Scan Detection: A Data Mining Approach

György J. Simon
Computer Science
Univ. of Minnesota
gsimon@cs.umn.edu

Hui Xiong
MSIS Dept.
Rutgers Univ.
hui@rbs.rutgers.edu

Eric Eilertson
Computer Science
Univ. of Minnesota
eric@cs.umn.edu

Vipin Kumar
Computer Science
Univ. of Minnesota
kumar@cs.umn.edu

Abstract

A precursor to many attacks on networks is often a reconnaissance operation, more commonly referred to as a scan. Despite the vast amount of attention focused on methods for scan detection, the state-of-the-art methods suffer from high rate of false alarms and low rate of scan detection. In this paper, we formalize the problem of scan detection as a data mining problem. We show how the network traffic data sets can be converted into a data set that is appropriate for running off-the-shelf classifiers on. Our method successfully demonstrates that data mining models can encapsulate expert knowledge to create an adaptable algorithm that can substantially outperform state-of-the-art methods for scan detection in both coverage and precision.

1 Introduction

A precursor to many attacks on networks is often a reconnaissance operation, more commonly referred to as a scan. Identifying what attackers are scanning for can alert a system administrator or security analyst to what services or types of computers are being targeted. Knowing what services are being targeted before an attack allows an administrator to take preventative measures to protect the resources e.g. installing patches, firewalling services from the outside, or removing services on machines which do not need to be running them.

Given its importance, the problem of scan detection has been given a lot of attention by a large number of researchers in the network security community. Initial solutions simply counted the number of destination IPs that a source IP made connection attempts to on each destination port and declared every source IP a scanner whose count exceeded a threshold [16]. Many enhancements have been proposed [18, 5, 15, 9, 14, 13], but despite the vast amount of expert knowledge spent on these methods, current, state-of-the-art solutions still suffer from high percentage of false alarms or low ratio of scan detection.

For example, a recently developed scheme by Jung [5] has better performance than earlier methods, but it requires that scanners attempt connections to several hosts on a destination port before they can be detected. This constraint limits the scheme's applicability because a sizable portion of the scanners attempt connections to only one host on each port in a typical observation window giving a net result of poor coverage.

Data mining techniques have been successfully applied to the generic network intrusion detection problem[8, 2, 10], but not to scan detection.[1]In this paper, we present a method for transforming network traffic data into a feature space that successfully encodes the accumulated expert knowledge. We show that an off-the-shelf classifier, Ripper[3], can achieve outstanding performance both in terms of missing only very few scanners and also in terms of very low false alarm rate. While the rules generated by Ripper make perfect sense from the domain perspective, it would be very difficult for a human analyst to invent them from scratch.

1.1 Contributions

This paper has the following key contributions:

- We formalize the problem of scan detection as a data mining problem and present a method for transforming network traffic data into a data set that classifiers are directly applicable to. Specifically, we formulate a set of features that encode expert knowledge relevant to scan detection.
- We construct carefully labeled data sets to be used for training and test from real network traffic data at the University of Minnesota and demonstrate that Ripper can build a high-quality predictive model for scan detection. We show that our method is capable of very early detection (as early as

[1]Scans were part of the set of attacks used in the KDD Cup '99 [1] data set generated from the DARPA '98/'99 data sets. Nearly all of these scans were of the obvious kind that could be detected by the simplest threshold-based schemes that simply look at the number of hosts touched in a period of time or connection window.

the first connection attempt on the specific port) without significantly compromising the precision of the detection.

- We present extensive experiments on real-world network traffic data. The results show that the proposed method has substantially better performance than the state-of-the-art methods both in terms of coverage and precision.

2 Background and Related Works

Until recently, scan detection has been thought of as the process of counting the distinct destination IPs talked to by each source on a given port in a certain time window [16]. This approach is straightforward to evade by decreasing the frequency of scanning. With a sufficiently low threshold (to allow capturing slow scanners), the false alarm rate can become high enough to render the algorithm useless. On the other hand, higher thresholds can leave slow and stealthy scanners undetected. A number of more sophisticated methods [9, 18, 15, 5, 4] have been developed to address the limitations of the basic method.

Robertson [15] assigns an anomaly score to a source IP based on the number of failed connection attempts it has made. This scheme is more accurate than the ones that simply count all connections since scanners tend to make failed connections more frequently. However, the scanning results still vary greatly depending on the definition of a failed connection and how the threshold is set. Lickie [9] uses a statistical approach to determine the likelihood of a connection being normal versus being part of a scan. The main flaw of this algorithm is that it generates too many false alarms when access probabilities are highly skewed (which is often the case.) SPICE [18] is another statistical-anomaly based system which sums the negative log-likelihood of destination IP/port pairs until it reaches a given threshold. One of the main problems with this approach is that it will declare a connection to be a scan simply because it is to a destination that is infrequently accessed.

The intuition behind SPICE is partly correct. It is true that destinations that are accessed only by scanners are rare, but the converse is not true. A scheme proposed by Ertöz et al. [4] assigns a scan score to each source IP on each destination. If the requested service is offered – regardless of how infrequently it is used – the score is not increased. If the requested service does not exist, the score is increased by the reciprocal of the log frequency of the destination. This scheme achieves fairly good performance, and is generally comparable to the TRW scheme in precision and recall that we describe next.

The current state-of-the-art for scan detection is Threshold Random Walk (TRW) proposed by Jung et al. [5]. It traces the source's connection history performing sequential hypothesis testing. The hypothesis testing is continued until enough evidence is gathered to declare the source either scanner or normal. Assuming that the source touched k distinct hosts, the test statistics (the likelihood ratio of the source being scanner or normal) is computed as follows:

$$\Lambda = \prod_{i=1}^{k} \begin{cases} \gamma_0 & \text{if the first connection to} \\ & \text{host } i \text{ fails} \\ \frac{1}{\gamma_1} & \text{otherwise,} \end{cases}$$

where γ_0 and γ_1 are constants. The source is declared a scanner, if Λ is greater than an upper threshold; normal, if Λ is less than a lower threshold. The thresholds are computed from the nominal significance level of the tests.

It is worth pointing out that in a logarithmic space, when $\log \gamma_0 = 1$, then $\log \Lambda$ is increased by one every time a first connection fails and is decreased by one every time a first connection succeeds. The threshold in this logarithmic space (the log of the threshold in the original space) is number of consecutive first-connection failures required for a source to be declared a scanner. For simplicity, *in the rest of the paper, when we say* threshold, *we will refer to the log-threshold.*

The authors of TRW recommend using a threshold of 4 (that is TRW is going to declared an IP scanner only after having made at least 4 observations). At this threshold, TRW can achieve high precision. Reducing the threshold to 1 will result in an unacceptably high rate of false alarms rendering TRW unable to reliably detect scans that only make one connection attempt within the observation period.

These false alarms are primarily caused by P2P and backscatter traffic. In recent years when the legality of certain uses of file sharing networks (based on P2P) has been questioned in the courtrooms, P2P networks started increasingly utilizing random ports to avoid detection. Peers agree upon a (practically) randomly chosen destination port that they conduct their communications on. They also maintain a list of the <IP address, destination port> pairs of their peers. Upon trying to re-connect to a P2P network, the host makes connection attempts to the hosts on its list of peers. The hosts on the list may not offer the service any more – e.g. they may be turned off or their dynamic IP address changed. To a scan detector, a P2P host unsuccessfully trying to to connect to its peers may appear as a scan [7, 6].

Backscatter traffic is another type of network traffic that can be easily mistaken for scan. Backscatter traffic typically originates from a server under denial-of-service (DoS) attack. In the course of a DoS attack, attackers send a server (the victim) such a large amount of network packets that the server becomes unable to render its usual service to its legitimate clients. The attackers also falsify the source IP and source port fields in these packets, so when the server responds,it sends its replies to the falsified (random) addresses. For a sufficiently large network, there can be enough falsified addresses that fall within the network space, such that replies from the victim will make it seem like a scanner: the victim is unsolicitedly sending packets to hosts that may not even exist [11].

3 Definitions and Method Description

In the course of scanning, the attacker aims to map the services offered by the target network. There are two general types of scans (1) **horizontal scans**, where the attacker has an exploit at his disposal and aims to find hosts that are exploitable by checking many hosts for a small set of services; and (2) **vertical scan**, where the attacker is interested in compromising a specific computer or a small set of specific computers. They often scan for dozens or hundreds of services.

Source IP, destination port pairs (SIDPs) are the basic units of scan detection; they are the potential scanners. Assume that a user is browsing the Web (destination port 80) from a computer with a source IP S. Further assume that S is infected and is simultaneously scanning for port 445. Our definition of a scanner allows us to correctly distinguish between the user surfing the Web (whose SIDP $< S, 80 >$ is not a scanner) from the SIDP $< S, 445 >$ which is scanning.

Scan Detection Problem Given a set of network traffic records (network traces) each containing the following information about a session (source IP, source port, destination IP, destination port, protocol, number of bytes and packets exchanged and whether the destination port was blocked), scan detection is a classification problem in which each SIDP, whose source IP is outside the network being observed, is labeled as `scanner` if it was found scanning or `non-scanner` otherwise.

Overview of the Solution The essence of our proposed method is the assumption that given a properly labeled training data set and the right set of features, data mining methods can be used to build a predictive model to classify SIDPs (as `scanner` or `non-scanner`).

In case of the scan detection problem, we will observe SIDPs over as long a time period as our computational resources allow and label them with high precision[2]. We will train a classifier – any off-the-shelf classifier – on this precisely labeled data and let the classifier learn the patterns characteristic of scanning behavior. Then we can apply this classifier to unlabeled data collected over a much shorter observation period and (as we will demonstrate later) successfully detect scanners.

The success of this method depends on (1) whether we can label the data accurately and (2) whether we have derived the right set of features that facilitate the extraction of knowledge. Section 3.1 and 3.2 will elaborate on these points.

Choice of classifier. Although most classifiers are applicable to our problem, some classifiers are better suited than others.

Our understanding of data mining classifier algorithms guided us towards choosing Ripper. We chose Ripper, because (a) the data is not linearly separable, (b) most of the attributes are continuous, (c) the data has multiple modes and (d) the data has unbalanced class distribution. Ripper can handle all of these properties quite well. Furthermore, it produces a relatively easily interpretable model in the form of rules allowing us to assess whether the model reflects reality well or if it is merely coincidental. An additional benefit is that classification is computationally inexpensive [3]. The drawback of Ripper is its greedy optimization algorithm and its tendency to overfit the training data at times. These drawbacks did not set us back too much; we encountered the overfit problem with visible effect only on one occasion.

3.1 Features The key challenge in designing a data mining method for a concrete application is the necessity to integrate the expert knowledge into the method. A part of the knowledge integration is the derivation of the appropriate features. Table 1 provides the list of features that we derived.

The first set of features (`srcip, scrport, dstport`) serve to identify a record; these features are not used for classification.

The second set of features contains statistics about the destination IPs and ports. These features provide an overall picture of the services and hosts involved in the source IP's communication patterns. The first feature in this group, `ndstips`, is the revered heuristic that defined early scan detection schemes. In addition, we provide features to show whether the source IP was using a few

[2]As we will explain in Section 3.2, despite our claims about the poor performance of the current scan detection schemes, labeling at a very high precision is possible under certain circumstances.

[3]Building the model is computationally expensive, but it can be performed off-line. It is the actual classification that needs to be carried out in real-time.

Table 1: The List of Features Extracted from the Network Trace Data

Feature	Description
`srcip`	Source IP
`srcport`	Source port or 0 for multiple source ports
`dstport`	Destination port
Destination IP and Port Statistics	
`ndstip`	Number of distinct destination IPs touched by the source IP
`ndstports`	Number of distinct destination ports touched by the source IP
`avgdstips`	Number of distinct destination IPs averaged over all destination ports touched by the source IP.
`maxdstips`	Maximum number of distinct destination IPs over all destination ports that the source IP touched.
Statistics Aggregated over All Destination Ports	
`server`	The ratio of (distinct) destination IPs that provided the service that the source IP requested.
`client`	The ratio of (distinct) destination IPs that requested service from the source IP on the destination port during the 24 hours preceding the experiment time.
`nosrv`	The ratio of (distinct) destination IPs touched by the source IP that offered no service on `dstport` to any source during the observation period.
`dark`	The ratio of (distinct) destination IPs that has been inactive for at least 4 weeks prior to the experiment date.
`blk`	The ratio of (distinct) destination IPs that were attempted connections to by the source IP on a blocked port during the experiment.
`p2p`	The ratio of (distinct) destination IPs that have actively participated in P2P traffic over the 3 weeks prior to the test date.
Statistics on Individual Destination Ports	
`i_ndstips`	Same definitions as above except measured on a single `dstport`.
`i_none`	
`i_dark`	
`i_blk`	

services on many hosts, or many services on a few hosts (`avgdstips, maxdstips`).

The third set of features have two goals. First, they help determine the role of the source IP: high values of `client` (percentage of inside IPs that are clients of the source IP) and low values of `server` (percentage of inside IPs that offer service to the source IP) indicate that the source IP is a server, otherwise it is a client. The remaining features `nosrv, dark, blk, p2p` allow us to assess the source IP's behavior.

The fourth set of features describe the role and behavior of the source IP on a specific destination port. The individual features serve the same purpose as their siblings in the third set. The importance of including the fourth group lies in the observation that certain source IPs exhibit vastly different behavior on some ports than on the majority of the ports they touch. An example could be a P2P host which is infected by a worm: the host is engaged in its usual P2P activity on a number of ports and it is also scanning on some ports.

3.2 Labeling the Data Set The goal of labeling is to assign each $<$ source IP, destination port $>$ pair (SIDP) a *label* that describes its behavior best. We distinguish between the following broad *behavior classes* (1) scanning (labeled as *SCAN*), (2) traditional client/server applications (*NRML*), (3) P2P (*P2P*) and (4) Internet noise (*NOISE*) [12].

While general scan detection schemes have been criticized for their high false alarm rates or low coverages, we claim that we are able to reliably label a set of SIDPs that appear within a short time window by observing their behavior over a long period. This is possible because our labeling scheme is different from earlier scan detection methods in the following key respects.

The most crucial difference lies in the length of the observation period. The sharp-eyed Readers may have noticed that some of the features in Table 1 include information about inactive IPs and P2P hosts. Automatically constructing an accurate list of inactive IPs or P2P hosts requires very long observations. We use 22 days of traffic data (95 GB of compressed netflow data) to construct these lists. Fortunately, the changes to these lists are marginal and hence they can be continuously updated during normal operation.

In sharp contrast to the above lists, features in Table 1 are so dynamic that under the performance constraints of production use (real-time classification), we can only maintain information within a 20-minute window: scan detection must be done based on 20 minutes of observation. On the other hand, the labeling of the training set can be performed off-line. Upon training, we select a 20-minute window and observe the source IPs that were active in that 20-minute window for 3 days. The extra observations we obtained by watching the IPs for 3 days enables us to label them at considerably higher confidence.

Even existing schemes are capable of scan detection at high precision – at high thresholds. High thresholds require more observations, causing the coverage to become intolerably small. By drastically increasing the time window (from 20 minutes to 3 days), we provide additional observations that helps increase the coverage.

Second, existing scan detection methods observe the behavior of the source IPs on specific ports separately. In our labeling scheme, on top of examining the activities of a source IP on each individual destination port separately, we also correlate their activities across ports. Certain types of traffic, most prominently P2P and backscatter, can only be recognized when information is correlated across destination ports.

Third, practical scan detection schemes have requirements such as being able to run in real-time. As we have discussed before, training (labeling the training data and building the data mining model) can be performed off-line. This allows us to perform more expensive computations including tracing the connection history of source IPs for 3 days instead of 20 minutes.

For the details of the labeler, the Reader is referred to [17].

4 Evaluation

4.1 Description of the Data Set For our experiments, we used real-world network trace data collected at the University of Minnesota between the 1st and the 22nd March, 2005. The University of Minnesota network consists of 5 class-B networks with many autonomous subnetworks. Most of the IP space is allocated, but many subnetworks have inactive IPs. We collected information about inactive IPs and P2P hosts over 22 days and we used 03/21/2005 and 03/22/2005 for the experiments. To test generalizability (in other words to reduce possible dependence on a certain time of the day), we took samples every 3 hours and performed our experiments on each sample.

As far as the length of each observation period (sample) is concerned, longer observations result in better performance but delayed detection of scans. Therefore, in production use, the scan detector will operate in a streaming mode, where the periods of observation will vary in length across SIDPs: lengths will be kept to the minimal amount sufficient for conclusive classification of each SIDP. Now, the system works in batch mode, so we keep as many observations as possible. Our memory (1 GB) allows us to store and process 4 million flows, which approximately corresponds to 20 minutes of traffic.

Table 2 describes the traffic in terms of number of <source IP, destination port> (SIDP) combinations pertaining to scanning-, P2P- and normal traffic and Internet noise.

The proportion of normal traffic appears small. This has two reasons: (a) arguably the dominant traffic of today's Internet is P2P [7] and (b) even though P2P is also "normal" traffic, according to our definition, the normal behavior class consists of the traditional client/server type traffic which excludes P2P.

Table 2: The distribution of (source IP, destination ports) (SIDPs) over the various traffic types for each traffic sample

ID	Day.Time	Total	scan	p2p	normal	noise	dknw
01	0321.0000	67522	3984	28911	6971	4431	23225
02	0321.0300	53333	5112	19442	9190	1544	18045
03	0321.0600	56242	5263	19485	8357	2521	20616
04	0321.0900	78713	5126	32573	10590	5115	25309
05	0321.1200	93557	4473	38980	12354	4053	33697
06	0321.1500	85343	3884	36358	10191	5383	29527
07	0321.1800	92284	4723	39738	10488	5876	31459
08	0321.2100	82941	4273	39372	8816	1074	29406
09	0322.0000	69894	4480	33077	5848	1371	25118
10	0322.0300	63621	4953	26859	4885	4993	21931
11	0322.0600	60703	5629	25436	4467	3241	21930
12	0322.0900	78608	4968	33783	7520	4535	27802
13	0322.1200	91741	4130	43473	6319	4187	33632

Other than the distribution of the SIDPs over the different behavior classes, the traffic distribution is as expected. The proportion of normal traffic is highest during business hours, P2P – for the largest part attributed to students living in the residential halls – peaks late afternoon and during the evening, and scans are most frequent in the early morning hours. The number of "don't know"s, that is traffic with insufficient observation, seems to be more correlated with normal and P2P traffic than with scanning traffic. The patterns repeat during the next day.

Evaluation Measures The performance of a classifier is measured in terms of precision, recall and F-measure. For a contingency table of

	classified as Scanner	classified as not Scanner
actual Scanner	TP	FN
actual not Scanner	FP	TN

$$\begin{aligned} \text{prec} &= \frac{\text{TP}}{\text{TP}+\text{FP}} \\ \text{recall} &= \frac{\text{TP}}{\text{TP}+\text{FN}} \\ \text{Fm} &= \frac{2*\text{prec}*\text{recall}}{\text{prec}+\text{recall}}. \end{aligned}$$

Less formally, precision measures the percentage of scanning (source IP, destination port)-pairs (SIDPs) among the SIDPs that got declared scanners; recall measures the percentage of the actual scanners that were discovered; F-measure balances between precision and recall.

4.2 Comparison to the State-of-the-Art To obtain a high-level view of the performance of our scheme, we built a model on the 0321.0000 data set (ID 1) and tested it on the remaining 12 data sets. The performance results were compared to that of Threshold Random Walk (TRW), which is considered to be the state-of-the-art. Figure 1 depicts the performance of our proposed scheme and that of TRW on the same data sets.

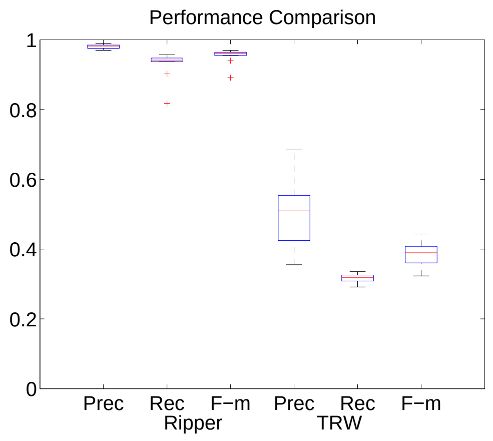

Figure 1: Performance comparison between the proposed scheme and TRW. From left to right, the six box plots correspond to the precision, recall and F-measure of our proposed scheme and the precision, recall and F-measure of TRW. Each box plot has three lines corresponding (from top downwards) to the upper quartile, median and lower quartile of the performance values obtained over the 13 data sets. The whiskers depict the best and worst performance.

One can see that not only does our proposed scheme outperform the competing algorithm by a wide margin, it is also more stable: the performance varies less from data set to data set (the boxes on Figure 1 appear much smaller).

Figure 2 shows the actual values of precision, recall and F-measure for the different data sets. The performance in terms of F-measure is consistently above 90% with very high precision, which is important, because high false alarm rates can rapidly deteriorate the usability of a system. The only jitter occurs on data set # 7 and it was caused by a single source IP that scanned a single destination host on 614(!) different destination ports meanwhile touching only 4 blocked ports. Not only is such behavior uncharacteristic of our network (the only other vertical scanner we saw was the result of a 'ScanMyComputer' service touching as many blocked ports as expected), but it is more characteristic of P2P traffic causing the misclassification of this scanner.

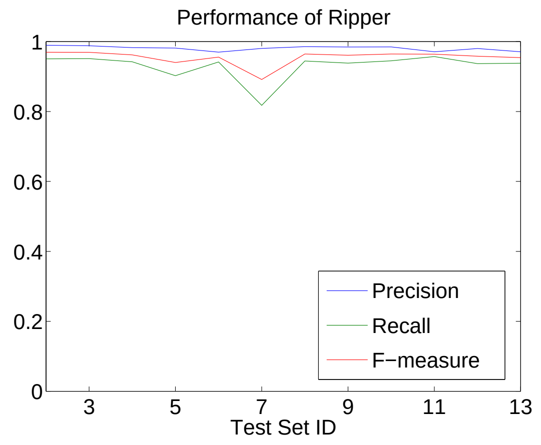

Figure 2: The performance of the proposed scheme on the 13 data sets in terms of precision (topmost line), F-measure (middle line) and recall (bottom line). The model was built on data set ID 1.

As discussed earlier, the authors of TRW recommend a threshold of 4. In our experiments, we found, that TRW can achieve better performance (in terms of F-measure) when we set the threshold to 2, this is the threshold that was used in Figure 1, too.

In Figure 3, we examine how the TRW's performance depends on the threshold. The five box plots represent the performance at the five thresholds between 1 and 5. The horizontal lines of the boxes correspond to the upper quartile, median and lower quartile of the 13 performance measurements (one for each of the 13 data sets) and whiskers extend out to the extreme performance values.

Figure 3 shows that the threshold balances between precision and recall. At low thresholds (such as 1), TRW achieves high recall by declaring anything a scanner that makes one failed connection more than successful connections. As a consequence, a lot of false alarms are generated, typically caused by P2P and backscatter. The other extreme is the high threshold of 5, where only the most obvious scanners are caught. P2P and backscatter are not declared scanners at such high thresholds, because the chances of them making connection attempts to 5 distinct destination IPs on the same random destination port are slim.

In the followings we show that the success of the data mining approach stems from its ability to correlate the behavior of a source IP on multiple hosts. This allows both early detection and accurate categorization of scanning-like non-scanning traffic.

To illustrate the benefits of data mining and to explain the above results, let us consider the concrete ex-

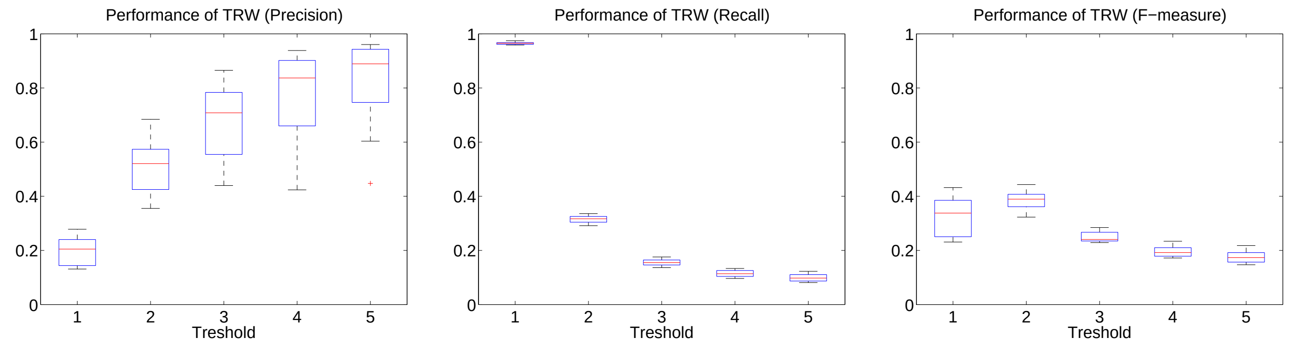

Figure 3: Box plots of the precision (left), recall (middle) and F-measure (right) of TRW at five different thresholds (horizontal axis).

ample of the data set 0321:1200, where our proposed approach achieved close to median of the 13 performance values.

Claim 1. *The correlation of evidence across destination ports allows the data mining approach to detect scanners earlier, in some cases as early as the first connection attempt on a destination port (provided that the same source IP previously made connection attempts to other ports).*

Intuitively, this claim is true, since positive evidences from different data sources can certainly increase the confidence in our decisions. This is widely recognized as joint learning.

In order to experimentally demonstrate our claim, we look at a portion of the traffic that made connection attempts to no more than one destination IP on any destination port within the 20-minute window.

Out of the 93,557 SIDPs, 66,872 satisfy this condition, which is a very large portion of the traffic. Out of the 66,872 SIDPs, we were unable to label 29,397. The SIDPs that we were able to label have the following distribution:

SCAN	*P2P*	*NRML*	*NOISE*	Total
3,014	28,078	5,995	388	37,475

In Table 3, we present the results of our proposed scheme and that of TRW at a threshold of 1. Since no SIDP made connection attempts to two distinct destination IPs on the same port, TRW at higher thresholds would not find scanners.

The results show that TRW1 found more scanners, but its false alarm rate was unacceptably high. Most of the false alarms are due to P2P and Internet noise.

For completeness, in Table 4 we show the number of SIDPs that we were unable to label but got declared scanners by the two schemes.

Table 3: The performance of the data mining approach and some competing schemes on the portion of the data set that contains source IPs that never made connection attempts to more than 1 destination IP on a single destination port. 'TP' denotes the number of SIDPs classified correctly as scanner, 'FN' the scanning SIDPs that did not get identified and 'FP' the SIDPs that were predicted to scan but they did not.

Scheme	TP	FN	FP	Prec	Rec	Fm
TRW1	2908	106	23760	0.1090	0.9648	0.1959
Ripper	2630	384	54	0.9799	0.8726	0.9231

Table 4: The number of SIDPs whose true nature is unknown classified as scanner by the different schemes

Scheme	TRW1	Ripper
Unlabeled Classified as Scanner	17,407	1,144

Although not directly relevant to proving Claim 1, in order to present the Reader with a full set of results, in Table 5, we show the performance of the above schemes on the complementary data set, namely the portion of the data set that contains the SIDPs that made connection attempts to at least two distinct destination IPs on at least one port.

There are 26,685 SIDPs satisfying this condition, out of which we managed to label 22,385. The distribution of traffic classes over these 22,385 SIDPs is as follows.

SCAN	*P2P*	*NRML*	*NOISE*	Total
1,459	10,902	6,359	3,665	22,385

Since it can be advantageous for TRW to use higher thresholds on this portion of the data, we present results for TRWk, where the threshold is k, $k = 1, \dots, 5$.

TRW exhibits the behavior described earlier: higher thresholds result in increasing precisions but rapidly

Table 5: The performance of the data mining approach and the competing schemes on the complementary (to Table 3) portion of the data set.

Scheme	TP	FN	FP	Prec	Rec	Fm
TRW1	1389	70	8952	0.1343	0.9520	0.2354
TRW2	1304	155	1760	0.4256	0.8938	0.5766
TRW3	645	814	261	0.7119	0.4421	0.5455
TRW4	485	974	45	0.9151	0.3324	0.4877
TRW5	447	1012	18	0.9613	0.3064	0.4647
Ripper	1406	53	23	0.9839	0.9637	0.9737

deteriorating coverage. In terms of F-measure, it achieved its peak performance at a threshold of 2, but still falling behind Ripper by a wide margin.

Claim 2. *Through correlating evidence across destination ports, the data mining approach is capable of filtering out scanning-like non-scanning traffic types such as P2P or backscatter.*

Certain types of traffic, including P2P and Internet noise – most prominently backscatter – apart from some obvious cases (touching only P2P hosts on a certain ports) can only be recognized by observing the behavior of the source IP across ports. Table 6 presents the performance of the data mining based scheme and some competing schemes on P2P and scanning traffic only.

We are using the same model (built on the first data set, 0321:1200) as in case of Claim 1. We continue using the same test set (0321.1200), but we removed all instances that are not scanners or P2P. There are 38,980 P2P and 4,473 scanning SIDP's in this portion of the test set.

The competing schemes are TRW at a threshold of 1 (TRW-1), TRW at a threshold of 2 (TRW-2) and TRW-P, which is basically the TRW-1 scheme but we applied a simple P2P filtering: any source IP that makes connection attempts to a P2P hosts is not considered a scanner on that destination port.

Table 6: Performance Comparison on P2P and Scanning Traffic.

Scheme	TP	FN	FP	Prec	Rec	Fm
TRW-1	1715	1519	16414	0.0946	0.5303	0.1606
TRW-2	137	3097	1039	0.1165	0.0424	0.0621
TRW-P	594	2640	6774	0.0806	0.1837	0.1121
Ripper	2878	356	62	0.9789	0.8899	0.9323

We evaluated the same set of schemes on backscatter traffic, too. We use the same model and the same test set (0321:1200), but this time we removed all instances that were not noise or scanners from the original test set. There are 4,473 scanner and 4,053 noise SIDPs left. Table 7 presents the results.

Table 7: Performance Comparison on Internet Noise and Scanning Traffic.

Scheme	TP	FN	FP	Prec	Rec	Fm
TRW-1	1562	1323	2325	0.4019	0.5414	0.4613
TRW-2	118	2767	163	0.4199	0.0409	0.0745
TRW-P	538	2347	747	0.4187	0.1865	0.2580
Simple	219	2666	452	0.3264	0.0759	0.1232
Simple-P	184	2701	404	0.3129	0.0638	0.1060
Ripper	2549	336	0	1.0000	0.8835	0.9382

The results for P2P and backscatter are similar. While our proposed scheme's performance is characterized by very high precision, high recall, it is evident that without recognizing P2P, high performance is unattainable.

This comparison was not exactly fair, since in the original paper, TRW does not have P2P information at its disposal. However our experiments on the TRW-P scheme show, that even when information about P2P is available, it is not straightforward how one can use it. One needs to correlate ports in order to successfully distinguish P2P from scanners.

Claim 3. *The data mining approach was capable of extracting the characteristics of scanning behavior from the long (3-day) observation and it was subsequently capable of applying this knowledge to successfully classify data from short (20-minute) observation.*

Table 8 depicts the rules generated by Ripper on 0321:000. In general, the rules make sense. In fact, Rule #2, which is responsible for the lion's share of the work is essentially an enhanced version of TRW's heuristic (of failed vs. successful first connections). Similarly to TRW, Rule #2 also takes failure (and success) into account but it goes further by correlating this knowledge across ports. It does not stop there, it also correlates it with other evidence, the usage of blocked ports.

Rule #1 did not get invoked at all possibly indicating to the Reader that it may not be as useful in practice. Nevertheless, Rule #1 does makes sense, too. It did not get invoked simply because it is geared towards blocked scanners – SIDPs who scan multiple hosts on multiple ports. This test data set contained no such scanners; our scanners (in this data set) either scanned vertically (with half of the ports blocked) or horizontally, where less than half of the connection attempts were blocked (but may have touched inactive IPs). The former scanners were caught in large by Rule #2 and the horizontal scanners were caught by the rest of the rules.

From the `ndstips>=2` and `ndstports<=2` conditions, it is easy to see that Rule #3 is responsible for

Table 8: The rules and their interpretations for identifying scanners built on 0321.0000. 'Cnt' stands for the number of invocations on the test set 0321.1200.

ID	Cnt	Rule
1	0	`blk >= 0.5 nosrv >= 0.6667 indstips >= 2` More than half of the destination IPs touched were touched on blocked destination ports, mostly service was not offered and the source IP touched at least 2 distinct IPs on the current destination port.
2	3764	`blk >= 0.5 nosrv >= 0.7778` More then half of the IPs were touched on blocked destination ports and service is almost never offered.
3	35	`iblk >= 1 ndstips >= 2 ndstports <= 2` Connection attempts were made on no more than 2 distinct destination ports to at least 2 distinct destination IPs and the current destination port is blocked.
4	23	`iblk >= 1 pp <= 0.75 ndstips <= 4 ndstports >= 5` The source IP touched at least 5 destination ports out of which at least the current port is blocked, the number of distinct destination IPs is no more then 4 and at least one of them is not a P2P host. q
⋮	6	[truncated]

identifying horizontal scanners, where the requirement of one of the (possibly two) ports being blocked provides the evidence of scanning activity.

In our scheme, we included a whole set of features to enable the recognition of exceptional behaviors. In this rule set, Rule #4 has the ability to recognize such source IPs. Imagine a source IP, that engages in normal traffic on a variety of ports but perform scans on one. As long as this source IP made connection attempts on more than 5 destination ports, and scans on one (which is blocked), Rule #4 will catch it. The caveat is that P2P traffic can also exhibit this kind of behavior. P2P false alarms are avoided by the `pp<=.75` clause.

In the followings, let us have a look at concrete examples describing how the rule set supports our first two claims.

Early Detection. One of the most difficult tasks is to correctly recognize scanners who make connection attempts to only one destination IP on each destination port. The only hard evidence we can have against such a source IP is the unexpectedly large number of connection attempts on blocked ports. Either Rule #1 or Rule #2 will identify such scanners.

Rule # 3 is another example of correlating information. It says that if a source IP makes connection attempts to a blocked port then making connection attempts on only one other port to at least two distinct hosts is suspicious. Only P2P (using random ports) and backscatter are non-scanning traffic types making connection attempts to blocked ports. However, the probability of randomly selecting the same port for two distinct destination IPs is negligible, hence such behavior is indeed very suspicious.

Detection of P2P. Rule # 4 is the only rule that explicitly uses the P2P feature. P2P feature is not required for the detection of P2P, false alarms due to P2P can be avoided by using the blocked ports. While P2P can possibly make connection attempts on blocked ports (randomly chosen port), it is highly unlikely that it would attempt a large portion of its communication on blocked ports. The first three rules use 50% as the 'large portion', while Rule #4 explicitly uses the P2P feature.

Backscatter. There is no explicit feature provided to help recognize backscatter traffic. Fortunately, (most) P2P and backscatter traffic share the commonality of randomly choosing destination ports, so the mechanism used to filter out P2P can also be used to filter out backscatter: it is highly unlikely that a large portion of the randomly chosen set of destination ports for backscatter traffic will be blocked.

Table 9: Variants of Rule # 2 in Models Built on Different Time Intervals

02	`blk >= 0.5 nosrv >= 0.5476 .`
03	`iblk >= 1 nosrv >= 1 blk >= 0.5 .`
04	`blk >= 0.3333 nosrv >= 0.8889 .`
08	`blk >= 0.5 nosrv >= 1 .`
09	`blk >= 0.4 nosrv >= 0.75 .`
11	`blk >= 0.3333 nosrv >= 0.5 iblk >= 1 .`
12	`blk >= 0.3333 nosrv >= 0.5714 .`
13	`blk >= 0.3333 nosrv >= 0.875 .`

4.2.1 Variations to Rule #2

As mentioned before, Rule #2 has the largest contribution to the high performance of our proposed scheme. Since this rule is so dominant, yet appears very generic, it is reasonable to assume that models built on other data sets contain similar rules. Indeed, examination of the models built on the other 12 data sets revealed that 8 other data sets have close variants of Rule #2. Table 9 shows these rules.

The fact that the rule appears with a wide range of thresholds (blk varying between .3 and .5 and nosrv varying between .54 and 1) reassures us that the rule is indeed generic.

Figure 4 depicts the performance of Ripper (in terms of F-measure) on the test set 0321.1200 for nosrv and blk thresholds varying between 0 and 100 %. F-measure values less than .85 were encountered only when nosrv or blk was 0. To enhance visibility, we replaced all values (101 values) less than .85 with .85.

The results show that the dependence of the performance upon a correctly chosen threshold is minimal. For blk, the performance is practically constant as long as blk$\geq$ 50% (and experiences a maximum of 2% drop according to F-measure and precision for blk>1%); for nosrv, it performs best between 1% and 85%.

These results are not surprising. In [5], Jung has pointed out that network traffic has a bimodal distribution in terms of the ratio of $<$ destination IP, port$>$ pairs offering the requested service: either almost all of them or almost none of destinations offered the requested service. As far as the blk feature is concerned, there is a logical explanation for its characteristics. There are $N = 2^{16}$ ports, out of which B are blocked. If a source IP randomly selects a set of n ports, then the number b of blocked port in the set follows Binomial(n, B/N). Since $B << N$, even for small b's the probability of observing such a random port set will be very small. Hence the scanning-like non-scanner source IPs (P2P, backscatter), who select ports randomly, will have very small b values, while scanners, who do not select ports randomly and tend to make connection attempts to blocked ports will have larger b values. (The step at blk=50% is caused by the relatively frequent case of 1 blocked port out of 2.)

Now that we have concluded that the performance is stable for a wide range of blk and nosrv thresholds on 0321:1200, we investigate the variability in performance across data sets.

Figure 5 illustrates the variation in performance observed for a wide range of thresholds over the 13 data sets. For the boxplots in the left pane, blk was fixed at 50 % (a threshold that many models built on various data sets found best) and nosrv was varied. The set of values it took was ({0, 5, 10, 50, 75, 80, 85, 90, 95, 100}) reflecting our prior experience that the performance varied only at low values (0 to 1%) and at around 85%. The small sizes of the boxes indicate that the variation in performance from data set to data set is negligible. For the boxplots in the right pane, nosrv was fixed at 75% (in accordance with our observation that the performance is best between 1% and 85%; 75% is close to 77.78% chosen by Ripper) and selected the set {0, 1, 5, 45, 50, 55, 90, 95, 100} reflecting our interest in the change from 0 to 1% and the slight improvement around 50%. Once again, the variation from data set to data set is negligibly small.

In summary, we gave a logical explanation why Rule #2's performance should not change from data set to data set and why it is expected to be high over a wide range of thresholds. We also verified it experimentally. In addition we showed – from a security perspective – that the rules built by Ripper are reasonable and we also verified this claim experimentally. Consequently, we can conclude that our Claim 3 is proven. What we discovered, however, is more far-reaching than just an experimental proof. We saw that Ripper, using our framework, automatically discovered a heuristic (Rule #2) that is more powerful than the heuristics we have seen in the literature so far.

4.3 Generalizability We approach the question of generalizability from two angles. First, we investigate how the performance of a model changes when we apply it to different test sets. Second, we investigate how consistent with each other are the models built on different data sets.

In Section 4.2 we gave an answer to the first question. The results are consistently good with only data set ID 7 experiencing a slight drop due to the misclassification of the unexpected vertical scanner.

Next, we demonstrate that Ripper performs well regardless of which data set we selected for building the model on. Figure 6 depicts the performance of various models on the 13 test sets. The horizontal axis corresponds to the ID of the data set the model was built on, the vertical axis corresponds to the data set ID that the model was tested on. The performance is measured in terms of precision, recall and F-measure, lighter colors indicating better performance.

In general, the results are good and the differences between the models are marginal (In Figure 6 the color within each row does not change much), typically no more than 1% on the same data set.

The variability of the performance from data set to data set is higher. Particularly, data set ID 7 (0321:1800) sticks out. This is the data set that contained the vertical scanner. Since no other data set contained such a scanner, all models experienced a 2% drop in precision on this data set. Conversely, the model built on this data set (column 7 in Figure 6) was naturally able to recognize the vertical scanner and at the same time its performance on other data sets was comparable to that of other models (with the exception of test set ID 9, where a pair of IPs were talking to each other on 872 different ports appearing to Ripper as vertical scanners).

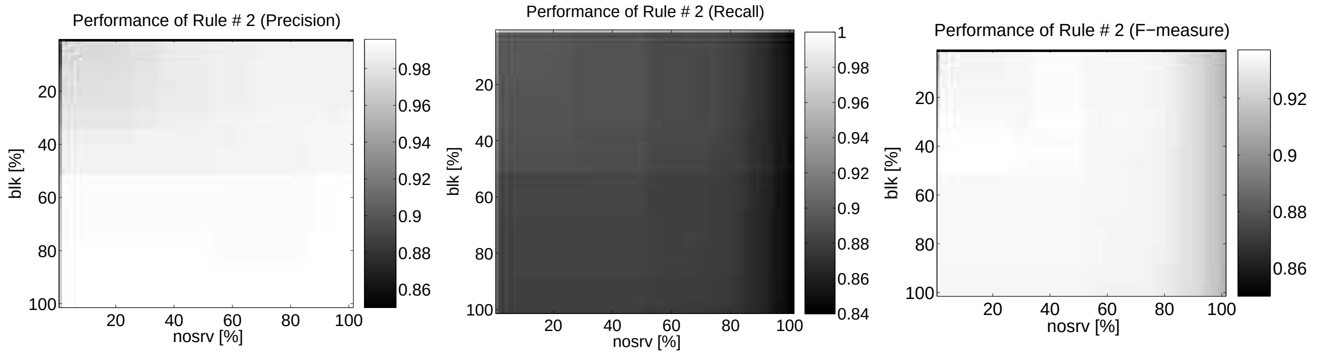

Figure 4: The performance of Rule #2 on 0321.1200 at `blk` and `nosrv` thresholds varying between 0 and 100% in terms of precision (left), recall (middle) and f-measure (right)

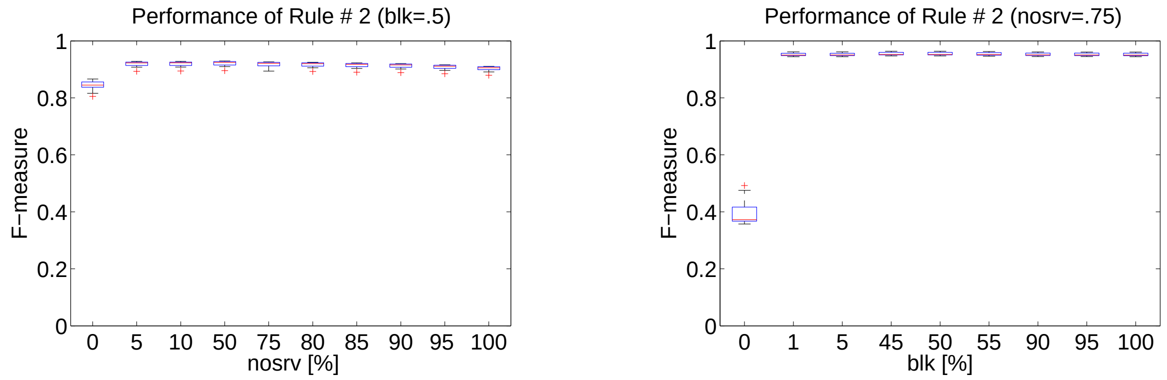

Figure 5: Illustration of the variation of Rule #2's performance at different thresholds over the 13 data sets. The figure on the left depicts the variation for a set of `nosrv` thresholds at fixed `blk=50%`, while the figure on the right depicts the variation for a set of `blk` thresholds at fixed `nosrv=75%`

As this example illustrates, the limitation of the approach is that (i) only scanning modes that are present in the data set can be learnt and that (ii) rare scanning modes (scanning modes that occur in few data sets) appearing in a single data set in large amounts (e.g. the massive vertical scan in 0321:1800) can cause Ripper to overfit the training data.

5 Summary and Conclusion

In this paper, we have introduced a method for formalizing the scan detection problem as a classification problem solvable with data mining techniques. We proposed a method for transforming network trace data into data sets that off-the-shelf classifiers can be run on. We selected Ripper, a fast, rule-based classifier, because it is particularly capable of learning rules from multi-modal data sets and it provides results that are easy to interpret. Moreover, we demonstrated that data mining methods, given a carefully labeled training set and an appropriate set of features, are capable of extracting knowledge from observations made over a long period of time and applying this knowledge successfully to observations made over only a short period of time.

We found that by using our data mining framework, we achieved a substantial improvement in coverage, a factor of close to 9, at improved precision over the state-of-the-art heuristic-based scan detector, TRW at its author's suggested threshold of 4.

We demonstrated that the gain stems from the classifier's ability of detecting scanners early (as early as the first connection attempt on a specific port) at high precision and recall, and its ability to avoid false detection of scanning-like non-scanning traffic such as P2P and backscatter. We demonstrated that Ripper indeed extracted the behaviors characteristic of scan detection: the model it built made perfect sense from a network security perspective although due its complexity it would be difficult for a human analyst to discover it. We zoomed in onto the rule that dominated the rule set (in terms of number of invocations) and found that the heuristic (encoded in form of a rule) was so powerful that it alone could outperform the competing schemes

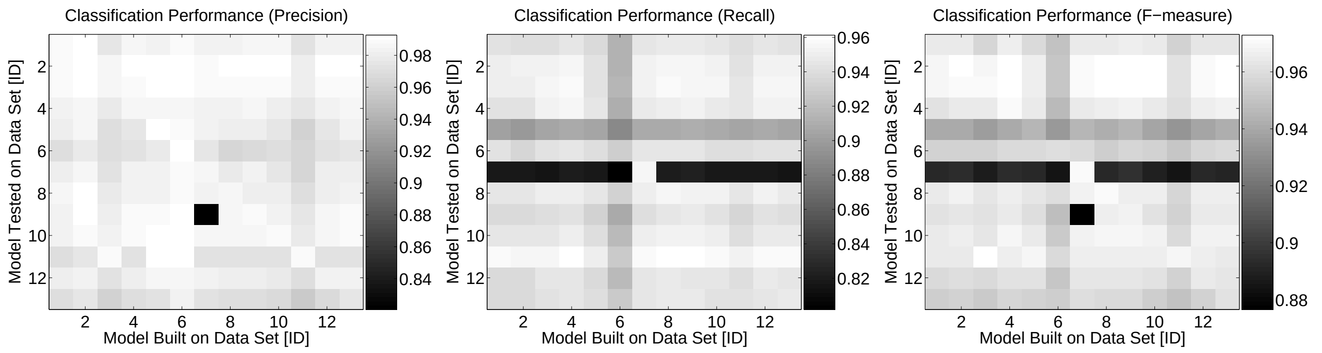

Figure 6: Generalizability of the scheme. The performance of different model on different data sets in terms of precision, recall and F-measure (from left to right)

(TRW at various thresholds).

References

[1] Kdd cup '99 data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[2] Daniel Barbara, Ningning Wu, and Sushil Jajodia. Detecting novel network intrusions using bayes estimators. In *SDM*, 2001.

[3] William W. Cohen. Fast effective rule induction. In *ICML*, 1995.

[4] Levent Ertoz, Eric Eilertson, Paul Dokas, Vipin Kumar, and Kerry Long. Scan detection - revisited. Technical Report AHPCRC 127, University of Minnesota – Twin Cities, 2004.

[5] Jaeyeon Jung, Vern Paxson, Arthur W. Berger, and Hari Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *IEEE Symposium on Security and Privacy*, 2004.

[6] Thomas Karagiannis, Andre Broido, Nevil Brownlee, and kc claffy. Is p2p dying or just hiding? In *IEEE Globecom 2004 "Emerging Technologies Applications and Services"*, 2004.

[7] Thomas Karagiannis, Andre Broido, Michalis Faloutsos, and kc claffy. Transport layer identification of p2p traffic. In *International Measurement Conference (IMC)*, 2004.

[8] Wenke Lee, Salvatore J. Stolfo, and Kui W. Mok. Mining audit data to build intrusion detection models. In *KDD*, 1998.

[9] C. Lickie and R. Kotagiri. A probabilistic approach to detecting network scans. In *Eighth IEEE Network Operations and Management*, 2002.

[10] Matthew V. Mahoney and Philip K. Chan. Learning rules for anomaly detection of hostile network traffic. In *ICDM*, 2003.

[11] David Moore, Geoffrey M. Voelker, and Stefan Savage. Inferring internet denial-of-service activity. In *USENIX Security Symposium*, 2001.

[12] Ruoming Pang, Vinod Yegeswaran, Paul Barford, Vern Paxson, and Larry Peterson. Characteristics of internet background radiation. In *Internet Measurement Conference (IMC)*, 2004.

[13] V. Paxon. Bro: a system for detecting network intruders in real-time. In *Eighth IEEE Network Operators and Management Symposium (NOMS)*, 2002.

[14] Phillip A. Porras and Alfonso Valdes. Live traffic analysis of tcp/ip gateways. In *NDSS*, 1998.

[15] Seth Robertson, Eric V. Siegel, Matt Miller, and Salvatore J. Stolfo. Surveillance detection in high bandwidth environments. In *DARPA DISCEX III Conference*, 2003.

[16] Martin Roesch. Snort: Lightweight intrusion detection for networks. In *LISA*, pages 229–238, 1999.

[17] Gyorgy Simon, Hui Xiong, Eric Eilertson, and Vipin Kumar. Scan detection: A data mining approach. Technical Report AHPCRC 038, University of Minnesota – Twin Cities, 2005.

[18] Stuart Staniford, James A. Hoagland, and Joseph M. McAlerney. Practical automated detection of stealthy portscans. *Journal of Computer Security*, 10(1/2):105–136, 2002.

Acknowledgements. This work was partially supported by the Army High Performance Computing Research Center contract number DAAD19-01-2-0014, by the ARDA Grant AR/F30602-03-C-0243 and by the NSF grants IIS-0308264 and ITR-0325949. 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.

Special thanks to Paul Dokas, the chief security analyst at the University of Minnesota for his help in labeling the data and understanding the problem.