2006

Black Box Anomaly Detection: is it Utopian?

Shobha Venkataraman
Carnegie Mellon University

Juan Caballero
Carnegie Mellon University

Dawn Song
Carnegie Mellon University

Avrim Blum
Carnegie Mellon University

Jennifer Yates
AT&T Labs-Research

Black box anomaly detection: is it utopian?

Shobha Venkataraman*, Juan Caballero*, Dawn Song*, Avrim Blum*, Jennifer Yates†
*Carnegie Mellon University †AT&T Labs-Research

ABSTRACT

Automatic identification of anomalies on network data is a problem of fundamental interest to ISPs to diagnose incipient problems in their networks. ISPs gather diverse data sources from the network for monitoring, diagnostics or provisioning tasks. Finding anomalies in this data is a huge challenge due to the volume of the data collected, the number and diversity of data sources and the diversity of anomalies to be detected.

In this paper we introduce a framework for anomaly detection that allows the construction of a black box anomaly detector. This anomaly detector can be used for automatically finding anomalies with minimal human intervention. Our framework also allows us to deal with the different types of data sources collected from the network. We have developed a prototype of this framework, TrafficComber, and we are in the process of evaluating it using the data in the warehouse of a tier-1 ISP.

1 INTRODUCTION

ISPs collect large amounts of data from their networks into warehouses and use this information for provisioning, analysis and generally to guarantee the health of the network. Given this wealth of information, ISPs are interested in using anomaly detection techniques on this collected data to diagnose incipient problems before they can significantly impact the network.

The network data collected varies depending on the ISP's storage resources and monitoring capabilities but is generally characterized by its volume and diversity. The volume of the data collected, which can be in the order of gigabytes per day for a large national ISP, makes manual analysis of the data infeasible. In addition to the volume, the diversity of the data is also daunting. An ISP could expend significant time studying and modeling one feature from a single data source and only gain insight about a drop in a sea of anomalies. ISPs need a scalable approach that automates this process and requires no previous knowledge or analysis of the behavior of the data.

ISPs currently collect measurements such as: byte counts, link error counts and CPU utilization from SNMP data; periodic snapshots of the network topology; system logs from the servers and routers; end to end path measurements such as loss or delay; physical measurements such as current through an optical amplifier; configuration files; and many others. Previous approaches for anomaly detection have usually focused on a small set of data sources, usually packet traces or SNMP data, and a small set of features like volume, byte counts, or IP header features [2, 5, 7, 8, 18, 20, 23, 24]. This kind of approach only explores a few points in the anomaly detection space and would require an ISP to operate multiple specific anomaly detection systems working in parallel, requiring network analysts to work with different outputs from each system, resulting in high operating costs.

In this paper we propose a different approach. We seek answers to the following question: to what extent can we build a black box anomaly detector that automatically finds anomalies in any data source and any feature gathered from the network? This approach would allow an ISP to deal with the volume and diversity of the collected data in a scalable and comprehensive way. We introduce a general framework that splits the problem into two: 1) transforming multiple data sources into a common input and 2) building a black box anomaly detector that, given that input, automatically outputs multiple types of anomalies. In this paper, we tackle the following problems:

Finding novel anomalies not yet seen in the data: The traditional approach to automatic detection of anomalies is to use machine learning algorithms on samples of both normal data and anomalies, and train a good classifier to separate these samples [4, 8, 13, 16]. The basic problem with this approach is that it limits us to the detection of anomalies *which are already present in the samples*. Thus, our approach is different – we aim to find features of the data that consistently behave in the same way in normal data. Then, when we see new, possibly mixed data, we can use this behavior to decide if the data is normal or contains anomalies. In machine learning terminology, we view the problem as *learning only with positive examples* [17].

Identifying features of interest: Given a data source, we are interested in detecting changes in the behavior of the data source. One could use several features from the same data source for detecting the change but not all of them may be equally useful. For example, it is difficult to detect anomalies in a feature that exhibits a lot of fluctuation and very little regularity in normal data. Our framework automates the exploration of multiple features from the same data source and the identification of those that have a consistent behavior in the normal data, according to the models and metrics defined, and thus can be used by the black box anomaly detector to detect changes in the data source.

Detecting anomalies without domain knowledge: Given a set of features of interest, our black box anomaly detector finds anomalies in those features, using no apriori domain knowledge about the behavior of those features. This is done by automatically extracting the ex-

pected behavior from the normal data and then flagging significant deviations from the expected behavior as anomalies. Thus, our anomaly detector flags changes in behavior with respect to the training data.

Handling multiple data source types: Our framework takes into account the diversity of data sources currently collected by ISPs and how they can be transformed into the proper input for the black box anomaly detector.

Gradually increasing the scope of the detection: The black box anomaly detector provides feedback on when a model does not apply to a feature, which allows us to gradually add new models as they are needed. Since each model allows to detect specific families of anomalies, it also allows us to gradually add new models that allow detection of more sophisticated families of anomalies.

2 FRAMEWORK OVERVIEW

We split the question of how to build a black box anomaly detector that handles multiple data sources and features into two smaller questions, that we address in turn. The first one is: can we transform the different types of data sources collected from the network into a common input that can be used by our black box anomaly detector? The second one is: how do we build that black box anomaly detector? In Section 3 we deal with the issue of different data source types and then in Section 4 we explain how to build the black box anomaly detector.

Now, we briefly introduce the four components of our framework: transformations, features, models and metrics. Transformations allow us to convert from data sources of different types to features of a single type. Features represent characteristics of the data that we are interested in. In our framework they are instantiated as a time series of real values that can be used as input to our black box anomaly detector. For example, a data source might be a packet trace captured from the network. From this data source we can extract multiple features such as the traffic volume per connection or the number of destinations contacted by every source on port 80. We discuss both transformations and features in Section 3.

A model is an abstraction that allows us to represent some property of the data. For example, we can model the distribution of a feature. Finally, the metrics are used to evaluate how good the model is representing that property of the data and also allow us to find deviations from the model, that is anomalies. We discuss both models and metrics in Section 4.

3 HANDLING DATA SOURCE TYPES

In this section we address the question of how to transform the different types of data sources collected from the network into a common input type, that can be used by our black box anomaly detector.

3.1 Data source type classification

A time series is a time-ordered sequence of data points. We classify time series into four different classes according to two properties: data point values and data point time spacing. With respect to the data point values, we classify a time series into *real-valued*, when the data point is a point in $\Re^k$, or *structured* when the data point is a more complex structure. With respect to the data point time spacing, we classify a time series into *constant-spaced* when data points are equally spaced, or *variable-spaced* when the spacing between data points is not constant.

	Constant-spaced	Variable-spaced
Real-valued	SNMP measurements End-to-End path measurements	SNMP measurements with missing data
Structured	Periodically sampled topology information	Packet traces Netflow logs Configuration files Syslogs

Figure 1: Data source type classification

We consider that data sources can be associated with timestamps, e.g. of the time when they are collected. Thus, a data source can be considered a time series and classified into one of the four classes shown in Figure 1. We can then define functions that convert between the different classes of time series, which we describe in Section 3.3.

3.2 Features

We define a *feature* to be a representation of some network characteristic that is instantiated as a k-dimensional real-valued time series. Some of the data sources gathered from the network can be used as features themselves but others cannot. For example, a common SNMP measurement such as packets per second on a link, collected every 5 minutes, is a data source and can also be used as a feature, since it represents a network characteristic and has the format of a real-valued constant-spaced time series. On the other hand, a packet trace is a data source but is not a feature. In fact, many features can be extracted from this data source such as the traffic volume per connection or the number of destinations contacted by every source on a specific port.

Note that the input to the black box anomaly detector is a feature represented as a k-dimensional time series. In our current implementation, we focus on modeling features that are represented as a one-dimensional real-valued time series. However, the framework allows the anomaly detector to take as input k-dimensional time series (which we could use to represent graphs, matrices, etc.) in order to support more complex models [3].

3.3 Transformations

We define a *transformation* as a function that takes as input a time series, and outputs another time series. We are interested in transformations that take as input a time series belonging to one of the four classes shown in Figure 1 and output a time series that belongs to a different class. Then, when given a data source we can define a sequence of transformations that will extract a feature represented as a real-valued time series.

This allows us to reduce the problem of anomaly detection with different data source types to finding the proper sequence of transformations, for each data source, and dealing with features of a single data type.

Whether the input time series needs to be constant-spaced or variable-spaced depends on how the anomaly

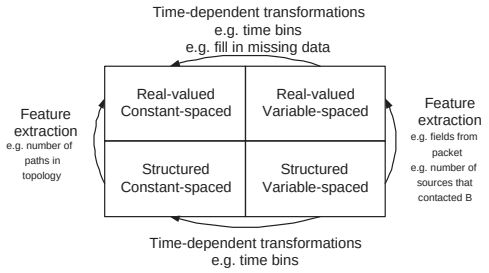

Figure 2: Example data sources transformations

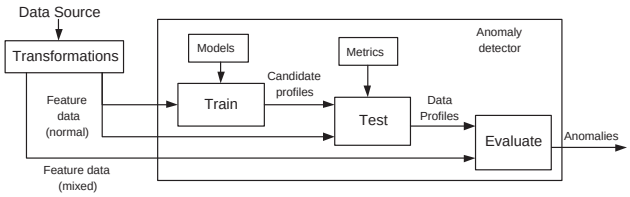

Figure 3: TrafficComber Overview

detector uses it. For example, we might be interested in removing periodic spikes because they hide other smaller spikes of interest, and then a constant-spaced time series will be needed. But if the model simply defines an upper bound on the average then the input time series can be either constant-spaced or variable-spaced.

We can divide transformations into two groups: time-dependent transformations and feature-extraction transformations. Time-dependent transformations operate within a single row of the matrix shown in Figure 1. As examples of time-dependent transformations, we describe two that we currently use to convert time series from variable-spaced to constant-spaced. The first is a bin transformation that divides time into equal size intervals, and bins together data points falling into each interval. The other is a generic missing data transformation, designed to fill in missing data points in the series [10].

Feature-dependent transformations operate within a single column of the matrix shown in Figure 1. They allow the extraction of network characteristics from more complex structures such as netflow logs, configuration files or packet traces. Figure 2 shows the different groups of transformations.

4 BLACK BOX ANOMALY DETECTOR

In Section 3, we have explained how to transform the data source into a set of real-valued time series, that we use as input to the blackbox anomaly detector. We now discuss how to build the blackbox anomaly detector.

4.1 Overview

An anomaly is a deviation from an expected behavior. This naturally poses a fundamental question: do we know the expected behavior of our data? In other words, can we predict the behavior of our data?

One approach to anomaly detection is to compare the given data to some domain knowledge of how the data should behave. However, this approach does not scale well, and is especially unsatisfactory with network data, where most often we do not know how the data should behave or even when we think we do, experimentation often proves us wrong. Without any domain knowledge on the data, we need to extract the expected behavior from the data itself.

Another approach is to analyze the behavior of the data, computing some measurements on the data, and examine how these measurements hold in future data. However, not every measurement will be applicable to future data. How do we know which ones, if any, will be applicable and indicative of future data? For example, we could measure the maximum value that appears in the normal data and use that value as an upper bound on future data. But how do we know whether that specific value will really be an upper bound? In order to be able to relate the past measurements with the future, we need to know what the relationship is between the data seen so far and the future data. Since we do not use domain knowledge about the data, we make the assumption that a specific property holds on the data, and then we test if the assumption is true, that is, if the data really behaves as implied by that underlying property.

Our approach is the following: we *search* for general properties of the data, such as independence across time intervals, and build models of the data based on these properties. For example, one model we could build is an upper bound on the data values, while another model could be an upper bound on the 50th percentile value of the distribution. Then, we assume that the data follows that model and *test* if this assumption is true by verifying if the data agrees with the model according to some metrics. If it does not, then the underlying property used to build the model does not hold and we need to test another model, built on a different property. If the model holds, then we can assume, to the extent tested, that the underlying property used to build the model holds on the data. We call this a *search-and-test* approach because we search for general properties of the data and test if they hold.

4.2 Train, Test, Evaluate

Our approach for implementing the black box anomaly detector is shown in Figure 3. We begin with a class of models and a set of normal data in the form of a time series of real values. For each model that we want to test, we use the input normal data to compute the values of the model parameters. We call the parameterized models *candidate profiles* and refer to this as the *Training* phase.

Next, in the *Testing* phase, we assess the candidate profiles on a new set of normal data. The candidate profiles are tested using metrics related to the model and only candidate profiles that satisfy the metrics are kept, the rest are discarded. Given sufficient normal traffic data, this approach will automatically generate profiles that characterize the input data and automatically discard all profiles which do not.

The data profiles generated through Training and Testing can then be used during the *Evaluation* phase for real-time anomaly detection, by applying them on a set of mixed data and flagging any deviations from the profile that the mixed data might present.

4.3 Modeling the data

In this section we describe the last two components of our framework: the models with their underlying assumptions and the evaluation metrics.

4.3.1 Models and Assumptions

In order to extract the profile of the traffic and have some confidence bounds on its prediction, we need to have some assumptions on the relationship between the data that we have seen so far, and the data that we expect to see in the future. Otherwise, the profile cannot imply anything about the new data. For example, each month's traffic might be drawn *iid* from a distribution over months, that has a 50% chance of being a high-traffic month, and 50% chance of being a low-traffic month. Clearly, in this case, training over a month of traffic would not yield a useful model over the next month.

With each assumption that we consider, we can define a class of models based on that assumption. We then build the models by computing appropriate values for the model parameters, and showing that these computed parameters are *tight*, i.e., they are not too far from the (unknown) true parameter values. For every model built for a given feature, we now need to test that the relevant underlying assumptions hold; if they do not, the model is not valid. We handle this in the next section with the last component of our framework, the evaluation metrics. [1]

There are always tradeoffs associated with the choice of assumptions to use for the data. If we choose models with very strong assumptions, the data may not obey these assumptions. On the other hand, if we restrict ourselves to models with very weak assumptions, the guarantees we would get from the model would be very weak. For example, if the only assumption that we make is that the values have an upper bound, then any guarantee we can make on the values can only involve this upper bound. The rest of the distribution might change significantly, but we would not be able to detect it with models involving only this assumption.

Therefore, we explore models based on a range of assumptions in our anomaly detector, and we build models for the features starting from the strongest assumptions, weakening them as needed to fit the data.

4.3.2 Evaluation Metrics

We need to be able to evaluate the candidate profiles in order to check if they fit the data, that is to test the validity of the model assumptions they are based on. In addition, we need to evaluate how likely a deviation from the model is to determine an anomaly.

For every model, we can define some properties that should be satisfied by feature values that fit this model. We refer to these as *model evaluation properties*. Some example model evaluation properties are the mean, the variance and the 90% percentile. There may not be a small (or even finite) set of properties that are sufficient to ensure that the model holds, so we may have to pick a subset of these properties to examine.

[1] Note that it is not always possible to say when an assumption does hold; the best we can sometimes say is whether data is consistent with the assumption.

In order to evaluate the model, we examine the values of these properties, and test how likely they are to have been generated from the model that we are testing. Thus, our *evaluation metrics* for the model are the likelihoods of the evaluation properties. So, for example, when we have built a model and our evaluation property for testing it is the variance, we compute the empirical variance on the data, and test how likely it is that this value of the variance was generated from the model that we built.

Thus, our evaluation of the validity of the model can be only as good as the properties of the model that we consider. For this reason, we will ask whether a model and the relevant evaluation property *together* are valid for the feature, rather than ask if a particular model is valid.

Since we can have multiple evaluation properties and associated metrics for a particular model, we may find that some of them are violated while others are never violated. If any of the evaluation properties do not hold during the testing phase, this indicates that the underlying assumption generating the model is violated and that model does not represent the feature accurately.

Thus, the space of anomalies we can detect is defined by the classes of models and their evaluation metrics that we use in the anomaly detector. These allow us to gradually increase the space of anomalies the detector can detect, e.g. as more sophisticated models and metrics are added to the anomaly detector, more families of anomalies will be detected. Also, it allows a different evaluation of the models: the larger the family of anomalies that a model can detect, the more suited it is for anomaly detection.

4.3.3 Application of the Framework

As a concrete example of the application of our framework, we now discuss some of the assumptions, models and the metrics that we consider.

We explore models that work on a single feature of one-dimensional real-values and consider two examples of assumptions on the values: *interval-independence* and *source-independence*. For the interval-independence assumption, we assume that each value in the time series is generated *iid* from the model (so, independently of the other values of the time series). For the source-independence assumption, we assume that each value of the time series is generated as a sum of independent processes. For example, the time series may represent the number of sources that are active on a particular port, and it may be reasonable to assume that each source acts independently of the other sources. With these two assumptions, we build two models: the first uses only interval independence, while the second uses both.

When we use only the first assumption, the data values could come from any distribution. So, the model we build consists of upper bounds for the various percentiles of the distribution. That is, we compute an upper bound for say, the 50th percentile, 70th percentile, etc. of the distribution. We evaluate this model by examining if the upper bound of each percentile is obeyed (under probabilistic guarantees). We refer to this model as the *percentile model*.

When we use both assumptions, the class of distributions the values can belong to is the generalized binomial distribution. The model we build is an estimate of the relevant parameters of the generalized binomial distribution. The evaluation properties we examine are properties that hold on this distribution, and we test how likely the property observed on the data is to come from a binomial distribution with the estimated parameters. We refer to this model as the *coin-tossing model*.

While the percentile model is clearly more general than the coin-tossing model, the guarantees we can get from the percentile model are weaker than those we can get from the coin-tossing model. For example, in the percentile model, any single value could be arbitrarily high without being anomalous (e.g., we may have an estimate on the 90th percentile of the distribution, but any individual value could be arbitrarily large), whereas in the coin-tossing model, we estimate how likely a particular value is to come from the distribution.

These two models capture many aspects of the behaviour of a feature, and so metrics can be defined to find changes in these aspects of the behaviour. For example, some data sources exhibit periodicity: there is a spike in their value periodically. As long as the distribution itself does not change, this can be modelled using the percentile model. However, there are other patterns of behaviour that these models cannot capture. For example, if the normal data exhibits an increasing trend, then the distribution of the values is no longer fixed, and therefore, we cannot model it with the coin-tossing or the percentile models. We plan to explore models that are able to capture these trends using non-parametric models, time series analysis and forecasting.

5 Related Work

There has been a wealth of research on anomaly detection. We focus here on the work that we feel is the closest to ours. One line of previous work has focused on specific features. McDaniel et al. [12] proposed profiles that capture the set of peers with which a host communicates and used the profiles for worm containment. Lakhina et al. [7] compute the entropy of the distribution of different IP header fields, and use it for automatically classifying network anomalies through unsupervised learning. LERAD [11] use a different approach that considers each byte of a packet header as a different attribute. Barford et al [2] use wavelet analysis to find anomalies. Thottan et al [20] propose a statistical signal processing technique to detect abrupt changes. Another line of previous work has presented frameworks for anomaly detection. Lee et al. [8] presented a framework that uses both normal and anomalous data to find the characteristic features of anomalies. Zhang et al. [24] presented a framework for network anomography that builds in the linear relationship between link loads and traffic matrices. Our work differs in that our framework considers different data sources types, uses only normal data, and allows for different features and models with different kinds of assumptions to be used for anomaly detection.

There have also been a number of studies that applied network profiles to intrusion detection. EMERALD [15] combines signature analysis with statistical profiling, MINDS [9] and ADAM [1] use data mining techniques to build profiles through learning network connections assumed to be normal. SPADE [19] generates a packet table based on connection history for each port and hosts and raise an alarm for packets rarely seen. eBayes TCP [21] employs Bayesian inference to categorize connections into pre-defined models, and WSARE [22] adds temporal attributes which allow to detect periodic and seasonal anomalies. Finally, Pang et al. [14] and Labovitz et al. [6] show anomalies present in different types of network data.

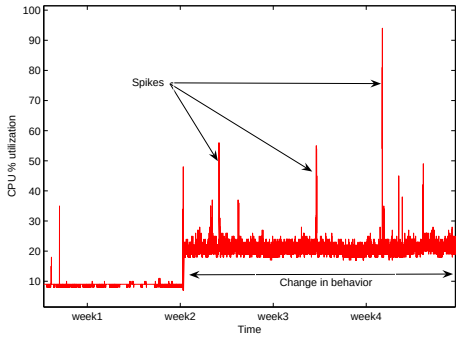

Figure 4: One month of CPU utilization in a backbone router

6 Preliminary Results

We have implemented a prototype of our framework, TrafficComber, and we are currently in the process of evaluating it.

Data sources and features We have started evaluation using two different data sources: packet traces captured at the border router of a departmental network and SNMP measurements from the network of a tier-1 ISP.

As a proof-of-concept, we are currently testing two features extracted from the packet traces: (1) the number of sources contacting more than k destinations on a fixed port, and (2) the number of ports in which a fixed source contacts more than k destinations. We refer to these features as *port* and *src* respectively. The first feature looks at the outbound traffic on a single port aggregated over all of the hosts in the network. The second feature models the outbound traffic of each individual host in the network. Both features aim to detect events characterized by one or a few sources having large fan-out. Some example applications of these features could be detecting worm outbreaks or finding hosts with heavy P2P usage located inside the monitored network.

Given the large number of active hosts and ports in the network, a manual approach is infeasible. For example, the number of hosts in our network is above 1000 and so far the number of active hosts for which we have automatically built profiles varies from 300 to 975 hosts depending on the feature and time period.

From the SNMP measurements, we are currently modeling the router CPU utilization. Figure 4 shows one month of CPU utilization from a backbone router. It in-

Data Source	Feature	% invalid models
Packet traces	port	3%
Packet traces	src	7%
SNMP measurements	cpu	25%

Table 1: Percentage of invalid models when applying the coin-tossing model on different features

cludes two types of anomalies: spikes and changes in behavior. We have found that changes in the behavior of the CPU utilization are widespread among the backbone routers and so far we have identified two main causes: software upgrades and hardware replacements.

Models and metrics To validate that the anomaly detector truly tells us which models are valid for a specific feature, we apply the coin-tossing model introduced in Section 4.3.3 to the three different features. We know that the coin-tossing model uses a source-independence assumption that does not really apply to the cpu and src features. If the system determines that the model does not fit the data it will discard the model.

Table 1 shows how often the anomaly detector discarded models when applying the coin-tossing model for the different features. Note that 25% of models were discarded for the cpu feature, implying that the coin-tossing model is not able to create valid profiles for that fraction of the routers. This in turn suggests that it would be better to use a different model (with different assumptions) for this feature, and we want to evaluate other models for this feature. The fact that the anomaly detector is able to tell us when a model does not apply to a feature allows us to gradually add new models as they are needed.

7 CONCLUSION

Anomaly detection is a fundamental tool for ISPs to maintain the health of their networks. But the volume and diversity of the data currently gathered from the network requires a comprehensive and automatic approach, rather than a set of individual solutions. In this paper we have shown that it is possible to build a black box anomaly detector that handles the diversity of data sources and features collected from the network. We have introduced a framework that splits the problem into two: handling different data sources and building a black box anomaly detector. We deal with diverse data sources through sequences of transformations that convert them into sets of features with a well defined data type. Then, our search-and-test black box anomaly detector automatically tests for the presence of underlying properties in those features that allow us to detect changes in the behavior. This approach allows us to detect novel anomalies not yet seen in the data and to explore the multiple features of interest, while gradually increasing the scope of the detection. We have developed a prototype of our framework, TrafficComber, and we are in the process of evaluating it using the data in the warehouse of a tier one ISP.

8 ACKNOWLEDGEMENTS

We would like to thank Tamraparni Dasu, Zihui Ge, Ajay Mahimkar, Eric Vance and Jia Wang for many discussions; Min Gyung Kang and Pongsin Poosankam for their help with the project, and Vyas Sekar and the anonymous reviewers for their insightful comments.

REFERENCES

[1] D. Barbara, J. Couto, S. Jajodia, L. Popyack, and N. Wu. Adam: detecting intrusions by data mining. *IEEE Workshop on Information Assurance and Security 2001*.

[2] P. Barford, J. Kline, D. Plonka, and A. Ron. A Signal Analysis of Network Traffic Anomalies. *Internet Measurement Workshop 2002*.

[3] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. An Information-Theoretic Approach to Detecting Changes in Multi-Dimensional Data Streams. *38th Symposium on the Interface of Statistics, Computing Science, and Applications*.

[4] G. Giacinto and F. Roli. Intrusion Detection in Computer Networks by Multiple Classifier Systems. *International Conference on Pattern Recognition 2002*.

[5] Y. Gu, A. McCallum, and D. Towsley. Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation. *Internet Measurement Conference 2005*.

[6] C. Labovitz, G.R. Malan, and F. Jahanian. Internet Routing Instability. *ACM SIGCOMM 1997*.

[7] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. *ACM SIGCOMM 2005*.

[8] W. Lee and S. Stolfo. A Framework for Constructing Features and Models for Intrusion Detection. *ACM Transactions on Information and System Security*, **3**(4).

[9] L.Ertz, E. Eilertson, A. Lazarevic, P. Tan, J. Srivastava, V. Kumar, and P. Dokas. The MINDS - Minnesota Intrusion Detection System. In *Next Generation Data Mining*. MIT Press, 2004.

[10] R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, 1987.

[11] M. V. Mahoney and P. K. Chan. Learning Rules for Anomaly Detection of Hostile Network Traffic. *Third IEEE International Conference on Data Mining*.

[12] P. McDaniel, S. Sen, O. Spatscheck, J. Van der Merwe, B. Aiello, and C. Kalmanek. Enterprise Security: A Community of Interest Based Approach. *Network and Distributed Systems Security 2006*.

[13] A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. *ACM SIGMETRICS 2005*.

[14] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson. Characteristics of Internet Background Radiation. *ACM Internet Measurement Conference 2004*.

[15] P. A. Porras and P. G. Neuman. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. *National Information Systems Security Conference 1997*.

[16] M. Sabhnani and G. Serpen. Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context. *International Conference on Machine Learning, Models, Technologies and Applications 2003*.

[17] H. Shvaytser. A Necessary Condition for Learning from Positive Examples. *Machine Learning journal*, **5**(1):101-113.

[18] A. Soule, K. Salamatian, and N. Taft. Combining Filtering and Statistical Methods for Anomaly Detection. *Internet Measurement Conference 2005*.

[19] S. Staniford, J. A. Hoagland, and J. M. McAlerney. Practical automated detection of stealthy portscans. *7th ACM Conference on Computer and Communications Security*.

[20] M. Thottan and C. Ji. Anomaly Detection in IP Networks. *IEEE Transactions on Signal Processing*, **51**(8).

[21] A. Valdes and K. Skinner. Adaptive, Model-Based Monitoring for Cyber Attack Detection. *Third International Workshop on Recent Advances in Intrusion Detection*.

[22] W. Wong, A. Moore, G. Cooper, and M. Wagner. Bayesian Network Anomaly Pattern Detection for Disease Outbreaks. *Twentieth International Conference on Machine Learning*.

[23] K. Xu, Z. Zhang, and S. Bhattacharyya. Profiling internet backbone traffic: behavior models and applications. *ACM SIGCOMM 2005*.

[24] Y. Zhang, Z. Ge, M. Roughan, and A. Greenberg. Network anomography. *Internet Measurement Conference 2005*.