Anomaly Extraction in Backbone Networks using Association Rules

Daniela Brauckhoff, Xenofontas Dimitropoulos, Arno Wagner, and Kave Salamatian

TIK-Report 309
ftp://ftp.tik.ee.ethz.ch/pub/publications/TIK-Report-309.pdf

***Abstract*—Anomaly extraction refers to automatically finding in a large set of flows observed during an anomalous time interval, the flows associated with the anomalous event(s). It is important for several applications ranging from root cause analysis, to attack mitigation, and testing anomaly detectors. In this work, we use meta-data provided by several *histogram-based detectors* to identify suspicious flows and then apply *association rule mining* to find and summarize event flows. Using rich traffic data from a backbone network (SWITCH/AS559), we show that our technique triggers a very small number of false positives, on average between 2 and 8.5, which, in addition, exhibit specific patterns and can be trivially sorted out by an administrator. Our anomaly extraction method significantly reduces the work-hours needed for analyzing alarms making anomaly detection systems more practical.**

I. Introduction

A. Motivation

Anomaly detection techniques are the last line of defense when other approaches fail to detect security threats or other problems. They have been extensively studied since they pose a number of interesting research problems, involving statistics, modeling, and efficient data structures. Nevertheless, they have not yet gained widespread adaptation, as a number of challenges, like reducing the number of false positives or simplifying training and calibration, remain to be solved.

In this work we are interested in the problem of identifying the traffic flows associated with an anomaly during a time interval with an alarm. We call finding these flows the anomalous flow extraction problem or simply *anomaly extraction*. At the high-level, anomaly extraction reflects the goal of gaining more information about an anomaly alarm, which without additional meta-data is often meaningless for the network operator. Identified anomalous flows can be used for a number of applications, like root-cause analysis of the event causing an anomaly, improving anomaly detection accuracy, and modeling anomalies.

B. Anomaly Extraction

In Figure 1 we present the high-level goal of anomaly extraction. In the bottom of the figure, events with a network-level footprint, like attacks or failures, trigger *event flows*, which after analysis by an anomaly detector may raise an alarm. Ideally we would like to extract exactly all triggered event flows; however knowing or quantifying if this goal is realized is practically very hard due to inherent limitations in finding the precise ground truth of event flows in real-world traffic traces. The goal of anomaly extraction is to find a set of *anomalous flows* coinciding with the event flows.

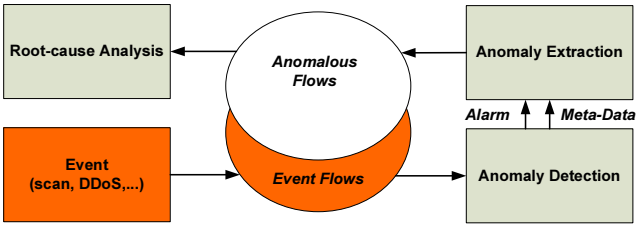


Fig. 1. The high-level goal of anomaly extraction is to filter and summarize the set of anomalous flows that coincide with the flows caused by a network event such as Denial of Service attacks or scans.

An anomaly detection system may provide meta-data relevant to an alarm that help to narrow down the set of candidate anomalous flows. For example, anomaly detection systems analyzing histograms may indicate the histogram bins an anomaly affected, *e.g.*, a range of IP addresses or port numbers. Such meta-data can be used to restrict the candidate anomalous flows to these that have IP addresses or port numbers within the affected range. In Table I we outline useful meta-data provided by various well-known anomaly detectors.

To extract anomalous flows, one could build a model describing normal flow characteristics and use the model to identify deviating flows. However, building such a microscopic model is very challenging due to the wide variability of flow characteristics. Similarly, one could compare flows during an interval with flows from normal or past intervals and search for changes, like new flows that were not previously observed or flows with significant increase/decrease in their volume [14], [7]. Such approaches essentially perform anomaly detection at the level of individual flows and could be used to identify anomalous flows.

C. Contributions

In this work, we take an alternative approach to identify anomalous flows that combines and consolidates information from multiple histogram-based anomaly detectors. Compared to other possible approaches, our method does not rely on past data for normal intervals or normal models. Intuitively, each

Meta-data	Anomaly detection technique
Protocol	Maximum-Entropy [10]
	Histogram [12], [23]
IP range	Defeat [16]
	MR-Gaussian [8]
	DoWitcher [21]
	Histogram [12], [23]
Port range	Maximum-Entropy [10]
	Histogram [12], [23]
	DoWitcher [21]
TCP flags	Maximum-Entropy [10]
	Histogram [12], [23]
Flow size	DoWitcher [21]
Packet size	Histogram [12], [23]
Flow duration	Histogram [12], [23]

TABLE I
USEFUL META-DATA PROVIDED BY VARIOUS ANOMALY DETECTORS. THE LISTED META-DATA CAN BE USED TO IDENTIFY SUSPICIOUS FLOWS.

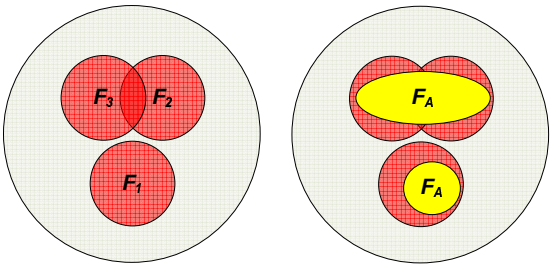


Fig. 2. Each detector j supplies a set of suspicious flows F_j. We filter the union set of suspicious flows $\cup F_j$ and apply association rule mining to extract the set of anomalous flows F_A.

histogram-based detector provides an additional view of network traffic. A detector may raise an alarm for an interval and provide a set of candidate anomalous flows. This is illustrated in Figure 2, where a set F_j represents candidate flows supplied by detector j. We then use association rules to extract from the union $\cup F_j$ a summary of the anomalous flows F_A. The intuition for applying rule mining is the following: anomalous flows typically have similar characteristics, *e.g.*, common IP addresses or ports, since they have a common root-cause, like a network failure or a scripted Denial of Service attack. We test our anomaly extraction method on rich network traffic data from a medium-size backbone network. The evaluation results show that our solution reduced the classification cost in terms of items that need to be manually classified by several orders of magnitude. In addition, our approach effectively extracted the anomalous flows in all 31 analyzed anomalies and on average it triggered between 2 and 8.5 false positives, which can be trivially filtered out by an administrator.

D. Outline

The rest of the paper is structured as follows. Section 2 describes our techniques for extracting anomalous traffic from Netflow traces using histogram-based detectors and association rules. In Section 3, we describe the datasets used for this study and present evaluation results. Related work is discussed in Section 4. Finally, Section 5 concludes the paper.

II. METHODOLOGY

In the following section we give an overview of our approach to anomaly extraction. Further, we discuss the details of each functional block, namely histogram cloning and detection, meta-data generation with different voting strategies, pre-filtering, and association rule mining.

A. Overview

An overview of our approach to the anomaly extraction problem is given in Figure 3. It contains two subfigures that illustrate the individual steps of our approach. The upper subfigure depicts the anomaly detection and meta-data generation steps. These steps are applied for each traffic feature. The lower subfigure shows how association rule mining is applied to suspicious flows. A subtle point of our approach is filtering flows matching *any* meta-data (in other words we take the *union* of the flows matching meta-data) instead of flows matching *all* meta-data, *i.e.*, the intersection of the flows matching meta-data. Assume for example the Sasser worm that propagated in multiple stages: initially a large number of SYN flows scanned target hosts, then additional flows attempted connections to a backdoor on port 9996 of the vulnerable hosts, and finally a third set of frequent flows resulted from downloading the 16-Kbyte worm executable. *Anomalies often result in such distinct sets of frequent flows with similar characteristics*. In addition, different meta-data can relate to different phases of an anomaly. In our example, the anomaly could be annotated with meta-data about the SYN flag, port 9996, and the specific flow size. The intersection of the flows matching the meta-data would be empty, whereas the union would include the anomalous flows.

Our approach consists of four main functional blocks.

- **Histogram cloning:** To obtain additional traffic views the distribution of a traffic feature is tracked by multiple histogram clones. Each clone randomizes the distribution using one of k independent hash functions. Upon detection of a disruption in the distribution each clone compiles a list V_k of traffic feature values that are associated with the disruption.
- **Voting:** Meta-data is compiled from the individual feature value lists of clones by voting. More specifically, if a certain feature value is selected by at least l out of k clones, it is included in the final meta-data. We analyze the impact of different parameter settings for l and k on the accuracy of our approach.
- **Flow pre-filtering:** We use the union set of meta-data provided by n different traffic features to pre-filter a set of suspicious flows. This pre-filtering is necessary since it typically eliminates a large part of the normal flows.
- **Association rule mining:** A summary report of the most frequent item-sets in the set of suspicious flows is generated by applying association rule mining algorithms. The basic assumption behind this approach is that the most frequent item-sets in the pre-filtered data are often related to the anomalous event. A large part of this work is devoted to the verification of this assumption.

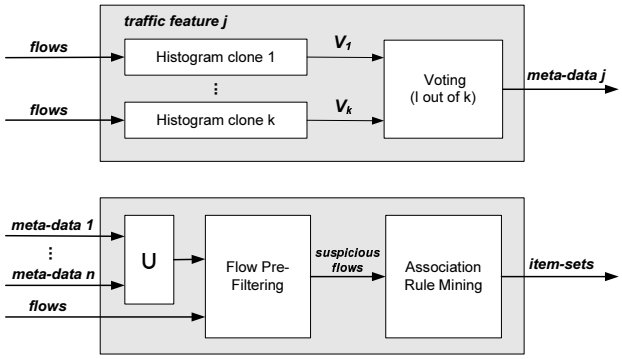


Fig. 3. Overview of our approach to the anomaly extraction problem. The upper figure illustrates how meta-data for a single traffic feature j is generated by voting from k histogram clones. The lower figure illustrates how the meta-data for filtering flows is consolidated from n traffic features by taking the union, and how suspicious flows are pre-filtered and anomalous flows are summarized n item-sets by association rule mining.

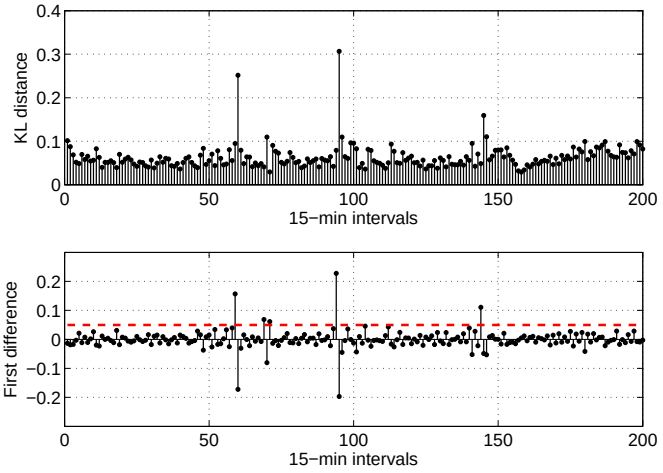


Fig. 4. Upper plot: KL distance time series for the source IP address feature for roughly two days. Lower plot: First difference of the KL distance for the same period. The dashed line corresponds to the anomaly detection threshold.

In the remainder of this section we describe each functional block in more detail.

B. Histogram Cloning and Detection

Histogram-based anomaly detectors [12], [23], [16], [20] have been shown to work well for detecting anomalous behavior and changes in traffic distributions. We build a histogram-based detector that (i) applies *histogram cloning*, *i.e.*, maintains multiple randomized histograms to obtain additional views of network traffic; and (ii) uses the Kullback-Leibler (KL) distance to detect anomalies. Each histogram detector monitors a flow feature distribution, like the distribution of source ports or destination IP addresses. We assume n histogram-based detectors that correspond to n different traffic features and have m histogram bins. Histogram cloning provides alternative ways to bin feature values. Classical binning groups adjacent feature values, *e.g.*, adjacent source ports or IP addresses. A histogram clone with m bins uses a hash function to randomly place each traffic feature value into a bin. Each histogram-based detector $j = 1 \ldots n$ uses k histogram clones with independent hash functions[1].

During time interval t, an anomaly detection module constructs histogram clones for different traffic features. At the end of each interval, it computes for each clone the KL distance between the distribution of the current interval and a reference distribution. The KL distance has been successfully applied for anomaly detection in previous work [10], [20]. It measures the similarity of a given discrete distribution q to a reference distribution p and is defined as

$$D(p||q) = \sum_{i=0}^{m} p_i \, log \, (p_i/q_i)$$

[1]Note that histogram cloning uses random projections as they are commonly used in sketch data structures that have been proposed in the literature. Sketches aim at summarizing a data stream in a compact data structure, which can be used for answering various queries. In contrast, histogram cloning is a method to randomly bin histograms that does not target summarization. As we discuss in the related work section, similar methods have been proposed in the literature [8], [16].

Coinciding distributions have a KL distance of zero, while deviations in the distribution cause larger KL distance values. In general, the KL distance is asymmetric $D(p||q) \neq D(q||p)$.

Instead of training and recalibrating distributions that represent normal behavior, we use the distribution from the previous measurement interval as reference distribution p. Hence, we will observe a spike in the KL distance time series each time the distribution changes. Assuming an anomalous event that spans multiple intervals, the KL distance will generate spikes at the beginning and at the end of an anomalous event. On the other hand, changes in the total number of flows that do not have an impact on the distribution will not result in large KL distance values. The KL distance time series for the source IP address feature over roughly two days is depicted in Figure 4 in the upper plot.

We have observed that the first difference of the KL distance time series is approximately normally distributed with zero mean and standard deviation σ. This observation enables to derive a robust estimate, the median absolute deviation, of the standard deviation $\hat{\sigma}$ and of the anomaly detection threshold $3\,\hat{\sigma}$ from a limited number of training intervals. We generate an alert when

$$\Delta_t \, D(p||q) \geq 3\,\hat{\sigma}$$

In Figure 4, we show the $\Delta_t \, D(p||q)$ time series for the source IP address feature and the corresponding threshold. An alarm is only generated for positive spikes crossing the threshold, since they correspond to significant increases in the KL distance.

If we detect an anomaly during interval t we need to identify the set B_k of affected histogram bins and the corresponding set V_k of feature values that hash into the affected bins. The set V_k is then used to determine meta-data for filtering suspicious flows.

To find the contributing histogram bins for each clone, we use an iterative algorithm that simulates the removal of suspicious flows until $\Delta_t \, D(p||q)$ falls below the detection threshold. In each round the algorithm selects the bin i with

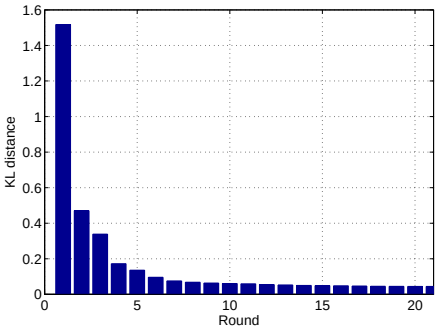


Fig. 5. This Figure illustrates our incremental method for determining the anomalous bins. The KL distance converges to zero as in each round the bin with the largest absolute difference is aligned with its counterpart in the reference distribution. Already after the first round the KL distance decreases significantly.

the largest absolute distance $\max_{i\in[0,m]} |p_i - q_i|$ between the histogram of the previous and current interval. The removal of flows falling into bin i is simulated by setting the bin count in the current histogram equal to its value in the previous interval ($q_i = p_i$). The iterative process continues until the current histogram does not generate an alert any more. This procedure is illustrated in Figure 5, where we plot the KL distance computed in each round. Already after the first round, the KL distance decreases significantly. Having identified the set of anomalous histogram bins B_k for each clone, we obtain the corresponding set of feature values V_k by keeping a map between values and corresponding bins.

C. Voting and Meta-data Generation

The cardinality of V_k is typically much larger than the cardinality of B_k, *e.g.*, the 65'000 unique port numbers are distributed evenly over 1024 bins if we use a 10-bit hash function for randomization. Therefore the set of feature values V_k provided by each clone is likely to contain normal feature values colliding on anomalous bins. Using k clones a normal feature value has a small probability of $(1/m)^k$ to appear in an anomalous bin in all k clones.

In our previous work [3], we keep feature values that have been identified by all histogram clones $M_j = \bigcap_k V_k$ in order to minimize false positives. We generalize this approach to a more flexible scheme that is based on voting. In particular, voting keeps a feature value if it has been selected by at least l out of k clones. With this approach the trade-off between false positives and false negatives can be adjusted via the parameters k and l.

Let's assume that each of the k clones has detected a disruption in the distribution of feature j in interval t, and has identified b responsible bins. Each clone includes an anomalous feature value in the set V_k with probability p_a, while a normal feature value is selected only if it collides on one of the selected bins, and has thus a selection probability of $p_n = b/m$ where m is the total number of bins.

If an anomalous value is included by one clone it is likely that it will also be included by the other clones as these events are not independent. Consequently, we can derive a lower bound for the probability that an anomalous feature value is included by more than l out of k clones

$$P_a \geq \sum_{i=l}^{k} \binom{k}{i} p_a^i \, (1-p_a)^{k-i} \tag{1}$$

and an upper bound for the probability that an anomalous feature value is missed

$$P_{\bar{a}} \leq 1 - P_a \tag{2}$$

The probability that a normal feature value is included by more than k clones, on the other hand, is given by

$$P_n = \sum_{i=l}^{k} \binom{k}{i} (p_n)^i \, (1-p_n)^{k-i} \tag{3}$$

Here we do not derive a bound since the considered events are not correlated.

To sum up, the meta-data M_j for feature j obtained after the voting process contain feature values representing normal and anomalous traffic. The ratio between the two classes depends on the parameters k and l, on the initial probability p_a, and the hash function length m.

D. Flow Pre-filtering

In the pre-filtering step we select all flows in time interval t that match the *union* of the meta-data provided by n detectors, *i.e.*, all flows that match $\cup M_j$ where $j = 1, ..., n$ are filtered. Pre-filtering usually removes a large part of the normal traffic. This is desirable for two reasons. Firstly, it generates a substantially smaller dataset that results in faster processing in the following steps and secondly it improves the accuracy of association rule mining by removing flows that might cause false positive item-sets.

E. Association Rule Mining

Association rules describe items that occur frequently together in a dataset and are widely-used for market basket analysis. For example, a rule might reflect that 98% of customers that purchase tires also get automotive services [1]. Formally, let a transaction T be a set of h items $T = \{e_1, \ldots, e_h\}$. Then the disjoint subsets $X, Y \subset I$ define an association rule $X \Longrightarrow Y$. The support s of an association rule is equal to the number of transactions that contain $X \cup Y$.

The problem of discovering all association rules in a dataset can be decomposed into two subproblems: (i) discover the frequent item-sets, *i.e.*, all item-sets that have a support above a user-specified minimum support; and (ii) derive association rules from the frequent item-sets.

Our motivation for applying association rules to the anomaly extraction problem is that anomalies typically result in a large number of flows with similar characteristics, *e.g.*, IP addresses, port numbers, or flow lengths, since they have a common root-cause like a network failure, a bot engine, or a scripted Denial of Service (DoS) attack. Each transaction T corresponds to a NetFlow record and the items e_i to the following seven ($h = 7$) flow features: srcIP, dstIP, srcPort, dstPort, protocol, #packets, #bytes. For example, the item $e_1 = \{srcPort : 80\}$ refers to

l	srcIP	dstIP	srcPort	dstPort	#packets	#bytes	support	what
1	*	*	*	*	2	*	10,407	
1	*	*	*	25	*	*	22,659	
2	Host A	*	*	80	*	*	11,800	HTTP Proxy
2	*	*	*	80	6	*	35,475	
2	Host B	*	*	80	*	*	14,477	HTTP Proxy
2	*	*	*	80	7	*	16,653	
2	Host C	*	*	80	*	*	15,230	HTTP Cache
2	*	*	*	80	5	*	58,304	
3	*	*	*	80	1	46	17,212	
3	*	*	*	80	1	48	11,833	
3	*	*	*	80	1	1024	23,696	
3	*	*	*	7000	1	48	12,672	Dist. Flooding
4	*	Host D	*	9022	1	48	22,573	Backscatter
5	*	Host E	54545	7000	1	46	23,799	Dist. Flooding
5	*	Host E	45454	7000	1	46	15,627	Dist. Flooding

TABLE II
FREQUENT ITEM-SETS COMPUTED WITH OUR MODIFIED APRIORI ALGORITHM. THE INPUT DATA SET CONTAINED 350,872 FLOWS AND THE MINIMUM SUPPORT PARAMETER WAS SET TO 10,000 FLOWS. IP ADDRESSES HAVE BEEN ANONYMIZED.

a source port number equal to 80, while $e_2 = \{dstPort : 80\}$ refers to a destination port number equal to 80. An l-item-set $X = \{e_1, \ldots, e_l\}$ is a combination of l different items. The largest possible item-set is a 7-item-set that contains a feature-value pair for each of the seven features. A transaction or an l-item-set cannot have two items of the same feature, *e.g.*, $X = \{dstPort : 80, dstPort : 135\}$ is not valid. The support of an l-item-set is given by the number of flows that match all l items in the set. For example, the support of the 2-item-set $X = \{dstIP : 129.132.1.1, dstPort : 80\}$ is the number of flows that have the given destination IP address and the given destination port.

Apriori Algorithm The standard algorithm for discovering frequent item-sets is the Apriori algorithm by Agrawal and Srikant [1]. Apriori makes at most h passes over the data. In each round $l = 1 \ldots h$, it computes the support for all candidate l-item-sets. At the end of the round, the frequent l-item-sets are selected, which are the l-item-sets with frequency above the minimum support parameter. The frequent item-sets of round l are used in the next round to construct candidate $(l+1)$-item-sets. The algorithm stops when no $(l+1)$-item-sets with frequency above the minimum support are found.

By default, Apriori outputs all frequent l-item-sets that it finds. We modify this to output only l-item-sets that are not a subset of a more specific $(l+1)$-item-set. More specific item-sets are desirable since they include more information about a possible anomaly. This measure allows us to significantly reduce the number of item-sets to process by a human expert. We denote the final set of l-item-sets as I. The Apriori algorithm takes one parameter, *i.e.*, the *minimum support*, as input. If the minimum support is selected too small, many item-sets representing normal flows (false positives) will be included in the output. On the other hand, if the minimum support is selected too large, the item-sets representing the anomalous flows might be missed (false negative).

Apriori Example In the following we give an example of using Apriori to extract anomalies. In the used 15-minute trace, destination port 7000 was the only feature value that was flagged by all histogram clones. It contributed 53,467 candidate anomalous flows. To make the problem of extracting anomalies more challenging, we manually added to the candidate set $\cup F_j$ flows that had one of the three most frequent destination ports but had not been flagged by all histogram clones. In particular, the most popular destination ports were port 80 that matched 252,069 flows, port 9022 that matched 22,667 flows, and port 25 that matched 22,659 flows. Thus, in total the input set $\cup F_j$ contained 350,872 flows. For our example, we set the minimum support parameter to 10,000 flows and applied our modified Apriori to the flow set $\cup F_j$.

The final output of the algorithm is given in Table II, which lists a total of 15 frequent item-sets. In the first iteration, a total of 60 frequent 1-item-sets were found. 59 of these were, however, removed from the output as subsets of at least one frequent 2-item-set. In the second iteration, a total of 78 frequent 2-item-sets were found. Again, 72 2-item-sets could be removed since they were subsets of frequent 3-item-sets. In the third iteration, 41 frequent 3-item-sets were found, of which four item-sets were not deleted from the output. In the fourth round, 10 frequent 4-item-sets were found but only one of them remained after removal of redundant 4-item-sets. Two frequent 5-item-sets were found in round five. Finally, the algorithm terminated as no frequent 6-item-sets satisfying the minimum support were found.

Three out of the 15 frequent item-sets had destination port 7000. We verified that indeed several compromised hosts were flooding the victim host E on destination port 7000. Regarding the other frequent item-sets, we verified that hosts A, B, and C, which sent a lot of traffic on destination port 80, were HTTP proxies or caches. The traffic on destination port 9022 was backscatter since each flow has a different source IP address and a random source port number. The remaining item-sets refer to combinations of common destination ports and flow sizes and are thus not likely of anomalous nature. These item-sets can be easily filtered out by an administrator.

F. Parameter Estimation

The various parameters associated with our approach, and their range as used in the evaluation of this work, are summarized in Table III. Although most of the parameters are associated with the detection part of our approach, they also

Parameter	Description	Range
n	Number of detectors	5
w	Interval length	[5,10,15] min
m	Hash function length	[512,1024,2048]
k	Number of clones	1-50
l	Voting parameter	1-k
s	Minimum support	3,000-10,000 flows

TABLE III
PARAMETERS INCLUDING DESCRIPTION AND RANGE AS USED IN THE EVALUATION SECTION OF THIS WORK.

have an impact on the extraction part. In the following we describe each parameter in detail and provide criteria for selecting the correct parameter settings.

Number of detectors n: As the number of detectors increases further information can be exploited for identifying anomalies and therefore a large n is generally desired. In this work we use five detectors, where each detector monitors one of the following features: source IP address, destination IP address, source port number, destination port number, number of packets per flow. Other features that might be useful for anomaly detection purposes are the number of packets per flow, the average packet size, or the flow duration.

Interval length w: The interval length w determines the detectable anomaly scale, *i.e.,* it becomes harder to detect short disruptions that contain only few flows with longer intervals. On the other hand, it is not always desirable to detect such short disruptions. Hence, the desired number of daily or weekly anomalous alarms can be used to set the interval length w. The desired number of alarms depends on the available human resources for investigating alarms. Some studies report that actionable alarms require on average 60 minutes investigation time [19], which would correspond to 8 alarms per day assuming a full-time employ for analyzing alarms. Another issue related to the interval length is the detection delay as an anomaly can only be detected at the end of a given interval. Typically used intervals correspond to delays of few minutes, e.g., 5 to 15 minutes. However, a sliding window mechanism can shorten this delay. Finally, one last implication is that a larger w results in more flows to be processed by association rule mining and in higher computational overhead. Nevertheless, the overhead of association rule mining is low as we discuss in the next section.

Hash function length m: The hash function length m is also involved in a detection sensitivity versus aggregation trade-off we discussed for parameter w. The smaller the hash function length the more flows are aggregated per hash function bin. In addition, a larger m is desired for anomaly extraction as it decreases the probability P_n that a normal feature value remains in the meta-data after voting and, thus, the number of candidate flows for rule mining. Finally, the parameter also affects the required memory resources. Assuming that the available memory resources do not drive the choice of m, then an acceptable range of values can be first determined via simulation using Equation 3 and a target range for P_n. Then, m should be selected together with w based on a desired number of daily/weekly anomalous alarms. Among the possible (m, w) choices realizing a desired number of alarms, the solutions with larger m, *i.e.,* smaller bins, are preferable for anomaly extraction.

Voting parameters l **and** k: The parameter k determines the total number of histogram clones used. The computational requirements in terms of memory and CPU scale linearly with k. Moreover, the parameter k has an impact on the probability that a feature value remains in the meta-data after voting and thus on accuracy. The parameter l determines the lower bound for the number of clones that need to select a feature value for it to be included in the final meta-data. Therefore, l can vary between 1, corresponding to the union, and k, representing the intersection. Just like k, the parameter l impacts the number of flows selected in the pre-filtering step and thus the accuracy of our approach. The parameter settings for l and k can also be obtained by simulation using Equation 1 and 3. Simulation results for $P_{\bar{a}}$ and P_n for different settings of l and k will be presented in the evaluation section.

Minimum support s: The parameter s determines the frequency threshold above which an item-set is extracted by Apriori as a possible set of anomalous flows. A large s extracts no or few item-sets, which in our experiments were almost always associated with anomalous events. On the other hand, decreasing s results in more item-sets and in a small but higher rate of false positives. By progressively decreasing s the administrator can create a rank of suspecious item-sets in decreasing order of confidence. Therefore, the parameter s does not require any special calibration but is manipulated by the administrator to progressively extract and analyze additional item-sets. The parameter s is set by the user in an iterated fashion. One starts off with a larger value and decreases s in each round until sufficient anomalous flow-sets have been investigated.

In summary, the parameters n and s are the simplest as n should generally be large involving additional useful features and s should be varied by the user to investigate different anomalies. The parameters w and m are mainly involved in a detection sensitivity versus aggregation trade-off. This trade-off should be settled based on the average number of daily or weekly anomalous alarms. Having set this trade-off, then a large m, *i.e.,* smaller bins, is desired for anomaly extraction, which should be balanced by a larger w, *i.e.,* 15 minutes in our experiments, to achieve sufficient aggregation. Finally, the parameters l and k serve to balance the number of false and true positives produced by pre-filtering. A range of acceptable values can be determined by simulations using the discussed analytical models.

III. EVALUATION

In this section we first describe the traces we used for our experiments and then evaluate each step of our approach for different parameter settings. In particular, we evaluate the accuracy of our approach, as well as the reduction in classification cost, in terms of flows or item-sets.

A. Data Set and Ground Truth

To validate our approach we used a Netflow trace coming from one of the peering links of a medium-sized ISP

Anomaly class	Occurrences	Mean #flows
Flooding	5	163'139
Backscatter	5	85'716
Network Experiment	3	27'606
DDoS	5	132'509
Scanning	16	96'375
Spam	1	33'765
Unknown	1	23'360
Total	36	99'688

TABLE IV
IDENTIFIED ANOMALIES IN TWO WEEKS OF NETFLOW DATA SEPARATED BY ANOMALY CLASS. FOR EACH CLASS WE GIVE THE NUMBER OF OCCURRENCES AND THE AVERAGE NUMBER OF FLOWS CAUSED BY THIS CLASS OF ANOMALY.

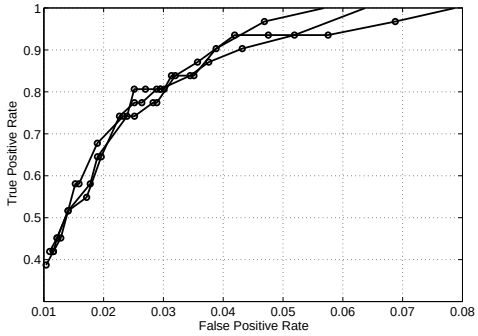


Fig. 6. ROC curves plotting the false positive rate versus the true positive rate for different thresholds. The three curves correspond to different histogram clones.

(SWITCH/AS559). SWITCH is a backbone operator connecting all Swiss universities and various research labs, *e.g.*, CERN, IBM, PSI, to the Internet. We have been collecting non-sampled and non-anonymized NetFlow traces from the peering links of SWITCH since 2003. The SWITCH IP address range contains approximately 2.2 million IP addresses. On average we see 92 million flows and 220 million packets per hour crossing the peering link we used for our experiments. The dataset used for this study was recorded during December 2007 and spans two continuous weeks.

To generate datasets for evaluating the Apriori algorithm, we computed the KL distance timeseries for the two weeks of data for the following feature distributions: source IP address, destination IP address, source port number, destination port number, and flow size in packets. We manually identified 31 anomalous intervals by visual inspection and top-k queries on the data. To determine the root cause of each anomaly, we extracted all flows in an anomalous interval and analyzed the timeseries and distribution of the five features, the the number of packets and bytes per flow, the flow inter-arrival times, and the flow durations. We found a total of 36 different events within the 31 the anomalous intervals. The identified anomalies, their class, and the average number of flows per class are listed in Table IV.

Subsequently, we computed the set of candidate anomalous flows $\cup F_j$ for each anomalous interval using our modified Apriori algorithm. After applying Apriori, we manually analyzed the found frequent item-sets and identified true positives, which matched the identified events, and false positives, which matched benign traffic.

B. Accuracy of Histogram Clones

As a first step we evaluated the *detection accuracy* of our histogram-based detector for different values of the interval length w and the hash function length m. We found small differences in the detection results for m equal to 512, 1024, and 2048. We also found that the number of detections decreases with the interval length w. In particular, setting m to 1024 and w to 5, 10, and 15 minutes, we detected 62, 52, and 31 anomalous intervals, respectively. Based on these number and the parameter selection guidelines we analyzed in Section II-F, we set w conservatively to 15 minutes, which corresponds to 2.2 alarms per day, and m to 1024.

To assess the detection accuracy, we used ROC curve analysis. We computed the number of false positives, *i.e.*, intervals that have an alarm but are not in the ground truth set, and true positives, *i.e.*, intervals that are in the ground truth set and have an alarm. A ROC curve plots the false positive rate (FPR), the ratio between the number of false positives and the total number of intervals that are not in the ground truth set versus the true positive rate (TPR), the ratio between the number of true positives and the total number of intervals with an alarm. Different points in the ROC space are obtained by varying the detection threshold.

In Figure 6 we plot ROC curves for three histogram clones, *i.e.*, using three different hash functions. A detection rate of 0.8 corresponds to a false positive rate of 0.03, while a detection rate of 1 (100%) to a false positive rate between 0.05 and 0.08 for different clones. With a false positive rate as low as 0.01 only 40% of the anomalies are detected. These results are a lower bound on the performance of our detector. This is because some of the false-positive intervals might contain unknown anomalous traffic.

C. Impact of Voting

After the correct interval has been determined, each clone selects b histogram bins that are suspected to contain anomalous flows. The number of responsible bins is determined by the detection threshold and the nature of the anomaly, *i.e.*, whether it is distributed over many feature values or concentrated on a single or few feature values. The probability p_a that a clone correctly identifies an anomalous feature value is equal to the probability that an anomalous feature value has caused the disruption in the histogram, and that the disruption in the respective interval has been detected.

We analyze the impact of voting using simulations. Each clone includes an anomalous feature value in the set V_k with probability p_a, while a normal feature value is selected only if it collides on one of the selected bins with probability $p_n = b/m$. For simulating the impact of different voting strategies on the error probabilities according to Equation 1 and 3, we set $p_a = 0.8$, corresponding to a false positive rate of approximately 0.03 ($m = 1024$) and varied b in the range $[1, 25]$.

In Figure 7 the upper bound for the probability $P_{\bar{a}}$ that an anomalous feature value is missed is plotted for different

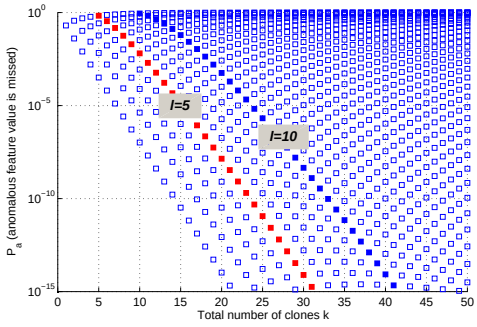
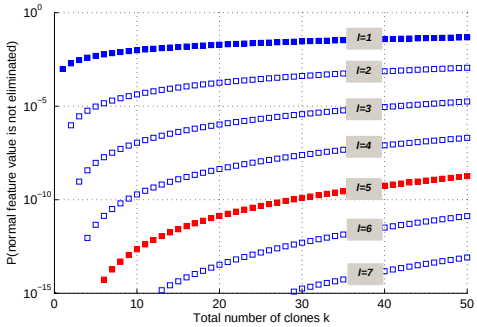
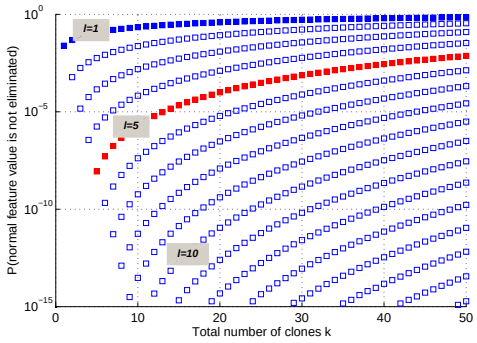


Fig. 7. Upper bound for the probability $P_{\bar{a}}$ that an anomalous feature value is eliminated by voting for different values of l and k in logarithmic scale. The results for $l = 5$, $l = 10$ are marked for better readability. For a given value of k, $P_{\bar{a}}$ increases with l, e.g., for $l = 5$, $k = 10$ we obtain $P_{\bar{a}} = 0.006$ while for $l = 10$, $k = 10$ the probability increases to $P_{\bar{a}} = 0.89$.

values of l and k in logarithmic scale. The results for $l = 5$, $l = 10$ are marked for better readability. For a given value of k, $P_{\bar{a}}$ increases with l, *e.g.*, for $l = 5$ and $k = 10$ we obtain $P_{\bar{a}} = 0.006$, while for $l = 10$ and $k = 10$ the probability increases to $P_{\bar{a}} = 0.89$. Consequently, the upper bound for a fixed number of histogram clones k increases with the number of clones l that are required to agree on a feature value. In particular, it has its minimum for $l = 1$, which corresponds to the union, and is maximized for $l = k$, which corresponds to the intersection.



(a) Number of anomalous bins b =1



(b) Number of anomalous bins b =25

Fig. 8. Probability P_n that a normal feature value is not eliminated by voting for different values of l and k in logarithmic scale. The number of anomalous bins is $b = 1$ (upper plot) and $b = 25$ (lower plot) and the number of total bins is $m = 1024$.

In Figure 8(a) and Figure 8(b) we plot the probability P_n that a normal feature value is not eliminated by voting for different values of l and k in logarithmic scale. The number of selected bins is $b = 1$ and $b = 25$, respectively. The number of total bins is $m = 1024$ for both plots. The results for $l = 1$, $l = 5$ are marked for better readability. For a given value of k, P_n decreases with l, *e.g.*, for $l = 1$ and $k = 10$ the probability for including a normal feature value is $P_n = 10^{-2}$ for $b = 1$ and $P_n = 0.22$ for $b = 25$. For $l = 5$ and $k = 10$ the probability decreases to $P_n = 10^{-13}$ for $b = 1$ and to $P_n = 10^{-6}$ for $b = 25$. Moreover, we observe that the probability of including a normal feature value in the meta-data increases dramatically with the number of anomalous bins b. Consequently, assuming a fixed setting of the voting parameters we have to tolerate higher false positive rates for anomalies affecting multiple bins, *e.g.*, distributed anomalies. Alternatively, the parameter l could be adapted based on the estimated number of bins b to achieve a target probability P_n. The average number of false positive feature values can be determined by multiplication of P_n with the average number of feature values observed within one interval, *e.g.*, between one and 65'536 for port numbers.

The simulation results show that a variety of operating points $[P_{\bar{a}}, P_n]$ can be achieved by setting the voting parameters l, k appropriately. In order to determine the parameters that provide the best overall performance, in terms of accuracy and computational overhead, the following rule mining step needs to be taken into account. The essential questions to answer are i) how is the accuracy impacted by the number of normal feature values included in the meta-data that is used for pre-filtering the candidate flows, and ii) how does the rule mining performance decrease with the number of candidate flows.

D. Accuracy of Rule Mining

After the meta-data has been identified by voting, the corresponding flows are filtered and subsequently send to the rule mining process. The accuracy in terms of correctly identified item-sets depends on three parameters: the accuracy of the meta-data used for per-filtering flows, the frequency of the pre-filtered normal and anomalous flows, and the minimum support parameter s.

An interesting question concerning the accuracy of meta-data is: What is the probability that a normal value in the meta-data results in a false positive item-set. Recall that an item-set will be generated if more than s flows matching the meta-data have one (1-item-set) or more (l-item-set) common feature values. We have observed that the probability for generating a false positive item-set from a normal feature value is highly skewed. For example, if port number 80 is included in the meta-data it is likely that webservers with high load will appear as false positive 2-item-sets in the output of Apriori. Nevertheless, they will be easy to identify as such. On the other hand, if other less frequent port numbers are chosen, few flows will match the feature value and no false positive item-set will be generated.

To further study the rule mining accuracy, we used the flow data of the 31 anomalous intervals. To generate the input data sets for Apriori, we set k to 3, l to 3, and m to 1024. This

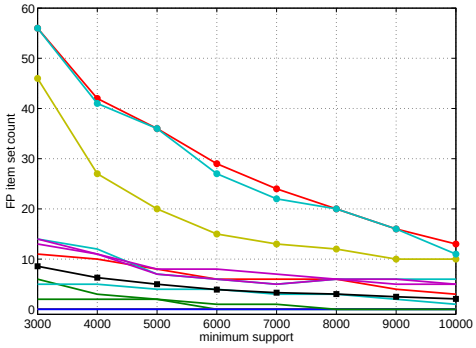


Fig. 9. Number of false positive (FP) item-sets generated by Apriori for different minimum support parameter values for 10 anomalous intervals (30%). For 21 anomalous intervals (70%) we obtain no FP item-sets at all. The average FP item-set count over all 31 anomalous intervals is marked with squares.

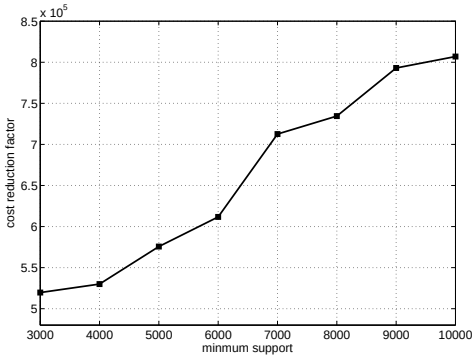


Fig. 10. Average decrease in classification cost vs. minimum support.

corresponds to $P_{\bar{a}}$ =0.488 and $P_n = 10^{-4}$ for $b = 25$. Despite the large value for $P_{\bar{a}}$ none of the 31 anomalies were missed. This illustrates the fact that $P_{\bar{a}}$ is an upper bound that was derived under the assumption of independence between clones. On the other hand, as P_n is very low, only few normal feature values are included in the meta-data.

For 21 anomalous intervals (70%) we obtained no FP item-sets at all. The number of FP item-sets for the remaining 10 anomalous intervals is plotted in Figure 9 together with the average number of FP item-sets over all 31 anomalous intervals (marked with squares). The number of FP item-sets decreases with the minimum support since less FP item-sets satisfy the minimum support condition. Figure 9 shows that on average between 2 and 8.5 FP item-sets are generated for minimum support values between 3,000 and 10,000 flows, respectively. The top three lines in the figure correspond to anomalies with higher numbers of FP item-sets. The observed FP item-sets are exclusively caused by *anomalous* heavy-hitter feature values such as common ports, *e.g.*, port 80, or short flow lengths. Hence, if an anomaly happens to involve such a heavy-hitter feature value the number of FP item-sets automatically increases even if no *normal* feature values are included in the meta-data. However, most of the FP item-sets can be sorted out rather easily by a network administrator.

An important question is which types of anomalies are captured with our rule mining approach. There are two requirements for extracting an anomaly. The anomaly should: i) be detected by causing a deviation in a traffic feature distribution and ii) trigger a large number of flows with similar characteristics. For many anomalies that originate from or terminate to a single or few IP addresses these requirements are met. Scanning, flooding, and spamming activity, (distributed) Denial of Service attacks as well as related backscatter fall into this category. Although the rule mining approach is not targeted at botnet detection, anomalous activities such as spamming, scanning or flooding are often caused by compromised hosts. Other anomalies may not be concentrated on a single or few IP addresses like network outages, routing anomalies, or distributed scanning. Distributed scanning activity typically has a common destination port and often a fixed flow length. Therefore, it will appear as a frequent item-set. Anomalies that affect certain network ranges, such as outages or routing anomalies can be either captured by using IP address prefixes as additional dimensions for rule mining, or by applying concepts from the hierarchical heavy-hitter detection domain [6].

E. Computational Overhead of Rule Mining

The exact computational overhead of Apriori depends highly on the implementation used. Progressive implementations that use FP-trees and database partition techniques [13] have been shown to outperform standard hash tree implementations [1]. Nevertheless, for all implementations the computational overhead increases with the number of transactions and the number of frequent 1-item-sets. Since both, the number of transactions and the number of frequent 1-item-sets increase as more normal flows are included in the input data set, the performance of Apriori will decrease with the size of the input data set, *e.g.*, when we lower the threshold of the histogram-based detectors or do not use the meta-data at all. Moreover, some implementations show considerably longer computation times as the relative minimum support decreases [13], which is equivalent to increasing the data set size and keeping the absolute minimum support constant. In our experiments using a non-optimized implementation in Python, the computation overhead was small requiring few seconds up to minutes in the worst case.

F. Decrease in Classification Cost

Using association rules we obtain a summarized view that is based on frequent item-sets instead of flows. As a consequence, the problem of manually classifying flows can be reduced to the problem of classifying item-sets. To quantify this decrease in classification cost, we assume that the classification cost is a linear function of the number of items that need to be classified. Accordingly, we define the reduction in classification cost r for a given dataset as $r = |F|/|I|$ where $|F|$ denotes the number of flows in the flagged interval and $|I|$ the number of item-sets in the output of Apriori. The number of flows in 15-minute intervals ranges between 700,000 and 2.6 million flows. Since the cardinality of I depends on the minimum support parameter, we plot in Fig.

10 the reduction in classification cost for different values of the minimum support parameter. The average cost reduction increases with the minimum support and ranges between $600,000$ and $800,000$. The cost reduction saturates for larger minimum support parameters as the minimum number of itemsets is reached. This result illustrates that association rule mining can greatly simplify root-cause analysis and attack mitigation.

IV. Related Work

Substantial work has focused on dimensionality reduction for anomaly detection in backbone networks [2], [22], [24], [15], [10], [4], [12]. These papers investigate techniques and appropriate metrics for detecting traffic anomalies, but do not focus on the anomaly extraction problem we address in this paper.

Closer to our work, Dewaele *et al.* [8] use sketches to create multiple random projections of a traffic trace, then model the marginals of the sub-traces using Gamma laws and identify deviations in the parameters of the models as anomalies. In addition, their method finds anomalous source or destination IP addresses by taking the flows matching the intersection of the addresses hashing into anomalous sub-traces. DoWitcher [21] is a scalable system for worm detection and containment in backbone networks. Part of the system automatically constructs a flow-filter mask from the intersection of suspicious attributes provided by different detectors. Li *et al.* [16] use sketches to randomly aggregate flows as an alternative to origin-destination (OD) aggregation. They show that random aggregation can detect more anomalies than OD aggregation in the PCA subspace anomaly detection method [15]. In addition, the authors describe that their method can be used for anomaly extraction, however, the work primarily focuses on anomaly detection.

Association rules have been successfully applied to different problems in networking. Chandola and Kumar [5] use clustering and heuristics to find a minimal set of frequent itemsets that summarizes a large set of flow records. Mahoney and Chan [17] use association rule mining to find rare events that are suspected to represent anomalies in packet payload data. They evaluate their method on the 1999 DARPA/Lincoln Laboratory traces [18]. Their approach targets edge networks where mining rare events is possible. In massive backbone data, however, this approach is less promising. Another application of rule mining in edge networks is eXpose [11], which learns fine-grained communication rules by exploiting the temporal correlation between flows within very short time windows.

Hierarchical heavy-hitter detection methods [9], [25], [6] group traffic into hierarchical clusters of high resource consumption. For example, they have been used to identify clusters of Web servers in hosting farms. Our focus is different as we are interested in extracting anomalous traffic flows.

V. Conclusion

In this paper, we have studied the problem of anomaly extraction that is of uttermost importance to several applications such as root-cause analysis, anomaly mitigation, and detector testing. We presented a histogram-based detector that provides fine-grained meta-data for filtering suspect flows. Further, we introduced a method for extracting and summarizing anomalous flows. Our method models flows as itemsets and mines frequent subsets. It finds large sets of flows with identical values in one or more features. Using datasets from a backbone network we showed that rule mining is very effective, extracting in all studied cases the involved event flows and triggering a low nuumber of false positives in certain cases that could be trivially sorted out. Though we have tested our method with a specific detector, the presented anomaly extraction approach is generic and can be used with other detectors providing useful meta-data about identified anomalies.

References

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*. Morgan Kaufmann, 1994, pp. 487–499.

[2] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. New York, NY, USA: ACM Press, 2002, pp. 71–82.

[3] D. Brauckhoff, X. Dimitropoulos, A. Wagner, and K. Salamatian, "Anomaly extraction in backbone networks using association rules," in *IMC'09*, November 2009.

[4] D. Brauckhoff, M. May, and K. Salamatian, "Applying PCA for Traffic Anomaly Detection: Problems and Solutions," in *IEEE INFOCOM Mini Conference*, 2009.

[5] V. Chandola and V. Kumar, "Summarization - compressing data into an informative representation," *Knowl. Inf. Syst.*, vol. 12, pp. 355–378, 2007.

[6] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava, "Finding hierarchical heavy hitters in streaming data," *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 4, pp. 1–48, 2008.

[7] G. Cormode and S. Muthukrishnan, "What's new: finding significant differences in network data streams," vol. 13, no. 6. Piscataway, NJ, USA: IEEE Press, 2005, pp. 1219–1232.

[8] G. Dewaele, K. Fukuda, P. Borgnat, P. Abry, and K. Cho, "Extracting hidden anomalies using sketch and non gaussian multiresolution statistical detection procedures," in *LSAD '07: Proceedings of the 2007 workshop on Large scale attack defense*. New York, NY, USA: ACM Press, 2007, pp. 145–152.

[9] C. Estan, S. Savage, and G. Varghese, "Automatically inferring patterns of resource consumption in network traffic," in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM Press, 2003, pp. 137–148.

[10] Y. Gu, A. McCallum, and D. Towsley, "Detecting anomalies in network traffic using maximum entropy estimation," in *IMC '05: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*. Berkeley, CA, USA: USENIX Association, 2005, pp. 32–32.

[11] S. Kandula, R. Chandra, and D. Katabi, "What's going on?: learning communication rules in edge networks," in *SIGCOMM*, 2008, pp. 87–98.

[12] A. Kind, M. P. Stoecklin, and X. Dimitropoulos, "Histogram-based traffic anomaly detection," *IEEE Transactions on Network and Service Management*, vol. to appear, 2009.

[13] W. A. Kosters, W. Pijls, and V. Popova, "Complexity analysis of depth first and fp-growth implementations of apriori," in *MLDM*, 2003, pp. 284–292.

[14] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: methods, evaluation, and applications," in *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM Press, 2003, pp. 234–247.

[15] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2005, pp. 217–228.

[16] X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, and A. Lakhina, "Detection and identification of network anomalies using sketch subspaces," in *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2006, pp. 147–152.

[17] M. V. Mahoney and P. K. Chan, "Learning rules for anomaly detection of hostile network traffic," in *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*. Washington, DC, USA: IEEE Computer Society, 2003, pp. 601–604.

[18] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Trans. Inf. Syst. Secur.*, vol. 3, pp. 262–294, 2000.

[19] P. E. Proctor, "Marketscope for network behavior analysis, 2h06," Gartner Inc., Gartner Research Report G00144385, November 2006.

[20] K. H. Ramah, K. Salamatian, and F. Kamoun, "Scan surveillance in internet networks," in *Networking*, 2009, pp. 614–625.

[21] S. Ranjan, S. Shah, A. Nucci, M. M. Munafò, R. L. Cruz, and S. M. Muthukrishnan, "Dowitcher: Effective worm detection and containment in the internet core," in *INFOCOM*, 2007, pp. 2541–2545.

[22] A. Soule, K. Salamatian, and N. Taft, "Combining filtering and statistical methods for anomaly detection," in *IMC'05: Proceedings of the 5th Conference on Internet Measurement 2005, Berkeley, California, USA, October 19-21, 2005*. USENIX Association, 2005, pp. 331–344.

[23] M. P. Stoecklin, J.-Y. L. Boudec, and A. Kind, "A two-layered anomaly detection technique based on multi-modal flow behavior models," in *PAM: Proceedings of 9th International Conference on Passive and Active Measurement*, ser. Lecture Notes in Computer Science. Springer, 2008, pp. 212–221.

[24] 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.

[25] Y. Zhang, S. Singh, S. Sen, N. Duffield, and C. Lund, "Online identification of hierarchical heavy hitters: algorithms, evaluation, and applications," in *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2004, pp. 101–114.