

Histogram-Based Traffic Anomaly Detection

Andreas Kind, IBM Zurich Research Laboratory
 Marc Ph. Stoecklin, IBM Zurich Research Laboratory
 Xenofontas Dimitropoulos, ETH Zurich

Abstract—Identifying network anomalies is essential in enterprise and provider networks for diagnosing events, like attacks or failures, that severely impact performance, security, and Service Level Agreements (SLAs). Feature-based anomaly detection models (ab)normal network traffic behavior by analyzing different packet header features, like IP addresses and port numbers. In this work, we describe a new approach to feature-based anomaly detection that constructs histograms of different traffic features, models histogram patterns, and identifies deviations from the created models. We assess the strengths and weaknesses of many design options, like the utility of different features, the construction of feature histograms, the modeling and clustering algorithms, and the detection of deviations. Compared to previous feature-based anomaly detection approaches, our work differs by constructing detailed histogram models, rather than using coarse entropy-based distribution approximations. We evaluate histogram-based anomaly detection and compare it to previous approaches using collected network traffic traces. Our results demonstrate the effectiveness of our technique in identifying a wide range of anomalies. The assessed technical details are generic and, therefore, we expect that the derived insights will be useful for similar future research efforts.

I. INTRODUCTION

IDENTIFYING network anomalies is critical for the timely mitigation of events, like attacks or failures, that can affect the security, SLAs, and performance of a network. In addition, network anomalies pose a significant threat to many services that rely on networks. For example, in 2007 a faulty network card caused a nine-hour outage in the network of a major airport leaving many flights and travelers on the ground [1]. For these reasons, research on improving anomaly detection techniques is especially important.

Traditional approaches to anomaly detection use attack signatures built in an Intrusion Detection System (IDS) that can identify attacks with known patterns. Significant research efforts have focused on building IDS's and, therefore, related production systems are presently employed in many networks. Although signature-based detection finds most known attacks, it fails to identify new attacks and other problems that have not appeared before and do not have known signatures.

To address this problem, anomaly detection systems model the normal behavior of network traffic and identify anomalies as deviations from the normal behavior. A number of volume-based detection techniques exist that monitor the aggregate or per-link traffic load of a network, to identify anomalies that trigger significant traffic volume changes. Such anomalies include flooding attacks and certain types of Denial of Service (DoS) attacks. However, not all network incidents result in significant traffic volume shifts. Low traffic rate attacks, for example, produce limited change in traffic load and are, therefore, undetectable with volume-based detection systems.

Feature-based anomaly detection seeks to address the limitations of volume-based systems by examining a range of network traffic features, instead of relying solely on traffic volume. Common network traffic features are IP header fields, like destination/source IP addresses, destination/source port numbers, and TCP flags. Feature-based anomaly detection techniques rely on the observation that under normal conditions traffic features exhibit regular patterns that may be distorted by anomalies. In this work, we describe a new approach for feature-based anomaly detection. Our approach builds histograms describing the detailed characteristics of traffic features. For example, a histogram could give the number of flows associated with different destination ports over a period of time. Each histogram is then embedded into a metric space so that (dis)similar histograms are positioned close (apart) in the space. Data mining techniques are used to identify patterns of typical behavior. These patterns are finally compared to the online behavior of a network to identify deviations and trigger anomaly alarms. Compared to the few feature-based detection techniques presently available, the introduced scheme has two main differences. Firstly, we model the detailed characteristics of histograms, which enables us to identify a wider range of anomalies. In contrast, previous anomaly detection approaches only coarsely model the characteristics of traffic features, which render certain anomalies undetectable. Secondly, the proposed scheme mainly focuses on identification of anomalies in enterprise networks rather than large backbone networks, which have been the focus of most previous studies. We assess and evaluate a number of different design options, which reflect common technical challenges towards building effective anomaly detection systems. In particular, we discuss, among other important topics, the utility of different traffic features, clustering parameters that affect the quality of the derived models, the construction of histograms, and the selection of metric space. We expect that our findings and insights will provide useful guidelines for similar future studies.

The remaining of this paper is organized as follows. In the next section we provide background information. In Section III, we review the related work. Section IV presents our anomaly detection framework. Sections IV-A to IV-C discuss many technical details relevant to the design of an effective anomaly detection system. Finally, in Section V, we outline our evaluation results and conclude in Section VI.

II. BACKGROUND

Feature-based anomaly detection uses traffic monitoring data, which are usually collected with Cisco's NetFlow technology. Typically-used features are the source/destination IP

addresses, the source/destination port numbers, the TCP flags, and the transport protocols of network flows. In addition, other traffic data sources, namely packet (header) traces, may be used. The main difference between NetFlow and packet (header) data is that the former can be easily collected after configuring, appropriately, a router or switch. Whereas, online collection of packet (header) data typically requires special-purpose monitoring devices and substantially more resources for transmitting, storing, and analyzing collected data.

A feature-based anomaly detection system analyzes collected traffic data and raises an alarm on observing an unusual behavior. The system needs to distinguish between normal and abnormal traffic variability. Seasonal, weekly, time-of-day effects, and protocol dynamics result in legitimate traffic changes. These changes should be identified and separated from traffic variations due to anomalous events, like attacks, failures, and misconfiguration.

A problem in training and evaluating anomaly detection systems is the general lack of traces with labeled anomalies (“ground-truth”). Certain well-known anomalies, like worm outbreaks, can be easily identified and labeled. However, manually labeling all anomalies in a traffic trace is impossible, because a trace may also contain footprints of other unknown anomalies. For this reason, heuristics are necessary for distinguishing between normal and abnormal traffic patterns. In addition to heuristics, an administrator may provide useful feedback to an anomaly detection system by labeling known anomalous events. However, an effective system should not necessitate such feedback or assume anomaly-free training data. Clearly, the lack of ground-truth data does not only complicate training of anomaly detection systems, but also makes their evaluation more difficult.

Tuning an anomaly detection system to detect anomalies in arbitrary network environments is also challenging. Present feature-based anomaly detection systems have been shown to effectively find anomalies mainly in laboratory experiments. However, it is not clear how they perform in different settings and how they can be automatically tuned to yield accurate predictions in arbitrary production networks.

III. RELATED WORK

A number of studies have focused on developing volume-based anomaly detection schemes [2], [3], [4], [5], [6], [7]. For example, Barford *et al.* [2] used wavelets to distinguish between predictable and anomalous traffic volume changes. More recently, Zhang *et al.* [6] introduced a general framework that aims to identify anomalies from network-wide link load traffic data. These studies are successful in identifying anomalies that result in (network-wide) traffic volume deviations. However, they are not so effective in detecting stealth attacks, such as low-rate port scanning, that do not result in notable traffic volume changes.

Closer to our approach, the seminal work by Lakhina *et al.* [8] introduced the idea of using traffic feature distributions to identify a wider range of anomalies. The proposed anomaly detection scheme uses Principal Component Analysis (PCA) to identify an orthogonal basis along which the measurement

data exhibit the highest variance. The principal components with high variance model the normal behavior of a network, whereas the remaining components of small variance are used to identify and classify anomalies. The proposed scheme aims at finding anomalies in large backbone networks and, consequently, aggregates traffic into origin-destination (OD) flows between network ingress and egress points. Ringberg *et al.* [9] discussed a number of shortcomings of PCA-based anomaly detection. In particular, it is hard to select the right number of principal components to achieve: 1) a low false-positive rate and 2) a subspace of PCA components that is anomaly-free. The anomaly detection scheme by Gu *et al.* [10] uses a single composite feature distribution to characterize traffic and computes a parametric model of the distribution using training data. Observed network traffic is, then, compared to the constructed model to identify anomalies. The authors assume that the training data-set does not contain any anomalies. Wagner *et al.* [11] studied the compressibility, which reflects entropy, of different IP header fields in traffic traces. They found that during well-known worm outbreaks the compressibility of traffic features changes drastically.

The described studies on feature-based anomaly detection have in common the use of information theory and, in particular, of (relative) entropy to compare differences between traffic feature distributions. Entropy is a measure of the uncertainty of a probability distribution. A high (low) entropy value denotes a flat (concentrated) probability distribution. Entropy has two properties suitable for anomaly detection. Firstly, entropy conveniently summarizes a probability distribution into a single value, which can be used to compare certain qualitative differences of probability distributions. Secondly, some well-known attacks result in concentrating or dispersing probability distributions of traffic features. For example, during a worm outbreak the probability distribution of source IP addresses typically concentrates around the addresses of the infected hosts. Entropy reflects changes concentrating or dispersing a probability distribution and, thus, is suitable for identifying such shifts. On the other hand, entropy has one important limitation. Two completely different distributions can have the same entropy value. As a result, entropy very coarsely models the properties of a probability distribution and often cannot identify significant differences between two distributions.

A number of studies have focused on related problems. The recent work by Ringberg *et al.* [12] introduced Web-Class, an online repository of anomaly-labeled traffic traces that researchers can use for evaluating anomaly detection techniques. Soule *et al.* [13] studied how network traffic anomalies are observed in two adjacent backbone networks. They found that due to differences in the traffic collection infrastructure of the two networks, large-scale anomalies can leave substantially different footprints. Brauckhoff *et al.* [14] described how traffic sampling can influence the accuracy of anomaly detection systems. They showed that sampling can influence volume-based anomaly detection metrics, but does not significantly affect the distribution of traffic features. Scherrer *et al.* [15] introduced a long-range dependent non-Gaussian model of Internet traffic and proposed an anomaly detection method that identifies significant changes in the

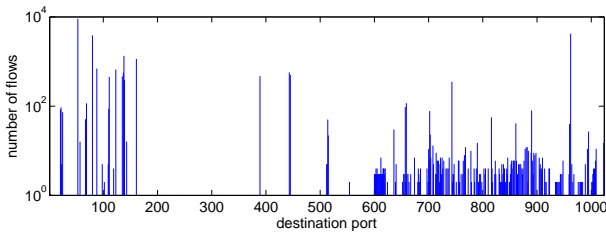


Fig. 1. Example histogram of destination-ports feature. For each destination port, the y -axis denotes the number of flows observed in a 5-minute trace. Port numbers larger than 1024 are clipped.

estimated parameters of the model.

In our recent work [16] we introduced a two-layered feature-based anomaly detection technique. The first layer models the typical values of different feature components. For example, it may model the typical number of flows connecting to a specific port. The second layer evaluates the differences between an observed feature distribution and a corresponding model. The main strength of the proposed scheme is the construction of fine-grained models that capture the details of feature distributions, instead of summarizing the details into an entropy value. The introduced technique is related to the more general approach we present in the current paper. Another related work [17] introduced a technique that uses histograms to visualize log files. The visualization enables administrators to easily spot anomalies. For a comprehensive survey of related studies the reader is referred to [18].

Our approach has two main differences to previous feature-based anomaly detection techniques. Firstly, it constructs detailed models of feature histograms, instead of using entropy to summarize distributions. As we show in the evaluation section, fine-grained modeling enables us to capture substantially more anomalies than with entropy-based modeling. The second main difference is that we focus on anomaly detection for enterprise networks, whereas most of the previous works focus on large backbone networks. PCA-based anomaly detection, for example, focuses on provider networks and applies PCA on OD flows between ingress and egress routers. In the context of enterprise networks, OD flows and the associated detection technique are not straightforward to use.

IV. HISTOGRAM-BASED ANOMALY DETECTION

A histogram is a distribution of the number of flows, packets, or bytes over the possible values of a traffic feature. Figure 1 illustrates a histogram of the destination-ports feature. A bin on the horizontal axis of a histogram may represent 1) a feature value, like a port number, 2) a set of feature values, like a range of IP addresses, or 3) joint values of multiple features, like a port number and a range of IP addresses. We collectively refer to the bins on the horizontal axis as *feature components*. If a feature has n components and component $i \in [1, \dots, n]$ has value $c_i \in \mathbb{N}_0$ flows, packets, or bytes, then we represent a histogram with a vector $\mathbf{c} = (c_1, c_2, \dots, c_n)$.

In Figure 2 we illustrate IP address histograms of five consecutive week days. All histograms correspond to five minute traces collected at the same time of the day. We make

the following observations. The histograms of the source IP address have similar patterns. Moreover, the histograms of the destination IP address also have similar patterns, with the exception of the second histogram collected on Tuesday. In this histogram, we can clearly see a network scanning attack that contacted all hosts in the monitored network. The attack substantially distorts the otherwise regular pattern. These two simple observations encompass the main ideas of our anomaly detection approach:

- 1) Feature histograms exhibit regular patterns that reflect the normal behavior of a network.
- 2) Network anomalies may distort the normal patterns of one or more features.

Histogram-based anomaly detection consists of the steps shown in Figure 3:

- 1) **Select features and construct histograms.** Some attacks may not be visible along one feature, but may be identified using more features or combinations of features. Generally, the more features one uses, the more likely one is to identify anomalies. Section IV-A discusses possible feature choices and their utility in identifying different types of anomalies.
- 2) **Map into metric space.** For each selected feature, we prepare a set of training histograms. We then map the vectors of training histograms into a metric space so that:
 - 1) two similar histograms are close in space, whereas
 - 2) two dissimilar histograms are far away.
 A number of different approaches can be used to quantify how similar two histograms are. For example, one commonly-used approach employs entropy. The dimensionality of a metric space can be up to n . The data may intrinsically be of lower dimensionality, for which dimensionality reduction techniques can be used to lower the order and complexity of the derived models. In Section IV-B we discuss alternative metric space choices and two dimensionality reduction techniques.
- 3) **Cluster and extract models.** In the training process, we cluster similar histograms together. Important details of clustering algorithms affect the characteristics of the constructed groups. In addition, known or unknown anomalies may be present in the training data, which deems the use of a filtering heuristic necessary. We examine clustering and filtering in Section IV-C.
- 4) **Classify.** In the detection process, the operation of a network is monitored and vectors for the different features are constructed. These vectors are, then, compared to the constructed models to measure how the observed online behavior differs. Two design choices are critical here. The first is how one computes the distance between a vector and a cluster of arbitrary shape. The second design option is the selection of necessary threshold(s) for raising an anomaly alarm. We discuss these issues in Section IV-D.

In the following subsections we discuss the high-level design aspects that determine histogram-based anomaly detection.

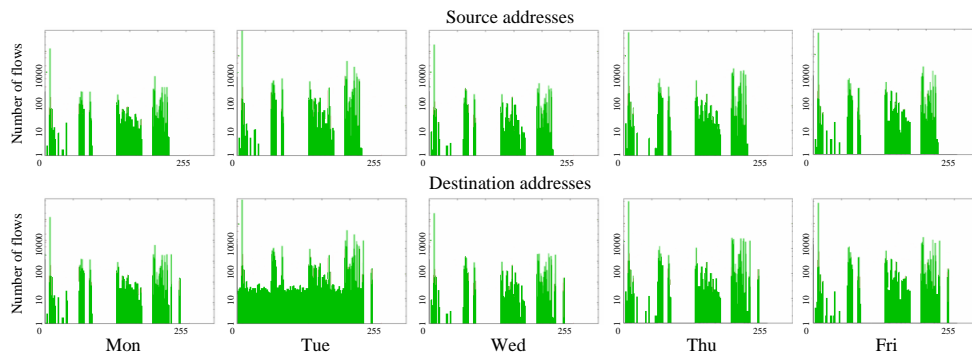


Fig. 2. IP/8 address usage histograms show host scan on Tuesday.

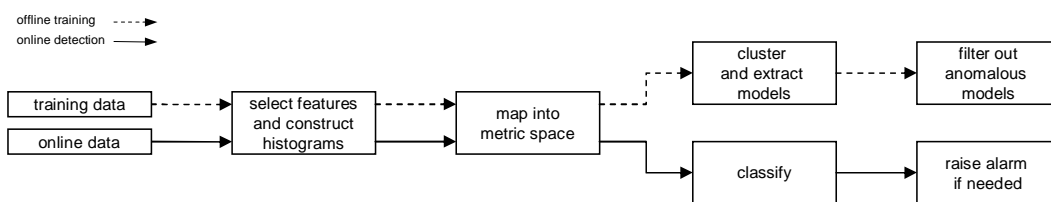


Fig. 3. Roadmap of anomaly detection framework. The dashed line illustrates the training process of an anomaly detection system. The solid line illustrates the online anomaly detection process.

TABLE I
TRAFFIC FEATURES AND THEIR UTILITY FOR ANOMALY DETECTION

Feature	Useful in identifying:
src IP address	<ul style="list-style-type: none"> one or more IP addresses associated with more traffic/connections than usual, e.g., due to worms, bots, or spoofing. failures that might disconnect certain servers or hosts from a network.
dst IP address	<ul style="list-style-type: none"> denial of service attacks towards one or more servers. network scanning attacks contacting most IP addresses in a network.
src port number	<ul style="list-style-type: none"> flows reflected from attacks to a specific port, e.g., scan for a vulnerable service, or a subset of ports, e.g., host scan or fingerprinting.
dst port number	<ul style="list-style-type: none"> targeted attacks to specific ports, that may happen for example during worm propagation. port scanning attacks searching for possible vulnerabilities.
TCP flags	<ul style="list-style-type: none"> SYN flooding attacks. server failures indicated by increased number of reset connections. port misconfigurations, where one or more clients attempt to connect to the wrong port number.
protocol number	<ul style="list-style-type: none"> rate anomalies and abnormal behavior in specific protocols, e.g., surge of UDP traffic, ICMP-based scanning. configuration anomalies, e.g., ICMP Destination Unreachable replies. use of unsolicited transport protocols.
packet size	<ul style="list-style-type: none"> increase in packets of certain size, for example increase of 40-Byte packets during SYN flooding attack.
flow duration	<ul style="list-style-type: none"> changes in flow duration patterns, e.g., unusual precense of flows of similar durations (scans, DoS attack), server overloads may cause an increase. configuration errors, e.g., observation of connection timeouts.

A. Feature selection

A number of traffic features can be used for identifying anomalies. In Table I, we describe how different traffic features are useful in identifying anomalies. For example, histograms of source IP addresses are useful for identifying attackers that create many connections to target hosts.

Additional features can be extracted from packet (header) traces. In particular, packet traces can be used to construct

histograms of packet sizes that can pinpoint sudden changes in the frequency of packets of certain size. Another source of additional traffic features is the recently introduced Flexible NetFlow (FNF). FNF is the new generation of Cisco’s NetFlow technology that can monitor up to 65 selected packet fields. It provides the means to increase the number and diversity of traffic features, however, it is still not widely used.

Besides plain feature histograms, combinations of features can reveal interesting traffic changes that are not visible otherwise. For example, a histogram combining source IP addresses with TCP flags, can pinpoint a sudden increase of reset packets sent by a server. This scenario may be the result of a crashed service resetting connection requests. Note, that the significant increase in reset packets sent from an IP address can be dwarfed when inspecting a feature histogram of the TCP flags alone. Among the different possible composite histograms, the combination of TCP flags with IP addresses or port numbers is especially informative for identifying problems with network servers or hosts.

An important issue in constructing a feature histogram is the number of feature components that lay on the horizontal axis. Feature components determine how traffic data is aggregated to yield a histogram. One extreme is to avoid aggregation and to use, for example, one component for each different IP address. However, this approach results in thousands of different components, which undermines the scalability of the system. In deciding the number of components, the following considerations are relevant. Firstly, aggregation affects the diagnostic capacity of the anomaly detection system. Fewer components make it harder to trace, for example, the IP address of an attacker. Secondly, a large number of components increases memory consumption and processing overhead. Thirdly, lower aggregation increases the variability of a component and,

therefore, requires fine-grained modeling to capture its normal behavior. Provided that a powerful modeling framework is used, we select a large number of components, subject to staying below the memory consumption and processing overhead limits imposed by the system.

A second issue that requires attention is histogram normalization. Histograms may be normalized, i.e., transformed into probability distributions. We do not find normalization useful, since it essentially loses information on the size of the components. To illustrate this, consider the following simple scenario. Assume that, normally, a certain feature has a uniform histogram. Furthermore, assume that an anomaly doubles the size of half of its components. Comparing the normal with the observed histogram easily highlights which components are affected. On the other hand, using normalized histograms, it is impossible to diagnose the affected components.

In this work we focus on the eight traffic features shown in Table I. These are typically available and are sufficient for detecting the anomalies considered in the evaluation. With the emergence of more sophisticated meters and metering protocols, other features will be available in future. We believe the results presented in this work are also valid for different feature sets that are more appropriate in other application domains.

B. Metric space

A central problem in identifying patterns of common behavior is quantifying how similar two histograms $\mathbf{x} = (x_1, x_2, \dots, x_n)$ and $\mathbf{y} = (y_1, y_2, \dots, y_n)$ are. If D is the set of all histograms, then a distance function $d: D \times D \rightarrow \mathbb{R}$ measures the similarity between two histograms $\mathbf{x}, \mathbf{y} \in D$ in the metric space $S = (D, d)$.

A number of metric space options exist that are defined by different distance functions. A straight-forward approach is to use the $L1$ -norm or $L2$ -norm distance functions. The $L1$ -norm distance, or the *Manhattan distance* as it is also known, is given by $d = \sum_{i=1}^n |x_i - y_i|$, whereas, the $L2$ -norm distance is $d = \sqrt{\sum_{i=1}^n |x_i - y_i|^2}$. These two distance functions have in common that they both weight equally the different components of histograms. The former is simply the sum of differences of the individual components and the latter is the well-known Euclidean distance.

Weighting equally all components does not capture the variance characteristics of a data-set. Assume, for example, two components, the first exhibiting high variability and the second being mostly stable in the observed data. Then, a change on the first component should not be as suspicious as a change on the second component. The variance characteristics of the different components can be captured with the *Mahalanobis distance*. Let C be the $n \times n$ covariance matrix of the n components of a feature. Then, the Mahalanobis distance between two vectors \mathbf{x} and \mathbf{y} is given by:

$$d = \sqrt{(\mathbf{x} - \mathbf{y})C^{-1}(\mathbf{x} - \mathbf{y})}. \quad (1)$$

The Mahalanobis distance takes into account the variance and covariance of the different components. A change to a component of high variance has a smaller weight than an

equal change to a component of smaller variance. In addition, covariance is important. Consider the joint distribution of two components of the TCP-flags feature in Figure 4. The samples are mainly randomly scattered showing that the two components are independent. In this case, the covariance between the two components is almost zero and the components are weighted by the inverse of their variance (marked on the figure). If the components are independent and σ_i is the standard deviation of a component i , then Mahalanobis distance reduces to:

$$d = \sum_{i=1}^n \sqrt{|x_i - y_i|^2 / \sigma_i^2}, \quad (2)$$

which is known as the *normalized Euclidean distance*. However, feature components are not always independent. In Figure 5, we illustrate the joint distribution of two components that clearly exhibit a dependence pattern. If one would ignore covariance, then the marked point A would be less “normal” than point B, since A has a larger distance from the mean along the two axes. The pattern observed in Figure 5, nevertheless, illustrates that point A is more “normal” than point B. Covariance factors-in the dependence between different components and, thus, the general form of the Mahalanobis distance in Equation 1 would assign point A a smaller distance than point B.

So far, the distance functions we discussed depend substantially on the size of the components, which requires dealing with the variability problem. An alternative approach is to encode into a distance function only the observed relationships between components. A relationship between three components may be that the first component is *higher than* the second component and that the second component is *smaller than* the third component. One can empirically observe that often these relationships are consistent. For example, in our experiments with the histograms of the destination port feature, the size of the component including port 80 was always larger than the two adjacent components. We may denote these relationships using a vector of $n-1$ bits, where the i -th bit has a value of one (zero) if the i -th component is larger (smaller) than the $(i+1)$ -th component. The *Hamming distance* between the bit-vectors of two histograms is equal to the number of bits in which the two vectors differ. Thus, the Hamming distance quantifies how consistent relationships between histogram’s components are.

Besides selecting a distance function, reducing the dimensionality of the metric space is needed to improve the parsimony of the models and the scalability of the classification. A first approach is simply to remove dimensions for feature components that remain constant in the training data. In our experiments, we find that such components are associated with unused feature values, for example, obsolete protocol numbers. Secondly, PCA is a well-known dimensionality reduction technique. Assuming that m observations of n components are denoted with an $m \times n$ matrix, PCA finds an orthogonal base of principal components, along which the data exhibit the highest variance. Specifically, the data exhibit the highest variance along the first principal component, the second highest variance along the second principal component, etc. We can reduce the number of dimensions by using the first $n' \leq n$

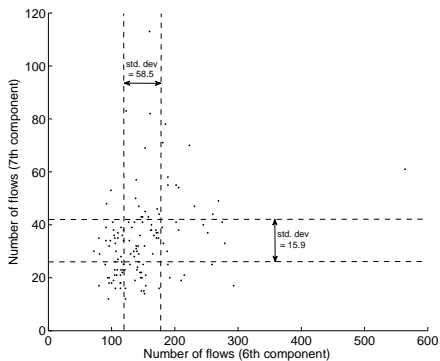


Fig. 4. Scatterplot of two independent components of the TCP-flags feature. The dashed lines mark the standard deviation of the two components around their mean.

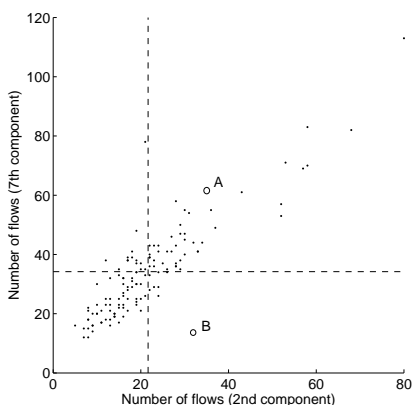


Fig. 5. Scatterplot of two strongly dependent components of the TCP-flags feature. The dashed lines mark the mean of the two components. Point A is more “normal” than point B, which is, however, closer to the mean along both axes. Taking into account the covariance of the two components, enables the correct classification of the two example points.

components that account for a significant fraction of the total variance. If the fraction is high enough, say 99.99%, then PCA realizes an almost lossless transformation of the data into a metric space of n' dimensions. In the described context, PCA is used for the traditional application of dimensionality reduction, which is different from using PCA to distinguish between the normal and abnormal behavior of a network [8].

C. Clustering

Clustering is needed for identifying and modeling patterns of normal behavior. The two most commonly-used clustering algorithms are hierarchical and k -means clustering. These algorithms can be used with the distance functions we discussed in the previous section.

Hierarchical clustering iteratively groups points or clusters forming new clusters. At each step a point/cluster is grouped with another point/cluster. The selected pair has the minimum distance among all possible choices. A number of options exist for measuring the distance to a cluster. Typically, distance is measured to the center of a cluster that is defined as the mean of the points in the cluster. Other possibilities are measuring the distance to the closest or furthest point of a cluster.

An important problem is finding the number of clusters that best describes the data. A number of decision rules have been proposed in the literature with the Calinski-Harabasz (CH) index [19] considered the most effective [20]. The CH index $H(k)$ of a data-set with k clusters uses the between groups sum of squares $BGSS(k)$, which measures the distance of the cluster centroids from the general mean of the data, to quantify how far apart the clusters are. In addition, it uses the within groups sum of squares $WGSS(k)$, which reflects the distance of m points from their centroids, to quantify the dispersion of the clusters. The optimal solution is the one resulting in the highest index:

$$H(k) = \frac{BGSS(k)}{\frac{k-1}{m-k} WGSS(k)}$$

Alternatively, one may use k -means clustering, which takes a top-down approach. It initially randomly divides points into a pre-selected number of clusters. Then, it iteratively refines the constructed clusters by computing their centers and assigning points to the closest centers. The process continues until the algorithm converges, i.e., the clusters do not change. In practice, k -means is substantially faster than hierarchical clustering. Nevertheless, it is often impractical as it requires prior knowledge of the number of clusters present in the data.

Having performed clustering, it is necessary to distinguish the clusters that correspond to the normal and to anomalous behavior. As we discussed previously, the training data used to construct clusters may contain unknown anomalies that are impossible to identify. A simple and straight-forward approach to filter anomalies is to remove clusters that correspond to a small fraction of the total observations. If the training data-set contains m histograms, then we remove all clusters that have fewer than $p \times m$ points. In our experiments, we set p to 0.05, which is a common filtering rule in statistics.

D. Classification

We call the set of clusters that model the normal behavior of a network *baseline*. Traffic patterns in a network change with the time of the day and the day of the week, which requires separate baselines for different time periods. The first challenge is determining how many baselines are necessary. One extreme is to construct a baseline for each hour in a week, yielding a total of $7 \times 24 = 168$ baseline models. The advantage of this extreme is that it yields baselines carefully crafted for different time periods. However, 168 baseline models substantially increase the complexity of the system and requires training data of multiple weeks to construct the models. The other extreme is to use a single baseline that includes models for the entire week. Though this extreme is computationally more approachable, it has the limitation that a normal behavior in a weekday may correspond to an anomalous behavior in a weekend. Since this approach merges different normal behaviors, the ability to discern anomalous from normal conditions decreases.

An intermediate approach is to map all 168 baselines into a single multidimensional space and to label the clusters of each baseline. Then, further clustering yields the weekly time

periods that exhibit similar behavior. In addition, it effectively reduces the number of distinct baselines and determines the time periods that can share training data.

Having constructed the baselines, the next step is to measure how the observed network behavior differs. For each feature, the anomaly detection system computes a vector that encodes the online behavior of the network. If the vector falls within the existing clusters, then the online behavior is considered normal. Otherwise, the behavior of the network is considered abnormal. We use the distance of the vector from the clusters to determine the severity of the alarm that is sent to the administrator. Two important issues are 1) when a vector is considered to fall within the clusters and 2) when an alarm is sent to the administrator.

A vector falls within the existing clusters if each vector component has a value within the normal component values defined by the cluster boundaries. When a vector falls outside the existing clusters along one or more components, then we measure its distance from the clusters of normal behavior. For each component that falls outside the clusters, we measure the distance from the closest cluster boundary. For the remaining components, we set the distance to zero. To quantify how large a distance value is, at this stage we only use the normalized Euclidean distance¹, since its value represents how many standard-deviations away from the clusters a vector falls, which is an informative quantity for classification. We normalize the distance along each component by the standard deviation of the component and combine the distance values of all components using Equation 2 to derive the overall distance. If the vector falls more than three standard-deviations away, then we rise an alarm, which we annotate with the computed distance to indicate the criticality of the alarm.

V. EVALUATION

A. Data-Sets

For our experiments we use three data-sets. The *campus trace* includes NetFlow packets collected from the campus of IBM Research at Zurich. The campus hosts more than 300 employees and at least as many networked workstations. NetFlow records have been collected and archived from two border routers since 2004.

The second trace comes from a *data center* that hosts a number of high-volume websites. NetFlow records are collected from a router that transfers more than 6 TBytes of data per day, with average sending and receiving rates of 550 Mb/s and 100 Mb/s, respectively. We have archived four months of NetFlow data collected during 2007.

The last trace is an intrusion detection benchmark called *DARPA IDEval* [21]. It contains two weeks of packet traces collected from a controlled test-bed network. The first week does not have any anomalies, whereas the second week includes a number of labeled anomalies.

TABLE II
DIMENSIONALITY REDUCTION

Feature	Dimensions		
	original / constant-removal / PCA		
	campus trace	data-center trace	IDEval trace
src IP address	256 / 145 / 33	256 / 133 / 2	256 / 60 / 37
dst IP address	256 / 152 / 38	256 / 137 / 2	256 / 62 / 37
src port number	1024 / 548 / 41	1024 / 461 / 3	1024 / 19 / 7
dst port number	1024 / 571 / 43	1024 / 469 / 3	1024 / 20 / 7
protocol number	256 / 8 / 3	256 / 8 / 2	256 / 20 / 4
number of Bytes	32 / 27 / 12	32 / 28 / 5	32 / 18 / 9
number of packets	32 / 22 / 8	32 / 19 / 3	32 / 12 / 4
flow duration	32 / 27 / 22	32 / 27 / 2	32 / 23 / 9

B. Baseline Analysis

We construct three sets of training data. The first set includes the campus traffic of 8 Monday mornings (9-12am) from August 13 until October 1, 2007. The second set includes data-center traffic of 11 Saturday afternoons (1-4pm) from October 6 until December 15, 2007. The third set consists of three workdays of the anomaly-free week (Week 1) of the DARPA IDEval data-set. Using 5-minute intervals for each histogram, the training sets result in 288, 396, and 789 histograms for each feature, respectively.

Having constructed the training sets, we reduce the dimensionality of the data using the constant-removal and PCA techniques discussed in Section IV-B. In Table II we summarize for each feature the number of dimensions in the beginning, after removing constant components, and after applying PCA with a conservative threshold of 99.99%. We first observe that both techniques significantly decrease the number of dimensions with a total decrease ranging between 31.2% to 99.7%. The dimensionality reduction is consistently higher in the data center trace than in the campus trace. The former exhibits lower variance as it reflects traffic patterns on Saturday afternoons, whereas, the campus trace reflects more volatile traffic patterns on Monday mornings. The dimensionality reduction of the selected three workdays in the IDEval data-set exhibits similar results as in the campus traces. Only in the port number features, we observe significantly lower dimensions in the IDEval data-set. This result was expected as the traffic mix in the IDEval test-bed shows a much lower diversity of applications compared to our campus network. The significant decrease in the number of dimensions underlines that 1) the data is inherently of low dimensionality, which indicates prevalent correlations between histogram components and 2) using a large number of components, which is desirable for the diagnostic capacity and the modeling detail of the system, does not severely impact its scalability, since the effective number of components after dimensionality reduction remains low.

We use hierarchical clustering and for each feature apply the CH rule to determine the optimal number of clusters. In Figure 6 we illustrate how $BGSS(k)$, $WGSS(k)$, and $H(k)$ change with the number of clusters for the flow-duration feature. Observe that as the number of clusters increases, $BGSS(k)$ becomes larger and $WGSS(k)$ decreases. $BGSS(k)$ increases with k simply because it is the sum of distances of more clusters from the general mean. On the other hand, $WGSS(k)$ decreases because the points are split in more clusters, which become more compact. The $H(k)$

¹Note that, as we discuss in the evaluation section, the Mahalanobis distance gives in practice identical results with the normalized Euclidean distance.

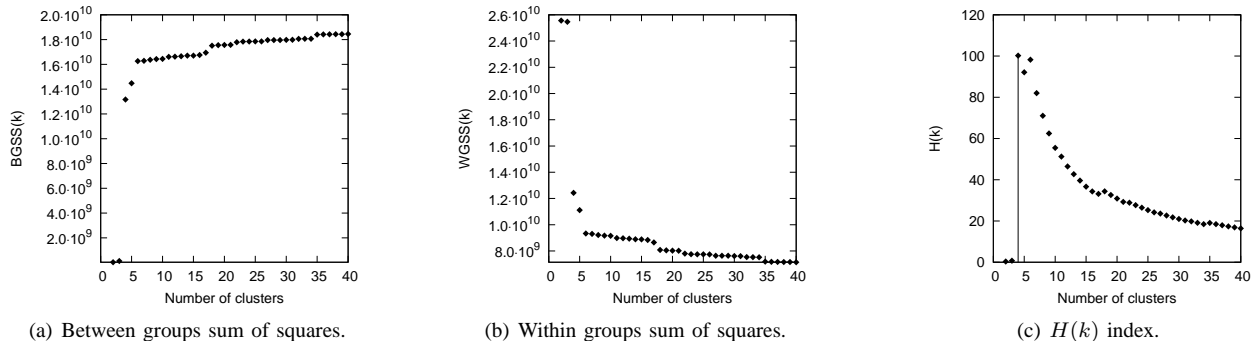


Fig. 6. $BGSS(k)$, $WGSS(k)$, and how they determine the $H(k)$ index.

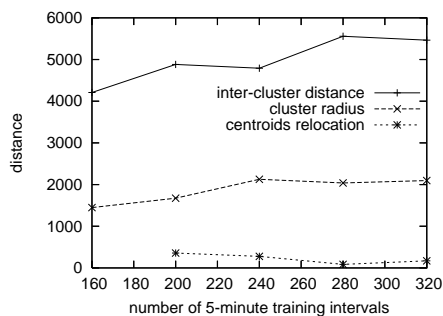


Fig. 7. Avg. inter-cluster distance, avg. cluster radius, and avg. relocation of centroids versus the length of the training data for the flow-size feature using the data-center trace.

index on Figure 6(c) indicates the optimal solution having four clusters. This solution corresponds to jumps in $BGSS(k)$ and $WGSS(k)$ denoting a transition that pinpoints the most representative number of clusters describing the data.

For each feature and distance function combination, we cluster the training data. The number of clusters we find ranges from as low as 2 up to 40. In most cases, the data formed one or two large clusters and a number of smaller clusters frequently containing one or few data points.

We next examine how the length of the training data affect the constructed clusters. We repeat clustering using subsets of the training data. In Figure 7 we examine how the two main clusters of the flow-size feature change as we increase the length of the training data. The two clusters account for between 91.2% and 98.7% of the total number of training points. We measure three quantities: 1) the average radius of the clusters defined as the distance of their furthest points to their centroids, 2) the average distance between the cluster centroids, 3) and the average relocation of the cluster centroids. We first observe that as the length of the training data increases, the radius of the main clusters increases slightly and tends to converge; the means that adding more training data from the same time period reinforces the existing clusters. The inter-cluster distance is larger than the cluster radius, which, in turn, is larger than the centroids relocation. In addition, the location of the cluster centroids is changing only minimally, which indicates that the clusters modeling normal behavior do not significantly change with the amount of training data.

Next we examine the effect of distance functions on the resulting clusters. We define the *similarity ratio* between two clustering solutions for a data-set, as the fraction of point pairs having a consistent cluster relationship in both solutions, i.e., where in both solutions the two points of a relationship either belong or do not belong to the same cluster. In Table III, we compare the mutual similarity ratios between clustering solutions obtained with the Euclidean, Manhattan, normalized Euclidean, and Mahalanobis distance functions. We do not include the Hamming distance, as we did not find it useful in identifying patterns of normal behavior.² For the remaining functions, we report the computed similarity ratios for different features and data-sets. Examining Table III yields the following observations. Generally, the similarity ratios for all function combinations and features are high, i.e., larger than 77.2% in 90% of the cases. This indicates that the distance functions do not result in dramatically different clusters or, in other words, two clustering solutions have similarities regardless of the distance function used. This is a desired property as it suggests that the derived models are not very sensitive to the distance functions. Besides, we observe very high similarity between the Euclidean and Manhattan pair and between the normalized Euclidean and Mahalanobis pair: the ratios are always higher than 87.5% and often equal to 100%. The functions of the two pairs have indeed common properties. The Euclidean and Manhattan functions weigh equally all components, whereas the normalized Euclidean and Mahalanobis factor-in the variance of the components and, thus, expose higher changes in relatively stable components.

In practice, the normalized Euclidean and the Mahalanobis function produce identical clusters in our experiments. The Mahalanobis distance improves the normalized Euclidean distance in making an orthogonal axes transformation using the covariance matrix. In our data, we apply PCA during pre-processing, which also achieves the same effect with the Mahalanobis distance of transforming the axes along the direction of highest variance. Due to this pre-processing step, we find that the normalized Euclidean and Mahalanobis distances result in identical clusters. Similarly, the Euclidean and Manhattan distances result in identical clusters in our experiments.

²After dimensionality reduction, the remaining components had a consistent order throughout the training data, which always resulted in a single point in the Hamming metric space.

In Figure 8 we illustrate the histograms of different clusters. Each subplot corresponds to a different cluster and overlays a histogram for each point in the cluster. The clusters are computed using the Euclidean distance function and small clusters containing a single point are not shown. The histograms correspond to the destination port feature. Observe that the data exhibit eight clusters, which have few versatile components and a large number of components with values close to zero. Each of the eight clusters encodes a distinct behavior observed in the training data. In Figure 9, we illustrate the corresponding clusters and histograms using the Mahalanobis distance function. In this case, the data includes only two clusters with size larger than one. We observe that the upper histogram of Figure 9 corresponds to a large cluster of 283 points. Using the Euclidean distance these points were split into multiple smaller clusters. This observation reflects the main difference we consistently observed between the Euclidean or Manhattan distance and the normalized Euclidean or Mahalanobis distance: The latter pair of distance functions is not sensitive to components of high variability, and thus a lot of the individual clusters in Figure 8 are merged into a single large cluster in Figure 9. The second cluster in Figure 9 is the result of an increase near the otherwise stable component 100^3 . This component has a small value in the training data and, therefore, a small change is highly penalized, forming a separate cluster.

In summary, the main findings of our evaluation of the baseline models are:

- Dimensionality reduction drastically reduces the effective number of histogram components (up to 99.7% reduction in our experiments) making our technique more scalable.
- Increasing the used training data does not change significantly the location and size of the resulting clusters, which confirms that the patterns of network traffic in the examined time intervals are similar and admit our modeling approach.
- The baseline clusters have certain similarities regardless of the distance function, i.e., the similarity metric was higher than 77.2% in 90% of the cases. The similarity of the resulting clusters is very high when using the Manhattan or Euclidean distance and when using the normalized Euclidean and Mahalanobis distance. The main difference between these two pairs is the sensitivity to components of high variability.

C. Detection Accuracy

We use the *IDeval* trace to analyze the detection accuracy of our system. The trace contains a number of labeled anomalies for evaluating signature-based as well as anomaly-based intrusion detection systems. We ignore attacks that do not affect traffic patterns, i.e., attacks that carry malicious payloads – these attacks are useful for evaluating signature-based systems. We focus on 15 attacks that can change network traffic patterns. These attacks include network and port scanning,

³Note that the values of the x -axes in Figures 8 and 9 correspond to component indices rather than port numbers due to the removal of constant components.

worm propagation, surveillance software probes, mail bombs, and system fingerprinting.

We use the first anomaly-free week to train our system and the second week containing the attacks to evaluate its accuracy. The features extracted are all the features described in Table I. We use the normalized Euclidean distance, as it takes into account component variation, remove constant components, apply PCA, and finally find clusters of normal behavior.

The system effectively raised alarms for 13 (86.7%) out of the 15 attacks. These alarms were consistently raised regardless of the distance function used during training. The missed labeled attacks were two network scans within a narrow IP prefix block, triggered by a surveillance system. These attacks did not produce too much traffic or trigger any notable pattern shifts. Overall, alarms were raised for 74 five-minute intervals during the second week of the trace. 18 of these alarms were related to the 13 identified attacks. Moreover, 2 alarms were related to incidents that were not among the labeled attacks, but were anomalous events of interest to network administrators. In particular, our manual investigation of the trace around the raised alarms revealed an excessive amount of DNS replies, all having a size of 172 Bytes, which could indicate an artificial load on the DNS server due to a misconfiguration. Finally, the remainder of the alarms can be considered false positives, as they do not correspond to a labeled attack or to a notable anomaly.

In Figure 10, we plot the distribution of alarm distances from the closest clusters of normal behavior distinguishing between verified and false-positive alarms. We observe that the vast majority of false-positive alarms have a small distance from the clusters of normal behavior, which indicates that these alarms are of low severity. To determine the severity of the alarms we use their distance from the closest clusters and split them into three classes: *warning*, *serious*, and *critical* alarms. Using the shape of the distribution, we set the class boundaries to 20 for switching between a warning and a serious alarm, and to 100 for switching to a critical alarm. 87% of the false-positives are warning alarms of low criticality, whereas 100% of the critical and 53.8% of the serious alarms correspond to verified attacks. Thus, high distances, i.e., critical and serious alarms, from the clusters of normal behavior mostly correspond to true anomalies or attacks. A medium number of false-positive alarms (46 during a week) are present, most of which have a small distance from the clusters of normal behavior, i.e., they are simply warnings, and could be reduced, as indicated by Figure 10, by optimizing the distance threshold for raising alarm. We leave this important and substantially deep problem as a subject for future studies.

D. Comparison with Entropy-based Anomaly Detection

The last set of experiments of our evaluation aim to compare histogram-based with entropy-based anomaly detection. We examine the widely-used sample entropy: if $p_i, i = 1, \dots, N$, are the probabilities of different values, then $\sum_{i=1}^N -p_i \log(p_i)$ gives the entropy of the sample. For each known anomaly in the *IDeval* data-set, we compute the entropy of different features over 5-minute intervals. Then we examine the entropy

TABLE III
SIMILARITY RATIOS OF DISTANCE FUNCTIONS FOR DIFFERENT FEATURES

Feature	Similarity of distance functions					
	Euclidean & Manhattan	Euclidean & norm. Euclidean	Euclidean & Mahalanobis	Manhattan & norm. Euclidean	Manhattan & Mahalanobis	norm. Euclidean & Mahalanobis
dst/src IP address	87.5%	65.4%	65.4%	77.2%	77.2%	100%
dst port number	96.6%	91.2%	91.2%	88.7%	88.7%	100%
protocol number	99.9%	79.7%	79.7%	79.7%	79.7%	100%
number of Bytes	100%	99.5%	99.5%	99.5%	99.5%	100%
number of packets	99.9%	97.6%	97.6%	97.6%	97.6%	100%
flow duration	99.9%	96.7%	96.7%	96.7%	96.7%	100%

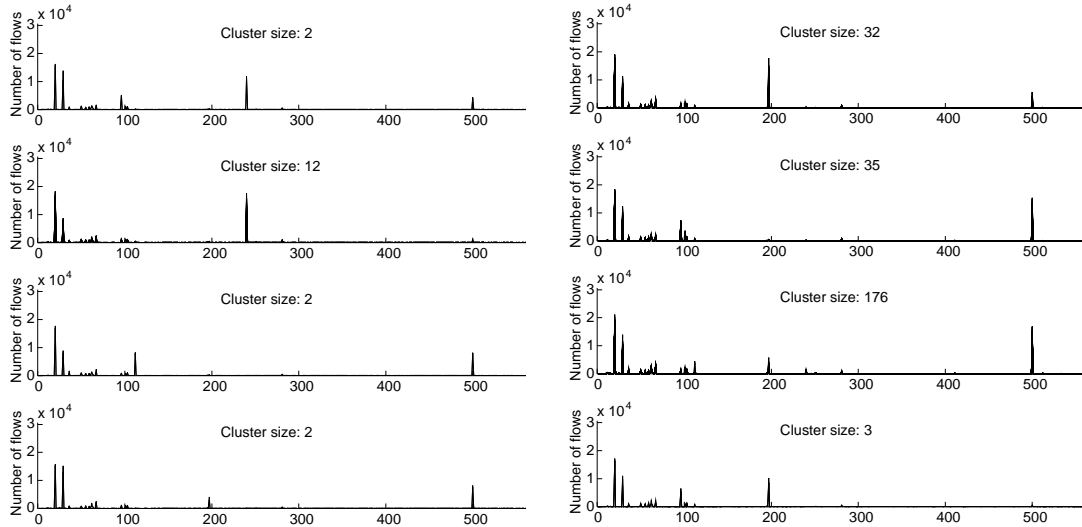


Fig. 8. Clusters using Euclidean distance.

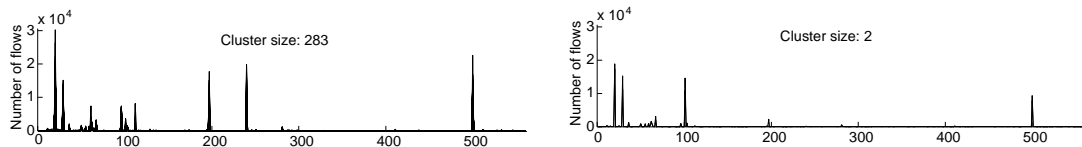


Fig. 9. Clusters using Mahalanobis distance.

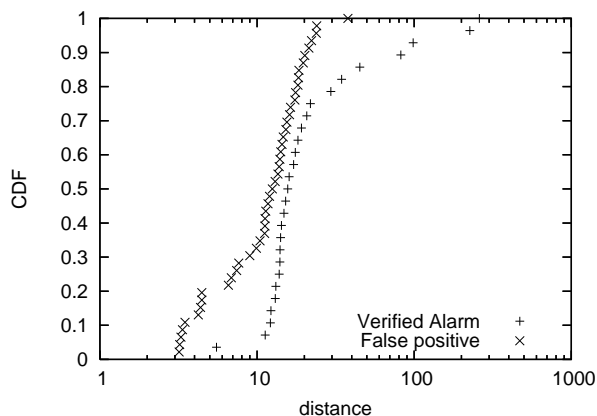
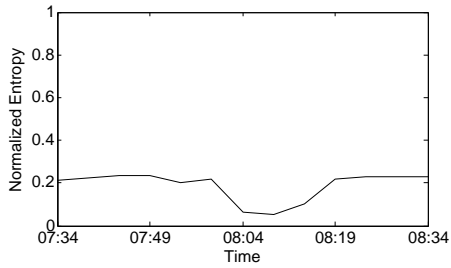


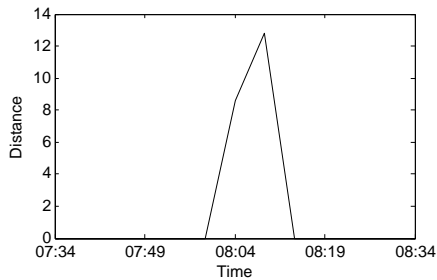
Fig. 10. Cumulative Distribution Function (CDF) of alarm distances from closest clusters of normal behavior. The two distributions correspond to verified alarms and two false alarms (false positives).

signal and report if an anomaly causes any visible change. In Figure 11(a) we illustrate the entropy signal for an anomaly that was visible with the entropy- and the histogram-based technique. The anomaly was a mailbomb attack that caused a significant change in the source and destination IP addresses features. The corresponding distance function of our technique and the notable change due to the anomaly is shown in Figure 11(b). In Figure 12 we show the corresponding plots for an anomaly that was only detected using our histogram-based approach. In this case, the anomaly is a limited-scope host scan. As shown in Figure 12(b), the anomaly was visible along the destination IP addresses feature using our histogram-based approach. However, it did not trigger any notable change along any feature using entropy, e.g., in Figure 12(a) we illustrate the entropy signal during the anomaly for the destination IP addresses feature.

We used the two techniques to examine the 15 anomalies of the *IDeval* data-set. Recall that the histogram-based technique could not identify two of these anomalies. These two

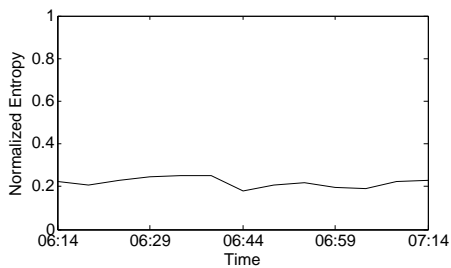


(a) Entropy signal of destination IP addresses feature during an attack.

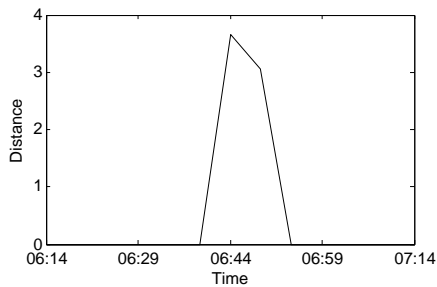


(b) Histogram distance from clusters of normal behavior of destination IP addresses feature during an attack.

Fig. 11. Mailbomb attack that is clearly visible using the histogram- and entropy-based techniques.



(a) Entropy signal of destination IP addresses feature during an attack.



(b) Histogram distance from clusters of normal behavior of destination IP addresses feature during an attack.

Fig. 12. Limited scope host scanning attack that is visible with the histogram-based technique.

anomalies were not identified with entropy as well. Entropy effectively identified 9 of the 15 anomalies. For the remaining 6 anomalies the change in the feature distributions, were such that did not affect the entropy of the distribution and, therefore, were not visible. In practice, we find that histogram-based detection outperforms entropy detection in identifying attacks that may affect a single or few components, distorting their usual values. Both techniques work equally well in identifying attacks that concentrate or flatten the feature distributions.

With respect to detection accuracy, the evaluation showed that techniques based on histograms can provide advantages over techniques based on entropy values. The price for higher accuracy is, however, paid by the additional effort in the preparation and maintenance of the histogram-based approach. Good training data and a thoughtful operation of the training phase is required. Shifts or seasonal effects on the workloads may also require to repeat the baselining on a regular basis. Root cause traceability is difficult with both, histogram- and entropy-based techniques. In both cases data is aggregated that would help to trace back towards the root cause of the anomaly. The distance of an actual histogram or entropy instance to its nearest cluster, that represents normal network condition, cannot be easily analyzed with respect to individual components, such as host addresses or port numbers. Furthermore, the adjustment of alarm thresholds might require experience.

VI. CONCLUSION

In this work we introduced a new feature-based anomaly detection approach that is based on modeling the characteristics of different traffic features for identifying anomalies. The presented methodology is generic and can be used to extract relevant information from arbitrary features, both from those presently proven to be useful as well as from possible future features that might become informative for revealing new anomalies. We discussed a number of technical alternatives in the process of constructing histograms, modeling their characteristics, and identifying deviations. Our work differs from previous feature-based anomaly detection techniques by directly modeling the detailed characteristics of histograms, instead of using entropy that identifies coarse-grained changes between distributions. Besides evaluating our approach and illustrating its effectiveness in identifying anomalies, we showed that 1) the dimensionality of histograms can be effectively reduced making their detailed modeling more tractable; 2) the clusters of normal behavior do not significantly change with amount of training data; and 3) different distance functions produce clusters that exhibit similarities.

An important issue remains open, which is the reduction, or ideally elimination, of false positives. False positives are an inherent problem with all anomaly detection systems, as both normal and abnormal conditions can occasionally result in the same observable characteristics. In our work, we found a medium number of false positives. The successful and widespread use of feature-based anomaly detection systems requires lasting efforts for reducing the occurrences of costly incorrect alarms. A number of promising paths exist that we plan to explore in our future research.

REFERENCES

- [1] J. Schwarz, "Who needs hackers?" *The New York Times*, Sep. 12, 2007.
- [2] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *Proc. ACM SIGCOMM Internet Measurement Workshop*, Nov. 2002, pp. 71–82.
- [3] J. Brutlag, "Aberrant behavior detection in timeseries for network monitoring," in *Proc. USENIX LISA*, Nov. 2002.
- [4] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," in *ACM SIGCOMM*, Aug. 2004.
- [5] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: Methods, evaluation, and applications," in *Proc. ACM SIGCOMM Internet Measurement Conference*, Oct. 2003.
- [6] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan, "Network anomography," in *Proc. ACM SIGCOMM Internet Measurement Conference*, Oct. 2005.
- [7] M. Thottan and C. Ji, "Anomaly detection in IP networks," *IEEE Trans. on Signal Processing*, vol. 51, no. 8, pp. 2191–2204, Aug. 2003.
- [8] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *ACM SIGCOMM '05*, 2005, pp. 217–228.
- [9] H. Ringberg, A. Soule, J. Rexford, and C. Diot, "Sensitivity of pca for traffic anomaly detection," in *SIGMETRICS '07: Proc. of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM, 2007, pp. 109–120.
- [10] Y. Gu, A. McCallum, and D. Towsley, "Detecting anomalies in network traffic using maximum entropy estimation," in *IMC'05: Proc. of the Internet Measurement Conference 2005 on Internet Measurement Conference*. Berkeley, CA, USA: USENIX Association, 2005, pp. 32–32.
- [11] A. Wagner and B. Plattner, "Entropy based worm and anomaly detection in fast ip networks," in *WETICE '05: Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 172–177.
- [12] H. Ringberg, A. Soule, and J. Rexford, "Webclass: adding rigor to manual labeling of traffic anomalies," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 1, pp. 35–38, 2008.
- [13] A. Soule, H. Larsen, F. Silveira, J. Rexford, and C. Diot, "Detectability of traffic anomalies in two adjacent networks," in *Proc. Passive and Active Measurements (PAM) Conference*, Apr. 2007.
- [14] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, and A. Lakhina, "Impact of packet sampling on anomaly detection metrics," in *IMC '06: Proceedings of the 6th ACM SIGCOMM on Internet measurement*. New York, NY, USA: ACM Press, 2006, pp. 159–164.
- [15] A. Scherrer, N. Larrieu, P. Owezarski, P. Borgnat, and P. Abry, "Non-gaussian and long memory statistical characterizations for internet traffic with anomalies," *IEEE/ACM Transactions on Dependable and Secure Computing*, vol. 4, no. 1, pp. 56–70, 2007.
- [16] M. Ph. Stoecklin, J.-Y. Le Boudec, and A. Kind, "A two-layered anomaly detection technique based on multi-modal flow behavior models," in *Proc. Passive and Active Measurements (PAM) Conference*, Apr. 2008.
- [17] A. Frei and M. Rennhard, "Histogram matrix: Log file visualization for anomaly detection," in *Proceedings of the Third International Conference on Availability, Reliability and Security*, 2008, pp. 610–617.
- [18] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection : A survey," *ACM Computing Surveys*, vol. to appear, 2009.
- [19] Calinski and Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics*, vol. 3, no. 1, pp. 1–27, 1974.
- [20] G. W. Milligan and M. Cooper, "An examination of procedures for determining the number of clusters in a data set," *Psychometrika*, vol. 50, pp. 159–179, 1985.
- [21] Lincoln Laboratory, "DARPA intrusion detection evaluation," <http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/>, MIT, 2001.