Anomaly Extraction in Backbone Networks Using Association Rules
Daniela Brauckhoff, Xenofontas Dimitropoulos, Arno Wagner, Kavé Salamatian

To cite this version:
Daniela Brauckhoff, Xenofontas Dimitropoulos, Arno Wagner, Kavé Salamatian. Anomaly Extraction in Backbone Networks Using Association Rules. Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference, 2009, pp.28-34. <hal-00527139>

HAL Id: hal-00527139
https://hal.archives-ouvertes.fr/hal-00527139
Submitted on 18 Oct 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
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]: Inter-networking

General Terms

Design, Experimentation, Measurement

Keywords

Anomaly extraction, association rules, histogram cloning

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

IMC’09, November 4–6, 2009, Chicago, Illinois, USA.
Copyright 2009 ACM 978-1-60558-770-7/09/11 ...$10.00.

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.
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 those 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.

<table>
<thead>
<tr>
<th>Meta-data</th>
<th>Anomaly detection technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>Maximum-Entropy [9]</td>
</tr>
<tr>
<td></td>
<td>Histogram [11, 21]</td>
</tr>
<tr>
<td>IP range</td>
<td>Defeat [15]</td>
</tr>
<tr>
<td></td>
<td>MR-Gaussian [7]</td>
</tr>
<tr>
<td></td>
<td>DoWitcher [19]</td>
</tr>
<tr>
<td></td>
<td>Histogram [11, 21]</td>
</tr>
<tr>
<td>Port range</td>
<td>Maximum-Entropy [9]</td>
</tr>
<tr>
<td></td>
<td>Histogram [11, 21]</td>
</tr>
<tr>
<td></td>
<td>DoWitcher [19]</td>
</tr>
<tr>
<td>TCP flags</td>
<td>Maximum-Entropy [9]</td>
</tr>
<tr>
<td></td>
<td>Histogram [11, 21]</td>
</tr>
<tr>
<td>Flow size</td>
<td>DoWitcher [19]</td>
</tr>
<tr>
<td>Packet size</td>
<td>Histogram [11, 21]</td>
</tr>
<tr>
<td>Flow duration</td>
<td>Histogram [11, 21]</td>
</tr>
</tbody>
</table>

Table 1: Useful meta-data provided by various anomaly detectors. The listed meta-data can be used to identify suspicious flows.

In this work, we take an alternative approach to identifying 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 \( \bigcup F_j \) 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.

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 \( p \) to a reference distribution \( q \) and is defined as

\[
D(p||q) = \sum_{i=0}^{m} p_i \log \left( \frac{p_i}{q_i} \right).
\]
In the current histogram equal to its value in the previous of flows falling into bin

The set

The standard algorithm for discov-

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 car-

describe items that occur frequently to-

Our motivation for applying association rules to the anomaly extraction problem is that anomalous flows typically have similar characteristics, e.g., IP addresses, port

Association rules describe items that occur frequently to-

2.2 Meta-Data Generation

If we detect an anomaly during interval \( t \) we want to iden-

tify the set \( B_k \) of affected histogram bins and the correspond-

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 asymmet-

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 distrib-

We have observed that the first difference of the KL distance time series is normally distributed with zero mean and standard deviation \( \sigma \). This observation enables to de-

\[ \Delta_t D(p||q) \geq 3 \hat{\sigma}. \]

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 iden-

tify the set \( B_k \) of affected histogram bins and the correspond-

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 \( \max_{i \in [p,q]} |p_i - q_i| \) 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.
quently 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 \textit{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).

\textbf{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.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline
\textbf{l} & \textbf{srcIP} & \textbf{dstIP} & \textbf{srcPort} & \textbf{dstPort} & \textbf{#packets} & \textbf{#bytes} & \textbf{support} & \textbf{event} \\
\hline
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 \\
\hline
\end{tabular}
\caption{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.}
\end{table}

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 evaluation.
anomalous flows each anomalous interval we computed the set of candidate originating from SWITCH-internal hosts, nine distributed event. The identified anomalies were then classified according to the 31 flagged intervals we found at least one anomalous #packets, #bytes, interarrival, and flow duration. In all of several flow features, we extracted the flows matching the meta-data provided by one of the detectors as anomalous. For manual verification, 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.
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 anomaly extraction, it provides a more general approach using a multi-purpose system. Our work focuses exclusively on anomaly extraction, however, the intersection operator missed the anomaly completely as the meta-data contained FP entries.

(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.

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. 210585 (INTERSECTION Project) and no. 223936 (ECODE Project).

7. REFERENCES


