Anomaly Extraction in Backbone Networks Using Association Rules

Daniela Brauckhoff, Xenofontas Dimitropoulos, Arno Wagner, Kavé Salamatian

Anomaly Extraction in Backbone Networks Using Association Rules

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

***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 root-cause analysis, network forensics, attack mitigation, and anomaly modeling. In this paper, we use meta-data provided by several *histogram-based detectors* to identify suspicious flows, and then apply *association rule mining* to find and summarize anomalous flows. Using rich traffic data from a backbone network, we show that our technique effectively finds the flows associated with the anomalous event(s) in all studied cases. In addition, it triggers a very small number of false positives, on average between 2 and 8.5, which 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.**

***Index Terms*—Association rules, computer networks, data mining, detection algorithms.**

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 paper, 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, collecting network forensics, improving anomaly detection accuracy, and modeling anomalies.

Manuscript received November 25, 2009; revised April 25, 2011 and December 10, 2011; accepted January 18, 2012; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Kodialam.

D. Brauckhoff and A. Wagner are with the Computing Department, ETH Zurich, Zurich 8092, Switzerland (e-mail: brauckhoff@tik.ee.ethz.ch; arno@wagner.name).

X. Dimitropoulos is with the Department of Information Technology and Electrical Engineering, ETH Zurich, Zurich 8092, Switzerland (e-mail: fontas@tik.ee.ethz.ch).

K. Salamatian is with LISTIC PolyTech, Université de Savoie Chambery Annecy, Annecy le Vieux Cedex 74944, France (e-mail: kave.salamatian@univ-savoie.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2012.2187306

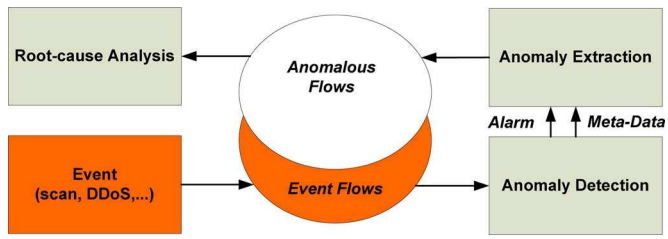

Fig. 1. High-level goal of anomaly extraction is to filter and summarize the set of anomalous flows that coincide with the flows caused by a network event such as denial-of-service (DoS) attacks or scans.

B. Anomaly Extraction

In Fig. 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.

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 that 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 some 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 [16], [8]. Such approaches essentially perform anomaly detection at the level of individual flows and could be used to identify anomalous flows.

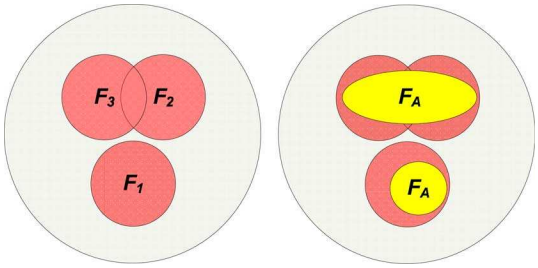

Fig. 2. Each detector j supplies a set of suspicious flows F_j. We filter the union set of suspicious flows $\cup F_j$ and apply association rule mining to extract the set of anomalous flows F_A.

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

Meta-data	Anomaly detection technique
Protocol	Maximum-Entropy [11] Histogram [14], [30]
IP range	Defeat [20] MR-Gaussian [9] DoWitcher [27] Histogram [14], [30]
Port range	Maximum-Entropy [11] Histogram [14], [30] DoWitcher [27]
TCP flags	Maximum-Entropy [11] Histogram [14], [30]
Flow size	DoWitcher [27]
Packet size	Histogram [14], [30]
Flow duration	Histogram [14], [30]

C. Contributions

In this paper, 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 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 Fig. 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: *Anomalies typically result in many flows with 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 (DoS) attack. We test our anomaly extraction method on rich network traffic data from a medium-sized backbone network. The evaluation results show that our approach effectively extracted the anomalous flows in all 31 analyzed cases and, on average, triggered between 2 and 8.5 false positives, which can be trivially filtered out by an administrator. In addition, our solution reduced the classification cost in terms of items that need to be manually classified by several orders of magnitude.

D. Outline

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

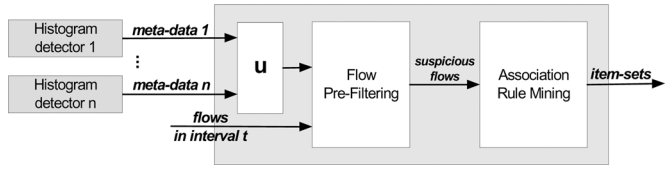

Fig. 3. Overview of our approach to the anomaly extraction problem. The figure illustrates how the meta-data for filtering flows is consolidated from n traffic features by taking the union and how suspicious flows are prefiltered and anomalous flows are summarized in item-sets by association rule mining.

II. METHODOLOGY

An overview of our approach to the anomaly extraction problem is given in Fig. 3. A number n of different histogram-based anomaly detectors monitor network traffic and detect anomalies in an online fashion. Upon detecting an anomaly, we use the union set of meta-data provided by the n detectors to prefilter a set of suspicious flows. This prefiltering is necessary since it eliminates a large fraction of the normal flows. A summary report of frequent item-sets in the set of suspicious flows is generated by applying association rule mining. The basic assumption behind this approach is that frequent item-sets in the prefiltered data are often related to the anomalous event. A large part of our evaluation results is devoted to the verification of this assumption and shows that this is indeed true. The entire anomaly extraction process is automated and can take place both in a online and offline fashion. In the online case, the anomaly detector triggers the anomaly extraction process upon detecting an anomaly. In the offline case, an administrator triggers the anomaly extraction process to analyze anomaly alarms in a post-mortem fashion and to determine their validity.

A. Flow Prefiltering

Assume a time interval t with an anomaly. Prefiltering selects all flows that match the *union* of the meta-data V_j provided by n detectors, i.e., all flows that match $\cup V_j$ where $j = 1, \ldots, n$ are filtered. Prefiltering usually removes a large part of the normal traffic. This is desirable for two reasons. First, it generates a substantially smaller dataset that results in faster processing in the following steps. Second, it improves the accuracy of association rule mining by removing flows that could result in false-positive item-sets.

An important detail of our approach is that we keep flows matching *any* of the meta-data instead of flows matching *all* the meta-data. In other words, we take the *union* of the flows matching meta-data rather than the intersection of the flows matching meta-data. Taking the union is important because identified meta-data can be flow-disjoint, meaning that they appear in different flows, in which case the intersection is empty. For example, consider 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-kB worm executable. In this example, an anomaly would

likely 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. Anomalies often have a multistage footprint, which highlights that taking the intersection of the flows would fail. A comparison between using the union and the intersection for analyzing actual anomalies can be found our previous work [3, Section 3.4]. It shows that the union results in fewer false positives than the intersection, which may entirely miss an anomaly.

B. Frequent Item-Set 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]. Let $I = \{i_1, i_2, \ldots, i_d\}$ be the set of all items in a market basket and $T = \{t_1, t_2, \ldots, t_N\}$ be the set of all transactions. Each transaction t_i contains a subset of items chosen from I. In association analysis, a collection of zero or more items is called an item-set. If an item-set contains l items, it is called an l-item-set [31].

The problem of discovering all association rules in a dataset can be decomposed into two subproblems: 1) the main and most challenging part is finding frequent item-sets, i.e., item-sets that appears in more than a threshold number of transactions; and 2) given the frequent item-sets, derive association rules. The second part is trivial as a frequent item-set implies a set of candidate association rules. For example, a frequent item-set $\{i_1, i_2, i_3\}$ gives the candidate rules $\{i_1, i_2\} \Rightarrow \{i_3\}$, $\{i_1, i_3\} \Rightarrow \{i_2\}$, and $\{i_2, i_3\} \Rightarrow \{i_1\}$. In this paper, we apply the first step of association rule mining, i.e., we find frequent item-sets to extract anomalous flows from a large set of flows observed during a time interval. We do not compute corresponding association rules as this second step does not provide any additional information for the purpose of anomaly extraction. Our assumption for applying frequent item-set mining 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 DoS attack.

We map each flow record into a corresponding transaction t_i. The transaction width is defined as the number of items present in a transaction. Each transaction has a width of seven since each flow record has seven associated features corresponding to its srcIP, dstIP, srcPort, dstPort, protocol, #packets, #bytes. For example, the item $i_1 = \{\mathrm{srcPort} : 80\}$ refers to a source port number equal to 80, while item $i_2 = \{\mathrm{dstPort} : 80\}$ refers to a destination port number 80. By construction, a transaction cannot have two items of the same feature type, e.g., two destination ports.

A transaction t_i is said to contain an item-set X if X is a subset of t_i. An important property of an item-set is its support count, which refers to the number of transactions (flow records) that contain a particular item-set. For example, the support of the 2-item-set $X = \{\mathrm{dstIP} : 129.132.1.1, \mathrm{dstPort} : 80\}$ is equal to the number of flow records that have the given destination IP address and destination port.

Apriori Algorithm: The standard algorithm for discovering frequent item-sets is the Apriori algorithm by Agrawal and Srikant [1]. Apriori computes in each round the support for all candidate l-item-sets. At the end of each round, the l-item-sets with frequency above the minimum support parameter are selected. 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. In our setup, Apriori makes at most seven passes over the dataset as each transaction (flow record) has exactly seven features.

By default, Apriori outputs all frequent item-sets that it finds. We modify this to output only maximal frequent item-sets, i.e., frequent l-item-sets that are not a subset of a more specific frequent $(l+1)$-item-set. Maximal item-sets are desirable since they significantly reduce the number of item-sets to process by a human expert. The Apriori algorithm takes one parameter, i.e., the *minimum support* threshold, 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 negatives).

Apriori Example: In the following, we give an example of using Apriori to extract anomalies. For the purpose of this example, we used a 15-min window of data extracted from our traces (2 weeks long). In this trace, destination port 7000 was the only feature value that was flagged by our detectors. It contributed 53 467 candidate anomalous flows. We forced Apriori to artificially generate false-positive frequent item-sets by manually adding to the candidate set $\cup F_j$ flows that had one of the three most frequent destination ports but had not been flagged by our detector. 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. Such frequent flow features can lead to false positives if they go through our prefiltering process due to collisions with anomalous features. 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. However, 58 of these were removed from the output as subsets of at least one frequent 2-item-set, i.e., these frequent item-sets were not maximal. 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-set satisfying the minimum support was 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,

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

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

which sent a lot of traffic on destination port 80, were HTTP proxies or caches. The traffic on destination port 9022 (22 573 flows) was backscatter since each flow has a different source IP address and a random source port number. This backscatter anomaly was flagged by the detector in an earlier interval where it had started. The remaining item-sets refer to combinations of common destination ports and flow sizes and illustrate false positives that we artificially added in this example for illustration. Such frequent features can lead to false positives if they go through our prefiltering process. However, due to their frequent nature, typically they can be easily identified and filtered our by administrator. A key feature of our anomaly extraction is that in addition to leading to a small number of false positives, as we also discuss in the evaluation section, often it is possible to easily spot and filter our false positives.

C. Histogram-Based Detector

Histogram-based anomaly detectors [14], [30], [20], [26], have been shown to work well for detecting anomalous behavior and changes in traffic distributions. We build a histogram-based detector for our evaluation that uses the Kullback–Leibler (KL) distance to detect anomalies. The KL distance has been successfully applied for anomaly detection in previous work [11], [26]. 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 each m histogram bins. Our approach for binning of feature values to m histogram bins will be described in the next section.

During time interval t, an anomaly detection module constructs histograms for the number of flows per traffic feature. At the end of each interval, it computes for each histogram the KL distance between the distribution of the current interval and a reference distribution. The KL distance measures the similarity of a given discrete distribution q to a reference distribution p and is defined as

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

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 $D(p\|q) \neq D(q\|p)$.

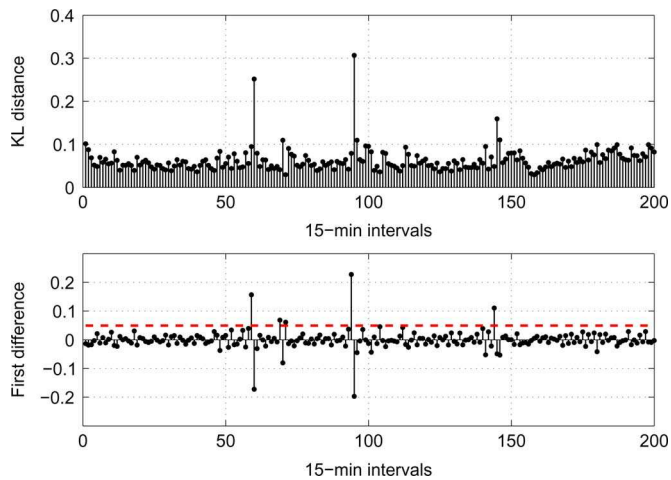

Fig. 4. *(top)* KL distance time series for the source IP address feature for roughly two days. *(bottom)* First difference of the KL distance for the same period. The dashed line corresponds to the anomaly detection threshold.

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 flow 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 Fig. 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 us 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

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

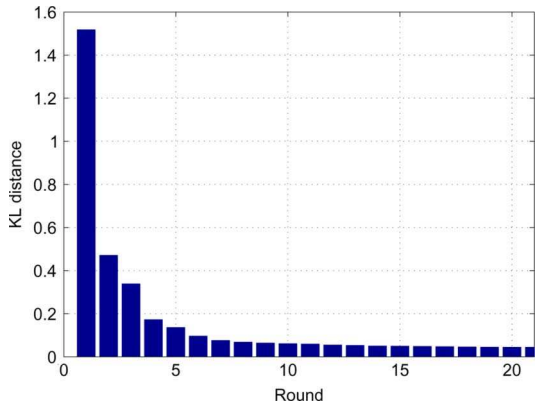

Fig. 5. Iterative method for determining the anomalous bins. The KL distance converges to zero as, in each round, the bin with the largest absolute difference is aligned with its counterpart in the reference distribution. Already after the first round, the KL distance decreases significantly.

In Fig. 4, we show the $\Delta_t\ D(p\|q)$ time series for the source IP address feature and the corresponding threshold. We deliberately use an one-sided threshold, i.e., an alarm is only generated for positive spikes crossing the threshold, since positive spikes correspond to a significant increase in the KL distance representing many additional similar flows whereas negative spikes typically denote the end of an anomaly.

If an anomaly is detected during interval t in the distribution of traffic feature j, we need to identify the set V_j of affected feature values, e.g., the IP addresses that have been targeted by a denial-of-service attack. In a first step, we identify the set of histogram bins B_j contributing to the KL distance spike. Then, in a second step, the corresponding feature values V_j, e.g., the DoS victim IPs, are identified. This two-step approach is taken since typically a large number of feature values are aggregated into a single bin.

In the first step, to find the contributing histogram bins, 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 the largest absolute distance $\max_{i\in[0,m]}\ |p_i - q_i|$ between the histogram of the previous and current interval. The motivation here is that anomalous flows, e.g., belonging to a denial-of-service attack, will cause such a difference in flow counts for certain bins. 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 "cleaned" histogram does not generate an alert any more. This procedure is illustrated in Fig. 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_j by simulation, we need to obtain the corresponding set of feature values V_j in the second step. This task is trivial if each bin contains only a single feature value. However, such an approach is not feasible with sparse traffic features like IP addresses, which are typically used for anomaly detection. Our approach for identifying the corresponding feature values is explained in the next section.

D. Histogram Cloning and Voting

Histogram binning typically groups in a rather ad hoc way adjacent feature values, e.g., adjacent IP addresses, into a desired number of bins. As an alternative to arbitrary binning, we introduce histogram cloning. With histogram cloning, different clones provide alternative ways to group feature values into a desired number of bins/groups creating effectively additional views along which an anomaly may be visible. The cloning mechanism is coupled with a simple voting scheme that controls the sensitivity of the detector and eventually affects a tradeoff between false positives and negatives.

In particular, 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] Upon detection of a disruption in the distribution, each clone compiles a list $V_{j,k}$ of traffic feature values that are associated with the disruption by keeping a map of bins and corresponding feature values. The advantage of histogram cloning is that we obtain additional traffic views that help us in identifying the correct feature values V_j in each anomalous bin using a voting scheme. In the short version [3] of this work, we only keep feature values that have been identified by all histogram clones $V_j = \bigcap_k V_{j,k}$. 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 tradeoff between false-positive and false-negative feature values can be adjusted via the parameters k and l.

Typically, a histogram bin corresponds to a large number of feature values, e.g., the 65 K 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 $V_{j,k}$ identified by each clone contains a large number normal feature values colliding on anomalous bins. Using the traffic views provided by k clones drastically reduces the probability $(1/m)^k$ that a normal feature value appear in an anomalous bin in all of the k clones.

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. Therefore, each clone includes the anomalous feature value in the set $V_{j,k}$ with probability p_a. A distinct normal feature value, on the other hand, is selected only if it collides on one of the selected bins and thus has 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 l (out of k) or more clones

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

[1]Note that histogram cloning uses random projections as they are commonly used in sketch data structures, e.g., [6], 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.

TABLE III
PARAMETERS INCLUDING DESCRIPTION AND RANGE AS USED IN SECTION III

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	1% - 10%

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

$$P_{\bar{a}} \leq 1 - \sum_{i=l}^{k} \binom{k}{i} p_a^i (1-p_a)^{k-i}. \quad (2)$$

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

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

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

To sum up, the meta-data V_j for feature j obtained after the voting process contain feature values representing normal and anomalous traffic. The ratio of normal and anomalous feature values depends on the parameters k and l, on the initial probability p_a, and the hash function length m. The impact of these parameters on the overall accuracy of our approach is analyzed next.

E. 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, some also impact the extraction part. In the following, we describe each parameter in detail and discuss selection criteria.

Number of Detectors n: In this paper, we use five detectors, which correspond to five features that are frequently used for network traffic anomaly detection: source IP addresses, destination IP addresses, source port numbers, destination port numbers, and number of packets per flow. In principle, if the computational overhead is reasonable, which is the case with our detector, more features are welcome since they provide additional views along which an anomaly may prevail. Other features that can be useful for anomaly detection are the following: the number of packets per flow, the average packet size, the duration of a flow, the source/destination autonomous system (AS) numbers, and the geographical distribution of IP addresses.

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 min investigation time [25], which would correspond to eight alarms per day assuming a full-time employee 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., 5–15 min. 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 after prefiltering is relatively low as we discuss in Section III.

Hash Function Length m: The hash function length m is also involved in a detection sensitivity versus aggregation tradeoff as 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 (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 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 prefiltering step and thus the accuracy of our approach. The parameter settings for l and k can also be obtained by simulation using (1) and (3). Simulation results for $P_{\bar{a}}$ and P_n for different settings of l and k will be presented in Section III.

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. The size of the top item-sets depends on many factors, like the used interval length, the monitored link rate(s), the type of filtering used, and the traffic mix among others. A specific value for s is unlikely to work in all cases. The value of s needs to be determined by trial and error. In our evaluation of the number of false positives in Section III, we have used a range between 3000 and 10 000 flows that resulted in a small number of item-sets. In general, through our extensive experimentation, we have learned that a suitable s is typically in the range between 1% and 10% of the total number of input flows. Starting with a value within this range, typically one can arrive to a suitable s within a small number of 2–3 trials. One possibility is to select a very low s that will

generate a large number of item-sets. Note that the generated item-sets can be ranked by their frequency. Then, one can keep only the top item-sets according to the frequency ranking. This could include, for example, the top 10 or top 20 item-sets as desired. The cost of a lower s is more overhead for running the algorithm. Alternatively, one can start with a high s and progressively decrease it until a sufficient number of anomalous item-sets has 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 low or variable. The parameters w and m are mainly involved in a detection-sensitivity-versus-aggregation tradeoff. This tradeoff should be settled based on the average number of daily or weekly anomalous alarms. Having set this tradeoff, then a large m, i.e., smaller bins, is desired for anomaly extraction, which should be balanced by a larger w, i.e., 15 min 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 prefiltering. 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. Dataset and Ground Truth

To validate our approach, we used a Netflow trace coming from one of the peering links of a medium-sized ISP (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 nonsampled and nonanonymized 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 time series 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 time series and distribution of the five features, the number of packets and bytes per flow, the flow interarrival 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. Determining the class of an anomaly is a complex manual process that combines hints extracted from visual inspection, like targeted ports or IP addresses, with the expertise of the analyst and with knowledge about malware from forums and threat expert reports to reason about the proper class of an anomaly. We classified anomalies based on a manual process into seven classes: Flooding, Backscatter, Network Experiment, DDoS, Scanning, Spam, and Unknown. Our class "Network Experiment" corresponds to anomalies we traced to a PlanetLab node running in our university. The "Spam" class corresponds to anomalies targeting SMTP servers, while "Flooding" differs from a standard "DDoS" in that it involves a small number of sources.

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

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

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 min, we detected 62, 52, and 31 anomalous intervals, respectively. Based on these numbers and the parameter selection guidelines we analyzed in Section II-E, we set w conservatively to 15 min, which corresponds to 2.2 alarms per day, and m to 1024.

To assess the detection accuracy, we used receiver operating characteristic (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. An 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 Fig. 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

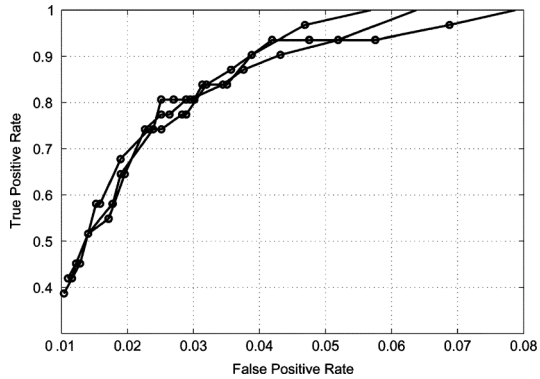

Fig. 6. ROC curves plotting the false positive rate versus the true positive rate for different thresholds. The three curves correspond to different histogram clones.

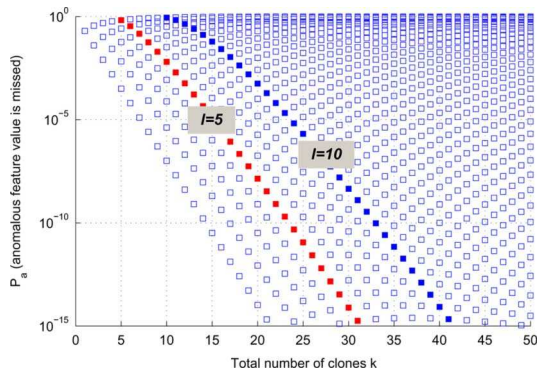

Fig. 7. Upper bound for the probability $P_{\bar{a}}$ that an anomalous feature value is eliminated by voting for different values of l and k in logarithmic scale. The results for $l = 5$, $l = 10$ are marked for better readability. For a given value of k, $P_{\bar{a}}$ increases with l, e.g., for $l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

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 the disruption 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 (1) and (3), we set $p_a = 0.8$, corresponding to a false positive rate of approximately 0.03 and varied b in the range [1, 25].

In Fig. 7, the upper bound for the probability $P_{\bar{a}}$ that an anomalous feature value is missed is plotted for different values of l and k in logarithmic scale. The results for $l = 5$ and $l = 10$ 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 $l = 1$ and is maximized for $l = k$.

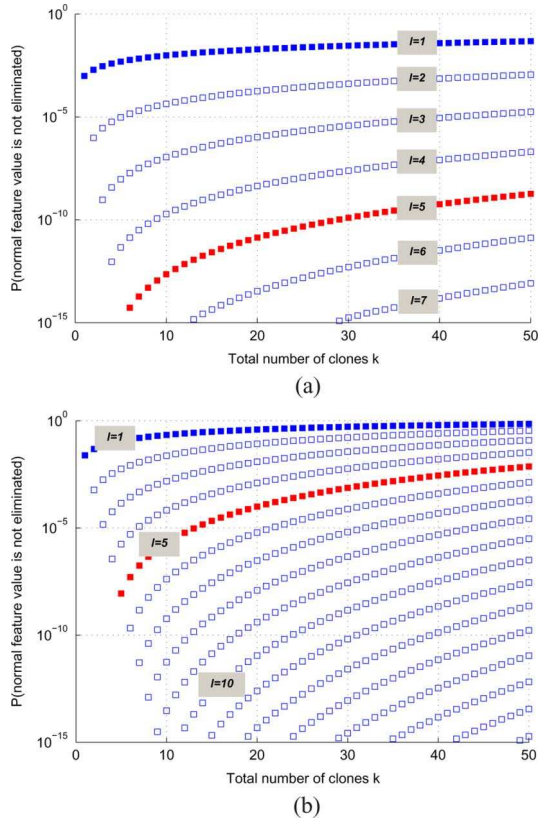

Fig. 8. Probability P_n that a normal feature value is not eliminated by voting for different values of l and k in logarithmic scale. The number of anomalous bins is (a) $b = 1$ and (b) $b = 25$, and the number of total bins is $m = 1024$.

In Fig. 8(a) and (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 $l = 1$ and $l = 5$ 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 1 and 65 536 for port numbers.

The simulation results show that a variety of operating points $[P_{\tilde{a}}, P_n]$ can be achieved by setting the voting parameters l, k appropriately. The selection of the parameters l and k can be further optimized taking into account the induced accuracy and overhead in the rule mining step. The essential questions to answer are the following: 1) how is the accuracy impacted by the number of normal feature values included in the meta-data that is used for prefiltering the candidate flows; and 2) how does the rule mining performance decrease with the number of candidate flows?

D. Accuracy of Frequent Item-Set Mining

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

An interesting question concerning the accuracy of meta-data is the following: 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 Web servers 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 item-set 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 corresponds to $P_{\tilde{a}} = 0.488$ and $P_n = 10^{-4}$ for $b = 25$. Despite the large value for $P_{\tilde{a}}$, none of the 31 anomalies were missed. This illustrates the fact that $P_{\tilde{a}}$ 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 Fig. 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. Fig. 9 shows that on average between 2 and 8.5 FP item-sets are generated for minimum support values between 3000 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 common feature values such as ports, e.g., port 80, or short flow lengths. Hence, if an anomaly happens to involve such a common 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.

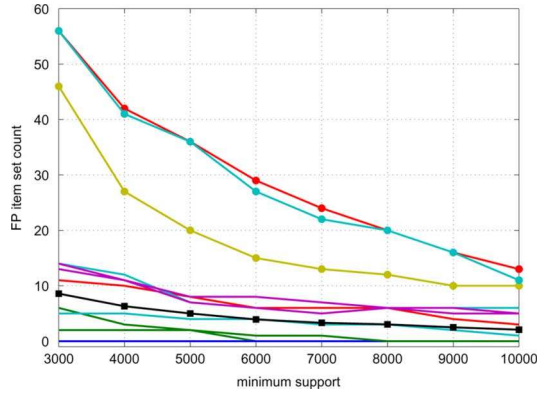

Fig. 9. Number of false positive (FP) item-sets generated by Apriori for different minimum support parameter values for 10 anomalous intervals (30%). For 21 anomalous intervals (70%), we obtain no FP item-sets at all. The average FP item-set count over all 31 anomalous intervals is marked with squares.

An important question is which types of anomalies are captured with our item-set mining approach. There are two requirements for extracting an anomaly. The anomaly should: 1) be detected by causing a deviation in a traffic feature distribution; and 2) trigger a large number of flows with similar characteristics. For many anomalies that originate from or are directed 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 can be identified by frequent item-sets. Although the item-set 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. However, distributed scanning activity typically has a common destination port and often a fixed flow length that 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 item-set mining, or by applying concepts from the hierarchical heavy-hitter detection domain [7].

E. Computational Overhead

The computational cost for updating histograms and for computing the KL-distance is linear to the number of histogram bins. The memory cost is also quite low. For example, 5 detectors with 3 clones and 1024 histogram bins require 472 kB of memory. The iterative method for determining anomalous bins converges fairly fast as shown in Fig. 5 and only needs to be executed when an anomaly is found. Frequent item-set mining is the most demanding step of our methodology both in terms of running time and memory overhead. The exact computational overhead of Apriori depends highly on the implementation used. Progressive implementations that use FP-trees and database partition techniques [15] 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

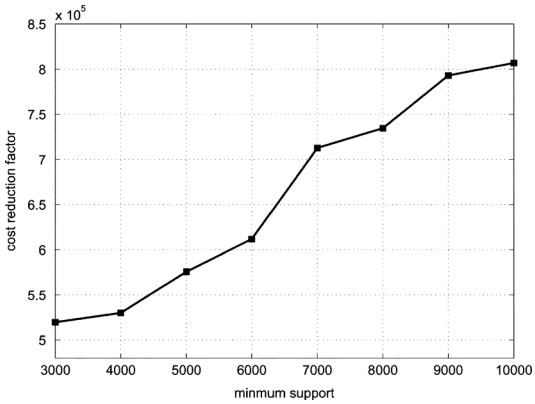

Fig. 10. Average decrease in classification cost versus minimum support.

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 meta-data at all. Moreover, some implementations show considerably longer computation times as the relative minimum support decreases [15], which is equivalent to increasing the dataset size and keeping the absolute minimum support constant. For our experiments, we used an unoptimized implementation of Apriori in Python. Even with the unoptimized implementation, we were able to find frequent item-sets within reasonable time: In the worst case, the algorithm required 5 min to compute frequent item-sets on a Dual Core AMD Opteron 275 processor. A large number of frequent item-set mining algorithms with better computational efficiently than Apriori have been proposed in the literature. A survey of findings can be found in [12]. We believe that it is possible to substantially optimize the computational overhead of our implementation. However, such optimization goes beyond the scope of this paper.

F. Decrease in Classification Cost

As a result of our approach, we obtain a summarized view that is based on frequent item-sets instead of flows. Hence, 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 $r = |F|/|O|$, where $|F|$ denotes the number of flows in the flagged interval and $|O|$ the number of item-sets in the output of Apriori. The number of flows in 15-min intervals ranges between 700 000 and 2.6 million flows. Since the cardinality of O 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 item-sets is reached. This result illustrates that association rule mining can greatly simplify root-cause analysis and attack mitigation.

IV. Related Work

A short version of our work has previously appeared in [3]. Most related to our work, Silveira and Diot [28] recently introduced a tool called URCA that searches for anomalous flows by iteratively eliminating subsets of normal flows. URCA also classifies the type of a detected anomaly. Nevertheless, it requires to repeatedly evaluate an anomaly detector on different flow subsets, which can be costly. Compared to this work, we show that simply computing frequent item-sets on prefiltered flows is sufficient to identify anomalous flows. DoWitcher [27] 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 (meta-data) provided by different detectors. We also leverage suspicious attributes from an anomaly detector and study the anomaly extraction problem in more depth. We highlight that using the intersection can miss anomalous flows and find that the union of the meta-data combined with association rule mining gives better results. Dewaele *et al.* [9] use sketches to create multiple random projections of a traffic trace, then model the marginals of the subtraces using Gamma laws and identify deviations in the parameters of the models as anomalies. In addition, their method finds possible anomalous source or destination IP addresses by taking the intersection of the addresses hashing into anomalous subtraces. Compared to this work, we introduce and validate techniques to address the more challenging problem of finding anomalous flows rather than IP addresses. Lakhina *et al.* [17] use SNMP data to detect network-wide volume anomalies and to pinpoint the origin–destination (OD) flow along which an anomaly existed. In contrast, our approach takes as input a large number of flow records, e.g., standard 5-tuple flows, and extracts anomalous flows. An OD flow may include millions of both normal and anomalous 5-tuple flows and, therefore, can form the input to our methodology. Li *et al.* [20], use sketches to randomly aggregate flows as an alternative to OD aggregation. The authors show that random aggregation can detect more anomalies than OD aggregation in the PCA subspace anomaly detection method [18]. In addition, the authors discuss how their method can be used for anomaly extraction. However, the work and evaluation focus primarily on anomaly detection.

Association rules have been successfully applied to different problems on networking. Lee and Stolfo [19] show how association rules can be used to extract interesting intrusion patterns from system calls and tcpdump logs. Vaarandi [32] introduces a tool called LogHound that provides an optimized implementation of Apriori and demonstrates how LogHound can be used to summarize traffic flow records. Yoshida *et al.* [34] also use frequent item-set mining to identify interesting events in traces from the MAWI traffic archive [23]. Li and Deng [21] outline a variant of the Eclat frequent item-set mining algorithm [35] that operates in a sliding window fashion and evaluate it using traffic flow traces from a Chinese university. Chandola and Kumar [5] describe heuristics for finding a minimal set of frequent item-sets that summarizes a large set of flows. Mahoney and Chan [22] 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 [24]. 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 [13], which learns fine-grained communication rules by exploiting the temporal correlation between flows within very short time windows. Compared to these studies, we show how association rule mining can be combined with anomaly detection to effectively extract anomalous flows.

Hierarchical heavy-hitter detection methods [10], [36], [7] group traffic into hierarchical clusters of high resource consumption and focus primarily on optimizing computational performance for summarizing normal traffic. For example, they have been used to identify clusters of Web servers in hosting farms. Hierarchical heavy-hitter detection is similar to frequent item-set mining in that both approaches find different forms of multidimensional heavy hitters. Compared to these studies, we learn that intelligently combining multidimensional heavy-hitters with anomaly detection enables us to extract anomalous flows. In addition, frequent item-set mining scales to higher dimensions much better than existing hierarchical heavy-hitter detection methods.

Finally, substantial work has focused on dimensionality reduction for anomaly detection in backbone networks [2], [29], [33], [18], [11], [4], [14]. 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.

V. Conclusion

Anomaly extraction takes as input a large set of flows and aims at finding the flows associated with the event(s) that triggered an observed anomaly. It is very useful for finding the root cause of detected anomalies, which helps in anomaly mitigation, network forensics, and anomaly modeling. In this paper, we first introduced a histogram-based detector that provides fine-grained meta-data for filtering suspect flows. Furthermore, we introduced a method for extracting and summarizing anomalous flows. Our method models flows as transactions and applies frequent item-set mining to find large sets of flows with identical values in one or more features. Using real anomalies and traffic traces from a medium-sized backbone network, we showed empirically that the extracted frequent item-sets pinpoint the root cause of the anomalies in all (31) studied cases. In addition, frequent item-set mining produced very few false positives, which could be trivially filtered out by an administrator. The presented anomaly extraction approach is generic and can be used with different anomaly detectors that provide meta-data about identified anomalies. It reduces the work-hours needed for the manual verification of anomaly alarms.

A number of possible directions for future research exist. Optimizing the scalability and efficiency of frequent item-set mining for dealing with big network traffic data including stream processing is one open problem. Association rule mining is likely the most well-studied data mining problem with a very large number of variants of the Apriori algorithm. Mining top-k item-sets; mining closed or maximal frequent item-sets; and mining on multilevel, multidimensional, or quantitative features are possible extensions to our work that could provide useful additional features for network traffic monitoring.

References

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proc. 20th VLDB*, Santiago de Chile, Chile, Sep. 12–15, 1994, pp. 487–499.

[2] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *Proc. ACM SIGCOMM IMW*, 2002, pp. 71–82.

[3] D. Brauckhoff, X. Dimitropoulos, A. Wagner, and K. Salamatian, "Anomaly extraction in backbone networks using association rules," in *Proc. 9th ACM SIGCOMM IMC*, 2009, pp. 28–34.

[4] D. Brauckhoff, M. May, and K. Salamatian, "Applying PCA for traffic anomaly detection: Problems and solutions," in *Proc. IEEE INFOCOM Mini Conf.*, 2009, pp. 2866–2870.

[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 and S. Muthukrishnan, "An improved data stream summary: The count-min sketch and its applications," *J. Algor.*, vol. 55, no. 1, pp. 58–75, 2005.

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

[8] G. Cormode and S. Muthukrishnan, "What's new: Finding significant differences in network data streams," *IEEE/ACM Trans. Netw.*, vol. 13, no. 6, pp. 1219–1232, Dec. 2005.

[9] G. Dewaele, K. Fukuda, P. Borgnat, P. Abry, and K. Cho, "Extracting hidden anomalies using sketch and non Gaussian multiresolution statistical detection procedures," in *Proc. LSAD*, 2007, pp. 145–152.

[10] C. Estan, S. Savage, and G. Varghese, "Automatically inferring patterns of resource consumption in network traffic," in *Proc. ACM SIGCOMM*, 2003, pp. 137–148.

[11] Y. Gu, A. McCallum, and D. Towsley, "Detecting anomalies in network traffic using maximum entropy estimation," in *Proc. 5th ACM SIGCOMM IMC*, 2005, pp. 32–32.

[12] J. Han, H. Cheng, D. Xin, and X. Yan, "Frequent pattern mining: Current status and future directions," *Data Min. Knowl. Discov.* vol. 15, pp. 55–86, Aug. 2007.

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

[14] A. Kind, M. P. Stoecklin, and X. Dimitropoulos, "Histogram-based traffic anomaly detection," *IEEE Trans. Netw. Service Manage.*, vol. 6, no. 2, pp. 110–121, Jun. 2009.

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

[16] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: Methods, evaluation, and applications," in *Proc. 3rd ACM SIGCOMM IMC*, 2003, pp. 234–247.

[17] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," in *Proc. ACM SIGCOMM*, 2004, pp. 219–230.

[18] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *Proc. ACM SIGCOMM*, 2005, pp. 217–228.

[19] W. Lee and S. J. Stolfo, "Data mining approaches for intrusion detection," in *Proc. 7th USENIX Security Symp.*, 1998, vol. 7, p. 6.

[20] 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 *Proc. 6th ACM SIGCOMM IMC*, 2006, pp. 147–152.

[21] X. Li and Z.-H. Deng, "Mining frequent patterns from network flows for monitoring network," *Expert Syst. Appl.* vol. 37, no. 12, pp. 8850–8860, 2010.

[22] M. V. Mahoney and P. K. Chan, "Learning rules for anomaly detection of hostile network traffic," in *Proc. 3rd IEEE ICDM*, 2003, pp. 601–604.

[23] MAWI, "The MAWI Working Group traffic archive," [Online]. Available: http://mawi.wide.ad.jp/mawi/

[24] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory," *Trans. Inf. Syst. Secur.*, vol. 3, pp. 262–294, 2000.
[25] P. E. Proctor, "Marketscope for network behavior analysis, 2H06," Gartner, Inc., Stamford, CT, Gartner Res. Rep. G00144385, Nov. 2006.
[26] K. H. Ramah, K. Salamatian, and F. Kamoun, "Scan surveillance in Internet networks," in *Proc. Netw.*, 2009, pp. 614–625.
[27] 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 *Proc. IEEE INFOCOM*, 2007, pp. 2541–2545.
[28] F. Silveira and C. Diot, "URCA: Pulling out anomalies by their root causes," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.
[29] A. Soule, K. Salamatian, and N. Taft, "Combining filtering and statistical methods for anomaly detection," in *Proc. 5th ACM SIGCOMM IMC*, Oct. 19–21, 2005, pp. 331–344.
[30] M. P. Stoecklin, J.-Y. L. Boudec, and A. Kind, "A two-layered anomaly detection technique based on multi-modal flow behavior models," in *Proc. 9th PAM*, 2008, Lecture Notes in Computer Science, pp. 212–221.
[31] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, 1st ed. Boston, MA: Addison-Wesley Longman, 2005.
[32] R. Vaarandi, "Mining event logs with SLCT and LogHound," in *Proc. IEEE NOMS*, Apr. 2008, pp. 1071–1074.
[33] A. Wagner and B. Plattner, "Entropy based worm and anomaly detection in fast IP networks," in *Proc. 14th IEEE WETICE*, 2005, pp. 172–177.
[34] K. Yoshida, Y. Shomura, and Y. Watanabe, "Visualizing network status," in *Proc. Int. Conf. Mach. Learning Cybern.*, Aug. 2007, vol. 4, pp. 2094–2099.
[35] M. Zaki, "Scalable algorithms for association mining," *IEEE Trans. Knowl. Data Eng.*, vol. 12, no. 3, pp. 372–390, May–Jun. 2000.
[36] Y. Zhang, S. Singh, S. Sen, N. Duffield, and C. Lund, "Online identification of hierarchical heavy hitters: Algorithms, evaluation, and applications," in *Proc. ACM SIGCOMM IMC*, 2004, pp. 101–114.

Daniela Brauckhoff received the Ph.D. degree in electrical and computer engineering from ETH Zurich, Zurich, Switzerland, in 2010.

She is a Security Architect in the financial industry. She joined the Communications Systems Group (CSG), ETH, in 2005 as a Research Assistant. During her Ph.D. studies, she worked on applying principal component analysis in the context of anomaly detection and on the anomaly extraction problem. Further contributions were made to the topic of anomaly detection on packet-sampled network data. She also participated in the development of FLAME, a collection of tools to facilitate the modification of flow traces. FLAME is published under GPLv2 and a modified BSD license. Her research interests include backbone network measurements on flow data, network security, and anomaly detection.

Xenofontas Dimitropoulos received the Ph.D. degree in electrical and computer engineering from Georgia Institute of Technology, Atlanta, in 2006.

He is a Senior Researcher and Lecturer with the Communication Systems Group (CSG), ETH Zurich, Zurich, Switzerland. During his Ph.D. studies, he conducted research on measuring the AS topology of the Internet and on building the BGP++ parallel-distributed simulator. In the past, he was a Post-Doc with IBM Research, Zurich, Switzerland, doing research on the Aurora network traffic profiling system (now known as Tivoli Netcool Performance Flow Analyzer), and a Visiting Scholar with the Cooperative Association for Internet Data Analysis (CAIDA), San Diego, CA, working on inferring and modeling AS topologies. His present research interests focus on network measurements. More information can be found at http://www.fontas.net.

Dr. Dimitropoulos has had various honors, such as a best paper award, a Fulbright scholarship, and a Marie Curie fellowship. In addition, he has been in the program committee of conferences such as the ACM Internet Measurement Conference (IMC).

Arno Wagner received the Ph.D. degree in Internet security from the Swiss Federal Institute of Technology at Zurich, Zurich, Switzerland, in 2008.

He is currently a Senior Security Consultant with Consecom AG and a Lecturer with the Zurich University of Applied Sciences (ZHAW), both in Zurich, Switzerland. His research interests include network anomaly detection, Internet security, and software security.

Kavé Salamatian received the Ph.D. degree in computer science from Paris SUD–Orsay University, Orsay, France, in 1998. He also received the M.B.A. degree from Isfahan University of Technology, Isfahan, Iran, in 1993.

He is a Professor with the University of Savoie, Annecy-le-Vieux, France. He was previously a Reader with Lancaster University, Lancaster, U.K., and an Associate Professor with the University Pierre et Marie Curie, Paris, France. He also worked on the market floor as a Risk Analyst and enjoyed being an Urban Traffic Modeler for some years. During his Ph.D. studies, he worked on joint source channel coding applied to multimedia transmission over Internet. His main areas of research are Internet measurement and modeling and networking information theory. These days, he is working on figuring out if networking is a science or just a hobby, and if it is a science, what are its fundamentals.