669 unique sources of

alarms, 20825 of them generated only 1 or 2 alarms. (More detailed information about the distributions is given in Table 5.) Although some of these alerts may indicate very slow stealthy scans, majority of these alerts are likely to be for legitimate connections on rarely used destination IP, destination port combination. Specifically, we can say that the alarms on p2p applications are false alarms (not scans) since p2p applications get the list of active servers from their super-nodes instead of scanning for machines running these applications. In any case, one cannot expect from a system administrator to investigate alarms for so many different sources in a short 10-minute window.

If we raise our threshold and look at only top 10,000 alerts, the distribution is very similar. If we look at the breakdown of alarms by type, in the top 10,000 (threshold = 12.6108), web alerts are 59% (5927), common services are 7.5% (754), p2p applications are 9% (923) and the remaining, hard-to-interpret ones are 24% (2396). There are 3306 unique sources that generate 10,000 alarms, 3159 of which raised either 1 or 2 alarms. If we look at top 1000, 497 out of 547 unique sources raised less than 3 alarms. If we look at top 100, 73 out of 78 unique sources raised less than 3 alarms.

If we do a similar analysis for alarms on web, P2P (peer to peer), common services and the remaining alarms separately, we still get a very similar picture; the number of unique sources generating very few alarms is very large. We argue that most of SPADE alarms are effectively false alarms as the number of unique sources generating the alarms is at a high percentage regardless of the threshold.

SPADE does find some stealthy scans that will be hard to find using simple scan detection schemes that look for source IP's that connect to more than X destination Ports / IP's in a specified time or connection window. For example, among the top 10,000 alarms 66 unique sources generated at least 10 or more alarms. Each one of these sources were either scanning a specific IP for 10 or more ports, or scanning 10 or more IP's for a specific port. But in the process of finding these hard to detect stealthy scans, SPADE generated false alarms for far too many legitimate connections.

Number of alarms	1	2	3	4	
Number of sources	19282	1543	394	151	
Number of alarms	5-6	7-20	21-50	51-150	151+
Number of sources	139	106	45	5	4

Table 5. Distribution of number of alarms from unique sources (inbound alarms)

For the same 10-minute period, we ran MINDS and asked our security expert to analyze top few hundred anomalous connections ranked by the anomaly detector. Most of these were scans, which were interleaved with few non-scan connections. Most of these scans targeted dozens of IPs inside the University of Minnesota networks. Among non-scan anomalous connections, our security expert identified a local machine running a web proxy open to everyone, and lots of people were browsing the web through the proxy to anonymize their connections. Proxy settings were fixed after the issue was identified. For privacy reasons, we cannot report the individual connections but only provide a high level summary. Note that stealthy scans that target only a few machines during the 10-minute window are not likely to receive high anomaly score by MINDS.

4.3 MINDS anomaly detection module versus SNORT

Here we compare general capabilities of SNORT and MINDS in detecting the following categories of attacks and irregular behavior:

- content-based attacks
- scanning activities
- policy violations

Content based attacks. These attacks are out of scope for our anomaly detection module since it does not consider the 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.

Scanning activities. When detecting various scanning activities SNORT and MINDS anomaly detection module have similar performance for certain types of scans, but they have very different detection capabilities for other types. There are two categories of scanning activities, where SNORT and our anomaly detection module might have different detection performance:

- Fast (regular) scans
- Slow scans

When detecting regular scans, 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 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

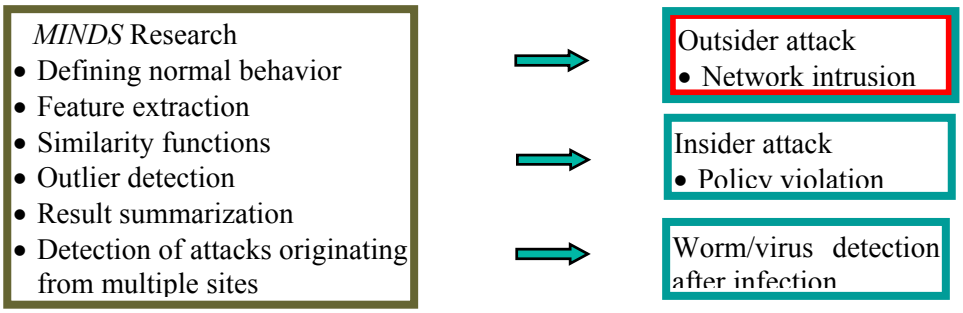

Figure 4 Three types of threats that can be detected by MINDS anomaly detection module

activities tend to have other features that are unusual (such as very small payload), which make additional contributions to the anomaly score.

A 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.

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.

5. Conclusions and Future Work

The overall goal for MINDS is to be a general framework and system for detecting attacks and threats to computer systems. Data generated from network traffic monitoring tends to have very high volume, dimensionality and heterogeneity. Coupled with the low frequency of occurrence of attacks, this makes standard data mining algorithms unsuitable for detecting attacks. 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 locations/networks in order to detect these distributed attacks. According to our initial analysis, the intrusions detected by MINDS are complementary to those of SNORT – a signature-based system. This implies that the two can be combined to increase overall attack coverage. In addition, MINDS will have a summarization and visualization tools to aid the analyst in better understanding anomalous/suspicious behavior detected by the anomaly detection engine.

The key anomaly detection approach used by MINDS is based on the analysis of unusual behavior, and is thus suitable for detecting many types of threats. Figure 4 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 our behavior-analysis based approach can pick it up as anomalous behavior, reporting it as a possible attack. Since no security mechanism is fool proof, an undetected successful outsider becomes 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.

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.

Acknowledgments

This work was partially supported by Army High Performance Computing Research Center contract number DAAD19-01-2-0014 and NSF grants IIS-0308264, ACI-9982274. 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.

References

1. C. C. Aggarwal, P. Yu, Outlier Detection for High Dimensional Data, Proceedings of the ACM SIGMOD Conference, 2001.
2. Anderson, D., Lunt, T. F., Javitz, H., Tamaru, A., Valdes, A.: 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, Menlo Park, CA, 1995.
3. D. Barbara, N. Wu, S. Jajodia, Detecting Novel Network Intrusions Using Bayes Estimators, First SIAM Conference on Data Mining, Chicago, IL, 2001.
4. E. Bloedorn, et al., Data Mining for Network Intrusion Detection: How to Get Started, MITRE Technical Report, August 2001.
5. M. M. Breunig, H.P. Kriegel, R. T. Ng, J. Sander, LOF: Identifying DensityBased Local Outliers, Proceedings of the ACM SIGMOD Conference, 2000.
6. Cabrera, J. B. D., Ravichandran, B., Mehra, R. K.: Statistical Traffic Modeling For Network Intrusion Detection, Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, San Francisco, CA, 2000.
7. D.E. Denning, An Intrusion Detection Model, IEEE Transactions on Software Engineering, SE-13:222-232, 1987.
8. E. Eskin, A. Arnold, M. Prerau, L. Portnoy, S. Stolfo, A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data, Data Mining for Security Applications, Kluwer 2002.
9. W. Fan, S. J. Stolfo, J. Zhang, P. K. Chan, AdaCost: Misclassification Cost-sensitive Boosting, Proceedings of the Sixteenth International Conference on Machine Learning, 97-105, Bled, Slovenia, 1999.
10. A. Ghosh, A. Schwartzbard, A study in Using Neural Networks for Anomaly and Misuse Detection, Proceedings of the Eighth USENIX Security Symposium, 141--151, Washington, DC, August 1999.
11. Incident Storm Center, www.incidents.org.
12. H.S. Javitz, and A. Valdes, The NIDES Statistical Component: Description and Justification, Technical Report, Computer Science Laboratory, SRI International, 1993.
13. K.S.Jones: A Statistical Interpretation Of Term Specificity And Its Application In Retrieval, Journal of Documentation, 28 (1) 11-21, 1972.
14. M. Joshi, V. Kumar, R. Agarwal, Evaluating Boosting Algorithms to Classify Rare Classes: Comparison and Improvements, First IEEE International Conference on Data Mining, San Jose, CA, 2001.
15. M. Joshi, R. Agarwal, V. Kumar, Predicting Rare Classes: Can Boosting Make Any Weak Learner Strong?, Proceedings of Eight ACM Conference ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Canada, 2002.
16. M. Joshi, R. Agarwal, V. Kumar, PNrule, Mining Needles in a Haystack: Classifying Rare Classes via Two-Phase Rule Induction, Proceedings of ACM SIGMOD Conference on Management of Data, May 2001.
17. M. Joshi, V. Kumar, CREDOS: Classification using Ripple Down Structure (A Case for Rare Classes), in review.

18. E. Knorr, R. Ng, Algorithms for Mining Distance-based Outliers in Large Data Sets, Proceedings of the VLDB Conference, 1998.

19. A. Lazarevic, N. Chawla, L. Hall, K. Bowyer, SMOTEBoost: Improving the Prediction of Minority Class in Boosting, AHPCRC Technical Report, 2002.

20. A. Lazarevic, L. Ertoz, A. Ozgur, V. Kumar, J. Srivastava: A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection, Proceedings of Third SIAM International Conference on Data Mining, May, San Francisco, 2003.

21. W. Lee, S. J. Stolfo, Data Mining Approaches for Intrusion Detection, Proceedings of the 1998 USENIX Security Symposium, 1998.

22. W. Lee, D. Xiang: Information-Theoretic Measures for Anomaly Detection, IEEE Symposium on Security and Privacy, 2001.

23. R. P. Lippmann, R. K. Cunningham, D. J. Fried, I. Graf, K. R. Kendall, S. W. Webster, M. Zissman, Re-sults of the 1999 DARPA Off-Line Intrusion Detection Evaluation, Proceedings of the Second International Workshop on Recent Advances in Intrusion Detection (RAID99), West Lafayette, IN, 1999.

24. R. Lippmann, R. Cunningham, Improving intrusion detection performance using keyword selection and neural networks, Computer Networks, 34(4):597--603, 2000.

25. R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. P. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation, Proceedings DARPA Information Survivability Conference and Exposition (DISCEX) 2000, Vol 2, pp. 12-26, IEEE Computer Society Press, Los Alamitos, CA, 2000.

26. J. Luo, Integrating Fuzzy Logic With Data Mining Methods for Intrusion Detection, Master's thesis, Department of Computer Science, Mississippi State University, 1999.

27. M. Mahoney, P. Chan: Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks, Proceedings of 8th International Conference on Knowledge Discovery and Data Mining, 376-385, 2002.

28. S. Manganaris, M. Christensen, D. Serkle, and K. Hermix, A Data Mining Analysis of RTID Alarms, Proceedings of the 2nd International Workshop on Recent Advances in Intrusion Detection (RAID 99), West Lafayette, IN, September 1999.

29. J. McHugh, The 1998 Lincoln Laboratory IDS Evaluation (A Critique), Proceedings of the Recent Advances in Intrusion Detection, 145-161, Toulouse, France, 2000.

30. Net flow tools, www.splintered.net/sw/flow-tools.

31. S. Ramaswamy, R. Rastogi, K. Shim, Efficient Algorithms for Mining Outliers from Large Data Sets, Proceedings of the ACM SIGMOD Conference, 2000.

32. J. Ryan, M-J. Lin, R. Miikkulainen, Intrusion Detection with Neural Networks, Proceedings of AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management, 72-77, AAAI Press, 1997.

33. C. Sinclair, L. Pierce, S. Matzner, An Application of Machine Learning to Network Intrusion Detection, Proceedings of the 15th Annual Computer Security Applications Conference, 371-377, Phoenix, AZ, December 1999.

34. R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, S. Zhou: Specification Based Anomaly Detection: A New Approach for Detecting Network Intrusions, ACM Conference on Computer and Communications Security, 2002.

35. SNORT Intrusion Detection System. www.snort.org.

36. S. Staniford, J. Hoagland, J. McAlerney, Practical Automated Detection of Stealthy Portscans, Journal of Computer Security, vol. 10, No. 1-2, 105-136, 2002.

37. K. Yamanishi, J. Takeuchi, G. Williams, P. Milne, On-line Unsupervised Oultlier Detection Using Finite Mixtures with Discounting Learning Algorithms, Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 320-324, Boston, MA, 2000.

38. S. Robertson, E. V. Siegel, M. Miller, S. J. Stolfo, Surveillance Detection in High Bandwidth Environments, DARPA DISCEX, 2003.

39. KDD cup 99, kdd.ics.uci.edu/databases/kddcup99/task.html