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
ETH Zurich
Zurich, Switzerland
brauckhoff@tik.ee.ethz.ch

Xenofontas Dimitropoulos
ETH Zurich
Zurich, Switzerland
fontas@tik.ee.ethz.ch

Arno Wagner
ETH Zurich
Zurich, Switzerland
wagner@tik.ee.ethz.ch

Kavè Salamatian
Lancaster University
Lancaster, United Kingdom
kave@lancaster.ac.uk

ABSTRACT

Anomaly extraction is an important problem essential to several applications ranging from root cause analysis, to attack mitigation, and testing anomaly detectors. Anomaly extraction is preceded by an anomaly detection step, which detects anomalous events and may identify a large set of possible associated event flows. The goal of anomaly extraction is to find and summarize the set of flows that are effectively caused by the anomalous event.

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 the event flows. Using rich traffic data from a backbone network (SWITCH/AS559), we show that we can reduce the classification cost, in terms of items (flows or rules) that need to be classified, by several orders of magnitude. Further, we show that our techniques effectively isolate event flows in all analyzed cases and that on average trigger between 2 and 8.5 false positives, which can be trivially sorted out by an administrator.

Categories and Subject Descriptors

C.2.6 [**Computer - Communication Networks**]: Internetworking

General Terms

Design, Experimentation, Measurement

Keywords

Anomaly extraction, association rules, histogram cloning

IMC'09, November 4–6, 2009, Chicago, Illinois, USA.

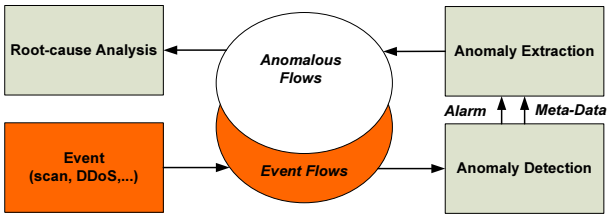

Figure 1: High-level goal of anomaly extraction.

1. INTRODUCTION

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.

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.

Meta-data	Anomaly detection technique
Protocol	Maximum-Entropy [9]
	Histogram [11, 21]
IP range	Defeat [15]
	MR-Gaussian [7]
	DoWitcher [19]
	Histogram [11, 21]
Port range	Maximum-Entropy [9]
	Histogram [11, 21]
	DoWitcher [19]
TCP flags	Maximum-Entropy [9]
	Histogram [11, 21]
Flow size	DoWitcher [19]
Packet size	Histogram [11, 21]
Flow duration	Histogram [11, 21]

Table 1: Useful meta-data provided by various anomaly detectors. The listed meta-data can be used to identify suspicious 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 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 1 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 [12]. Such approaches essentially perform anomaly detection at the level of individual flows and could be used to identify anomalous flows.

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 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.

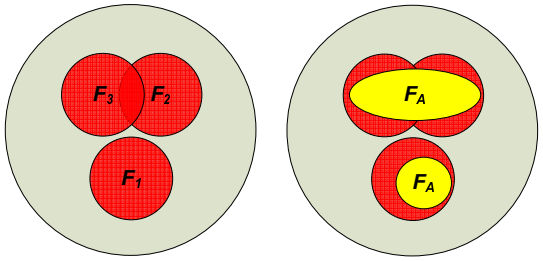

Figure 2: Each detector j supplies a set of suspicious flows F_j. We filter the union set of suspicious flows $\cup F_j$ and apply association rule mining to extract the set of anomalous flows F_A.

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 then present evaluation results. Related work is discussed in Section 4. Finally, Section 5 concludes the paper.

2. METHODOLOGY

In the following section we outline our approach for generating fine-grained meta-data with histogram-based detectors, and for finding the set of anomalous flows with the help of association rules.

2.1 Histogram-based Detection

Histogram-based anomaly detectors [11, 21, 15, 18] have been shown to work well for detecting anomalous behavior and changes in traffic distributions. We build our own 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 random bin. Each histogram-based detector $j = 1 \ldots n$ uses k histogram clones with independent hash functions.

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 [9, 18]. It measures the similarity of a given discrete distribution q to a reference distribution p and is defined as

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

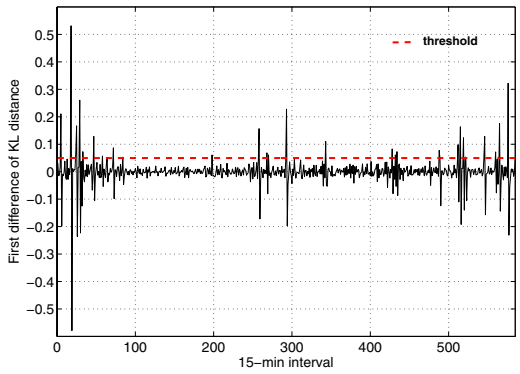

Figure 3: Time series of KL distance first difference for the source IP address feature. The dashed line shows the anomaly detection threshold.

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, our detector will generate an alert 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.

We have observed that the first difference of the KL distance time series is normally distributed with zero mean and standard deviation σ. This observation enables to derive a robust estimate 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 3, 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.

2.2 Meta-Data Generation

If we detect an anomaly during interval t we want 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 useful 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 the largest absolute distance $\max_{i\in[0,s]}\, |p_i - q_i|$ 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.

Having identified the set of anomalous histogram bins B_k for each clone, we obtain the corresponding set of feature values V_k. The cardinality of V_k is much larger than the cardinality of B_k and therefore the set of feature values V_k provided by each clone is likely to contain many normal feature values colliding on anomalous bins. In order to minimize false positives, we keep only those feature values that have been identified by all histogram clones $\cap V_k$. Using k clones a non-anomalous feature value has a rather small probability of $(1/m)^k$ to appear in an anomalous bin in all k clones. The candidate flows F_j of detector j are the flows matching the feature values $\cap V_k$. In the technical report [3] we also analyze and evaluate the use of voting techniques as an alternative to taking the intersection $\cap V_k$.

2.3 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 anomalous flows typically have 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. Further, the similarity between flows is represented by a binary function that is zero for different feature values and one for identical feature values. Other similarity functions, *e.g.,* Manhattan distance, are less suited for the specific problem at hand.

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 $e_1 = \{\texttt{srcPort} : 80\}$ refers to a source port number equal to 80, while $e_2 = \{\texttt{dstPort} : 80\}$ 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.,* $X = \{\texttt{dstPort} : 80, \texttt{dstPort} : 135\}$ 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 $X = \{\texttt{dstIP} : 129.132.1.1, \texttt{dstPort} : 80\}$ 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 fre-

l	srcIP	dstIP	srcPort	dstPort	#packets	#bytes	support	event
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 2: 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.

quent 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 2, 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 since they were 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 at least one frequent 3-item-set. In the third iteration, 41 frequent 3-item-sets are 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.

3. EVALUATION

In this section we first describe the traces we used for our experiments and then outline our evaluation results. We evaluated the accuracy of the generated item-sets and the reduction in classification cost.

3.1 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 (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. Our dataset was recorded during August 2007 and spans two continuous weeks. We focus on the more popular TCP flows that originate within AS559 and that do not terminate within AS559, though the same evalua-

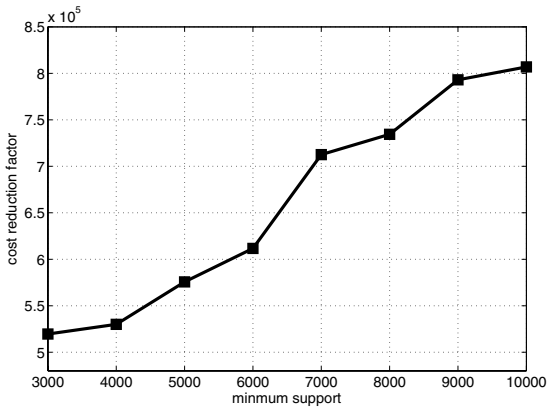

Figure 4: Average decrease in classification cost versus the minimum support parameter. The classification cost reflects the number of flows or item-sets that are manually classified.

tion methodology has been applied [3] to UDP flows and to traffic originating outside AS559.

To generate datasets for evaluating the Apriori algorithm, we fed the two weeks of Netflow data to five histogram-based detectors, *i.e.,* $n = 5$. Each detector monitors the distribution of one of the following features over 15-minute intervals: source IP address, destination IP address, source port number, destination port number, and flow size in packets. For histogram cloning we used five 10-bit hash functions, *i.e.,* $k = 5$ and $m = 1024$. In the companion technical report [3] we investigate thoroughly the impact of different parameter settings.

We manually verified the 31 intervals flagged by at least one of the detectors as anomalous. For manual verification, we extracted the flows matching the meta-data provided by our detectors and analyzed the time series and distribution of several flow features, *i.e.,* srcIP, dstIP, srcPort, dstPort, #packets, #bytes, interarrival, and flow duration. In all the 31 flagged intervals we found at least one anomalous event. The identified anomalies were then classified according to the criteria given in [13]. Our trace includes ten scans originating from SWITCH-internal hosts, nine distributed DoS attacks, five scan replies, four unusual network experiments, two DoS attacks, and one flash crowd event. For each anomalous interval we computed the set of candidate anomalous flows $\cup F_j$ on which we run our modified Apriori algorithm to find frequent item-sets. Finally, we manually analyzed the found frequent item-sets and identified true positives, which matched the identified anomalous events, and false positives, which matched benign traffic.

3.2 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. If we find the item-sets that match the anomalous flows, the anomaly extraction problem is solved.

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 r in classification cost of an anomaly 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 Figure 4 the average reduction in classification cost over all anomalous intervals versus the minimum support parameter. The maximum value that we allow is 10,000 so that in all cases the set I contains the anomalous flows. The average cost reduction increases with the minimum support and ranges between $520,000$ and $800,000$. The average 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.

3.3 Accuracy of Apriori

The output of Apriori is a mix of item-sets matching anomalous flows (true positives) and non-anomalous flows (false positives). As long as the minimum support parameter is set to a value that is lower than the support of the anomalous flow set, Apriori will not produce any false negatives. Hence, assuming a rather small minimum support, *e.g.,* at most 10,000 flows, we evaluate the number of false positive item-sets generated by Apriori.

In Figure 5 we plot the number of false positive (FP) item-sets generated by Apriori versus the minimum support parameter. For 21 anomalies (70%) we obtained no FP item-sets at all. The number of FP item-sets for the remaining 10 anomalies is plotted in the figure together with the average number of FP item-set over all 31 anomalies (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 5 shows that when restricting the input data set to flows that match the meta-data provided by the histogram-based detectors, 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 that have many more FP item-sets than the other anomalies. This is because these anomalies had an entry for single-packet flows in the meta-data. We found that the observed FP item-sets are often either due to common ports (*e.g.,* 80, 443, 25), short flow lengths, or SWITCH-internal hosts with a lot of traffic (*e.g.,* web servers/proxies/caches, mail hubs, or Planetlab nodes). Consequently, most of the FP item-sets can be sorted out rather easily by a network administrator.

3.4 Comparison with Intersection

Generating a filter rule based on the intersection of meta-data provided by different detectors is an alternative approach to identify anomalous flows [19]. In this work, we first use the intersection of the feature values provided by different clones, then take the union of the flow sets provided by different histogram detectors, and finally apply association rule mining to extract and summarize anomalous flows. We compared the use of the intersection as an alternative to the union operator in the second step of our methodology. For four out of the 31 anomalies (13%) the two approaches found the same anomalous item-sets. In two of these cases, the intersection performed slightly better since the union operator generated additional FP item-sets. However, for the majority of the cases, *i.e.,* for 27 out of 31 anomalies

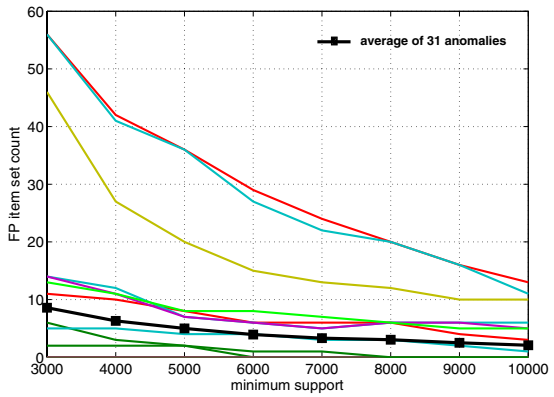

Figure 5: Number of false positive (FP) item-sets generated by Apriori for different minimum support parameter values. For 21 anomalies (70%) we obtain no FP item-sets at all. The FP item-set count for the remaining 10 anomalies is plotted in the Figure together with the average FP item-set count over all 31 anomalies (marked with squares).

(87%), the union operator followed by rule mining had superior results. In 25 cases our approach generated much more specific filter rules than the intersection operator leading to substantially fewer FP item-sets. In two cases, the intersection operator missed the anomaly completely as the meta-data contained FP entries.

4. RELATED WORK

Substantial work has focused on dimensionality reduction for anomaly detection in backbone networks [2, 20, 22, 14, 9, 4, 11]. 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.* [7] 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 intersection of the addresses hashing into anomalous sub-traces. DoWitcher [19] 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. These two papers look into the anomaly extraction problem as part of building a multi-purpose system. Our work focuses exclusively on anomaly extraction, it provides a more general approach using association rules, and has the core technical difference of working on the union set of flows (shown in Figure 2) instead of the intersection. Li *et al.* [15] also 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 [14]. In addition, the authors argue that their method can be used for anomaly extraction, however, the paper 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 item-sets that summarizes a large set of flow records. Mahoney and Chan [16] 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 [17]. Their approach targets edge networks where mining rare events is possible. In massive backbone data, however, this approach is less promising. Another application of association rule mining in edge networks is eXpose [10], which learns fine-grained communication rules by exploiting the temporal correlation between flows within very short time windows.

Hierarchical heavy-hitter detection methods [8, 23, 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.

5. 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 and detection system testing. We have presented a histogram-based detector that provides fine-grained meta-data for filtering suspect flows. Further, we have introduced a method for extracting anomalous flows that uses association rules to describe flows that have similar characteristics across several features.

We used a rich Netflow dataset captured in a backbone network to validate the proposed techniques. Our evaluation results show that the classification cost, in terms of items that need to be classified, can be reduced by several orders of magnitude using association rules. Further, we evaluated the accuracy of Apriori for several verified anomalies and found that it generates on average between 2 and 8.5 false positive item-sets, which can be trivially identified by an administrator. Further analysis and evaluation results can be found in the companion technical report [3].

6. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Program (FP7/2007-2013) under grant agreement no. 216585 (INTERSECTION Project) and no. 223936 (ECODE Project).

7. 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*, pages 487–499. Morgan Kaufmann, 1994.

[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*, pages 71–82, New York, NY, USA, 2002. ACM Press.

[3] D. Brauckhoff, X. Dimitropoulos, A. Wagner, and K. Salamatian. Anomaly extraction in backbone networks using association rules. TIK-Report 309, ETH Zurich, September 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.*, 12:355–378, 2007.

[6] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Finding hierarchical heavy hitters in streaming data. *ACM Trans. Knowl. Discov. Data*, 1(4):1–48, 2008.

[7] 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*, pages 145–152, New York, NY, USA, 2007. ACM Press.

[8] 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*, pages 137–148, New York, NY, USA, 2003. ACM Press.

[9] 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*, pages 32–32, Berkeley, CA, USA, 2005. USENIX Association.

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

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

[12] 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*, pages 234–247, New York, NY, USA, 2003. ACM Press.

[13] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 219–230, New York, NY, USA, 2004. ACM Press.

[14] 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*, pages 217–228, New York, NY, USA, 2005. ACM.

[15] 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*, pages 147–152, New York, NY, USA, 2006. ACM.

[16] 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*, pages 601–604, Washington, DC, USA, 2003. IEEE Computer Society.

[17] 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.*, 3:262–294, 2000.

[18] K. H. Ramah, K. Salamatian, and F. Kamoun. Scan surveillance in internet networks. In *Networking*, pages 614–625, 2009.

[19] 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*, pages 2541–2545, 2007.

[20] 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*, pages 331–344. USENIX Association, 2005.

[21] 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*, Lecture Notes in Computer Science, pages 212–221. Springer, 2008.

[22] 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*, pages 172–177, Washington, DC, USA, 2005. IEEE Computer Society.

[23] 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*, pages 101–114, New York, NY, USA, 2004. ACM.