Network Traffic Anomaly Detection and Evaluation

A dissertation submitted to
ETH ZURICH

for the degree of
Doctor of Sciences

presented by
DANIELA BRAUCKHOFF
Dipl. El.-Ing. TU Berlin
born December 23, 1978
citizen of Germany

accepted on the recommendation of
Prof. Dr. Bernhard Plattner, examiner
Prof. Dr. Kave Salamatian, co-examiner
Dr. Xenofontas Dimitropoulos, co-examiner
Dr. Andreas Kind, co-examiner

2010
Abstract

A worldwide Internet usage growth rate of 380% over the period from 2000, the year of the dot-com bubble burst, until present indicates that Internet technology has become a cornerstone of our daily life. In the same period, cyber-crime has seen an incredible professionalization that makes sophisticated protection mechanisms for computers and networks an absolute necessity. Firewalls as the major defense of the last decade do not provide sufficient protection anymore. This fact has given rise to the development of intrusion detection and prevention systems. Traditional intrusion detection systems are reactive in the sense that they use a set of signatures, which grows at the same rate as new vulnerabilities are discovered, to identify malicious traffic patterns. Anomaly detection systems are another branch of intrusion detection systems that act more proactively. They derive a model of the normal system behavior and issue alerts whenever the behavior changes; making a subtle assumption that such changes are frequently caused by malicious or disruptive events. Anomaly detection has been a field of intensive research over the last years as it poses several challenging problems.

In this thesis, we address three of these challenges. When working with large-scale network data from possibly multiple routers, the curse of dimensionality considerably complicates the problem of anomaly detection. Principal component analysis has been proposed to deal with it. However, as subsequent
work has discovered several deficiencies in the proposed PCA-method, there is room for improvement. A second challenge stems from the underlying assumption of anomaly detection mentioned above, which, unfortunately, does not always hold in practice. As a direct consequence of this circumstance, users are often overwhelmed with false alarms. To cope with high false alarm rates, one could either try to reduce the number of false alarms, or one could try to minimize the time that is required for resolving an alarm. This is where we see the largest discrepancy between research and practice, as the false alarm problem is broadly ignored by the scientific community. Finally, when a research field such as anomaly detection has reached a certain degree of maturity a sound evaluation of the proposed methods should be done. The major challenge with regard to evaluation is due to fact that there are practically no labeled real-world datasets available.

Our contributions are the following. In the first part of this thesis, we revisit the PCA-method and its underlying assumptions. We find that the assumption of independence between measurement points is not given as network traffic statistics typically exhibit strong temporal correlation. Therefore, we extend the PCA-method to stochastic processes and include the temporal as well as spatial correlation in the model. With our extended method we achieve an improvement in accuracy of up to 20 percent. In the second part of this thesis, we address the false alarm problem. We introduce a method that uses histogram-based anomaly detectors and association rules to help administrators with the identification of anomalous flows and event root causes. With our approach we are able to reduce the time for alarm resolution from typically one hour to a few minutes. The third part of this thesis describes several realistic anomaly models for simulation that we have derived directly from flow traces. Moreover, we introduce FLAME, a tool for anomaly injection into real-world traces, which has been used by several researchers for assessing the false negative rates of their algorithms.
Kurzfassung


In dieser Arbeit adressieren wir drei spezifische Herausforderungen. Da Modellierungsprobleme oft eine Vielzahl von Dimen-

In unserer Arbeit erarbeiten wir die folgenden Lösungen. Im ersten Teil dieser These, betrachten wir die Haptkomponentenanalyse und die ihr zugrunde liegenden Annahmen erneut. Dabei stellen wir fest, dass die Annahme der Unabhängigkeit zwischen einzelnen Messpunkten nicht erfüllt ist, da Netzwerkstatistiken oft stark zeitlich korreliert sind. Wir erweitern daher die Hauptkomponentenanalyse, um sie auf stochastische Prozesse anwenden zu können, indem wir sowohl die zeitliche als auch die räumliche Korrelation ins Modell einfließen lassen. Mit unserer erweiterten Methode erreichen wir eine Verbesserung der Genauigkeit um bis zu 20 Prozent. Im zweiten Teil dieser Arbeit, adressieren wir das Fehlalarmproblem. Wir entwickeln eine Methode, welche Histogramm-basierte Detektoren und Assoziierungsregeln nutzt, um Administratoren bei der Aufklärung eines Alarms zu
Contents

Contents ix
List of Figures xi
List of Tables xiii

1 Introduction 3
  1.1 Challenges in Anomaly Detection ............... 8
    1.1.1 The Curse of Dimensionality .......... 8
    1.1.2 Detection vs. Identification .......... 9
    1.1.3 Evaluation .................................. 9
  1.2 A Bird’s Eye View ................................. 10
  1.3 Research Problems ................................. 14
  1.4 Contributions .................................... 15
  1.5 Outline of the Thesis ............................. 16

2 Related Work 19
  2.1 Flow-based Anomaly Detection ............... 19
    2.1.1 Principal Component Analysis .......... 20
    2.1.2 Histogram-based Anomaly Detection .... 22
  2.2 Anomaly Extraction ............................... 25
    2.2.1 Meta-Data for Anomaly Extraction ...... 26
## Contents

4.2.4 Flow Pre-filtering .................................. 76
4.2.5 Association Rule Mining ............................... 77
4.2.6 Parameter Estimation ................................. 81

4.3 Evaluation .............................................. 84
4.3.1 Data Set and Ground Truth ............................ 84
4.3.2 Accuracy of Histogram Clones ......................... 86
4.3.3 Impact of Voting .................................... 87
4.3.4 Accuracy of Rule Mining .............................. 90
4.3.5 Computational Overhead of Rule Mining ............... 93
4.3.6 Decrease in Classification Cost ....................... 94

4.4 Summary ............................................... 95

5 Anomaly Modeling, Generation and Injection .............. 97
5.1 Problem Statement ..................................... 98
5.2 Anomaly Characterization ................................. 101
  5.2.1 Network Scans ................................... 102
  5.2.2 Spam ............................................. 112
  5.2.3 Denial of Service Attacks .......................... 115

5.3 Anomaly Models ......................................... 120
  5.3.1 Generation Functions ............................... 121
  5.3.2 Anomaly Rescaling ................................ 123
  5.3.3 Library of Anomaly Models ......................... 124

5.4 FLAME .................................................. 130
  5.4.1 Flow Input and Output .............................. 131
  5.4.2 Flow Generation .................................. 133
  5.4.3 Flow Modification ................................. 135
  5.4.4 Flow Merging ..................................... 137
  5.4.5 Injection Scenario ................................. 137

5.5 Discussion ............................................. 139
  5.5.1 Model Versatility .................................. 139
  5.5.2 Modeling and Injection Artifacts ................. 141
List of Figures

1.1 Illustration of the anomaly detection decision process and possible errors. .................................. 5

1.2 A map of the SWITCH backbone network from May 2009. ..................................................... 6

1.3 A bird’s eye view on the anomaly detection process including pre- and post-processing steps. The highlighted topics are subject to this thesis. ........ 11

2.1 PCA of a bi-variate normal distribution centered at (0,0). The arrows, representing the principal axes, point in the direction of maximum variance. 21

3.1 Time series of the preprocessed, i.e., demeaned and normalized, incoming traffic metrics that are used as input for PCA and KLE. Furthermore, we plot the labels with each timeseries. ................. 53

3.2 Time series of the preprocessed, i.e., demeaned and normalized, outgoing traffic metrics that are used as input for PCA and KLE. Furthermore, we plot the labels with each timeseries. ................. 54
3.3 Timeseries of the decision variable $D[k]$ for $L = 3$ and $M = 1$ in the upper plot and for $L = 3$ and $M = 3$ in the lower plot. Additionally, we have included the labels and a hypothetical threshold in each plot. ............................................ 56

3.4 ROC curves for $L=3$. The curve for $M=1$ corresponds to classical PCA. ................................. 58

3.5 ROC curves for $L=2$. The curve for $M=1$ corresponds to classical PCA. ................................. 60

3.6 ROC curves for $L=4$. The curve for $M=1$ corresponds to classical PCA. ................................. 61

4.1 The high-level goal of anomaly extraction is to filter and summarize the set of anomalous flows that coincide with the flows caused by malicious/disruptive network events such as Denial of Service attacks or scans. ................................. 67

4.2 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 $F_A$. ................................. 68

4.3 Overview of our approach to the anomaly extraction problem. The upper figure illustrates how meta-data for a single traffic feature $j$ is generated by voting from $k$ histogram clones. The lower figure illustrates how the meta-data for filtering flows is consolidated from $n$ traffic features by taking the union, and how suspicious flows are pre-filtered and anomalous flows are summarized in item-sets by association rule mining. ................................. 70
4.4 Upper plot: KL distance time series for the source IP address feature for roughly two days. Lower plot: First difference of the KL distance for the same period. The dashed line corresponds to the anomaly detection threshold. 73

4.5 This Figure illustrates our incremental method for determining the anomalous bins. The KL distance converges to zero as in each round the bin with the largest absolute difference is aligned with its counterpart in the reference distribution. Already after the first round the KL distance decreases significantly. 74

4.6 ROC curves plotting the false positive rate versus the true positive rate for different thresholds. The three curves correspond to different histogram clones. 87

4.7 Upper bound for the probability $P_{a}$ 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 $k$, $P_{a}$ increases with $l$, e.g., for $l = 5, k = 10$ we obtain $P_{a} = 0.006$ while for $l = 10, k = 10$ the probability increases to $P_{a} = 0.89$. 88

4.8 Probability $P_{n}$ that a normal feature value is not eliminated by voting for different values of $l$ and $k$ in logarithmic scale. The number of anomalous bins is $b = 1$ (upper plot) and $b = 25$ (lower plot) and the total number of bins is $m = 1024$. 89
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.9</td>
<td>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.</td>
<td>92</td>
</tr>
<tr>
<td>4.10</td>
<td>Average decrease in classification cost vs. minimum support</td>
<td>94</td>
</tr>
<tr>
<td>5.1</td>
<td>Scanning behavior and flow inter-arrival times of the Nachi scan</td>
<td>103</td>
</tr>
<tr>
<td>5.2</td>
<td>Scanning behavior and flow inter-arrival times of the SSH scan anomaly</td>
<td>105</td>
</tr>
<tr>
<td>5.3</td>
<td>Histogram of differences between consecutive IP addresses and inter-arrival times for the Radmin scan</td>
<td>107</td>
</tr>
<tr>
<td>5.4</td>
<td>Histogram of destination IP address differences between consecutive flows for the RCP scan anomaly</td>
<td>110</td>
</tr>
<tr>
<td>5.5</td>
<td>Target selection strategy and timing behavior of the Netbios scan</td>
<td>111</td>
</tr>
<tr>
<td>5.6</td>
<td>Target selection strategy of the Popup-Spam-A and Popup-Spam-B anomaly</td>
<td>113</td>
</tr>
<tr>
<td>5.7</td>
<td>Timing behavior of the BWFlood-A anomaly and destination port selection of the BWFlood-B anomaly</td>
<td>117</td>
</tr>
<tr>
<td>5.8</td>
<td>Source port selection for the TCPFlood-A and TCPFlood-B anomaly</td>
<td>119</td>
</tr>
<tr>
<td>5.9</td>
<td>Histogram of destination port differences for the TCP backscatter anomaly</td>
<td>121</td>
</tr>
<tr>
<td>5.10</td>
<td>Illustration of the impact of parameters $p$ and $v$ on the flow rate of an anomaly. We plot the time that is takes to send 1000 flows for different parameter settings</td>
<td>125</td>
</tr>
</tbody>
</table>
List of Figures

5.11 Application of FLAME for testing anomaly detection system. .......................... 130
5.12 Configuration setup example for FLAME: Generation and injection of two anomalies into a third flow stream in IPFIX format. .......................... 138
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Meta-data provided by existing anomaly detection systems.</td>
<td>27</td>
</tr>
<tr>
<td>3.1</td>
<td>List of the classified anomalies.</td>
<td>51</td>
</tr>
<tr>
<td>4.1</td>
<td>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 were anonymized.</td>
<td>79</td>
</tr>
<tr>
<td>4.2</td>
<td>Parameters including description and range as used in the evaluation section of this work.</td>
<td>81</td>
</tr>
<tr>
<td>4.3</td>
<td>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.</td>
<td>85</td>
</tr>
<tr>
<td>5.1</td>
<td>Overview of anomaly models extracted from three weeks of NetFlow traces captured in the SWITCH backbone network in August 2007. We provide the model (constant, random, or periodical) and parameter setting for 5 flow attributes. The remaining flow attributes are given in Table 5.2.</td>
<td>127</td>
</tr>
</tbody>
</table>
5.2 Overview of anomaly models extracted from three weeks of NetFlow traces captured in the SWITCH backbone network in August 2007. We provide the model (constant, random, or periodical) and parameter setting for 5 flow attributes. The remaining attributes are given in Table 5.1.

5.3 List of available keys to be used in generation and modification models.

5.4 Options for specifying additional information (start time and duration) for anomaly injection.

A.1 Mapping between flow information stored in internal FLAME format and NetFlow/IPFIX flow formats.

A.2 NetFlow v9 data template used by FLAME.

A.3 IPFIX data template used by FLAME.
Chapter 1

Introduction

In the past thirty years, the Internet has become a major driving factor for the worldwide communication and economy. Enterprises and governments across the globe, ranging from telecom providers to banks, universities, energy suppliers, and transportation carriers, rely on complex and inter-connected networks for their core operations. As a consequence of this development, more and more companies and institutions provide access to parts of their internal networks and offer services such as e-commerce to customers and partners. Moreover, broadband and flat-rate Internet connections have become the standard in private households over the last years. Thus, the variety of targets for cyber-criminals to choose from has become immense.

Cyber-crime itself has also experienced an increased professionalization over the last years. A black-market where cyber-criminals offer their goods and services has evolved [FSPT09]. For example, cyber-criminals offer networks of compromised hosts (botnets) that are abused for sending mass spam and for launching Denial of Service attacks. Under these conditions it is not a surprise that network-based attacks are on the rise. Financial losses due to cyber attacks are increasing likewise from
year to year [IC3, CSI]. Due to the prevalence of such threats, network security appliances such as intrusion detection systems and firewalls become indispensable.

Intrusion detection comes in two flavors: Deterministic systems that rely on matching received traffic with pre-defined patterns of malicious traffic and statistical systems that derive models of system properties under normal circumstances and compare predictions based on them with actual measurements.

**Deterministic Systems:** Most commercially available intrusion detection systems are signature-based systems that use predefined and frequently updated signature databases for detecting a wide range of known attacks. These signatures typically capture payload patterns or header and protocol fields. Such systems detect known attacks fairly reliably and with low false positive rates. The major drawback of signature-based intrusion detection systems, however, is their inability to detect unknown attacks.

**Statistical Systems:** The capability to detect unknown attacks is the strength of statistical anomaly detection systems. Anomaly detection systems derive a model of the normal behavior of a network or system and detect deviations from this normal profile. This enables them to detect known and unknown malicious activities likewise. The normal profile has been derived based on different statistics such as system calls on a single host, payload byte patterns in received traffic, or volume and entropy statistics over the traffic in a whole network. However, anomaly detection systems face their own problems that remain to be addressed before they can be widely deployed.

As the goal of this thesis is to address some of the open problems that exist with statistical anomaly detection, we elaborate a little more on the internals of anomaly detection systems. The
basic assumption of the anomaly detection paradigm is that malicious or disruptive behavior, which is of inherent interest for network administrators, is a subset of anomalous activity that is detected by anomaly detection systems. Malicious activities are for example compromised hosts that send unwanted traffic (spam, scans), Denial of Service attacks, or spreading Internet worms. Disruptive activities are for example network failures, outages, or routing flaws that have a negative impact on network stability and are thus likewise of interest. However, as shown in Fig. 1.1, malicious/disruptive activity is only a subset of anomalous activity that triggers anomaly detection alerts. Anomaly detection systems also report anomalous but benign activities that are
typically of less interest to network administrators, for example, newly introduced applications that alter the normal behavior profiles of the system. This results in a false positive or false alarm. Likewise, malicious/disruptive behavior might not be detected by the system, resulting in a false negative or misdetection. There is an inherent trade-off between the false alarm rate and the misdetection rate of each system that can be tuned by increasing or decreasing the sensitivity of the system. Designers and users of anomaly detection systems must be aware of this fact when developing and operating anomaly detection systems.

The datasets that we consider in this work are captured in the SWITCH backbone network and are thus very large in terms of data volume and multidimensional in terms of features that can be measured and evaluated. The SWITCH Internet back-
bone (AS559) connects all Swiss universities, various research labs, e.g., IBM, PSI, CERN, and other educational institutions to the Internet. A map of the SWITCH network is depicted in Fig. 1.2. The network traffic crossing a border router of the SWITCH network on an average working day in 2009 amounts to roughly 120 million flows and 600 GByte traffic per hour. In order to reduce the volume of data, we do not collect packet traces but store flow traces in the NetFlow v5 and recently v9 format [Cisb] that is exported by the Cisco [Cisa] border routers of SWITCH. Recently a standardized flow format called IPFIX has been developed by IETF based on Cisco’s Netflow v9 format [QBC+08]. A flow is essentially a unidirectional sequence of packets between two end points that is characterized by the 5-tuple of source and destination IP address, source and destination port number, and transport protocol (e.g., UDP or TCP).

Depending on the specific data format used, flow records contain additional information such as the number of packets and bytes transferred over the connection or the start time and duration of the connection.

Storage, processing, and visualization of such large datasets has become a challenge on its own. To cope with the increasing data amounts and the involved complexity different data reduction techniques have been developed. First of all random packet sampling, where each packet is sampled with a probability less than one, is applied to relieve routers from the additional burden of collecting measurement data. Secondly, time averaging is used for computing summary statistics (e.g., volume or entropy) on the network traffic to further reduce the information to be stored and processed. Thirdly, dimension reduction techniques such as Principal Component Analysis (PCA) are applied to deal with the still high dimensional summary statistics.
1.1 Challenges in Anomaly Detection

Anomaly detection systems face various problems that need to be addressed by researchers before they can be widely deployed. These rather hard problems include high false positive rates, the lack of representative training data, the difficulty of determining the specific event that has triggered an alert, which is also referred to as root-cause identification, as well as the calibration problem, \textit{i.e.}, determining the point in time when an anomaly detection system needs to be retrained. In the following we discuss the problems closely related to this thesis in more detail.

1.1.1 The Curse of Dimensionality

Curse of dimensionality as a term was coined by Richard Bellman in 1961. It refers to an exponential growth of the hypervolume as a function of dimensionality. Unsupervised learning algorithms as frequently used for anomaly detection on high-dimensional datasets are typically prone to this problem. See [Don00] for an extensive discussion on the curses and blessings of dimensionality in data analysis. Informally speaking, the higher the dimensionality of the input space, the more data may be needed by the algorithm and the more difficult it may become to distinguish the important dimensions from the noise. A commonly-applied and powerful solution to this problem is to preprocess the input data to find a lower-dimensional approximation that preserves the important properties of the input data. Dimension reduction techniques have been introduced to serve exactly this purpose. The most commonly used technique to analyze high-dimensional datasets is Principal Component Analysis (PCA). The challenge with PCA when used as a modeling method in anomaly detection is its reported sensitivity to certain parameter settings.
1.1 Challenges in Anomaly Detection

1.1.2 Detection vs. Identification

Upon detection of anomalous behavior, anomaly detection systems trigger an alert. This alert typically provides meta-data (e.g., the time interval or IP address range) that is associated with the detected anomaly. This information is then used by a network administrator to diagnose the root-cause that has triggered the anomaly detection alert. Root-cause identification is a manual process that on average takes up to one hour per alert according to the 2006 Gartner report [Pro06]. The difficulty of identifying the root-cause associated with an anomaly detection alert is one of central problems in anomaly detection. Knowing the root-cause is the basic requirement for deciding whether the alert is a false positive, or if not, to decide which countermeasures should be applied for handling the alert. Unfortunately, the problem of root-cause identification has not received sufficient attention from the research community so far.

1.1.3 Evaluation

Another challenge in the anomaly detection domain is the unavailability of labeled datasets for evaluation and training of anomaly detection systems. Since network traffic data is very privacy-sensitive, especially when it contains payloads or IP addresses, strict laws have been issued in many countries that prohibit the public sharing of network data. To circumvent these privacy issues, network traffic traces are typically made available to a small and closed group of researchers under the premise of non-disclosure agreements. This practice makes it extremely difficult to compare the performance of anomaly detection systems as each system is tested under different conditions. Moreover, the available datasets are usually not labeled, i.e., the instances of malicious or disruptive activity in the trace are not known beforehand and need to be added by a human expert. Unfortunately, there is no standard methodology or best practices available for
manual labeling. Hence, we have to believe in the ability and unbiasedness of the human expert to assign the correct labels.

### 1.2 A Bird’s Eye View

The larger context of this work is illustrated in Figure 1.3. Anomaly detection is surrounded by several pre- and postprocessing steps. First of all the metrics used for detection have to be extracted from the raw input data, then upon an alert the anomalous events have to be identified, and finally the gained information can be used for remediation, labeling, or for building anomaly models. Remediation of malicious/disruptive events is important in an on-line setting where the anomaly detection system is used to protect a network. Labeling is the act of assigning labels (e.g., DoS attack from 9.00PM to 9.15PM) to a data trace that can then be used for evaluating other anomaly detection systems. Anomaly modeling is important for simulation and evaluation.

#### Data Preprocessing

The most important motivation for data preprocessing is dimensionality reduction. Random packet sampling is applied in routers to significantly reduce the amount of information that is to be transferred through the network to the capturing point. Afterwards aggregation methods such as time-averaging are used to further consolidate the captured information. Clearly, this preprocessing step has an impact on the following anomaly detection step. We have studied this impact empirically in [BTW+06, TBM08] and found that metrics capturing traffic distributions (e.g., Shannon entropy) instead of traffic volumes are less disturbed by packet sampling. We have also studied the sampling problem from a more theoretical signal processing point of
1.2 A Bird’s Eye View

Figure 1.3: A bird’s eye view on the anomaly detection process including pre- and post-processing steps. The highlighted topics are subject to this thesis.

view in [BSM10] identifying aliasing as the main source of disruption. Further, we propose in this work to replace sampling and temporal aggregation with a specifically designed low-pass filter and show that this solution considerably improves anomaly detection results.

Another motivation for data preprocessing is to remove privacy-sensitive information such as IP addresses from the input data. However, there is a trade-off between gain in privacy
and reduction in utility when anonymization is applied. We have studied this trade-off for different anonymization techniques empirically in [BBMB08, BBM08].

The final purpose of data preprocessing is to derive meaningful metrics that anomaly detection systems can operate on. In general, there exists three types of metrics that have been used for anomaly detection: volume metrics, distributional metrics, and graph metrics. Our work is mainly concerned with volume metrics such as the flow count per time interval, and distributional metrics such as the Shannon entropy per time interval. Graph metrics such as the in- or out-degree of nodes per time interval are only considered briefly in this work.

**Anomaly Detection**

Statistical anomaly detection methods typically have two steps that we term entropy reduction and anomaly decision. In the entropy reduction step, a prediction derived from a trained model is used to filter the predictable and thus uninteresting part of the signal. The model can be based on a plethora of techniques ranging from auto-regressive processes to clustering mechanisms that distinguish different regimes or classes. In [BBMB08, BBM08] we have used a Kalman filter that in fact uses an autoregressive model to study the impact of sampling on anomaly detection. Another modeling technique that has been widely used due to its dimension reduction properties is Principal Component Analysis (PCA). In [BMS09a] and Chapter 3 of this thesis we considerably improve the previously published PCA method by including the temporal correlation within metrics in addition to the spatial correlation between metrics in the model. In [BDWS09a] and the first part of Chapter 4 we propose an anomaly detection technique that detects traffic changes in feature distributions such as the flow count per IP address over time. As the difference signal used in this work does not show significant trends, the en-
tropy reduction step is not required here. Traffic distributions are interesting for anomaly detection as they provide more information or meta-data such as affected IP addresses or ports to the administrator than volume-based methods that just specify the time interval when an anomaly has been detected.

The subsequent anomaly decision step takes the entropy-reduced metric and a threshold as input and makes a decision whether to output an alert for each time interval. The most common decision technique is to do a simple comparison that results in an alert when the metric exceeds the threshold. Typically the native distribution of the metric is transformed into a Gaussian distribution, so that the $3\sigma$ rule can be used to easily determine the threshold for a targeted error rate. We use this technique both in Chapter 3 and Chapter 4 to do the anomaly decision. If the detection metric distribution is unknown, one has to determine the error rates empirically.

### Anomaly Extraction

Anomaly extraction is an additional postprocessing step that we have introduced in [BDWS09b, BDWS09a]. The second part of Chapter 4 and [BDWS09b, BDWS09a] are devoted to the problem of anomaly extraction. The main goal of anomaly extraction is to decrease the time for manual handling of anomaly detection alerts which is currently 60 minutes per alert according to the Gartner Report 2006 [Pro06].

To do anomaly extraction, we use the meta-data provided by multiple histogram-based anomaly detectors to pre-filter a set of suspicious flows from the complete set of flows observed during the time interval with an alert. Subsequently, we apply association rule mining on the set of suspicious flows in order to extract and summarize the flows that have caused a malicious/disruptive event. The output of our tool is a small set of frequent flows and their common characteristics. We show that this set contains on
average as few as 2.0 to 8.5 false positive flows, which show recurring patterns and can thus be trivially filtered out by a network administrator. With our method at hand we can reduce the time for resolving an anomaly detection alert to a few minutes.

Anomaly Modeling

Anomaly detection and extraction can be applied with different goals in mind. In [BWM08] and Chapter 5 we make use of them for deriving parameterizable statistical models for different types of anomalies. Anomaly models are very useful for anomaly detection system evaluation as they allow us to inject anomalies with pre-defined characteristics such as flow rate or number of involved entities into background traffic traces. This allows us to test the sensitivity of anomaly detection systems and to compare different systems on the same set of anomalies. As part of this thesis we provide a tool for generating and injecting anomalies of various types and sizes.

1.3 Research Problems

In this thesis, we address the following research questions:

- **How to construct normal behavior models for anomaly detection?** Principal Component Analysis (PCA) is the optimal approach for modeling the spatial correlation between multiple random variables. However, as the timeseries data used for anomaly detection typically show a high temporal correlation as well, there might be a better solution than PCA that considers the correlation in both dimensions.

- **How to identify the specific events that have triggered an alert?** An inherent drawback of anomaly detection systems is that they typically provide only very rudimentary
information about the specific events that have triggered the alert. Histogram-based anomaly detection systems for example provide the time interval and a range of suspicious feature values such as IP addresses or ports. Diagnosing the specific event that has caused an anomaly detection alert is, however, absolutely necessary for being able to apply appropriate countermeasures.

- **How to model, generate and inject network traffic events for system testing?** We seek a method for generating synthetic malicious/disruptive network traffic based on parameterizable statistical models. Further, a method for injecting the synthetic traffic into existing background traffic is needed that does not cause injection artifacts.

### 1.4 Contributions

This thesis makes three main contributions to the research in the field of flow-based anomaly detection. Our work was published in peer-reviewed conferences and workshops.

- **Theoretical framework for applying PCA to the anomaly detection problem:** We propose a theoretical framework for applying Principal Component Analysis to the problem of anomaly detection [BMS09a]. In particular, we extend the standard PCA approach that applies to random variables to stochastic processes that are typically used for anomaly detection by applying the more general Karhunen-Loève transform. We illustrate the performance gain achieved with our solution on the basis of three weeks of NetFlow data from the SWITCH network.

- **Anomaly extraction using histogram-based detectors and rule mining:** We formulate the anomaly extraction problem and provide a solution that applies multiple
histogram-based detectors and association rule mining for identifying the set of flows that have caused an anomaly detection alert. We evaluate our approach on two continuous weeks of traffic data from the SWITCH network [BMS09b].

- **Tool for generation and injection of network traffic anomalies**: We develop a tool for injecting malicious network traffic flows, *e.g.*, Denial of Service attacks or scanning activity, into an existing background traffic dataset [BSM08]. The tool comes with a variety of anomaly models. Each anomaly is described by its characteristic traffic feature distributions. We further provide a method for scaling the traffic to be generated, for example, in order to modify the intensity of an anomaly.

### 1.5 Outline of the Thesis

This thesis is structured as follows:

**Chapter 2** reviews related work and highlights the novel aspects introduced in this thesis. Relevant research areas include flow-based anomaly detection systems, in particular PCA-based and histogram-based anomaly detectors, association rules and other change detection methods, as well as evaluation concepts and methods.

**Chapter 3** presents the Karhunen-Loeve theorem as an extension of classical PCA to stochastic processes such as time-series of monitored network behavior. We introduce a practical method to apply the Karhunen-Loeve theorem in an anomaly detection context. We illustrate the improvements in terms of detection accuracy that our approach achieves over classical PCA.

**Chapter 4** describes our approach for anomaly extraction. We discuss the developed histogram-based detector that applies
1.5 Outline of the Thesis

histogram cloning and flexible voting strategies to optimize the detection performance. Moreover, we introduce our approach for extracting anomalous flow sets based on metadata provided by the histogram-based detector that applies association rule mining concepts. An extensive evaluation of our approach is given.

Chapter 5 focuses on modeling network traffic related to security-relevant events at the flow-level. A method for scaling the characteristics of anomalous traffic is provided. We further present the FLAME tool that supports the generation and injection of synthetic malicious/disruptive events into existing background traffic.

Chapter 6 summarizes the contributions of this thesis and provides an outlook to open problems in the area that might be worth investigating.
Chapter 2

Related Work

This chapter reviews related work in the area of flow-based anomaly detection. We start by giving a general overview of the developments in this broad field. We provide more details on two specific anomaly detection techniques that are closely related to our work: detectors that rely on Principal Component Analysis for dimension reduction and histogram-based anomaly detectors. Next, we present related work that deals with anomaly extraction and root-cause identification. Finally, we survey existing datasets for testing anomaly detection systems.

2.1 Flow-based Anomaly Detection

A great deal of research effort has gone into developing anomaly detection systems since the anomaly detection paradigm was first proposed by Dorothy Denning in 1987 [Den87]. In several survey papers, e.g., [ETGTVD04, PP07, CBK09], the various proposed methods have been summarized. [PP07] distinguishes anomaly detection approaches using statistical or signal processing techniques (e.g., wavelets [BKPR02], Kalman filter [SST05]), concepts from the machine learning domain (e.g., PCA [LCD04c],
Bayesian networks [SHM02]), or methods applied for data mining tasks (e.g., clustering [XZB05]). An interesting and provocative discussion of the anomaly detection paradigm is presented in [GT06]. The paper discusses several underlying assumptions of the anomaly detection paradigm and questions whether these assumptions hold in practice.

2.1.1 Principal Component Analysis

Principal Component Analysis (PCA) is a well-known and often used dimension reduction technique in machine learning [JM79, MvDV92]. PCA in mathematical terms is a technique that transforms a set of correlated random variables to a new coordinate system that is given by the principal axes or components. In this new system the first principal axis points in the direction of the maximum variance, the second axis points in the direction of the remaining maximum variance, and so on (see Figure 2.1 for an example). The idea behind PCA is that in many datasets the first principal components contribute most to the variance in the original dataset, and the remaining components can be disregarded with minimal losses.

PCA in the context of anomaly detection became popular with the seminal papers of Lakhina et al. [LCD04c, LCD04a]. The authors develop a method based on PCA (called the subspace method) to detect network-wide volume anomalies such as Denial of Service attacks, port scanning, or network outages in Origin-Destination (OD) flow timeseries. An OD flow comprises all aggregated traffic that enters the network at a particular ingress router (origin) and leaves the network at another egress router (destination). PCA is applied to separate the OD flows into normal and anomalous components. The top-\(k\) components that capture the dominant temporal patterns of OD flows are considered as normal, the remaining components are considered anomalous. The normal components are used afterwards
for prediction and anomaly detection. The subspace method was evaluated on four weeks of NetFlow data from the Abilene backbone network. The top-$k$ parameter has been set to $k=4$ in all tests.

In [LCD05] the approach is extended to feature distributions. Shannon entropy is used as a summarization of source and destination IP address and port number distributions. The basic subspace method is extended to handle the four multivariate (multiple OD flows) entropy timeseries simultaneously. The multi-way subspace method is evaluated on three weeks of NetFlow data from the Abilene and Geant backbone networks.

It has been shown by Ringberg et al. [RSRD07] that the subspace method is very sensitive to its parameter settings, in particular the number of principal components $k$ that are included in the normal behavior model. They show that the false positive rate is very sensitive to small changes in the number of principal components $k$. Moreover, they find that the effectiveness of PCA
is sensitive to the level of aggregation used (ingress routers, OD flows, or input links), and that large anomalies might pollute the normal subspace.

In this thesis, we revisit PCA-based approaches for anomaly detection from a signal processing point of view. We show what kind of problems arise when PCA is not carefully applied in the anomaly detection context, i.e., when the data is not appropriately preprocessed and temporal correlation is ignored, we provide a profound theoretical explanation for the encountered problem, and improve the subspace method.

2.1.2 Histogram-based Anomaly Detection

Histogram-based anomaly detectors use information derived from traffic feature distributions, e.g., distribution summary statistics such as entropy, or individual bin counts, for anomaly detection purposes. Several detection methods using traffic feature histograms have been introduced in the last years.

Gu et al. [GMT05] divide packets into different classes according to their transport protocol information and destination port numbers. A baseline distribution of these classes is learned using Maximum Entropy estimation. This learning phase requires pre-labeled training data. In the detection phase the relative entropy of observed packet classes with respect to the baseline distributions is computed. An anomaly alert is generated for packet classes that contribute significantly to the relative entropy. The packet class information provides the transport protocol and destination port numbers related to the alert. The approach is evaluated on seven hours (one hour per day) of packet data captured at the UMASS Internet uplink. The data is labeled manually and the first hour is used for training the system.

Krishnamurthy et al. [KSZC03] were the first to explore efficient data stream computation methods developed in the database research community for anomaly detection purposes.
They introduce the k-ary sketch, which contains multiple random projections of a distribution and uses a small and constant amount of memory. The authors show that they can reconstruct lists of top flows accurately using k-ary sketches instead of per-flow information. Their approach consists of three parts: The sketch module that updates the sketch data structure for every received flow. The forecasting module that applies different univariate timeseries forecasting techniques such as Moving Average, Holt-Winters and ARIMA models for producing a forecast sketch for the next interval. And finally the change detection module that uses the forecast error sketch to generate an alert if the error, i.e., the difference between the measured and the forecast sketch, exceeds a preset threshold. They evaluate their approach on four hours of flow data from ten different routers of a tier-1 ISP. The evaluation shows that the accuracy of k-ary sketches is comparable to systems that keep per-flow state.

Li et al. [LBC+06] combine k-ary sketches with the PCA-subspace method introduced by Lakhina et al. [LCD05] in a tool called Defeat. In a first step, Defeat computes local sketches for source and destination IP addresses (first 21 bits) at each participating router. In the second step, global sketches are computed by taking the sum of the local sketches, and the entropy of each global random projection is computed giving a matrix with one line per time interval and one column per feature. These entropy matrices (one per hash function) are used as input for the subspace method. The subspace method generates alerts for time intervals and hash functions. To increase the confidence in alerts only time intervals that have been flagged by several hash functions are considered. To identify the corresponding IP flows the intersection of key sets over all hash functions is used. The paper contains a limited performance evaluation with one week of data from the Abilene and Geant networks.

Dewaele et al. [DFB+07] use random projection techniques (sketches) and multiresolution non-Gaussian marginal distribu-
tion modeling to extract anomalies at different aggregation levels. The involved methodology uses aggregated timeseries of hashed traffic as input for the modeling step. Non-Gaussian models of marginal laws are derived at different timescales, and the estimated Gamma law parameters are compared to average reference behaviors and typical variabilities using the Mahalanobis distance. An alert is generated if the Mahalanobis distance for a timeseries exceeds a pre-defined threshold. The attributes corresponding to the timeseries with an alert are identified by keeping a map of attributes and hash values, and taking the intersection of attributes corresponding to each hash function. The method is validated on a set of 15 minute-per-day packet traces from the MAWI trace repository that were collected on a trans-Pacific link from 2001 to 2006.

Researchers at the IBM Zurich Research Lab have developed a histogram-based detector that specifically targets enterprise networks. In [SBK08] the authors introduce a two-layered feature-based anomaly detector that constructs fine-grained models of feature distributions for different times of the day from given training data. In [KSD09] they extend their previous detection approach. A dimension reduction is achieved by applying PCA, then histogram clustering is used to obtain prevalent model classes, and in a last step anomalous models are filtered out by removing clusters that have only few members. For anomaly detection, observed histograms are processed likewise (PCA and clustering). An alert is generated if an observation does not fall within any of the baseline clusters. The normalized Euclidean distance of the observation to the closest cluster boundary is reported to the administrator as a measure of alert severity. They evaluate their approach on three data sets, a 9-day trace from the research lab border router, an 11-day trace from a data center, and the DARPA intrusion detection benchmark trace.

Ramah et al. [RSK09] have proposed a histogram-based detector targeted at scan detection. Their detection approach con-
sists in aggregating scan traffic collected over a fixed number of packets, computing the joint distribution of features of interest, and using these scan profiles for online detection afterwards. They analyze two metrics that have been previously used for change detection in histogram timeseries, namely entropy and relative entropy (or Kullback-Leibler distance), and find that entropy is much less suited for this purpose than relative entropy. They evaluate their approach on a manually labeled 2-day trace from the Tunisian university network as well as on synthetic traces generated with the nmap scanning tool.

In this thesis, we propose a histogram-based anomaly detector that re-uses established concepts from previous work. In particular, like [KSZC03] we do not train models of normal behavior from previous intervals. Instead, we regard a distribution as normal as long as it does not change significantly. Further, like [GMT05, RSK09] we use the Kullback-Leibler distance as distribution summary statistic. As a minor contribution, we introduce a random projection method called histogram cloning for obtaining additional traffic views and for improving the detection quality. We validate our approach on three weeks of NetFlow data from the SWITCH network [SWI].

2.2 Anomaly Extraction

The anomaly extraction problem has not received sufficient attention from the research community so far as most previous approaches concentrate on the anomaly detection part. Nevertheless, we discuss some previous applications of association rule mining and related heavy-hitter detection methods that use concepts similar to our approach. Moreover, we present some insights and related work on the problem of meta-data extraction.

Complementary to our two-step anomaly extraction approach that first detects a disruption in a summary statistic and then
tries to identify the flows that have caused the disruption, are
methods that perform anomaly or change detection directly at
the level of individual traffic features. Cormode et al. [CM05]
present a method for finding significant differences in network
data streams between interfaces and routers and refer to them as
deltoids. Their algorithms use combinatorial group testing to find
deltoids very efficiently in space and time. Similar in spirit Xu et
al. [XZB05] and Wei et al. [WMK06] apply clustering techniques
for building behavior profiles of end-hosts and services.

2.2.1 Meta-Data for Anomaly Extraction

The goal of anomaly extraction is to identify and summarize
anomalous flows based on meta-data provided by anomaly detec-
tors. Anomaly detectors based on volume timeseries provide only
the point in time when an anomaly was detected as meta-data.
Histogram-based detectors may provide additional meta-data for
root-cause identification. The meta-data generated by various
existing approaches is summarized in Tab. 2.1.

However, the generation of meta-data from a histogram-based
detector is not straightforward for two reasons: First of all the
histogram bins that have caused a distribution change that has
manifested itself in a summary statistic such as entropy or relative
entropy need to be identified \(^1\). A second problem stems from the
fact that histograms typically aggregate multiple feature values in
a single bin to reduce the complexity. Aggregation is either done
randomly by applying a hash function [LBC\(^+\)06,DFB\(^+\)07,RSK09]
or by merging adjacent feature values [GMT05,SBK08]. Ran-
dom aggregation has the advantage that it allows for identify-
ing the responsible feature value(s) with a certain probability
when multiple randomized versions of a histogram are compared.

\(^1\)This is however not an issue for the set of more complex approaches that
keep a model for the entire distribution [DFB\(^+\)07,SBK08].


<table>
<thead>
<tr>
<th>Meta-data</th>
<th>AD methods in literature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>Maximum-Entropy [GMT05]</td>
</tr>
<tr>
<td></td>
<td>Histogram [SBK08, KSD09]</td>
</tr>
<tr>
<td>IP range</td>
<td>Defeat [LBC+06]</td>
</tr>
<tr>
<td></td>
<td>MR-Gaussian [DFB+07]</td>
</tr>
<tr>
<td></td>
<td>DoWitcher [RSN+07]</td>
</tr>
<tr>
<td></td>
<td>Histogram [SBK08, KSD09]</td>
</tr>
<tr>
<td>Port range</td>
<td>Maximum-Entropy [GMT05]</td>
</tr>
<tr>
<td></td>
<td>Histogram [SBK08, KSD09]</td>
</tr>
<tr>
<td></td>
<td>DoWitcher [RSN+07]</td>
</tr>
<tr>
<td>TCP flags</td>
<td>Maximum-Entropy [GMT05]</td>
</tr>
<tr>
<td></td>
<td>Histogram [SBK08, KSD09]</td>
</tr>
<tr>
<td>Flow size</td>
<td>DoWitcher [RSN+07]</td>
</tr>
<tr>
<td>Packet size</td>
<td>Histogram [SBK08, KSD09]</td>
</tr>
<tr>
<td>Flow duration</td>
<td>Histogram [SBK08, KSD09]</td>
</tr>
</tbody>
</table>

Table 2.1: Meta-data provided by existing anomaly detection systems.

Together with different benign feature values in each randomized histogram. Existing approaches [LBC+06, DFB+07, RSN+07] typically include only flows matching the intersection of feature values, i.e., flows matching all feature values, that have been flagged by several randomized histograms in the meta-data and disregard any feature values that have not been flagged in all histograms.

In this thesis, we propose a method for identifying histogram bins that have caused a distribution change that differs slightly from the one proposed in [RSN+07]. From now on we simply call these histogram bins anomalous bins. The main advantage of our method is that the number of anomalous bins is determined in an automated fashion and must not be known beforehand. Moreover, we theoretically analyze different voting strategies for determining the feature values to be included in the meta-data.
ulation results show that the previously used intersection might not provide the best extraction accuracy trade-off.

### 2.2.2 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 done [AS94]. Association rules have been successfully applied to different problems in networking. We summarize in the following the publications that are most similar in spirit to our approach.

Chandola and Kumar [CK07] study the summarization of a set of transactions with categorical attributes as an optimization problem that trades-off compaction gain (amount of reduction) for information loss (amount of information missing in the summary). They propose and evaluate techniques based on clustering and association rule mining for summarizing flow sets. The authors propose a heuristic-based bottom-up algorithm that can be used to generate an approximately optimal summary for a given set of transactions. Their approach is evaluated on the 1998 DARPA intrusion detection data set and on the SKAION Corp. dataset.

Mahoney and Chan [MC03] introduce the LERAD algorithm that uses association rules to mine rare events that are suspected to represent anomalous application behavior. The algorithm first learns conditional rules that describe normal application protocol behavior by feeding it with an attack-free packet data sequence. An alert is generated by LERAD if an attribute that does not match any of the trained rules is detected. They evaluate their method on the 1999 DARPA/Lincoln Laboratory traces [McH00].

Another application of rule mining in edge networks is eXpose [KCK08]. The authors present a technique that learns network communication rules from packet timing information. Com-
2.2 Anomaly Extraction

Communication rules are predicates that indicate flows that appear frequently in the same time window (of one second) in a network. They use generic templates to reduce the number of rules generated. Finally, the discovered rules are aggregated into a small number of clusters that an administrator can easily overlook. They evaluate eXpose in three different network settings, a lab environment, an enterprise network, and wireless hot-spots at conferences.

In this thesis, we use association rule mining, or more precisely frequent item-set identification, to mine anomalous flow sets in a set of candidate anomalous flows that have been filtered based on meta-data provided by histogram-based anomaly detectors. Hence, our approach is more efficient and scalable than [MC03, KCK08] and can be applied on larger datasets, for example, from backbone networks. In contrast to previous work, our application of association rules is motivated by the observation that in general anomalous flows have common traffic feature characteristics, while normal flows that have been accidentally included in the set of candidate anomalous flows show larger variability. In contrast to [CK07], we do not try to compute an optimal summary of all the flows.

2.2.3 Hierarchical Heavy-Hitter Detection

Hierarchical heavy-hitters represent traffic streams, which are responsible for a large fraction of the traffic. Detection of heavy-hitter flows is valuable in the context of network management. It is also related to the anomaly extraction problem as heavy-hitter flows can be caused by network anomalies.

The problem of identifying multidimensional heavy-hitters, i.e., flows that cause a significant percentage of traffic in a network, was first introduced by Estan et al. [ESV03]. Their approach consists in first computing high volume unidimensional clusters at different hierarchical levels (e.g., subnets). In a second
step they compress the unidimensional clusters by removing redundant clusters at different hierarchical levels. Finally, the compressed unidimensional clusters (trees) are combined into multidimensional clusters forming a graph, and the resulting graph is again compressed. Moreover, they provide delta reports that contain only clusters with significant changes, and compute an unexpectedness score for multidimensional clusters. The authors evaluate their AutoFocus tool on three private traces, a 31-day trace from a small network exchange point, a 39-day trace captured at the edge of a research institution, and an 8-hour trace from a backbone network.

Cormode et al. [CKMS03, CKMS04, CKMS08] present an online streaming algorithm to identify approximate one- and multidimensional heavy-hitters in one pass over the data and provide accuracy guarantees. They exploit the fact that the product of hierarchical dimensions forms a lattice structure to design efficient algorithms. They show that the proposed online algorithms perform as well as offline processing. The evaluation is done on synthetic and real flow and packet data.

Another seminal work on hierarchical heavy-hitter detection is [ZSS+04]. The authors develop algorithms for online identification of 1-dimensional and 2-dimensional hierarchical heavy hitters. The usage of an adaptive data structure that captures the estimated hierarchical aggregates of traffic activity allows for low worst-case update costs and thus very efficient algorithms. For the 1-dimensional case they use a trie data structure and dynamically increase the monitoring depth to ancestor nodes when the parent node becomes sufficiently large. Further, they propose to use heavy-hitter detection as a pre-filtering mechanism for anomaly detection. The approach is evaluated on three traces (3-min, 1-day, 1-month) collected at a Tier-1 ISP.

In this thesis, we apply algorithms and concepts similar to hierarchical heavy-hitter detection to a set of candidate anomalous flows that have been pre-filtered based on meta-data provided
by histogram-based anomaly detectors. However, our focus is different in the sense that we are particularly interested in highlighting malicious/disruptive traffic patterns. Nevertheless, aggregation at various hierarchical levels could be used to extend our approach.

2.3 Evaluation Datasets and Metrics

A major challenge in the anomaly detection domain poses the lack of evaluation data and standard procedures. In the following we give an overview of the various datasets and metrics that have been used and proposed for evaluating anomaly detection systems.

2.3.1 Captured Datasets

Captured datasets are typically very privacy-sensitive, especially when they contain packet payloads or IP addresses that identify users, and are thus subject to privacy laws in many countries. Although there is a large body of literature on anonymization techniques [SLL06, XFAM02, FXAM04] only few data owners make use of them and share their data within the community. This is due to the fact that even anonymized data leaks information that is valuable for attackers or competitors as shown by [PAPL06, CWM+07, RCMT08]. Moreover, the impact of anonymization on anomaly detection has been observed but not qualified by Soule et al. when they compare detection results from two adjacent networks that use different anonymization techniques [SRS+07]. In [BBMB08, BBM08] we have studied the inherent risk-utility trade-off for anomaly detection on anonymized data. We have shown that the utility decreases with anonymization strength since information important for anomaly detection is removed by the anonymization operation. Hence,
there are only few anonymized data sets shared within the community due to the limitations of anonymization techniques.

One notable data source are the packet traces from the WIDE backbone network [WID] that are maintained by the MAWI working group [MAW]. The repository includes a variety of anonymized traces from 1999 to 2009 from six different measurement points. The MAWI packet traces have been used for several studies, for example in [DFB+07, BDF+08]. Abilene [Abi] and Geant [GEA] are the other two backbone networks that provide researchers with anonymized NetFlow traces upon request. Traces from these networks have been used for evaluation in [UQLB06, LCD04c, LBC+06].

Another problem with captured datasets is that most of them do not contain the complete network traffic. To relieve routers from the additional burden of packet capturing, random packet sampling with low sampling rates (e.g., 1% or less) is used. Fortunately, the SWITCH border routers do not apply packet sampling and we are able to capture the complete network traffic at the flow level. We have studied the impact of packet sampling on the detection of several worms in [BTW+06, TBM08] empirically, and found that detection results are biased considerably by packet sampling. The authors of [MCS+06, SRS+07] report similar findings. Moreover, we have analyzed the impact of packet sampling from a signal processing point of view in [BSM10]. In this work, we also show that anomaly detection results can be significantly improved when aliasing is avoided by filtering the high-frequency part of the signal using a multistage low-pass filter.

By far the largest problem with captured traces is that they are typically unlabeled. Labels that identify when an anomaly has happened are, however, absolutely required for evaluating whether an anomaly detection system is accurate or not. Ringleberg et al. [RSR08] have recognized the problem and proposed a tool called Webclass that provides functionality to store and
2.3 Evaluation Datasets and Metrics

compare labels that have been assigned by different domain experts to a trace. Still trace labeling is a tedious and error-prone but unavoidable manual procedure. Very few researchers have made their labels available to the community. The only labeled data set that we are aware of is [DFB+07]. Gates et al. [GTB07] recently presented requirements for evaluation data and proposed a community-based approach for labeling released traffic traces. Although absolutely necessary, the approach has not been implemented to date.

In this thesis, we use unsampled and non-anonymized traces from the SWITCH backbone network [SWI] for evaluating our algorithms. The traces (three weeks from August 2007 and one week from December 2007) have been labeled manually and the identified labels are provided in this thesis. As the data has been made available to us by SWITCH based on a non-disclosure agreement it cannot be shared with third parties.

2.3.2 Synthetic Datasets

The need for simulation in network research in general, and in evaluating anomaly detection systems has been recognized over the last years [FK03, RRR08]. We present in the following synthetic datasets and simulation tools that contain or produce benign and/or malicious network traffic.

Probably the most widely used synthetic dataset for intrusion detection system evaluation is the DARPA data set from 1998, and its successors from 1999 and 2000 [LFG+00]. While the background traffic in the DARPA datasets is synthetic, attacks where injected into the dataset by launching real attack tools in an isolated test bed. However, the dataset has been heavily criticized by other researchers [McH00, MC99] for having unrealistic distributions of benign/background and attack traffic. Despite these well-known issues, and the fact that the dataset does not contain any recent attacks, it is still used for evaluation today since
better options are rare.

A more recent advance in dataset simulation and emulation is due to Sommers et al. [SB04, SYB04, SYB06]. In [SB04] the authors present Harpoon, a tool for generating representative benign packet traffic at the IP flow level. Unlike previous traffic generators, e.g., SURGE [BC98], Harpoon generates application-independent UDP and TCP traffic. In [SYB04] Sommers et al. present a tool called MACE that generates malicious packet traces for evaluating network intrusion detection systems such as Snort [SNO] and Bro [Pax99]. MACE supports different attacks, e.g., SYN flood, Welchia, Blaster, for which models have been constructed manually from trace analysis and literature. For each attack an exploit model, an obfuscation model, and a propagation model is provided.

In the same spirit, Mutz et al. [MVK03] present a tool called Mucus that automatically generates malicious network traffic based on signatures provided by Snort. Mucus is targeted at and used for testing other intrusion detection systems such as Symantec’s Net Prowler [Net].

Mirkovic et al. [MWH+07] have developed a tool called APProf that extracts the traffic caused by Denial of Service attacks from packet traces based on connection heuristics. The extracted attack traces are used as benchmarks for experimentation in the DETER testbed [BBK+07]. The main focus of their work is the evaluation of DDoS countermeasures in a variety of scenarios characterized by attacks, benign traffic, and topology.

In our work, we focus on modeling, generation, and injection of realistic network-wide anomalies at the flow level. Similar to [MWH+07,SYB04] we extract our models from captured network traces. However, unlike [MWH+07] we do not restrict ourselves to Denial of Service attacks, but provide in addition models for several scan tools and spam. In contrast to [SYB04,MVK03] that target packet-level traffic characteristics such as byte patterns and exploits we concentrate on modeling flow-level characteristics.
2.3 Evaluation Datasets and Metrics

2.3.3 Evaluation Metrics

Finally, we want to shed some light into the evaluation metrics jungle. Meaningful and consistent metrics are required for comparing the detection performance of different systems.

The central premise of anomaly detection is that malicious/disruptive activity is a subset of anomalous activity that is detected and triggers alerts. Malicious/disruptive activity that triggers an alert is called a true positive (TP), whereas malicious/disruptive activity that does not trigger an alert is called a false negative (FN). Activity that is neither malicious nor disruptive is called benign activity. Benign activity that does not trigger an alert is called a true negative (TN), whereas benign activity that triggers an alert is called a false positive (FP). The false negative rate (FNR) measures the percentage of missed malicious/disruptive activities, while the false positive rate (FPR) measures percentage of benign activities that have triggered an alert and thus have been misclassified as anomalous. Two related accuracy measures are precision and recall. Precision is defined as the number of true positives (TP) identified by a detector versus the total number of detected anomalies (TP+FP). Recall, on the other hand, is defined as the number of true positives (TP) identified by a detector versus the total number of malicious/disruptive events in a trace (TP+FN).

The Receiver Operating Characteristic, or ROC curve [Ega75, Swe73], has long been used in other fields such as radiology [HM82]. A ROC curve plots the true positive rate against the false positive rate as the detection threshold is varied. There are two types of classifiers: discrete classifiers, which output only a class label (anomalous or not anomalous), and scoring classifiers, which also emit a score that represents a class membership likelihood. The detection result of a discrete classifier corresponds...
to a single point in the ROC space, whereas scoring classifiers yield a curve in the ROC space where each point is generated by a different threshold setting.

Nevertheless, ROC curves should be used with caution. Fawcett [Faw04] points out that the variance of ROC curves across different datasets needs to be taken into account when one wants to derive conclusions about detector superiority. The unit of measure problem associated with ROC curves was first raised by McHugh [McH00]. Existing detection systems base their decisions on different units of measure (e.g., packets, flows, or often time intervals of different lengths). This makes evaluation results very difficult to compare. A second issue raised by McHugh regards the strategy that is used for matching the detection result with the given labels. He argues that the matching strategy (e.g., whether consecutive detections are counted as a single or multiple events) needs to be chosen with care. Axelson [Axe00] introduced the base-rate fallacy, i.e., benign activity is much more frequent than malicious/disruptive activity, and emphasizes that effective intrusion detection may require false alarm rates way below 0.1% to be effective.

One disadvantage of ROC curves is that they do not consider the notion of costs caused by false positives and false negatives. Stolfo et al. [SFL+00] propose a cost-based modeling approach for fraud and intrusion detection that considers costs due to damage, alert response, and detector operation. Adams and Hand [AH99] introduce a transformation of ROC curves that facilitates comparing detectors by cost. Gaffney and Ulvila [GJU01] compute an expected cost for each detector operation point using a decision tree model.

In this thesis, we use ROC curves for comparison of our improved PCA approach with the classical PCA-subspace method. We use the same unit of measure (15-minute intervals) for both approaches to avoid any bias. Moreover, we consider consecutive detections as a single event. For evaluating the detection ac-
curacy of our histogram-based anomaly detector we apply ROC curves as well.
Chapter 3

Applying PCA for Anomaly Detection

Dimensionality-reduction is an active area of anomaly detection research as the data sets used for anomaly detection are typically very large and multidimensional. Principal component analysis (PCA) is a technique that has been developed to tackle the problem of high dimensional data sets [Pea01]. PCA has been widely used in the domain of image compression, pattern recognition, and network intrusion detection. Recent publications [LCD04c, LCD05] have applied PCA to the anomaly detection problem with promising results. However, subsequent work [RSRD07] has identified several challenges of using PCA for traffic anomaly detection.

In this chapter, we review the classical Principal Component Analysis (PCA) from a signal processing point of view and reveal the problems that arise when PCA is applied to stochastic processes such as network timeseries data. Moreover, we review the Karhunen-Loeve expansion theorem that has been extensively used in model reduction for dynamical systems and show how it applies to the anomaly detection problem. Finally, we intro-
duce and evaluate an improved anomaly detection method based on multidimensional Karhunen-Loeve expansion that considers the spatial and temporal correlation structure in the modeling step. We show that our approach clearly outperforms the classical PCA.

3.1 Principal Component Analysis

A browse in the literature shows two closely related but different interpretations of PCA:

- As an efficient representation that transforms the data to a new coordinate system such that the projection on the first coordinate contains the greatest variance, the projection on second coordinate has the second greatest variance, and so on.

- As a modeling technique using a finite number of terms of an orthogonal serie expansion of the signal with uncorrelated coefficients.

Interestingly, networking literature mainly motivates the application of PCA to traffic anomaly detection by the first interpretation. However, the correct interpretation to use as we will show in the following is the second one. This mistake has resulted in some erroneous practices that have widely spread among the community. We are devoting this work to describe and correct these erroneous practices.

Principal Component Analysis (PCA) has been first proposed as a method for traffic anomaly detection in [LCD04c]. While PCA was used in other domains before, Lakhina et al. made its application very popular in the networking community [LCD04c,LCD04b,LCD05]. Only recently, it has been shown by Ringberg et al. [RSRD07] that PCA as applied [LCD04c] is
very sensitive to its parameter settings. The authors have reported about instability problems encountered when using PCA, however, they did not provide precise reasons for their observation.

3.1.1 PCA Theory

At first some notation: Throughout this chapter, matrices are denoted in upper case boldface letters and vectors are denoted in lower case boldface letters; all vectors are column vectors unless otherwise specified.

Suppose we have $j = 1, \ldots, K$ correlated random variables $(X_1, \ldots, X_K)$ that are observed together at $i = 1, \ldots, N$ observation points. The $N$ observation vectors are arranged in a $N \times K$ matrix $X$ where each row $i$ contains an observation vector $x_i = (x_{1i}, \ldots, x_{Ki})^T$ and each column contains the vector of $N$ observations of random variable $X_j$.

What PCA asks is whether there is another basis, which is a linear combination of the original basis, that best re-expresses a given data set. One basic assumption that PCA makes is that the directions with the largest variances contain the dynamics of interest of the measured system. Hence, the goal of PCA is to minimize redundancy measured by the covariance, and to maximize the signal measured by variance. PCA selects the easiest method for achieving this goal: PCA finds an orthonormal basis $P = (\phi_1, \ldots, \phi_K)$, which is a linear combination of the original basis, where $Y = PX$ such that the covariance matrix of $Y$

$$C_Y = \frac{1}{N-1} YY^T$$

is diagonalized, i.e., the variance along the principal components $\phi_1, \ldots, \phi_K$ is maximized and the off-diagonal components are minimized. Furthermore, the variances associated with each principal component provide a measure for the importance of
each principal component, where the component with the largest variance is most important. This process is illustrated in Fig. 2.1.

As the new basis \( \mathbf{P} \) is orthonormal one can rewrite the initial matrix of random variables \( \mathbf{X} \) in the new coordinate system as

\[
\mathbf{X} = \mathbf{Y} \mathbf{P}^T = \sum_{j=1}^{K} Y_j \phi_j
\]

(3.2)

where \( Y_j \) are jointly independent random variables. The true benefit of the orthonormality assumption, however, is that there exist efficient solutions in linear algebra for computing the principal components based on eigenvectors and singular value decomposition (SVD).

**Using Eigenvectors**

The first solution is based on the computation of eigenvectors. The covariance matrix \( \mathbf{C}_Y \) given in 3.1 can be rewritten as

\[
(N-1)\mathbf{C}_Y = \mathbf{YY}^T = (\mathbf{PX})(\mathbf{PX}^T) = \mathbf{P}(\mathbf{XX}^T)\mathbf{P}^T
\]

(3.3)

A symmetric matrix such as \( \mathbf{XX}^T \) is diagonalized by an orthogonal matrix of its eigenvectors

\[
\mathbf{XX}^T = \mathbf{VDV}^T
\]

(3.4)

where \( \mathbf{D} \) is a diagonal matrix of eigenvalues and \( \mathbf{V} \) is a matrix of the eigenvectors of \( \mathbf{XX}^T \).

The trick is to select the matrix \( \mathbf{P} \) such that each row of \( \mathbf{P} \) contains an eigenvector of \( \mathbf{XX}^T \). By this selection, \( \mathbf{XX}^T = \mathbf{P}^T \mathbf{DP} \) and substituting into Equation 3.3 we can now write

\[
(N-1)\mathbf{C}_Y = \mathbf{P}(\mathbf{P}^T \mathbf{DP})\mathbf{P}^T = (\mathbf{PP}^T)\mathbf{D}(\mathbf{PP}^T) = \mathbf{D}
\]

(3.5)
Hence, it is obvious that the choice of $P$ diagonalizes $C_Y$. In summary, the first solution of PCA is to select the matrix $P$ such that it contains the eigenvectors of $XX^T$ in its rows.

**Using Singular Value Decomposition**

A second more general solution is based on singular value decomposition (SVD). The SVD theorem

$$X = U S V^T$$

(3.6)

states that any matrix $X$ can be converted to an orthogonal matrix $U$, a diagonal matrix $S$, and another orthogonal matrix $V$.

We define the matrix $V$ such that it contains in its columns the eigenvectors of $XX^T$. The matrix $S$ is defined as a diagonal matrix that contains the rank-ordered set of singular values $\sigma_j$ where $\sigma_j = \sqrt{\lambda_j}$ and $\lambda_j$ are the eigenvalues of $XX^T$. And the matrix $U$ is defined such that $XV = US$.

From the last paragraph we know that the principal components of $X$ are the eigenvectors of $C_X$. Consequently, to obtain the eigenvectors of $C_X$, and thus the principal components of $X$, we simply need to compute the SVD of $X$. The principal components will be given by the columns of the orthonormal matrix $V$.

**3.1.2 Problems with Classical PCA**

We have learned that PCA is a powerful and easy-to-use technique for dimension reduction. However, one has to be aware of the fact that PCA is based on some assumptions made with respect to the input data. These assumptions are not so obvious but crucial for a correct operation of PCA. We will discuss these assumptions and their relevance to the particular application of anomaly detection in the following.
Linearity

The assumption of linearity requires that a random variable can be decomposed into a combination of independent linear random variables. If the assumption of linearity is not valid, a non-linear transformation can be applied to the data before PCA. This approach, termed Kernel PCA, is parametric since the user must incorporate prior knowledge of the kernel. However, it is in general quite difficult to infer such prior knowledge about network measurement data that is used for anomaly detection.

Mean and variance sufficiency

PCA requires that the first two moments entirely describe the probability distribution of a random variable. This assumption holds only for the family of exponential distributions. It guarantees that the most suitable basis is the one that maximizes the variance of each projected component. For non-Gaussian distributions PCA can be replaced with Independent Component Analysis (ICA) that could result in a non-orthogonal basis. The involved complexity, however, is much higher than with PCA. On the other hand, it has been shown that PCA is usually tolerant to slight deviations from this assumption.

Zero-mean Random Variables

The most basic but also most subtle assumption is that PCA must be applied to zero-mean random processes. The time-series data used for anomaly detection, however, are samples of stochastic processes that have temporal correlation. Therefore, this last assumption is certainly violated when PCA is applied for anomaly detection purposes. An expansion technique that applies to stochastic processes is the Karhunen-Loeve expansion, which will be described in the next section.
3.2 Extension to Stochastic Processes

The extension of PCA to stochastic processes is mandatory as the timeseries used for anomaly detection are samples of stochastic processes that have temporal correlation. The Karhunen-Loeve transform is the optimal orthogonal transform, equivalently to PCA for random processes, for stochastic processes that results in uncorrelated coefficients in the transform domain. In this section we briefly introduce the Karhunen-Loeve expansion theorem for continuous stochastic processes and its extension to discrete signals.

3.2.1 Karhunen-Loeve Expansion

The Karhunen-Loeve expansion was first introduced in 1946 [Kar46]. Let \( X(t) \) be a zero mean stationary stochastic process with autocorrelation function \( \sigma(s) = \mathbb{E}\{X(t)X(t-s)\} \) defined over an interval \([a,b]\). The Karhunen-Loeve expansion (KLE) theorem states that one can rewrite \( X(t) \) as an orthogonal series expansion

\[
X(t) = \sum_{i=1}^{\infty} Y_i \Phi_i(t) \tag{3.7}
\]

where \( Y_i \) are pairwise independent random variables and \( \Phi_i(t) \) are pairwise orthogonal deterministic functions defined on \([a,b]\), i.e., \( \int_a^b \Phi_i(t)\Phi_m(t)dt = 0 \) for \( i \neq m \) and in addition \( \int_a^b |\Phi_i(t)|^2 dt = 1 \).

The set of deterministic functions \( \Phi_i(t) \) is an orthonormal basis for the linear stochastic process in \( L_2(a,b) \) and the random variables \( Y_i \) are coordinates of the stochastic process \( X(t) \) in this new space. The random variables \( Y_i \) further satisfy

\[
Y_i = \int_a^b X(t)\Phi_i(t)dt \tag{3.8}
\]
3.2.2 Application to Measurement Data

In practice, however, we have only access to a finite set of samples observed each $T$ time unit. Let $X[k]$ denote the discrete version of a time continuous process sampled at times $kT$. The KLE theorem for discrete processes is given by

$$X[k] = \sum_{i=1}^{\infty} Y_i \Phi_i[k]$$  \hspace{1cm} (3.9)

The issue of finite sample set size can be addressed by approximating the original infinite-dimensional system by an $N$-dimensional system such as

$$X[k] = \sum_{i=1}^{N} Y_i \Phi_i[k]$$  \hspace{1cm} (3.10)

where $N \leq n$, and $n$ denotes the number of available samples from the stochastic process $X[k]$.

To compute the orthogonal eigenfunctions of $X[k]$ we need to construct a $N \times (n-N)$ matrix $X$ that contains in each row a time-shifted version of $X[k]$.

$$X = \begin{bmatrix} x(1) & \cdots & x(n-N) \\ x(2) & \cdots & x(n-N+1) \\ \vdots & \ddots & \vdots \\ x(N) & \cdots & x(n) \end{bmatrix}$$  \hspace{1cm} (3.11)

and apply the SVD technique described in the previous section to this matrix.

Up to now we have presented the KLE theorem that approximates a stochastic process by a limited number of orthogonal eigenfunctions, which have the largest eigenvalues. Next, we will show how this theorem can be applied for anomaly detection purposes.
3.3 KLE-based Anomaly Detection

Lakhina et al. [LCD04c, LCD05] have developed an anomaly detection method based on PCA that they call the subspace method. The subspace method applies PCA to derive a normal behavior model from the top-$k$ principal components. This model, however, considers only the spatial correlation between different Origin-Destination (OD) flows and different detection metrics, i.e., volume and IP/port entropy, but not the temporal correlation present in the data. In the following, we describe our detection method that considers both, temporal and spatial correlation, by applying a multivariate version of the KLE theorem to derive a normal behavior model.

### 3.3.1 Multidimensional KL Expansion

For the purpose of anomaly detection we have to consider the temporal correlation between sample $k$ and sample $k+s$ and the spatial correlation between $K$ zero mean stationary sampled stochastic processes $X_i[k]$, $i = 1,\ldots,K$ with covariance function

$$\sigma_{i,j}(s) = \mathbb{E}\{X_i[k]X_j[k-s]\}$$

Therefore, we extend the one-dimensional KLE theorem defined in the last section to the multi-dimensional case:

$$\begin{bmatrix}
X_1[k] \\
\vdots \\
X_K[k]
\end{bmatrix} = \sum_{i=1}^{K} \sum_{j=1}^{N} Y_i,j \Phi_{i,j}[k]$$

According to Equation 3.11 we need to construct a $NK \times (n-N)$ observation matrix $X$ as
3 Applying PCA for Anomaly Detection

\[
X = \begin{bmatrix}
  x_1(1) & \ldots & x_1(n-N) \\
  x_1(2) & \ldots & x_1(n-N+1) \\
  \vdots & \ddots & \vdots \\
  x_1(N) & \ldots & x_1(n) \\
  \vdots & \ddots & \vdots \\
  x_K(1) & \ldots & x_K(n-N) \\
  x_K(2) & \ldots & x_K(n-N+1) \\
  \vdots & \ddots & \vdots \\
  x_K(N) & \ldots & x_K(n)
\end{bmatrix}
\]  

(3.14)

for computing the eigenfunctions \( \Phi_{i,j} \) and the corresponding eigenvalues with the SVD technique.

It is noteworthy that because of temporal correlation one needs more data than in the independent case of PCA. However, this added complexity is unavoidable when one has to deal with correlated observations.

### 3.3.2 Normal Behavior Model

In the anomaly detection context, the KLE theorem is used for modeling the network behavior observed through \( j = 1, \ldots, K \) correlated stochastic processes \( X_j[k] \) under normal conditions. We make the same assumption here as the classical subspace method: The normal behavior of the system, e.g., the daily variation, is captured by the top principal components. Therefore, we obtain a linear approximation of the initial processes in a smaller dimension vector space as

\[
\begin{bmatrix}
  \hat{X}_1[k] \\
  \vdots \\
  \hat{X}_K[k]
\end{bmatrix} = \sum_{i=1}^{L} \sum_{j=1}^{M} Y_{i,j} \Phi_{i,j}[k]
\]  

(3.15)
3.3 KLE-based Anomaly Detection

where $L < K$ and $M < N$. This approximation has a noteworthy optimality property. Among all approximations defined over a linear space of dimension $LM$, this is the linear approximation with the smallest error variance $\text{Var}\{X - \hat{X}\}$. It is noteworthy that this formulation of the normal behavior model includes the case of classical PCA as applied by Lakhina et al. since the temporal correlation is completely disregarded for $M = 1$.

The basis change transform becomes a $KN \times LM$ matrix $V$ that contains the $LM$ eigenfunctions $\Phi_{i,j}[\cdot]$ in its columns. Consequently, the model prediction $\hat{X}$ can be written as a function of the input data $X$ and the basis change matrix $V$:

$$\hat{X} = V V^T X^T$$ (3.16)

Having derived a model for the normal behavior of the system, we now need to design a statistical test that can be applied to decide whether an observation adheres to the model or not.

### 3.3.3 Statistical Test

The residual signal $e[k]$ used for anomaly detection is obtained by taking the difference between model prediction and observation.

$$e[k] = X[k] - \hat{X}[k]$$ (3.17)

The realigned $^1$ squared prediction error

$$Q[k] = e[k]e[k]^T$$ (3.18)

provides a measure for the fit of an observation to the model. Jackson and Mudholkar [JM79] propose to use a non-linear function of the squared prediction error $Q[k]$ as decision variable $D[k]$

---

$^1$The temporal offset that was previously introduced in Equation 3.14 has to be removed before taking the square of the error components.
that is approximately normally distributed with zero mean and unit variance.

They have developed a statistical test for the residual vector. For a fixed type I error (false positive) rate $\alpha$, the upper limit for $Q$ may be approximated by

$$Q_{\alpha} = \theta_1 \left[ D_{\alpha} \sqrt{\frac{2\theta_2 h_0^2}{\theta_1}} + 1 + \frac{\theta_2 h_0 (h_0 - 1)}{\theta_1^2} \right]^{\frac{1}{h_0}}$$

(3.19)

where

$$h_0 = 1 - \frac{2\theta_1 \theta_3}{\theta_2^2}$$

(3.20)

and

$$\theta_i = \sum_{j=L+1}^{M} \lambda_{ij}$$

(3.21)

for $i = 1, 2, 3$. The authors of [JM79] refer to the work of Jensen [JS72] showing that the test is derived under the hypothesis that the elements of $e[k]$ are statistically independent and follow a Gaussian distribution.

### 3.4 Evaluation

Finally, we want to show that the theoretically derived results also hold in practice. Therefore, we manually identified a set of anomalies in a three-week NetFlow trace from the SWITCH network. Then we applied the classical PCA subspace method [LCD05] as well as our extended KLE-based detection method to the data. However, as our dataset comes from a stub network instead of a transit network like Abilene or GEANT, we do not consider the correlation across multiple OD flows but between incoming and outgoing traffic flows. We compare the detection
3.4 Evaluation

results with the help of ROC curves and find that our KLE approach significantly improves the detection results.

3.4.1 Data Set and Ground Truth

We evaluate the classical PCA and our KLE approach on three weeks of unsampled NetFlow data coming from the three peering links of a medium-sized ISP (SWITCH, AS559) located in Zurich, Basel, Geneva. A map of the SWITCH network is given in Figure 1.2. The data used in this study was recorded between August 19th 2007 and September 10th 2007. It comprises a variety of traffic anomalies happening in daily operation such as network scans, Denial of Service attacks, alpha flows, etc.

The paucity of labeled datasets and standardized labeling methods has been extensively discussed in the related work section of this thesis. Therefore, anomalies in the data were identified using available manual labeling methods: visual inspection of timeseries and top-n queries directly on the flow data [RSRD07].

<table>
<thead>
<tr>
<th>Classification</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denial of Service Attacks</td>
<td>8</td>
</tr>
<tr>
<td>Distributed DoS Attacks</td>
<td>7</td>
</tr>
<tr>
<td>Network Scans</td>
<td>10</td>
</tr>
<tr>
<td>Alpha flows</td>
<td>2</td>
</tr>
<tr>
<td>Outages</td>
<td>2</td>
</tr>
<tr>
<td>Total Anomalies</td>
<td>29</td>
</tr>
</tbody>
</table>

Table 3.1: List of the classified anomalies.

First we have identified the set of suspicious time intervals that contain either spikes or jumps in any of the timeseries used for detection by visual inspection. In a second step, we conducted top-n queries directly on the flow data to gain further confidence in our labels. In particular, we inspected the top-n IP addresses and ports that were responsible for most of the flows and verified
whether they have changed considerably over time. Subsequently, each verified anomaly was classified according to the criteria defined in [LCD04c]. This methodology resulted in the set of 29 anomalies that are listed in Table 3.1. While most anomalies last only for a single or few intervals, one of the anomalies, a long-lasting network scan, has a duration of a full day.

### 3.4.2 Timeseries Construction

Before applying PCA or KLE to a dataset we need to transform the data into timeseries and further pre-process the constructed timeseries. In particular, we compute three volume metrics for each direction (incoming or Not\_AS559->AS559 and outgoing or AS559->Not\_AS559) for all the traffic of a single border router, namely byte, packet, and flow counts resulting in a total of six timeseries. All timeseries are obtained by aggregating the traffic at 15-minute intervals resulting in a $2016 \times 6$ data matrix where each line contains the metric values for one interval.
Figure 3.1: Time series of the preprocessed, i.e., demeaned and normalized, incoming traffic metrics that are used as input for PCA and KLE. Furthermore, we plot the labels with each timeseries.
Figure 3.2: Time series of the preprocessed, i.e., demeaned and normalized, outgoing traffic metrics that are used as input for PCA and KLE. Furthermore, we plot the labels with each timeseries.
3.4 Evaluation

As PCA and KLE have to be applied to zero-mean signals it is important to further process the constructed timeseries. Therefore, we demean each timeseries $x$ by subtraction of its mean ($x_0 = x - \bar{x}$), and additionally normalize each timeseries by division with its standard deviation ($x_n = x_0 / \sigma_x$). The second step is required to avoid that the principal components are dominated by those timeseries that have the largest variance, e.g., the byte counts in our example.

The timeseries of the pre-processed metrics are plotted in Figure 3.1 for incoming traffic and in Figure 3.2 for outgoing traffic. Moreover, we highlight the labels, i.e., time intervals that have been identified as malicious/disruptive by manual classification, in each timeseries plot. The long-lasting network scan starts at interval 576 and ends 1 day later at interval 672. Moreover, we clearly observe the daily traffic variation in each of the timeseries.

3.4.3 Residuals

Having computed the timeseries used for detection over the three weeks of data, the next step is to train a PCA- and KLE-model over a limited number of data samples. In our evaluation we use the first two days corresponding to 192 time samples for calibrating the classical PCA and the KLE model varying the number of top components $L$ to be included in the model. We use the first two days for training as they contain only very few anomalies and we can apply the remaining days for validation. The complexity of selecting the optimal number of components $L$ to be included in the model, and the impact of $L$ on the detection result have been extensively discussed in [RSRD07]. We therefore do the same evaluation for $L = 2$, $L = 3$, and $L = 4$.

In Figure 3.3 we show the time series of the normalized prediction error $D[k]$ for $L = 3$. The upper plot shows the decision variable derived from a classical PCA model ($M = 1$), whereas the lower plot shows the same metric derived from a KLE model with
Figure 3.3: Timeseries of the decision variable $D[k]$ for $L = 3$ and $M = 1$ in the upper plot and for $L = 3$ and $M = 3$ in the lower plot. Additionally, we have included the labels and a hypothetical threshold in each plot.
Merely by visual comparison of the two plots it becomes already evident that classical PCA misses a significant part of the anomalies that are detected with the KLE approach. For example, the outage event at time interval 807 is not detected by PCA. This is due to the fact that the spatial correlation structure, that is essentially captured in the PCA model, does not change during this anomaly. The same applies for the network scan in interval 865 and various other anomalies. The temporal correlation structure during these events, on the other hand, is very well impacted and therefore KLE detects these anomalies with large confidence.

Furthermore, checking the decision variable $D[k]$ for PCA, we found that the assumptions of independence and Gaussianity of the error terms are violated as we observe a considerable bias and correlation between error terms. We further found that bias and correlation can be significantly reduced when applying the KLE method instead of PCA. This is a further indication that the Karhunen-Loeve transform is better suited for the problem at hand than the classical PCA subspace method.

### 3.4.4 ROC Curve Analysis

To validate this observation over a larger set of parameters we conducted a complete ROC curve analysis. The Neyman-Pearson theorem about statistical tests [LM01] defines a fundamental trade-off between true and false positive rates that is captured in a Receiver Operating Characteristics (ROC) curve. We computed the true and false positive rates, varying the number of components included in the model $L$, the correlation range $M$, and the detection threshold. In particular, for finding the false positive rate we determined the fraction of *intervals* in $D[k]$ without a positive label that have a value above the selected threshold for each setting. Likewise, for finding the true positive rate we determined the fraction of *events* that have at least one interval...
with a value larger than the selected threshold. Using intervals or events as unit of measure does not make much of a difference for short anomalies. However, since we have this long-lasting network scan event in our data we prefer to use events as unit of measure for the true positive rate since it would certainly falsify an interval-based measure.

In Figure 3.4 we show the four ROC curves for $L=3$ and $M=1,\ldots,4$. Note that all ROC curves are plotted on a logarithmic $x$-scale to increase their readability. The ROC curves for $L=2$ and $L=4$ are depicted in Figure 3.5 and Figure 3.6. A comparison of the three plots confirms the sensitivity of PCA to the top-$k$ parameter setting reported in [RSRD07]. In particular, we obtain the best detection results when the top three principal components are included in the model ($L=3$). If we use only
the top two components (see Figure 3.5) we obtain too many false positives as normal behavior is misclassified as anomalous, and on the other hand, if we use the top four components (see Figure 3.6) a relevant fraction of the anomalies is missed as they are included in the model. Moreover, we observe that KLE is more tolerant if the parameter $L$ is chosen too large. This is simply due to the fact that the number of principal components increases with $M$, i.e., $M = 1$ results in six principal components whereas $M = 2$ results in twelve principal components, and so on.

In Figure 3.4 we observe a constant offset of up to 20% between the ROC curves for classical PCA ($M = 1$) and those for KLE ($M > 1$). In particular, at a false positive rate of $10^{-3}$ PCA achieves a true positive rate of approximately 10% whereas KLE achieves a true positive rate of 30% for $M = 4$. Likewise, at a false positive rate of $2 \times 10^{-2}$ PCA detects only 50% of all anomalies whereas KLE detects 70% of all anomalies. This result confirms our hypothesis that a certain fraction of the anomalies is simply not detectable with the PCA approach. Obviously, the offset between the curves decreases as the false and true positive rates approach 100%. These ROC curves very nicely demonstrate the impressive performance gain that can be achieved when using KLE instead of PCA.

In Figure 3.5 we plot the results for $L = 2$, i.e., where only the top two principal components are included in the model. We find that all four ROC curves show worse results than those for $L = 3$. This can be explained by the fact that the normal subspace is too small as explained above. Nevertheless, even for this suboptimal setting of the subspace parameter $L$, we find that KLE gives better results than PCA. The performance gain, however, is smaller, i.e., up to 10%, than for the optimal subspace parameter setting of $L = 3$.

In Figure 3.6 we plot the ROC curves for $L = 4$ where we have used a model that includes the top four principal components. As we have outlined above, for PCA this setting means that we
include 4 out of 6 principal components in the model. Hence, a significant part of the anomalies are included in the model and are therefore not detectable anymore. This explains the bad performance of PCA for $L=4$. In contrast, the results for KLE degrade only slightly when compared to the optimal subspace setting of $L=3$. For different false positive rates we observe an acceptable degradation in the true positive rate for KLE between 3% and 10%, whereas the degradation for PCA of up to 30% is fatal.

Figure 3.5: ROC curves for $L=2$. The curve for $M=1$ corresponds to classical PCA.
3.5 Summary

In this chapter we have studied the eligibility of popular dimension reduction techniques, namely Principal Component Analysis (PCA) and Karhunen-Loeve Expansion (KLE), to the anomaly detection problem. The often overlooked difference between the two techniques is the following: PCA is applicable to random processes whereas KLE is used with stochastic processes that have temporal correlation. Hence, KLE is better suited for anomaly detection as the signals used for detection typically have temporal correlation. However, the additional cost of KLE is not justified when the input signals are random processes.

We verified, in theory and in practice, that temporal correlation cannot be ignored when applying dimension reduction.
techniques to the anomaly detection problem. We have proposed an alternative detection method based on multidimensional Karhunen-Loeve expansion that incorporates the temporal correlation as well as the spatial correlation into the normal behavior model. Finally, we have shown on three weeks of NetFlow data how clearly our method outperforms the classical subspace-PCA method developed in previous work. We expect these results to hold on other datasets as well, since temporal correlation has been successfully exploited for other datasets in previous work e.g., using a Kalman filter [SST05]. Nevertheless, also with our approach anomalies have to be sufficiently large in terms of flows in order to cause detectable disruptions in the temporal or spatial correlation structure of traffic.
Chapter 4

Anomaly Extraction

Having discussed dimension reduction techniques for anomaly detection in the last chapter, we now turn to the problem of anomaly extraction. We introduce anomaly extraction as an additional step between anomaly detection and root-cause analysis that takes the meta-data associated with an anomaly detection alert and provides a concise summary of the anomalous flow sets identified by the meta-data for further root-cause analysis. Hence, the goal of anomaly extraction is to aid a network administrator in his difficult task to make sense out of imprecise and vague anomaly detection alerts.

In this chapter, we formulate the anomaly extraction problem and provide a solution for identifying and summarizing the set of flows that have caused an anomaly detection alert. Our solution is based on association rules. For the detection part we use several histogram-based detectors as they provide more information about an alert than simple volume-based detectors (e.g., the PCA-based detector described in the previous Chapter). To improve the quality of the meta-data that is used for the extraction part, we introduce histogram cloning and different voting strategies among clones. Furthermore, we provide guide-
lines for calibrating the various parameters used in the detection and extraction part. We evaluate our approach with two weeks of traffic data from the SWITCH network that contains a total of 36 malicious/disruptive events to study.

4.1 Introduction

Anomaly detection techniques 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 4.1 we present the high-level goal of anomaly extraction. In the bottom of the figure, malicious/disruptive 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 2.1 we have outlined 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 [KSZC03, CM05]. 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 4.2, 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: 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 attack. We test our anomaly extraction method on rich network traffic data from a medium-size backbone network (SWITCH). 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.

4.2 Methodology

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

4.2.1 Approach Overview

An overview of our approach to the anomaly extraction problem is given in Figure 4.3. It contains two subfigures that illustrate the individual steps of our approach. The upper subfigure depicts the anomaly detection and meta-data generation steps. These steps are applied for each traffic feature. The lower subfigure shows how association rule mining is applied to suspicious flows.

A subtle point of our approach is filtering flows matching any meta-data (in other words we take the union of the flows matching meta-data) instead of flows matching all meta-data, *i.e.*, the intersection of the flows matching meta-data. Assume for example the Sasser worm that propagated in multiple stages: initially a large number of SYN flows scanned target hosts, then additional flows attempted connections to a backdoor on port 9996 of the vulnerable hosts, and finally a third set of frequent flows resulted from downloading the 16-Kbyte worm executable. *Anomalies often result in such distinct sets of frequent flows with similar characteristics.* In addition, different meta-data can relate to different phases of an anomaly. In our example, the anomaly could be annotated with meta-data about the SYN flag, port 9996, and the specific flow size. The intersection of the flows matching the meta-data would be empty, whereas the union would include the anomalous flows.
Our approach consists of four main functional blocks.

- **Histogram cloning**: To obtain additional traffic views the distribution of a traffic feature is tracked by multiple histogram clones. Each clone randomizes the distribution using one of \( k \) independent hash functions. Upon detection of a disruption in the distribution each clone compiles a list \( V_k \) of traffic feature values that are associated with the disruption.

- **Voting**: Meta-data is compiled from the individual feature value lists \( V_k \) by voting. Specifically, if a certain feature value is selected by at least \( l \) out of \( k \) clones, it is included in the final meta-data. We analyze the impact of differ-
ent parameter settings for $l$ and $k$ on the accuracy of our approach.

- **Flow pre-filtering:** We use the union set of meta-data provided by $n$ different traffic features to pre-filter a set of suspicious flows. This pre-filtering is necessary since it typically eliminates a large part of the normal flows.

- **Association rule mining:** A summary report of the most frequent item-sets in the set of suspicious flows is generated by applying association rule mining algorithms. The basic assumption behind this approach is that the most frequent item-sets in the pre-filtered data are often related to the anomalous event. A large part of our evaluation results are devoted to the verification of this assumption.

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

### 4.2.2 Histogram Cloning and Detection

Histogram-based anomaly detectors [KSD09, SBK08, LBC+06, RSK09] have been shown to work well for detecting anomalous behavior and changes in traffic distributions. In contrast to the PCA method, histogram-based detectors do not aim at dimension-reduction, but at providing additional information about anomalies such as affected IP addresses or ports. Hence, they are better suited than PCA for the purpose of anomaly extraction.

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

During time interval \( t \), an anomaly detection module constructs histogram clones for different traffic features. At the end of each interval, it computes for each clone the KL distance between the distribution of the current interval and a reference distribution. The KL distance has been successfully applied for anomaly detection in previous work [GMT05, RSK09]. It measures the similarity of a given discrete distribution \( q \) to a reference distribution \( p \) and is defined as

\[
D(p||q) = \sum_{i=0}^{m} p_i \log \left( \frac{p_i}{q_i} \right)
\]

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

Instead of training and recalibrating distributions that represent normal behavior, we use the distribution from the previous measurement interval as reference distribution \( p \). Hence, we will observe a spike in the KL distance timeseries each time the distribution changes. Assuming an anomalous event that spans multiple intervals, the KL distance will generate spikes at the

\(^1\)Note that histogram cloning uses random projections as they are commonly used in sketch data structures [10]. 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.
Figure 4.4: Upper plot: KL distance time series for the source IP address feature for roughly two days. Lower plot: First difference of the KL distance for the same period. The dashed line corresponds to the anomaly detection threshold.

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 timeseries for the source IP address feature over roughly two days is depicted in Figure 4.4 in the upper plot.

We have observed that the first difference $\Delta t$ of the KL distance timeseries is approximately normally distributed with zero mean and standard deviation $\sigma$. 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
Figure 4.5: This Figure illustrates our incremental method for determining the anomalous bins. The KL distance converges to zero as in each round the bin with the largest absolute difference is aligned with its counterpart in the reference distribution. Already after the first round the KL distance decreases significantly.

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

In Figure 4.4, we show the $\Delta_t D(p||q)$ timeseries for the source IP address feature and the corresponding threshold. An alarm is only generated for positive spikes crossing the threshold, since they correspond to significant increases in the KL distance.

If we detect an anomaly during interval $t$ we 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. This procedure is illustrated in Figure 4.5, where we plot the KL distance computed in each round. Already after the first round, the KL distance decreases significantly. Having identified the set of anomalous histogram bins $B_k$ for each clone, we obtain the corresponding set of feature values $V_k$ by keeping a map between values and corresponding bins.

4.2.3 Voting and Meta-data Generation

The cardinality of $V_k$ is typically much larger than the cardinality of $B_k$, e.g., the 65,536 unique port numbers are distributed evenly over 1024 bins if we use a 10-bit hash function for randomization. Therefore the set of feature values $V_k$ provided by each clone is likely to contain many normal feature values colliding on anomalous bins. 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.

It is common practice to keep only those feature values that have been identified by all histogram clones $M_j = \cap_k V_k$ in order to minimize false positives. We generalize this approach to a more flexible scheme that is based on voting. In particular, voting keeps a feature value if it has been selected by at least $l$ out of $k$ clones. With this approach the trade-off between false-positive and false-negative feature values can be adjusted via the parameters $k$ and $l$.

Assume that each of the $k$ clones has detected a disruption in the distribution of feature $j$ in interval $t$, and has identified $b$ responsible bins. Each clone includes an anomalous feature value
in the set $V_k$ with probability $p_a$, while a normal feature value is selected only if it collides on one of the selected bins, and has thus a probability of selection of $p_n = b/m$ where $m$ is the total number of bins.

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

$$P_a \geq \sum_{i=l}^{k} \binom{k}{i} p_a^i (1 - p_a)^{k-i}$$  \hspace{1cm} (4.1)

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

$$P_a^\bar{\ } \leq 1 - \sum_{i=l}^{k} \binom{k}{i} p_a^i (1 - p_a)^{k-i}$$  \hspace{1cm} (4.2)

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

$$P_n = \sum_{i=l}^{k} \binom{k}{i} p_n^i (1 - p_n)^{k-i}$$  \hspace{1cm} (4.3)

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

To sum up, the meta-data $M_j$ for feature $j$ obtained after the voting process contains feature values representing normal and anomalous traffic. The ratio between the two classes depends on the parameters $k$ and $l$, on the initial probability $p_a$, and the total number of bins $m$.

### 4.2.4 Flow Pre-filtering

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

4.2.5 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 [AS94]. Formally, let a transaction $T$ be a set of $h$ items $T = \{e_1, \ldots, e_h\}$. Then the disjoint subsets $X, Y$ define an association rule $X \Rightarrow Y$. The support $s$ of an association rule is equal to the number of transactions that contain $X \cup Y$.

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

Our motivation for applying association rules to the anomaly extraction problem is that anomalies typically result in a large number of flows with similar characteristics, e.g., IP addresses, port numbers, or flow lengths, since they have a common root-cause like a network failure, a bot engine, or a scripted Denial of Service (DoS) attack. Each transaction $T$ corresponds to a flow 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 $e_1 = \{\text{srcPort} : 80\}$ refers to a source port number equal to 80, while $e_2 = \{\text{dstPort} : 80\}$ refers to a destination port number equal to 80. An $l$-item-set $X = \{e_1, \ldots, e_l\}$ is
Anomaly Extraction

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, \( e.g. \), \( X = \{ \text{dstPort} : 80, \text{dstPort} : 135 \} \) 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 \( X = \{ \text{dstIP} : 129.132.1.1, \text{dstPort} : 80 \} \) 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 [AS94]. Apriori makes at most \( h \) passes over the data. In each round \( l = 1 \ldots h \), it computes the support for all candidate \( l \)-item-sets. At the end of the round, the frequent \( l \)-item-sets are selected, which are the \( l \)-item-sets with frequency above the minimum support parameter. The frequent item-sets of round \( l \) are used in the next round to construct candidate \((l+1)\)-item-sets. The algorithm stops when no \((l+1)\)-item-sets with frequency above the minimum support are found.

By default, Apriori outputs all frequent \( l \)-item-sets that it finds. We modify this to output only \( l \)-item-sets that are not a subset of a more specific \((l+1)\)-item-set. More specific item-sets are desirable since they include more information about a possible anomaly. This measure allows us to significantly reduce the number of item-sets to process by a human expert. We denote the final set of \( l \)-item-sets as \( I \). The Apriori algorithm takes one parameter, \( i.e., \) the *minimum support*, as input. If the minimum support is selected too small, many item-sets representing normal flows (false positives) will be included in the output. On the other hand, if the minimum support is selected too large, the item-sets representing the anomalous flows might be missed (false negative).
Table 4.1: 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 were anonymized.
**Apriori Example** In the following we give an example of using Apriori to extract anomalies. We use a 15-minute NetFlow trace from the SWITCH network. In this interval destination port 7000 was the only feature value that was flagged by all histogram clones. It contributed 53,467 candidate anomalous flows. We manually added to the candidate set $\bigcup F_j$ flows that had one of the three most frequent destination ports but had not been flagged by all histogram clones. We did this on purpose to generate and illustrate false positive item-sets. 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 $\bigcup F_j$ 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 $\bigcup F_j$.

The final output of the algorithm is given in Table 4.1, which lists a total of 15 frequent item-sets. In the first iteration, a total of 60 frequent 1-item-sets were found. 59 of these were, however, removed from the output as subsets of at least one frequent 2-item-set. In the second iteration, a total of 78 frequent 2-item-sets were found. Again, 72 2-item-sets could be removed since they were subsets of frequent 3-item-sets. In the third iteration, 41 frequent 3-item-sets were found, of which four item-sets were not deleted from the output. In the fourth round, 10 frequent 4-item-sets were found but only one of them remained after removal of redundant 4-item-sets. Two frequent 5-item-sets were found in round five. Finally, the algorithm terminated as no frequent 6-item-sets satisfying the minimum support were found.

Three out of the 15 frequent item-sets had destination port 7000. We verified that indeed several compromised hosts were flooding the victim host E on destination port 7000. Regarding the other frequent item-sets, we verified that hosts A, B, and C, which sent a lot of traffic on destination port 80, were HTTP proxies or caches. The traffic on destination port 9022
4.2 Methodology

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>Number of detectors</td>
<td>5</td>
</tr>
<tr>
<td>$w$</td>
<td>Interval length</td>
<td>[5,10,15] min</td>
</tr>
<tr>
<td>$m$</td>
<td>Number of bins</td>
<td>[512,1024,2048]</td>
</tr>
<tr>
<td>$k$</td>
<td>Number of clones</td>
<td>1-50</td>
</tr>
<tr>
<td>$l$</td>
<td>Voting parameter</td>
<td>1-$k$</td>
</tr>
<tr>
<td>$s$</td>
<td>Minimum support</td>
<td>3,000-10,000 flows</td>
</tr>
</tbody>
</table>

Table 4.2: Parameters including description and range as used in the evaluation section of this work.

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.

4.2.6 Parameter Estimation

The various parameters associated with our approach, and their range as used in the evaluation of this work, are summarized in Table 4.2. 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 $n$:** As the number of detectors increases further information can be exploited for identifying anomalies and therefore a large $n$ is generally desired. In this work we use five detectors, where each detector monitors one of the following features: source IP address, destination IP address, source port number, destination port number, number of packets per flow. Other features that might be useful for anomaly detection purposes are the number of packets per flow, the average packet size, or the flow duration.
Interval length $w$: The interval length $w$ determines the detectable anomaly scale, i.e., it becomes harder to detect short disruptions that contain only few flows with longer intervals. On the other hand, it is not always desirable to detect such short disruptions. Hence, the desired number of daily or weekly anomalous alarms can be used to set the interval length $w$. The desired number of alarms depends on the available human resources for investigating alarms. Some studies report that actionable alarms require on average 60 minutes investigation time [Pro06], which would correspond to 8 alarms per day assuming a full-time employ for analyzing alarms. Another issue related to the interval length is the detection delay as an anomaly can only be detected at the end of a given interval. Typically used intervals correspond to delays of few minutes, e.g., 5 to 15 minutes. However, a sliding window mechanism can shorten this delay. Finally, one last implication is that a larger $w$ results in more flows to be processed by association rule mining and in higher computational overhead. Nevertheless, the overhead of association rule mining is low as we discuss in the next section.

Number of bins $m$: The number of histogram bins $m$ is also involved in a detection sensitivity versus aggregation trade-off as discussed for parameter $w$. The smaller the number of bins the more flows are aggregated per bin. In addition, a larger $m$ is desired for anomaly extraction as it decreases the probability $P_n$ that a normal feature value remains in the meta-data after voting and, thus, the number of candidate flows for rule mining. Finally, the parameter also affects the required memory resources. Assuming that the available memory resources do not drive the choice of $m$, then an acceptable range of values can be first determined via simulation using Equation 4.3 and a target range for $P_n$. Then, $m$ should be selected together with $w$ based on a desired number of daily/weekly alarms. Among the possible $(m,w)$ choices realizing a desired number of alarms, the solutions with larger $m$, e.g., fewer flows per bin, are preferable for anomaly
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 pre-filtering step and thus the accuracy of our approach. The parameter settings for $l$ and $k$ can also be obtained by simulation using Equation 4.1 and 4.3. Simulation results for $P_a$ and $P_n$ for different settings of $l$ and $k$ will be presented in the evaluation section.

Minimum support $s$: The parameter $s$ determines the frequency threshold above which an item-set is extracted by Apriori as a possible set of anomalous flows. A large $s$ extracts no or few item-sets, which in our experiments were almost always associated with anomalous events. On the other hand, decreasing $s$ results in more item-sets and in a small but higher rate of false positives. In general 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. However, the choice of $s$ is not critically important. 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 slightly more overhead for running the algorithm. This however in our experiments was not very important as Apriori’s overhead
is low. Alternatively, the minimum support can be used as a user variable for zooming in and out of the most significant item-sets. The administrator can progressively decrease $s$ until sufficient anomalous item-sets have been investigated.

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

4.3 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.

4.3.1 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). The dataset used for this study was recorded during December 2007 and spans two continuous weeks.

To generate datasets for evaluating the Apriori algorithm, we computed the KL distance timeseries for the two weeks of data
4.3 Evaluation

<table>
<thead>
<tr>
<th>Anomaly class</th>
<th>Occurrences</th>
<th>Mean #flows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flooding</td>
<td>5</td>
<td>163,139</td>
</tr>
<tr>
<td>Backscatter</td>
<td>5</td>
<td>85,716</td>
</tr>
<tr>
<td>Network Experiment</td>
<td>3</td>
<td>27,606</td>
</tr>
<tr>
<td>DDoS</td>
<td>5</td>
<td>132,509</td>
</tr>
<tr>
<td>Scanning</td>
<td>16</td>
<td>96,375</td>
</tr>
<tr>
<td>Spam</td>
<td>1</td>
<td>33,765</td>
</tr>
<tr>
<td>Unknown</td>
<td>1</td>
<td>23,360</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>36</strong></td>
<td><strong>99,688</strong></td>
</tr>
</tbody>
</table>

Table 4.3: 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.

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-n queries on the data. To determine the root cause of each anomaly, we extracted all flows in an anomalous interval and analyzed the timeseries and distribution of the five features, and additionally of the number of packets and bytes per flow, the flow inter-arrival times, and the flow durations. We found a total of 36 different malicious/disruptive events within the 31 anomalous intervals. The identified anomalies, their class, and the average number of flows per class are listed in Table 4.3. We cover most known anomaly classes and have multiple instances for each class.

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.
4.3.2 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 number of bins $m$. We found small differences in the detection results for $m$ equal to 512, 1024, and 2048. We also found that the number of detections decreases with the interval length $w$. In particular, setting $m$ to 1024 and $w$ to 5, 10, and 15 minutes, we detected 62, 52, and 31 anomalous intervals, respectively. Based on these numbers and the parameter selection guidelines we analyzed in the previous Section, we set $w$ conservatively to 15 minutes, which corresponds to 2.2 alarms per day, and $m$ to 1024.

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

In Figure 4.6 we plot the ROC curves for three histogram clones, \textit{i.e.}, using three different hash functions. A detection rate of 0.8 corresponds to a false positive rate of 0.03, while a detection rate of 1 (100%) to a false positive rate between 0.05 and 0.08 for different clones. With a false positive rate as low as 0.01 only 40% of the anomalies are detected. These results are a lower bound on the performance of our detector. This is because some of the false-positive intervals might contain unknown anomalous traffic.
Figure 4.6: ROC curves plotting the false positive rate versus the true positive rate for different thresholds. The three curves correspond to different histogram clones.

4.3.3 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 Equation 4.1 and 4.3 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 Figure 4.7 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 $l = 5$, $l = 10$ are marked for better readability. For a given value of $k$, $P_\bar{a}$ increases with $l$, e.g., for $l = 5, k = 10$ we obtain $P_\bar{a} = 0.006$ while for $l = 10, k = 10$ the probability increases to $P_\bar{a} = 0.89$.

In Figures 4.8(a) and 4.8(b) we plot the probability $P_n$ that
4.3 Evaluation

Figure 4.8: Probability $P_n$ that a normal feature value is not eliminated by voting for different values of $l$ and $k$ in logarithmic scale. The number of anomalous bins is $b = 1$ (upper plot) and $b = 25$ (lower plot) and the total number of bins is $m = 1024$. 
a normal feature value is not eliminated by voting for different
values of $l$ and $k$ in logarithmic scale. The number of selected
bins is $b = 1$ and $b = 25$, respectively. The number of total bins
is $m = 1024$ for both plots. The results for $l = 1, l = 5$ are marked
for better readability. For a given value of $k$, $P_n$ decreases with
$l$, e.g., for $l = 1, k = 10$ 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
$l = 5, k = 10$ the probability decreases to $P_n = 10^{-13}$ for $b = 1$ and
$P_n = 10^{-6}$ for $b = 25$. Moreover, we observe that the probability of
including a normal feature value in the meta-data increases dra-
matically with the number of anomalous bins $b$. Consequently,
assuming a fixed setting of the voting parameters we have to tol-
erate higher false positive rates for anomalies affecting multiple
bins. Alternatively, the parameter $l$ could be adapted based on
the estimated number of bins $b$ to achieve a target probability
$P_n$. The average number of false positive feature values can be
determined by multiplication of $P_n$ with the average number of
feature values observed within one interval, e.g., between one and
65,536 for port numbers.

The simulation results show that a variety of operating points
$[P_\bar{a}, P_n]$ can be achieved by setting the voting parameters $l, k$
apropriately. 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 i) how
is the accuracy impacted by the number of normal feature
values included in the meta-data that is used for pre-filtering the
candidate flows, and ii) how does the rule mining performance
decrease with the number of candidate flows.

4.3.4 Accuracy of Rule Mining

After the meta-data has been identified by voting, the corre-
sponding flows are filtered and subsequently sent to the rule
mining process. The accuracy in terms of correctly identified
item-sets depends on: the accuracy of the meta-data used for pre-filtering flows, the structure of the pre-filtered normal and anomalous flows, and the minimum support parameter \( s \).

An interesting question concerning the accuracy of meta-data is: What is the probability that a normal value in the meta-data results in a false-positive item-set. Recall that an item-set will be generated if more than \( s \) flows matching the meta-data have one (1-item-set) or more (\( l \)-item-set) common feature values. We have observed that the probability for generating a false-positive item-set from a normal feature value is highly skewed. For example, if port number 80 is included in the meta-data it is very likely that many 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 rule mining accuracy, we used the flow data of the 31 anomalous intervals. To generate the input data sets for Apriori, we set \( k \) to 3, \( l \) to 3, and \( m \) to 1024. This corresponds to \( P_a = 0.488 \) and \( P_n = 10^{-4} \) for \( b = 25 \). Despite the large value for \( P_a \) none of the 31 anomalies were missed. This illustrates the fact that \( P_a \) is an upper bound that was derived under the assumption of independence between clones. On the other hand, as \( P_n \) is very low, only few normal feature values are included in the meta-data.

For 21 anomalous intervals (70%) we obtained no FP item-sets at all. The number of FP item-sets for the remaining 10 anomalous intervals is plotted in Figure 4.9 together with the average number of FP item-sets over all 31 anomalous intervals (marked with squares). The number of FP item-sets decreases with the minimum support since less FP item-sets satisfy the minimum support condition. Figure 4.9 shows that on average between 2 and 8.5 FP item-sets are generated for minimum support values between 3,000 and 10,000 flows, respectively. The
Figure 4.9: 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.

The top three lines in the figure correspond to anomalies with higher numbers of FP item-sets. The observed FP item-sets are exclusively caused by anomalous heavy-hitter feature values such as common ports, e.g., port 80, or short flow lengths. Hence, if an anomaly happens to involve such a heavy-hitter feature value the number of FP item-sets automatically increases even if no normal feature values are included in the meta-data. However, most of the FP item-sets can be sorted out rather easily by a network administrator.

An important question is which types of anomalies are captured with our rule mining approach. There are two requirements for extracting an anomaly. The anomaly should i) be detected by causing a deviation in a traffic feature distribution, and ii) trigger a large number of flows with similar characteristics. For
many anomalies that originate from or 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 fall into this category. Although the rule mining approach is not targeted at botnet detection, anomalous activities such as spamming, scanning or flooding are often caused by compromised hosts. Other anomalies may not be concentrated on a single or few IP addresses like network outages, routing anomalies, or distributed scanning. Distributed scanning activity typically has a common destination port and often a common flow length. Therefore, it will appear as a frequent item-set. Anomalies that affect certain network ranges, such as outages or routing anomalies can be either captured by using IP address prefixes as additional dimensions for rule mining, or by applying concepts from the hierarchical heavy-hitter detection domain [CKMS08].

4.3.5 Computational Overhead of Rule Mining

The exact computational overhead of Apriori depends highly on the implementation used. Progressive implementations that use FP-trees and database partition techniques [KPP03] have been shown to outperform standard hash tree implementations [AS94]. Nevertheless, for all implementations the computational overhead increases with the number of transactions and the number of frequent 1-item-sets. Since both, the number of transactions and the number of frequent 1-item-sets, increase as more normal flows are included in the input data set, the performance of Apriori will decrease if we lower the threshold of the histogram-based detectors or if we do not use pre-filtering at all. Moreover, some implementations show considerably longer computation times as the relative minimum support decreases [KPP03], which is equivalent to increasing the dataset size and keeping the absolute minimum support constant. In our experiments using a non-optimized im-
implementation in Python, the computation overhead was small requiring few seconds up to minutes in the worst case.

4.3.6 Decrease in Classification Cost

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

Since the cardinality of $I$ depends on the minimum support parameter, we plot in Figure 4.10 the reduction in classification cost.
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.

4.4 Summary

In this chapter, we have studied the anomaly extraction problem that is of uttermost importance to several applications such as root-cause analysis and detection system testing. We have introduced the concepts of histogram cloning and voting among clones for selecting the feature values to be included in the meta-data. Further, we have introduced a method for extracting anomalous flow sets that uses association rules to summarize flows that have similar characteristics across several traffic features.

We conducted an extensive evaluation of our approach using two weeks of NetFlow data from the SWITCH network captured in December 2007, as well as simulation. In particular, we evaluated the accuracy of our histogram-based detector using ROC curves. We used simulation to find the range of possible operating points that can be tuned via the voting parameters. We evaluated the performance of the rule mining approach on 31 datasets containing a variety of anomalies and found that the number of false positive item-sets is between 2 and 8.5 in the given setting. Moreover, we have discussed the computational overhead of rule mining and have shown that the classification cost comparing the flows in a 15-minute interval and the item-sets in the report generated by our approach is reduced by several orders of magnitude.
Chapter 5

Anomaly Modeling, Generation and Injection

An efficient method for anomaly extraction is not only useful for network operators, but also for testing anomaly detection systems. In the following we exploit the previously described method to extract and characterize a set of anomalies and build models from them.

Host-based anomaly detection systems [PSB06, CCC+05] monitor activity on a single host. Metasploit [Fos07] is a very handy tool for evaluating the detection performance of host-based anomaly detection systems since it allows for launching attacks with configurable payload against targets in a test environment. Network-based anomaly detection systems, on the other hand, mine unusual patterns in the traffic of a whole network. Numerous anomaly detection systems operating on flow data [LCD04c, SST05, LBC+06, DFB+07] have been proposed for detecting a wide range of anomalies such as Denial of Service attacks, network scans, heavy hitters, and botnet traffic. Unfortunately, there is no system comparable to Metasploit for generating realistic network-wide anomalies that can be used for evalu-
ating flow-based anomaly detection systems. One major obstacle is that at present we simply do not have realistic synthetic models of many anomalies for simulation.

In this Chapter, we make the following contributions to fill the described gap:

- **Extraction:** We extract a large number of anomalies and corresponding flows from three weeks of SWITCH NetFlow traces that were captured in August 2007.

- **Characterization:** We characterize the found anomalies and group similar anomalies into well-known classes such as scans, Denial of Service attacks, or spam.

- **Modeling:** We provide a basic set of parameterizable anomaly models that allow users to generate realistic anomalous traffic at the flow level.

- **Injection:** We develop a tool called FLAME for injecting anomalous flows into background traces based on the derived anomaly models.

FLAME has been used by numerous researchers from academic institutions and industry for testing their anomaly detection systems. For example, FLAME was used extensively to evaluate a novel anomaly detection method based on the Tsallis entropy spectrum [TBSM09]. Furthermore, Guavus [Gua], a start-up company based in the US and India, has used FLAME for evaluating their products.

### 5.1 Problem Statement

First of all anomaly modeling requires a clear understanding or definition of what an anomaly is. Unfortunately, no general anomaly definition exists. Soule et al. [SST05], for example, define anomalies as disturbances of different shapes such as
ramps, squares, or steps. The authors argue there is evidence that anomalies such as Denial of Service attacks, flash crowds, or alpha flows entail such changes in volume metrics. However, this level of abstraction is too high for realistic evaluation scenarios since the relation between traffic parameters such as scan rate or number of attackers and model parameters such as the slope of a ramp or height of a step is not evident. Nevertheless, such high-level models might be useful for understanding more generic limitations of anomaly detection systems.

We select an alternative approach that models anomalies at the level of individual flows. At this level, we define two main, fundamentally different classes of anomalies.

**Aliens:** Alien anomalies are caused by malicious/disruptive activity that generates traffic in addition to the normal network activity. Scans, Denial of Service attacks, spam, or other botnet-generated traffic are examples of this class. Modeling of alien anomalies thus requires flow-level models for the traffic generated due to such events.

**Changes:** Change anomalies are caused by changes to the normal network activity, *e.g.*, due to unintentional BGP routing loops, prefix hijacking [ZZHM07], or (partial) network failures. Modeling of change anomalies requires thus flow-level models of modifications to existing flows. For example, a flow-level model for a prefix hijack would contain instructions for deleting flows based on the BGP prefixes of their destination IP addresses.

In this thesis, we concentrate on alien anomalies since modeling of such anomalies requires a careful analysis of malicious/disruptive network traffic characteristics at the flow level.

Modeling anomalies at the flow level is a challenging task. As a first challenge, it is necessary to identify the attributes that our models should capture. To keep the involved complexity
at a manageable level, we decided to include three groups of flow attributes in our models: i) IP address and port number sequences per flow\(^1\), ii) flow size (in bytes and packets) and flow duration (in milliseconds), and iii) flow inter-arrival times. We consciously disregard other flow attributes such as the IP type of service (TOS) or SNMP input/output interfaces as they are not as important for anomaly detection purposes.

A second challenge is to decide whether to use deterministic or probabilistic models. Several attributes such as bytes, packets, flow duration, and inter-arrival times are clearly of probabilistic nature and thus best captured by stochastic models. IP address and port sequences on the other hand, have deterministic, e.g., a scan with regular IP address selection scheme, as well as stochastic components. Therefore, our first attempt was to use Markov chains to model these attributes. However, we quickly discovered that a large number of anomalies resulted in chains with an excessive number of states. If we use stochastic models for individual attributes we also have to decide whether to capture dependencies between different attributes in the model or if we assume they are independent. In general, most attributes are independent. Merely, the number of bytes, packets, and the flow duration have clear dependencies. Hence, we include the correlation between those attributes in the model by conditioning the bytes and flow duration with respect to packets.

A third challenge stems from the fact that we derive our models from traffic traces that have been captured in a particular network, but we want to use them for injecting anomalies in traffic traces captured in other networks with different topologies, network address ranges and so on. Therefore, our goal is to build

\(^1\)To be network-transparent we use relative values rather than absolute values for these two attributes. For ports we use the port difference between consecutive flows. For IP addresses we first convert them to Integers using a function like `inet-aton` and then take the difference between Integer addresses of consecutive flows.
network-independent models. Additionally, we have only a partial view of the network, as we only observe flows that cross the border of SWITCH but we do not observe any internal or external flows. Consequently, we have to be extremely careful especially when extracting sequential distributions such as IP addresses, port numbers, or inter-arrival times as part of the traffic might be missing in our datasets.

The last challenge is to provide a suitable set of parameters so that users of our models can modify certain characteristics of the malicious/disruptive traffic. Users need to be able to adapt traffic characteristics to the target network in order to avoid injection artifacts. For example, network address ranges must be a configurable parameter. Additionally, to test the sensitivity of a detection method it is useful to provide a parameter that gives the user control over the intensity of a malicious/disruptive event. The intensity of an event as observed by an anomaly detection system is typically proportional to two parameters: the flow rate, \( i.e., \) the inter-arrival times, and the event duration, \( i.e., \) events that are much shorter than the time scale used by the detector are harder to detect.

5.2 Anomaly Characterization

We have extracted the flows caused by several malicious/disruptive events from three weeks of NetFlow traces captured in August 2007 at the SWITCH border routers. These events fall in three general classes of anomalies: network scans, spam, and Denial of Service attacks. Moreover, we have observed multiple instances of most events. In this section we present a detailed characterization of all relevant flow attributes for each event.\(^2\)

\(^2\)Unfortunately, we cannot analyze TCP flags since SWITCH border routers do not export them.
5.2.1 Network Scans

We have captured and analyzed the characteristics of five different network scans: the Nachi scan, an SSH scan, an Radmin scan, a DCOM/RPC scan, and a Netbios scan. Network scans are by far the most commonly observed class of anomalous activity [PYB+04, CJB08]. They typically originate at a single source IP address and target many different destinations. While the time of large worm outbreaks is passed, we still observe occasionally scanning activity related to worms such as Nachi, Blaster, or Welchia. Network scans may use TCP, UDP, or ICMP as transport protocol. Scans are of importance to network administrators when they originate from internal hosts, since this might be a sign of an infection on the affected machine.

Nachi Scan

We found in our traces several instances of an ICMP scan that can be attributed to the Nachi worm released in 2003. Nachi uses fixed size 92-byte ICMP echo packets to scan for vulnerable hosts and is thus easy to recognize. The Nachi scan flow attributes have the following characteristics.

Transport protocol: is set to ICMP.
Source IP addresses: are set to the address of the scanning host.
Source port numbers: are set to ICMP code 0 type 8, which corresponds to echo-request messages.
Destination port numbers: are set to 0 or not set.
Flow sizes: are set to 1 packet and 92 bytes.
Flow durations: are set to 0 msecs, which is the default for flows that contain 1 packet.
Destination IP addresses: show a periodic pattern. In Figure 5.1(a) we plot the difference in integer IP addresses between consecutive flows versus the flow index. The flows are sorted with
5.2 Anomaly Characterization

(a) Difference in Integer IP addresses between consecutive flows. If a flow has an IP address of 192.168.1.0 and the next flow has an IP address of 192.168.1.100 the Integer difference would be 100.

(b) Flow inter-arrival times in msec for 100 Nachi scan flows. Flows at regular intervals have an inter-arrival time of 61 msec while the remaining flows have an inter-arrival time of 0 or 1 msec.

Figure 5.1: Scanning behavior and flow inter-arrival times of the Nachi scan.
regard to their flow start times. We observe an interesting periodic pattern that resembles a fishbone with two different cycles. Every 200 flows we observe a few positive and negative shifts by 400 IP addresses. Moreover, every 800 flows we observe periods with shifts of 45 to 110 IP addresses. All remaining flows have IP address differences in the range [-40:40].

**Inter-arrival times:** show a periodic behavior that has a stochastic component. Figure 5.1(b) shows for different flows with index $i$ their inter-arrival time $t_i$ measured relative to the previous flow. In particular,

$$t_i = \begin{cases} 
[0, 1] \text{msec}, & \text{if } i \neq n \times 5, n \in \mathbb{N} \\
61 \text{msec}, & \text{if } i = n \times 5, n \in \mathbb{N}
\end{cases}$$

After five scans that arrive at the router with an inter-arrival time of 0 or 1 msec \(^3\), no flow is observed for 61 msec until the next scan period starts. Furthermore, we do not see any flows in reply to the Nachi scans in our traces as the targeted network is probably filtering these packets.

**SSH Scan**

Password-probing scans for SSH are quite common in today's networks. We have extracted two SSH scan instances from the SWITCH traces. Both instances show a very similar behavior.

**Transport protocol:** is set to TCP.

**Source IP addresses:** are set to the address of the scanning host.

**Destination port numbers:** are set to 22.

**Destination IP addresses:** show a very irregular pattern that contains many atomic scan periods of 30 to 200 flows that each cover a range of approximately 400 IP addresses. In Figure 5.2(a)\(^3\)

\(^3\)The maximum temporal resolution in our data are milliseconds.
5.2 Anomaly Characterization

(a) Histogram of the destination IP address difference between consecutive SSH scan flows within one atomic scan period.

(b) Flow inter-arrival times for 8000 SSH scan flows. We observe an increased inter-arrival time every 800 flows.

Figure 5.2: Scanning behavior and flow inter-arrival times of the SSH scan anomaly.
we plot the histogram of the difference in destination IP addresses between consecutive flows within such an atomic scan period. After each period a positive or negative shift by 200 to 400 IP addresses occurs.

**Source port numbers:** are selected randomly from the range [32,000:61,000].

**Flow sizes:** Each flow contains between 1 and 4 packets where approximately 80% of the flows have a size of 2 packets and 120 bytes \(^4\).

**Inter-arrival times:** show a periodic behavior with two cycles and a stochastic component. The inter-arrival times for 8000 SSH scan flows are plotted in Figure 5.2(b).

\[
t_i = \begin{cases} 
0 \text{ msec}, & \text{if } i \neq n \times 300, n \in \mathbb{N} \\
[10 - 50] \text{ msec}, & \text{if } i = n \times 300, n \in \mathbb{N} \\
5000 \text{ msec}, & \text{if } i = n \times 800, n \in \mathbb{N}
\end{cases}
\]

Every 800th flow has an inter-arrival time of 5 seconds, every 300th flow has an inter-arrival time between 10 and 50 msec, while all other flows have inter-arrival times of either 0 or 1 msec.

**Radmin Scan**

We have observed one instance of a scan on destination port number 4899. This port is used by the Radmin remote administration application. A remotely exploitable vulnerability in the Radmin server version 2.0 and 2.1 that allows for code execution was reported in July 2004. The Radmin scan has the following characteristics.

**Transport protocol:** is set to TCP.

**Source IP addresses:** are set to address of scanning host.

\(^4\)This percentage depends on the network that is scanned and thus we make it a configurable parameter in our model.
5.2 Anomaly Characterization

(a) Histogram of the difference in destination IP addresses between successive Radmin scan flows.

(b) Timing behavior of the Radmin scan. The plot shows the inter-arrival times for 400 flows.

**Figure 5.3:** Histogram of differences between consecutive IP addresses and inter-arrival times for the Radmin scan.
**Destination port numbers:** are set to 4899.

**Destination IP addresses:** show a highly irregular pattern. The difference between consecutive IP addresses of the majority of flows varies in the range [-40:40] as shown in the histogram given in Figure 5.3(a). In addition we observe positive or negative shifts that exhibit no particular patterns.

**Source port numbers:** also show a highly irregular pattern, but in addition they are limited to the interval [1,000:5,000]. The distribution of the difference between port numbers of consecutive flows is very similar to the distribution of IP address differences.

**Flow sizes:** 89% of all scan flows contain 2 packets and 96 bytes, while 10% of the scan flows contain 1 packet and 48 bytes.

**Flow durations:** 2-packet flows have a duration of either 46*64 msec or 47*64 msec $^5$.

**Inter-arrival times:** show a periodical behavior with two cycles and a stochastic component. The timing behavior of the Radmin scan anomaly is illustrated in Figure 5.3(b) that shows the inter-arrival times for 400 flows.

$$t_i = \begin{cases} [0, 1] \text{ msec}, & \text{if } i \neq n \ast [25, 4000], n \in \mathbb{N} \\ [25 - 30] \text{ msec}, & \text{if } i = n \ast 25, n \in \mathbb{N} \\ [1K - 2K] \text{ msec}, & \text{if } i = n \ast 4000, n \in \mathbb{N} \end{cases}$$

Every 25th flow is received with a delay 25 to 30 msec, and every 4000th flow is received with a delay of 1 to 2 seconds. All remaining flows have an inter-arrival time of either 0 or 1 msec.

**DCE-RPC Scan**

Destination port 135 is one of the top-scanned ports as various vulnerabilities have been reported in the RPC service running on this port. Also the famous Blaster worm used port 135 for

$^5$We reason that the flow duration granularity of 64 msec, which we observe in all multi-packet flows, is a NetFlow implementation artifact.
propagation. DCE-RPC flows have the following characteristics.

**Transport protocol**: is set to TCP.

**Source IP addresses**: are set to the address of the scanning host.

**Destination port numbers**: are set to 135.

**Flow sizes**: are set to 3 packets and 144 bytes as port 135 is open on most machines.

**Flow durations**: are set to either 19*64, 20*64, or 21*64 msec.

**Destination IP addresses**: do not show any regular patterns. The difference between successively scanned IP addresses varies in the range [-256:256]. Their distribution is depicted in Figure 5.4. Note that this distribution differs from the ones we have previously encountered. Additionally, we find random shifts at irregular times.

**Source port numbers**: are irregular as well, but in addition they are limited to the range starting at 1,200 and ending at 4,800. The range of variation between port numbers of successive flows is approximately [-200:200]. Again, the distribution for source port differences resembles the distribution of IP address differences. Additionally, we have a periodic component that introduces a positive shift of 250 to 500 source port numbers after 300 to 600 received flows and has the effect that certain port ranges are skipped.

**Inter-arrival times**: show a periodical behavior. The timing behavior of the RPC scan is periodical and has a stochastic component:

\[
t_i = \begin{cases} 
[0, 1] \text{ msec}, & \text{if } i \neq n \times 256, n \in \mathbb{N} \\
2000 \text{ msec}, & \text{if } i = n \times 256, n \in \mathbb{N}
\end{cases}
\]

Every 256th flow has a delay of 2 seconds, while all other flows have an inter-arrival time of either 0 or 1 msec.
Figure 5.4: Histogram of destination IP address differences between consecutive flows for the RCP scan anomaly.

Netbios Scan

We found two instances of scans for the netbios service that runs on UDP port 137 in our traces. Several vulnerabilities for the netbios service exist.

Transport protocol: is set to UDP.
Source IP addresses: are set to the address of the scanning host.
Destination port numbers: are set to 137.
Source port numbers: are set to a fixed value larger than 10,000.
Flow sizes: are set to 1 packet and 78 bytes.
Flow durations: are set to 0 msec.
Destination IP addresses: show a periodic behavior. The IP addresses for 100 to 200 flows are selected sequentially until a
5.2 Anomaly Characterization

(a) Target selection mechanism of the Netbios scan. The difference between consecutive IP addresses for 200 flows is shown.

(b) Timing behavior of the Netbios scan. The plot shows the inter-arrival times for 100 flows.

Figure 5.5: Target selection strategy and timing behavior of the Netbios scan.
negative shift of 60 to 70 IP addresses occurs. The sequential target selection behavior within each scan interval of the Netbios scan is plotted in Figure 5.5(a). Most of the time the scanner simply increases the destination IP address by one. However, from time to time we observe a positive shift of 2, i.e., one IP address is skipped, followed by a negative shift of 1, i.e., the missed IP address is scanned, followed by a positive shift of 2, i.e., the normal scanning continues.

**Inter-arrival times:** show a periodic behavior. The timing behavior of the Netbios scan is plotted in Figure 5.5(b).

\[
I_i = \begin{cases} 
[60 - 70] \text{msec}, & \text{if } i \neq n \times 5, n \in \mathbb{N} \\
0 \text{msec}, & \text{if } i = n \times 5, n \in \mathbb{N}
\end{cases}
\]

Every 5th scan has a delay of 0 msec, while all other scans have an inter-arrival time between 60 and 70 msec. Hence, this Netbios scan is considerably slower than the previously analyzed scans.

### 5.2.2 Spam

We did not find any anomalies related to e-mail spam such as massive spam campaigns caused by botnets in the three analyzed weeks of data. Instead, we detected several instances of Windows Messenger pop-up spam. We call them variant A and variant B. Windows Messenger Popup spam targets UDP destination ports 1026 and 1027. This type of anomalous behavior has also been reported in [PYB+04].

**Popup Spam Variant A**

**Transport protocol:** is set to UDP.

**Source IP addresses:** are set to the address of host that is sending the spam.

**Destination port numbers:** are set to 1026 or 1027.

**Flow sizes:** are set to 1 packet and 925 bytes.
5.2 Anomaly Characterization

(a) Histogram of IP address differences between consecutive flows of the Popup-Spam-A anomaly.

(b) Histogram of IP address differences between successive flows of the Popup-Spam-B anomaly.

Figure 5.6: Target selection strategy of the Popup-Spam-A and Popup-Spam-B anomaly.
Flow durations: are set to 0 msec.

Inter-arrival times: show a periodical behavior with two cycles and a stochastic component:

\[ t_i = \begin{cases} 
[0, 1] \text{ msec}, & \text{if } i \neq n \times [200, 550], n \in \mathbb{N} \\
64 \text{ msec}, & \text{if } i = n \times 200, n \in \mathbb{N} \\
250 \text{ msec}, & \text{if } i = n \times 550, n \in \mathbb{N}
\end{cases} \]

Approximately every 200th flow has a delay of 64 msec, and every 550th flow has a delay of 250 msec. The remaining flows have an inter-arrival time of either 0 or 1 msec.

Destination IP addresses: show no regular patterns. The difference distribution of variant A is shown in Figure 5.6(a). IP address difference values vary in the range [-200:200].

Source port numbers: Variant A selects the source port sequentially from the range [32,000:61,000]. The difference between source ports of consecutive flows varies in the range [-1,000:1,000] and resembles the distribution of IP address differences. Additionally we observe a periodical component: After 550 flows a positive shift of 2,000 source port numbers occurs.

Popup Spam Variant B

We only report the attributes that differ from the popup-spam variant A anomaly in the following.

Destination IP addresses: Variant B selects destination IP addresses more or less randomly from blocks of 3000 IP addresses according to the difference distribution given in Figure 5.6(b). In this distribution the spikes at multiples of 256 IP addresses are interesting. However, no regular pattern involving multiples of 256 IP addresses is visible. After approximately 300 flows the next block of IP addresses is used.

Source port numbers: Variant B uses a different mechanism for sequential port selection. It randomly chooses a source port
number to start with. After 550 flows have been sent with the same source port, it increases the port number by 1 to 4 ports.

### 5.2.3 Denial of Service Attacks

The third large group of anomalies that we have found are denial of service (DoS) attacks. DoS attacks have been extensively studied in previous work. Jung et al. [JKR02] analyze the properties that can be used to distinguish flash crowd events and DoS attacks for better protection. In [MR04] the authors provide a taxonomy of DDoS attacks and defense mechanisms. Methods for distinguishing single- and multi-source DoS attacks based on spectral analysis has been proposed in [HHP03]. We complement this work by providing a detailed analysis of the network behavior for different types of denial of service attacks such as UDP bandwidth flood or TCP SYN flood.

#### UDP Bandwidth Flood Variant A

We have found three instances of two one-to-one UDP bandwidth flood variants. Again, we call them variant A and variant B. In the following we report the characteristics for variant A.

**Transport protocol:** is set to UDP.

**Source IP addresses:** are set to the address of attacking host.

**Destination IP addresses:** are set to the address of the victim host.

**Flow sizes:** are set to 1 packet and 540 bytes for variant A.

**Source port numbers:** are selected uniformly from the range \([x.x + 19]\) where \(x\) is randomly chosen.

**Destination port numbers:** are selected sequentially between 20 and 1024. For each flow the destination port number is increased by 1 port every time a flow is sent.

**Inter-arrival times:** show a periodical behavior. The flow
inter-arrival time distribution of variant A is depicted in Figure 5.7(a).

\[ t_i = \begin{cases} 
  [0, 1] \text{ msec}, & \text{if } i \neq n \cdot 40, n \in \mathbb{N} \\
  [60, 120] \text{ msec}, & \text{if } i = n \cdot 40, n \in \mathbb{N}
\end{cases} \]

Every 40th flow has a delay of 60 or 120 msec, while the remaining flows have shorter inter-arrival times of 0 or 1 msec.

**UDP Bandwidth Flood Variant B**

Again, we report only differences to variant A of this attack.

**Flow sizes:** are set to 1 packet and 1028 bytes for variant B.

**Source port numbers:** are selected randomly from the interval [1:6,000].

**Destination port numbers:** are selected from the range [1,000:5,000]. The distribution of port number differences between consecutive flows is shown in Figure 5.7(b). The positive and negative difference values of 200 to 300 port numbers stem from the fact that two processes with smaller port differences run in parallel.

**Inter-arrival times:** show a periodical behavior. In particular,

\[ t_i = \begin{cases} 
  [0, 1] \text{ msec}, & \text{if } i \neq n \cdot 75, n \in \mathbb{N} \\
  60 \text{ msec}, & \text{if } i = n \cdot 75, n \in \mathbb{N}
\end{cases} \]

Every 75th flow is received with a delay of 60 msec, while all other flows have inter-arrival times of 0 or 1 msec.

**TCP Flood Variant A**

We have observed two instances of one-to-one TCP floods on destination port 80. Both attacks target the same web server.
5.2 Anomaly Characterization

(a) Inter-arrival times for 1,500 BWFlood-A flows.

(b) Histogram of IP address differences between consecutive flows for the BWFlood-B anomaly.

Figure 5.7: Timing behavior of the BWFlood-A anomaly and destination port selection of the BWFlood-B anomaly.
Transport protocol: is set to TCP.
Source IP addresses: are set to the address of the attacking host.
Destination IP addresses: are set to the address of the victim host.
Destination port numbers: are set to 80.
Flow sizes: are set to 3 packets and 128 bytes.
Flow durations: are set to either 11*64 or 12*64 msec.
Source port numbers: are selected from the interval [1,000:3,000] and the difference between consecutive flows shows the regular but rather complex pattern depicted in Figure 5.8(a).
Inter-arrival times: Every 10th flow of TCPFlood-A has a delay of either 60 or 120 msec, while all other flows are sent with an inter-arrival time of either 0 or 1 msec.

TCP Flood Variant B

Flow sizes: are set to 1 packet (26.4%) or 2 packets (73.6%).
Flow durations: 2-packet flows have lengths between 2*64 msec and 15*64 msec. 1-packet flows have a length of 0 msec.
Source port numbers: show no particular patterns and are selected from the interval [49,000:65,400]. The difference between consecutive flows has the distribution shown in Figure 5.8(b).
Inter-arrival times: Every 10th flow of TCPFlood-B has a delay between 21 and 35 msec, while the remaining flows are sent with inter-arrival times less or equal to 1 msec.

TCP Backscatter

We found 11 instances of TCP backscatter in the SWITCH traces. Backscatter flows are replies of a DoS victim that has been overflown by packets with spoofed source IP addresses. The replies of the victim are then routed towards the owner of the spoofed address space.
5.2 Anomaly Characterization

![Graph](image)

(a) Source port differences between consecutive flows of TCPFlood-A.

![Histogram](image)

(b) Histogram of source port differences between consecutive flows of TCPFlood-B.

Figure 5.8: Source port selection for the TCPFlood-A and TCPFlood-B anomaly.
Transport protocol: is set to TCP.
Source IP addresses: are set to the victim of the DoS attack.
Flow sizes: are set to 1 packet and 44 or 46 bytes.
Destination port numbers: are selected randomly from the interval [1,000:2,000] according to the distribution given in Figure 5.9.
Source IP addresses: show no regular pattern. The difference in source IP addresses between consecutive flows varies in the range [-600:600].
Inter-arrival times: show a periodical behavior with three cycles:

\[
t_i = \begin{cases} 
0 \text{ msec}, & \text{if } i \neq n \times [720, 3000, 8000], n \in \mathbb{N} \\
1 \text{ msec}, & \text{if } i = n \times 720, n \in \mathbb{N} \\
60 \text{ msec}, & \text{if } i = n \times 3000, n \in \mathbb{N} \\
380 \text{ msec}, & \text{if } i = n \times 8000, n \in \mathbb{N}
\end{cases}
\]

Approximately every 720th flows has an inter-arrival time of 1 msec, every 3000th flow has a delay of 60 msec, and every 8000th flow has a delay of 380 msec. The remaining flows have inter-arrival times of 0 msec.

5.3 Anomaly Models

Based on the anomaly characterization presented in the last section, we have derived 12 anomaly model templates. A template contains a generation function for each of the 10 flow attributes: transport protocol, source IP address, destination IP address, source port number, destination port number, packets per flow, bytes per flow, inter-arrival times, flow duration, and TCP flags.\(^6\)

\(^{6}\)As we cannot derive TCP flags from our data, we have to use heuristics based on our knowledge of the TCP protocol for modeling TCP flags.
5.3 Anomaly Models

5.3.1 Generation Functions

In the anomaly characterization we found three basic generation functions for flow attributes: constant, random, and periodical. These three functions completely cover the behavior of the characterized anomalies. In the following we describe each generation function and its parameters in detail and illustrate their usage with examples. To create more complex patterns, one can also combine different functions with if, then, else statements. Other anomalies might not be covered by the basic generation

Figure 5.9: Histogram of destination port differences for the TCP backscatter anomaly.
functions presented here and combinations thereof. Hence, our FLAME tool supports the addition of other generation functions if necessary.

**Constant:** The simplest generation function is the constant routine. It takes a single value \( s \) as parameter. Although simple, it is quite common for anomalous traffic to have fixed values for certain traffic features. For example, the transport protocol is typically constant and can, for example, be set to \( s = \text{UDP} \) or \( s = \text{TCP} \). Also the source IP address is constant for several anomalies, *e.g.*, scans or one-to-one floods, where \( s \) is set to the IP address of the attacking host.

**Random:** The random routine takes an empirical distribution \( d \) as parameter and selects values randomly according to the given distribution. A given empirical distribution can either contain absolute values or relative delta values. In the latter case, additionally a start value \( s \) needs to be specified. If one also wants to restrict the value range for relative distributions, a range \( r \) can be specified. The distribution \( d \) is given as a list of value-probability pairs \( d = [v_1 : p_1, \ldots, v_n : p_n] \). We have observed uniform random selection strategies for source and destination port numbers. Three out of the twelve observed anomalies randomly select the source port number from a uniform distribution with varying ranges, *e.g.*, \( r = [1 : 6,000] \) or \( r = [1 : 19] \). Relative empirical distributions are also used to model destination IP addresses of scanning and backscatter anomalies, or source IP addresses of spoofed denial of service attacks.

**Periodical:** The periodical routine is used to model traffic features that have certain periodicities based on the flow index. It takes a list of period-value pairs \( l = [p_1 : v_1, \ldots, p_n : v_n] \) as input and uses the first condition that matches for each
5.3 Anomaly Models

flow. For example, if we have \( I = [100 : 20] \) the corresponding attribute of every 100th flow is set to 20. Alternatively, one can specify relative values to periodically increase or decrease a flow attribute. We have observed periodical behavior for example for the inter-arrival times of all anomalies independent of the transport protocol used. The observed frequencies vary between \( p = 10 \) flows and \( p = 8000 \) flows, whereas inter-arrival times of \( v = 60 - 65 \) msec appear frequently. Moreover, we have observed periodically increasing/decreasing ranges for destination IP addresses of scanning anomalies (e.g., \( v = +2,000 \)).

5.3.2 Anomaly Rescaling

The intensity of an anomaly as perceived by a detection system depends on two parameters: the flow size and the duration of the malicious/disruptive event where the intensity increases with the flow rate and the anomaly duration. The flow size is not modified as it is an important anomaly characteristic (see Section 5.2 for examples). Additionally, it is also possible to increase the intensity of an event by injecting multiple attacks in parallel, e.g., scans from different sources. The duration of an anomaly is a generic user parameter that we describe in the next section where our FLAME tool is introduced. The flow rate, on the other hand, being proportional to flow inter-arrival times is a model parameter. The rescaling function thus depends on the generation function that is used for inter-arrival times in the respective model. Since all analyzed anomalies have periodic generation functions, we describe in the following a method for rescaling the periodic generation function.

\footnote{We only observe this behavior for traffic that originates from a single source, but not for flows from multiple sources. Hence, we conclude that this behavior is not related to the flow export mechanism, but is either imposed by the network stack or the attack tool that is used for generating the traffic.}
The periodic generation function has two parameters for each periodic component or cycle, the period $p$ and the value $v$, one can tweak to modify the timing behavior of an anomaly. To make an anomaly slower (faster) one can either increase (decrease) the inter-arrival time values $v$, decrease (increase) the periods $p$ of long inter-arrival times or increase (decrease) the periods $p$ of short periods $p$. Of course one can also modify multiple parameters at the same time.

We illustrate this effect in Figure 5.10 where we plot the time that it takes to send 1000 flows for three different settings of $p$ and $v$. All three plots have one static periodic component that is not modified: every 20th flow has an inter-arrival time of 1 msec, while all other flows have inter-arrival times of 0 msec. The second periodic component is modified as follows: In the reference plot $p$ is set to 100 and $v$ is set to 60, i.e., every 100th flow has an inter-arrival time of 60 msec. With these settings applied it takes 640 msec to send 1000 flows. In the second plot we set $p$ to 50 and keep $v$ at 60. As we can see in the Figure the time for sending 1000 flows doubles as the period $p$ is halved. Also in the third plot where $p$ is kept at 100 but $v$ is set to 120, i.e., the inter-arrival time value $v$ is doubled, the time for sending 1000 flows doubles.

### 5.3.3 Library of Anomaly Models

An overview of the anomaly model templates that we provide is given in Tables 5.1 and 5.2. For each template we give the generation function and its parameters for each of the 10 flow attributes. When setting the parameters, we have to settle a trade-off between user freedom and model accuracy. To do this we fix values that are characteristic for the anomaly and leave the remaining parameters user-configurable.

In Table 5.1 we give the transport protocol, source IP address, destination IP address, source port number, and destination port
5.3 Anomaly Models

Figure 5.10: Illustration of the impact of parameters \( p \) and \( v \) on the flow rate of an anomaly. We plot the time that is takes to send 1000 flows for different parameter settings.

number. The *transport protocol* is constant across all anomalies and the parameter of the constant function is pre-assigned. The *source IP address* is as well constant across all anomaly models. However, the parameter of the constant function is a user parameter. The *destination IP address* is either random or constant. In both cases, the parameters can be set by the user depending on the network ranges of the target network. For random generation functions a start value, an empirical distribution, and a range needs to be given. Additionally, we have irregular or periodical range shifts for some anomalies \(^8\). For the *source port number* we have a variety of generation functions. For ICMP anomalies the source port is fixed since it represents the ICMP code and type of the corresponding packet. Also for TCP backscatter we have

\(^8\)These are not given in the table due to space restrictions. See the FLAME documentation for further details.
fixed the source port since the observed traffic characteristics are particular to web traffic. Further, we have uniform random selection from a given range, where the range is either fixed or configurable. Finally, some models specify empirical delta distributions and additionally a value fixed range. The destination port number is fixed for all scan anomalies as these target particular protocols or applications. Furthermore, we have a random uniform generation function for the popup spam anomalies and the backscatter. The UDP flooding anomalies apply random selection from an empirical delta distribution where the distribution and the start value can be configured by the user.
<table>
<thead>
<tr>
<th>Name</th>
<th>Protocol</th>
<th>srcIP</th>
<th>dstIP</th>
<th>srcPort</th>
<th>dstPort</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Network Scans</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nachi</td>
<td>con[IMCP]</td>
<td>con[x]</td>
<td>ran[s,d,r]</td>
<td>con[8]</td>
<td>con[0]</td>
</tr>
<tr>
<td>SSH</td>
<td>con[TCP]</td>
<td>con[x]</td>
<td>ran[s,d,r]</td>
<td>ran[d]</td>
<td>con[22]</td>
</tr>
<tr>
<td>Radmin</td>
<td>con[TCP]</td>
<td>con[x]</td>
<td>ran[s,d,r]</td>
<td>ran[s,d,1K:5K]</td>
<td>con[4899]</td>
</tr>
<tr>
<td>DCOM/RPC</td>
<td>con[TCP]</td>
<td>con[x]</td>
<td>ran[s,d,r]</td>
<td>ran[s,d,1K:5K]</td>
<td>con[135]</td>
</tr>
<tr>
<td>Netbios</td>
<td>con[UDP]</td>
<td>con[x]</td>
<td>ran[s,d,r]</td>
<td>con[&gt;10K]</td>
<td>con[137]</td>
</tr>
<tr>
<td><strong>Spam</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Popup-Spam-A</td>
<td>con[UDP]</td>
<td>con[x]</td>
<td>ran[s,d,r]</td>
<td>ran[s,d,32K:61K]</td>
<td>ran[d]</td>
</tr>
<tr>
<td>Popup-Spam-B</td>
<td>con[UDP]</td>
<td>con[x]</td>
<td>ran[s,d,r]</td>
<td>ran[s,d,1:65K]</td>
<td>ran[d]</td>
</tr>
<tr>
<td><strong>Denial of Service</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BWFlood-A</td>
<td>con[UDP]</td>
<td>con[x]</td>
<td>con[y]</td>
<td>ran[s,d,1:19]</td>
<td>ran[s,d,20:1K]</td>
</tr>
<tr>
<td>BWFlood-B</td>
<td>con[UDP]</td>
<td>con[x]</td>
<td>con[y]</td>
<td>ran[d]</td>
<td>ran[s/s+200,d,1K:5K]</td>
</tr>
<tr>
<td>TCPFlood-A</td>
<td>con[TCP]</td>
<td>con[x]</td>
<td>con[y]</td>
<td>ran[s,d,1K:3K]</td>
<td>con[80]</td>
</tr>
<tr>
<td>TCPFlood-B</td>
<td>con[TCP]</td>
<td>con[x]</td>
<td>con[y]</td>
<td>ran[s,d,49K:65K]</td>
<td>con[80]</td>
</tr>
<tr>
<td>TCPBackscat</td>
<td>con[TCP]</td>
<td>con[x]</td>
<td>ran[s,d,r]</td>
<td>con[80]</td>
<td>ran[d]</td>
</tr>
</tbody>
</table>

Table 5.1: Overview of anomaly models extracted from three weeks of NetFlow traces captured in the SWITCH backbone network in August 2007. We provide the model (constant, random, or periodical) and parameter setting for 5 flow attributes. The remaining flow attributes are given in Table 5.2.
In Table 5.2 we give the number of packets per flow, number of bytes per flow, flow duration, inter-arrival times, and TCP flags where applicable.

The number of packets and bytes per flow, the flow duration, and the TCP flags are typically correlated. For several 1-packet anomalies, e.g., the Nachi and Netbios scan, the number of bytes per flow and the flow duration are fixed. For multi-packet anomalies the number of packets has a higher variation where the particular distribution of flow sizes depends on the distribution of hosts in the network, i.e., whether scan requests are filtered or not. The number of packets and bytes, and the flow durations of these anomalies are represented by random generation functions where the bytes and the flow duration are additionally conditioned on the flow size in packets. The same applies for TCP flags. The empirical distribution of the flow size and flow duration can be set by the user to match the target network. The flow inter-arrival times of all anomalies are represented by a combination of a periodical generation function and a random or constant generation function. The random/constant function is used when the periodical function does not apply for a flow, i.e., the flow is not a multiple of the period specified in the model.
### Table 5.2: Overview of anomaly models extracted from three weeks of NetFlow traces captured in the SWITCH backbone network in August 2007. We provide the model (constant, random, or periodical) and parameter setting for 5 flow attributes. The remaining attributes are given in Table 5.1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Pkts</th>
<th>Byts</th>
<th>dur</th>
<th>iar</th>
<th>flags</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Network Scans</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nachi</td>
<td>con[1]</td>
<td>con[92]</td>
<td>con[0]</td>
<td>per[5:61]/ran[d]</td>
<td>-</td>
</tr>
<tr>
<td>SSH</td>
<td>ran[d]</td>
<td>ran[d]</td>
<td>ran[d]</td>
<td>per[300:r,800:5K]/ran[d]</td>
<td>ran[d]</td>
</tr>
<tr>
<td>Radmin</td>
<td>ran[d]</td>
<td>ran[d]</td>
<td>ran[d]</td>
<td>per[25:r1,4K:r2]/ran[d]</td>
<td>ran[d]</td>
</tr>
<tr>
<td>Netbios</td>
<td>con[1]</td>
<td>con[78]</td>
<td>con[0]</td>
<td>per[5:r]/con[0]</td>
<td>-</td>
</tr>
<tr>
<td><strong>Spam</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Popup-Spam-A</td>
<td>con[1]</td>
<td>con[925]</td>
<td>con[0]</td>
<td>per[200:64,550:250]/ran[d]</td>
<td>-</td>
</tr>
<tr>
<td>Popup-Spam-B</td>
<td>con[1]</td>
<td>con[925]</td>
<td>con[0]</td>
<td>per[200:64,550:250]/ran[d]</td>
<td>-</td>
</tr>
<tr>
<td><strong>Denial of Service</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BWFlood-A</td>
<td>con[1]</td>
<td>con[540]</td>
<td>con[0]</td>
<td>per[40:60/120]/ran[d]</td>
<td>-</td>
</tr>
<tr>
<td>BWFlood-B</td>
<td>con[1]</td>
<td>con[1028]</td>
<td>con[0]</td>
<td>per[75:60]/ran[d]</td>
<td>-</td>
</tr>
<tr>
<td>TCPFlood-B</td>
<td>ran[d]</td>
<td>ran[d]</td>
<td>ran[d]</td>
<td>per[10:r]/ran[d]</td>
<td>ran[d]</td>
</tr>
<tr>
<td>TCPBackscat</td>
<td>con[1]</td>
<td>ran[d]</td>
<td>con[0]</td>
<td>per[720:1:3K:60,8K:380]/con[0]</td>
<td>con[ACK]</td>
</tr>
</tbody>
</table>
5.4 FLAME

FLAME (short for Flow-Level Anomaly Modeling Engine) is a framework that provides facilities for testing anomaly detection systems. Its core functionality is the generation and modification of flows according to given models, e.g., the 12 anomaly models described above. Furthermore, it provides the means for reading and writing flows in different formats from files or sockets, and for merging multiple flow streams.

The envisioned usage of FLAME for anomaly detection system testing is illustrated in Figure 5.11. An injection schedule or script defines the anomalies, i.e., the templates and their parameterization including start time and duration that are to be injected into background traffic. The background traffic can be read from sockets, e.g., if generated with the D-ITG traffic generator [BDP07], or from captured files. An anomaly detection system that is fed with the flow traces modified by FLAME generates alarms for detected anomalous activities. For evaluating the performance of the detector, one compares the injection schedule with the actual detected anomalies to identify anomalies that have been missed. In this way one is able to obtain an estimate of the false negative rate for a given detector. For synthetic
background traffic, an estimate of the false positive rate can be obtained in addition.

The FLAME framework consists of four modules that are implemented as C++ components and communicate via named pipes. The flow input/output modules convert flow streams in standardized flow formats such as NetFlow or IPFIX to an internal flow format and vice-versa. All other modules use only the internal flow format that is based on NetFlow v5. The flow generation module takes an anomaly model as input and generates flows according to the generation functions and parameters specified in the model. Similarly, the flow modification module takes a modification model and changes all received flows according to the modification routines given in the model. Finally, the flow merging module takes multiple flow streams as input and joins them into a single stream. In the following we describe each FLAME module in more detail 9.

5.4.1 Flow Input and Output

Interfaces are not the most interesting part of a software, but they are essential. In its latest version (2.2) the FLAME prototype supports the following three flow formats: NetFlow version 5, NetFlow version 9, and IPFIX. Furthermore, it supports two input/output options. Flat files are practical for batch-processing of stored data. For demonstrations sockets can be used to allow for more interactive scenarios with live streaming of flows from a network. FLAME uses an efficient data structure for communication between modules that captures only per-flow information that is needed by all modules. This data structure contains ten fields, e.g., source and destination IP address and port, and is based on the NetFlow v5 flow record format. The corresponding fields in NetFlow v5, NetFlow v9, and IPFIX format are listed in

9The maximum resolution provided by FLAME are milliseconds.
the Appendix. The libipfix library by Fraunhofer FOKUS provides a complete IPFIX implementation, but this is actually not required for the limited functionality that is used by FLAME. However, the interoperability of FLAME with the libipfix library applied in the OpenIMP flow meter has been extensively tested within the INTERSECTION EU project [INT].

**NetFlow version 5**

A NetFlow version 5 export packet contains a header that is followed by up to 30 flow records. The header provides additional information such as the NetFlow version number, the SysUptime, and a flow sequence number. The header information of the first received packet is relevant for the writer module in order to obtain a consistent output stream. Thus the initial header information is communicated through the module chain as bulk data to initialize the flow sequence number and the SysUptime in the writer module. All following NetFlow v5 headers are not communicated through the chain but are re-generated by the writer module.

The processing for NetFlow v5 flow records is straightforward as they have a pre-defined structure. At the beginning of the module chain a reader module converts the NetFlow v5 flow records to the internal format, and at the end of the module chain a writer converts the records in the internal format back to NetFlow v5 flow records.

**NetFlow version 9 and IPFIX**

NetFlow version 9 defined in RFC 3954 [Cla04] and IPFIX defined in RFC 5102 [QBC+08] have a more flexible flow format than NetFlow version 5. A NetFlow Version 9 export packet consists of a packet header followed by at least one or more template FlowSet or data FlowSet. A template FlowSet provides
a description of the fields that will be present in future data FlowSets. Data FlowSets contain the actual flow records we are interested in. IPFIX, as it is based on NetFlow v9, has a very similar structure but uses a different terminology. An IPFIX Message consists of a Message Header, followed by one or more Sets, where Sets are either Data Sets, Template Sets, or Options Template Sets.

The FLAME reader module processes NetFlow v9 and IPFIX data template sets and data sets. IPFIX options template sets are currently ignored by FLAME. In particular, it stores received template sets in a hash table. Upon receipt of a data set, FLAME searches the hash table for the corresponding template set and uses the information provided by the template set to extract the ten fields that it needs for the internal flow record format from the received data set. All other fields possibly contained in the data set are ignored by FLAME. If the corresponding template set is not found, the data set is dropped. Moreover, data sets that do not contain the required ten fields are ignored by FLAME as well.

As for NetFlow v5, the initial header information is passed as bulk data to the writer module for initialization purposes. The writer module receives flow records in the internal format and converts them back to NetFlow v9 or IPFIX format. However, it replaces the original template with its own version (templateID is 2222) that is inspired by NetFlow v5. The data templates used by FLAME for NetFlow v9 and IPFIX are given in the Appendix.

### 5.4.2 Flow Generation

The flow generation module is one of the core modules of FLAME. It takes an anomaly generation model that specifies the generation functions (GF) and their parameters for all ten flow features as input and outputs flow records in the FLAME-internal format. Thus, it implements the three generation functions (con-
stant, random, and periodical) that have been described in the last section.

The flow generation module first parses a given anomaly model and then - if no errors occur - it generates flows according to the model until a stop conditions is met. Generated flows are inserted in a sorted list, and exported with increasing flow end timestamps. This procedure was introduced to mimic the flow export behavior of routers. In the following we describe in detail the flow generation model format.

Generation Model Format

Generation models are text files that must follow a certain format so that they can be parsed by the flow generation module. In each file the generation functions for all ten flow features are defined. Each line in the generation model text file contains a key:value pair. A list of possible keys is given in Table 5.3. The TCPflags key is not required for protocols other than TCP. All remaining keys are mandatory.

<table>
<thead>
<tr>
<th>key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sourceIPAddress</td>
<td>Source IP address GF</td>
</tr>
<tr>
<td>destIPAddress</td>
<td>Destination IP address GF</td>
</tr>
<tr>
<td>protocol</td>
<td>Protocol GF</td>
</tr>
<tr>
<td>sourcePort</td>
<td>Source port GF</td>
</tr>
<tr>
<td>destPort</td>
<td>Destination port GF</td>
</tr>
<tr>
<td>inter-arrival-msec</td>
<td>Inter-arrival time GF</td>
</tr>
<tr>
<td>packets</td>
<td>Number of packets GF</td>
</tr>
<tr>
<td>octets</td>
<td>Number of bytes GF</td>
</tr>
<tr>
<td>flowDur-msec</td>
<td>Flow duration GF</td>
</tr>
<tr>
<td>TCPflags</td>
<td>TCP flags GF</td>
</tr>
</tbody>
</table>

Table 5.3: List of available keys to be used in generation and modification models.
As values the generation functions and their parameters are given. The following generation functions are accepted as values: constant($s$), random($p$), periodical($l$). The GF parameters have to adhere to the format as defined in Section 5.3.1. Instead of specifying distributions inline, they can be defined in separate files and a reference to the file can be given as parameter to the generation function.

Additionally, different GF can be combined with if-then-else statements. Two possible constructs can follow if-clauses:

- A GF does not result in a match:
  if not [GF] (e.g., if not periodical($l$)),

- Another key has a given value:
  if [key] equal [value] (e.g., if packets equal 2).

Then- and else-statements must always be followed by a GF. For example, if not periodical($l$) then random($p$) selects a feature value randomly if the periodical GF did not match. Moreover, multiple if clauses can be combined with or/and connectors. For example, if packets equal 2 and/or bytes equal 128 then random($p$) selects the value randomly from $p$ if the packets flow field has been set to 2 and/or the bytes flow field has been set to 128.

### 5.4.3 Flow Modification

The flow modification module can be used to modify certain fields of a flow, to delete entire flows, or to split a flow stream based on certain criteria. Thus, it can be applied to implement sampling functions, e.g., where every flow gets deleted with a certain probability, or to model routing and outage anomalies where flows that match a given prefix are either deleted or shifted. The flow modification module can have multiple output interfaces.
The modification module takes a modification model as input that specifies the required modification actions. If a flow is to be deleted the flow modification module simply drops the flow and continues with the next flow. If a flow is to be modified, the respective fields are re-written according to the model and the flow record is passed to the next module in the chain.

**Modification Model Format**

Modification models are text files that contain one modification action per line. There is no limit on the number of modification actions in a model. Modification actions are given as if-then-else statements.

If-statements can either be applied to any of the keys given in Table 5.3 or any of the available generation functions (GF). The structure of if-statements is defined as: if [key/GF] [operator] [value]. Possible operators are equal, less than, less equal, larger than, and larger equal. For example, the statement

```
if random(0:1) less than 0.1
```

could be used randomly select flows with a probability of 10%. Again, it is possible to connect multiple conditions with and/or. For example, the if-clause

```
if sourceIPaddress larger equal x and sourceIPaddress less equal y
```

matches all flows that have a source IP address in the given range.

We define three different types of then-statements:

- Modification of flow fields:
  - then [key] equal [value] (e.g., then packets equal 3),

- Deletion of flow records:
  - then delete-flow

- Splitting of flow records:
  - then output[1-n]
5.4 FLAME

Multiple modifications can be triggered by combining assignments with the and connector, e.g., \texttt{then packets equal 3 and bytes equal 128} updates both the packet and byte field.

5.4.4 Flow Merging

The flow merging module has multiple input interfaces and a single output interface. It can for example be used to merge the flows that have been exported from different routers or to merge a background trace with flows exported by the flow generation module into a single stream.

It takes flows in the FLAME-internal format from \( n \) inputs, merges them based on flow end timestamps, and outputs the sorted flow records. It uses one queue per input stream and selects in each round the flow record with the smallest flow end timestamp. Thus, the flow merging module assumes that the flows from each input stream arrive in order of increasing flow end timestamps.

5.4.5 Injection Scenario

Different use cases for FLAME require different module chains. In Figure 5.12 we show the module setup for an injection scenario: Two anomalies are generated and injected into a third flow stream that is read either from file or socket in IPFIX format. Additionally, one could add a modification module after the merging module to inject a prefix hijacking anomaly or an outage anomaly. Finally, the merged flow stream is written in IPFIX format to a file or socket.

The injection schedule for this example scenario could look like this:

- \texttt{FlowGenerator -s now+60 -d 600 RPC-scan1.txt}
- \texttt{FlowGenerator -s now+120 -f 100'000 RPC-scan2.txt}
Each flow generation module takes as input the text file that contains the model. In the example, we inject two instances of an RCP scan anomaly that are defined in the given files.

Furthermore, FLAME provides three command line options for specifying additional information that is not captured in the particular model such as the start time and the duration. The possible options and values are listed in Table 5.4.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>-s</td>
<td>Start time in UTC</td>
<td>now, now + offset in sec, yyyy-mm-dd hh:ss</td>
</tr>
<tr>
<td>-d</td>
<td>Duration in sec</td>
<td>any positive number</td>
</tr>
<tr>
<td>-f</td>
<td>Maximum flow number</td>
<td>any positive number</td>
</tr>
</tbody>
</table>

Table 5.4: Options for specifying additional information (start time and duration) for anomaly injection.

The -s option defines the start time. It can take either a
5.5 Discussion

Finally, we want to assess our work on FLAME and anomaly models with respect to two criteria: model versatility and injection artifacts. Model versatility analyzes whether our modeling approach is able to correctly present a large body of anomalies. With respect to injection artifacts we assess whether a detector is able to tell that an observed anomaly is not genuine but has been generated artificially and injected subsequently.

5.5.1 Model Versatility

We demonstrate the versatility of our modeling approach through application to various network traffic anomalies. A classification of network anomalies has been provided in [LCD04c]. This list contains a total of 8 different anomaly types. In the following, we describe for each of these eight anomaly types how they could be generated using the FLAME framework.

**Alpha flows** are caused by an unusually high rate point to point byte transfer, which can for example be caused by band-
width measurement experiments. Alpha flows can be represented in FLAME using two generation models, one for the source and one for the destination. The destination needs a separate generation model since the reply traffic generated by the destination might as well be massive and thus lead to an alert.

**DoS and DDoS attacks** are caused by a single or multiple sources that send large amounts of traffic, in terms of packets and/or bytes to a single victim. We have presented the generation models for several denial of service attacks with and without spoofing. We have also presented a model for backscatter traffic that is a secondary effect caused by a denial of service attack. Distributed DoS attacks can be generated with FLAME by using a DoS generation model that selects the source IP address from a given list of attackers.

**Flash crowd events** are caused by an unusually large demand for a resource or service, *e.g.*, a web server that sells football tickets. As flash crowds are very similar to distributed denial of service attacks, they are also represented in FLAME in a similar fashion. One could use a web traffic generation model that selects the source IP address from a given list of clients. Multiple such instances that are started at different points in time could be used to model increasing or decreasing user demands.

**Network scans** are caused by a single host that scans the network for a target port. We provide five scan models (Nachi, SSH, Radmin, RCP, and Netbios) with FLAME. These models select the destination IP address sequentially, but random target selection is possible as well.

**Worms** are due to self-propagating code that spreads across a network by exploiting security flaws. Every newly infected host starts to scan for other victims. Therefore, worms behave much like distributed scans from a network traffic perspective. To model worms in FLAME, one could use a scan model and modify the source IP generation function in a way that it selects sources randomly from a given range.
Point to Multipoint anomalies are due to content distribution from one server to many users. They can be generated in FLAME using a model that selects the destination IP address periodically from a given list of target systems. Furthermore, a sequential generation function can be used for modeling the inter-arrival times for multiplexed traffic.

Outage events are caused by scheduled router/server down-times, or by failures in the measurement infrastructure. This type of anomalies can be injected with FLAME using a modification model that for example deletes all flows that have a certain IP address or IP address range, and that arrive in a given time window where the outage event is supposed to take place.

Ingress shifts are caused by traffic engineering where a network provider shifts traffic from one ingress point to another ingress point. To model this type of anomaly one could read the flow streams exported at two ingress points in parallel with FLAME, use a modification model to split the flows that are to be shifted from the first stream (e.g., based on source IP prefixes) and merge them into the second stream.

5.5.2 Modeling and Injection Artifacts

Artifacts are errors or modifications introduced by FLAME that allow a networking expert to identify the flow records that have been altered subsequently. Artifacts are unwanted as they can be misused for detecting anomalies injected by FLAME and for manipulating detection results by using the artifact as basis for detection.

There are two processes that can cause such errors, the model building process and the injection process itself. When generating anomaly models from observed traffic, one has to be very cautious not to include features that are specific to the observed trace into a model. The best strategy would be of course to use multiple traces that were captured at different observation points and at
different times as basis for building an anomaly model. Likewise, when one injects anomalies into a given trace one has to assure that the injected flows match the particular characteristics of the background trace.

First of all the model designer has to be aware that each router has a limited view that is imposed by routing policies, i.e., only certain prefixes are routed via a particular router, and the total number of prefixes that are owned by the network operator. Typically, prefixes are assigned in continuous blocks. Therefore, a router might only observe an anomaly partially if it extends to prefixes not owned by the network operator. Extracting correct models for largely distributed events that span many prefixes such as scans or DDoS attacks requires thus traces from routers with a broad view, e.g., border routers of medium or large backbone networks. When injecting traffic for such events into a particular trace on the other hand, we have to consider the prefixes that are available in the given network. The generation functions used in FLAME provide a range parameter that allows for specifying prefixes that are valid for generating source or destination IP addresses.

Another source of modeling artifacts are certain traffic characteristics that are caused by specific settings of the router that has exported the flows used for modeling. For example, a router that applies packet sampling drops a certain percentage of 1-packet flows completely and does not account for some packets of longer flows. This alters the traffic characteristics considerably. Accounting for this loss in the model design process is a challenging issue. Fortunately, the SWITCH traces that we have used to extract our models are unsampled. On the other hand, injecting anomalies into a packet sampled trace is less of a problem as packet sampling can be implemented using the FLAME modification module.

Furthermore, there are implementation differences among router manufacturers such as Cisco, Juniper, Huawei, and even
among models from of the same manufacturer. For example, the implementation of ICMP traffic handling varies greatly from model to model although there exists a recommendation for implementing it correctly [Cla04]. Another example are the flow export artifacts discovered by Cunha et al. [tCSO+09]. The authors found that Juniper’s flow export tool J-Flow introduces periodic patterns into flow traces. When injecting anomalies into flow traces that have been generated by a router with known implementation artifacts one has to assure that the traffic generated by FLAME matches this behavior. If we would not include the artifacts then flows injected by FLAME would be very easy to detect. This is of course not an issue for simulated or emulated background traffic that does not have such artifacts. Currently, the router adaptation has to be included in each anomaly model. A nice extension of FLAME would be to separate the implementation issues from the model layer.

5.6 Summary

In this chapter, we have presented the FLAME tool that can be used for evaluating anomaly detection systems operating on flow data. FLAME comes with a library of parameterizable anomaly model templates that have been derived from real-world datasets. One interesting feature of FLAME is that it can be used to inject anomalies of different scale for testing the sensitivity of anomaly detectors with respect to the strength of an anomalous event.

Moreover, FLAME provides several basic generation functions that can be assembled to form novel anomaly templates. This makes the FLAME tool easily extensible. We have shown that our flow-based modeling approach is very flexible and covers well-known classes of anomalies. Further, we provided a discussion on injection artifacts and how they can be avoided in the model design process and in the anomaly injection process itself.
Finally, the utility of FLAME has been verified by the various research and industry projects that have used it to test and improve their anomaly detection systems.
Chapter 6

Conclusions and Future Work

We conclude our work on anomaly detection systems and evaluation methods in this chapter. We first give a review of the main contributions made in this thesis. Then, we mention possible shortcomings and weaknesses of our work. Finally, we identify and discuss open research issues in the field of anomaly detection that deserve further attention.

6.1 Review of Contributions

In this thesis, we make three core contributions that we will discuss in the following.

Application of Dimension Reduction Techniques for Anomaly Detection

The first part of our work was devoted to the problem of how to apply dimension reduction techniques to the anomaly detection problem. We studied two different dimension reduc-
Anomaly Extraction in Backbone Networks using Association Rules

In the second part of our work we went beyond the limits of anomaly detection and formulated the anomaly extraction problem that is essentially an intermediate step toward the ultimate goal of root-cause identification. We basically asked ourselves how we can support network administrators in their daily task of making sense out of imprecise and vague anomaly detection alerts. We developed a three-stage approach that first uses multiple histogram-based detectors to identify anomalies in traffic data and derive meta-data that identifies suspicious flows, second we filter all flows that match the union of meta-data provided by several detectors, and thirdly we apply association rule mining to summarize sets of anomalous flows and provide a concise anomaly report to the network administrator. In particular, we have introduced the concept of histogram cloning to obtain multiple views...
on a single traffic stream. Moreover, we have studied how voting among different clones can improve overall anomaly extraction results. While association rules have been applied to many problems in networking before we are the first to use them for the purpose of anomaly extraction. We have shown on real-world datasets that our approach can effectively isolate and summarize anomalous flow from real-world backbone datasets with low false positive rates.

Anomaly Modeling, Generation, and Injection

The third part of our work was dedicated to the evaluation problem. We developed the FLAME tool that provides the functionality for generating and injecting anomalous traffic into background traffic. FLAME has been used by several researchers for testing and improving their anomaly detection systems. It has also been employed in the INTERSECTION project that is co-funded by the European Commission in the context of the Seventh Framework Program. At the heart of FLAME is an anomaly model library containing several anomaly templates for network scans, denial of service attacks, and spam, as well as a set of basic generation functions that can be applied to easily construct additional anomaly templates. These models and generation functions have been established based on numerous instances of anomalies that have been extracted from three weeks of backbone flow traces captured on the SWITCH border routers. To evaluate our modeling approach we have assessed its versatility and shown that it accommodates the various types of anomalies that have been reported in previous work. Finally, we have discussed the potential for artifacts caused by FLAME and provided guidelines on how they can be avoided in the modeling and injection process.
6.2 Critical Assessment

The goal of this thesis was to address three eminent problems that hinder the deployment of anomaly detection systems in practice: the high dimensionality of data, the difficulty of root-cause identification for once detected anomalies, and the paucity of evaluation methods and tools. In the following we will assess to which extent we have reached these challenging objectives.

In the first part of this thesis we have provided an extensive discussion on the pitfalls of applying dimension reduction techniques to the anomaly detection problem, and subsequently have proposed an improved detection method. Nevertheless, there exist other open challenges related to the high dimensionality of network data that have not been addressed in this thesis. In particular, the irreversible information loss introduced by packet sampling remains an important issue that has yet to be solved by the research community.

The second part of this thesis introduced anomaly extraction as an intermediate step toward the ultimate goal of root-cause identification. Although we have explicitly illustrated that our proposed methodology can greatly reduce the manual effort that is needed for root-cause identification, a human in the loop is still needed to interpret the concise report that is generated in an automated fashion with our approach. However, as false positives in the report are quite easy to spot for an experienced network administrator, an interesting extension of our work would be to implement a learning-based mechanism for exactly this purpose.

In the third part of this thesis we have developed a tool for injecting anomalous traffic into present background traffic that comes with an extensible anomaly model library and a set of parameterizable generation functions. Although the anomaly injection approach is very helpful for the difficult task of evaluating the rate of false positives/negatives generated by an anomaly detection system, it does not solve the complementary and likewise
demanding problem of assigning labels to captured traffic traces. Furthermore, two open issues are the separation of implementation details from the model layer and the generation of models from sampled flow data.

6.3 Future Work

The field of flow-based anomaly detection is still in its beginnings. Two issues that should receive attention from the research community are discussed in the following.

Up to now, numerous anomaly detection systems and mechanisms have been proposed and evaluated in isolation. However, to really compare and understand the shortcomings and advantages of the different approaches standardized benchmark datasets and evaluation methods as for example those available in the more mature field of database research are required. The anomaly model library we have developed in this thesis provides only the first ingredient to such benchmark traces. To obtain the second ingredient, \textit{i.e.}, the background traffic, we need improved methods for generating realistic benign network traffic at large scale. Until these are available we have to rely on captured and manually labeled network traffic traces. In order to guarantee at least a minimum level of transparency, standardized rules or directives for manual labeling should be introduced.

Many of the proposed detection systems make two assumptions that are actually unlikely to hold in practice. They rely on anomaly-free datasets for training purposes and they imply stationarity of the timeseries used for detection. Future work should be considered with the development of methods that are tolerant to i) non-stationarities of timeseries during operation of the anomaly detection system, and ii) anomalies present in the data used for training the system. This is an essential prerequisite for the acceptance of anomaly detection systems in practice.
Appendix A

Appendix

A.1 NetFlow and IPFIX Conversion

The mapping for the 10 flow features captured in the FLAME-internal flow record format and NetFlow v5, NetFlow v9, and IPFIX format is provided in Table A.1. NetFlow version 9 is defined in RFC 3954 [Cla04]. The IPFIX field specifiers are defined in RFC 5102 [QBC+08]. NetFlow version 5 is a Cisco standard and not defined in a separate RFC.
<table>
<thead>
<tr>
<th>Description</th>
<th>FLAME</th>
<th>NF v5</th>
<th>NF v9</th>
<th>IPFIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source IP address</td>
<td>addr</td>
<td>srcaddr</td>
<td>IPV4_SRC_ADDR</td>
<td>sourceIPv4Address</td>
</tr>
<tr>
<td>Dest. IP address</td>
<td>dstAddr</td>
<td>dstaddr</td>
<td>IPV4_DST_ADDR</td>
<td>destinationIPv4Address</td>
</tr>
<tr>
<td>Source Port</td>
<td>port</td>
<td>sraddr</td>
<td>L4_SRC_PORT</td>
<td>sourceTransportPort</td>
</tr>
<tr>
<td>Destination Port</td>
<td>dstPort</td>
<td>dstport</td>
<td>L4_DST_PORT</td>
<td>destinationTransportPort</td>
</tr>
<tr>
<td>Layer 4 Protocol</td>
<td>prot</td>
<td>prot</td>
<td>PROTOCOL</td>
<td>protocolIdentifier</td>
</tr>
<tr>
<td>Number of packets</td>
<td>dPkts</td>
<td>dPkts</td>
<td>IN_PKTS</td>
<td>packetDeltaCount</td>
</tr>
<tr>
<td>Number of bytes</td>
<td>dOctets</td>
<td>dOctets</td>
<td>INBYTES</td>
<td>octetDeltaCount</td>
</tr>
<tr>
<td>Flow start (ms)</td>
<td>first</td>
<td>f(first)</td>
<td>f(LAST_SWITCHED)</td>
<td>flowStartMilliseconds</td>
</tr>
<tr>
<td>Flow end (ms)</td>
<td>last</td>
<td>f(last)</td>
<td>f(FIRST_SWITCHED)</td>
<td>flowEndMilliseconds</td>
</tr>
<tr>
<td>TCP flags</td>
<td>tcpFlags</td>
<td>tcp_flags</td>
<td>TCP_FLAGS</td>
<td>tcpControlBits</td>
</tr>
</tbody>
</table>

**Table A.1:** Mapping between flow information stored in internal FLAME format and NetFlow/IP-FIX flow formats.
A.2 FLAME NetFlowv9/IPFIX DataTemplates

The data template that is used by FLAME for exporting NetFlow v9 data is given in Table A.2. The data template that is used by FLAME for exporting IPFIX data is given in Table A.3.

<table>
<thead>
<tr>
<th>Field ID</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlowSet ID</td>
<td>0</td>
<td>48</td>
<td>IPv4 source address</td>
</tr>
<tr>
<td>Length</td>
<td>2222</td>
<td>4</td>
<td>IPv4 destination address</td>
</tr>
<tr>
<td>Field Count</td>
<td>10</td>
<td>2</td>
<td>L4 source port</td>
</tr>
<tr>
<td>Field 1 Type</td>
<td>8</td>
<td>4</td>
<td>L4 destination port</td>
</tr>
<tr>
<td>Field 1 Length</td>
<td>12</td>
<td>2</td>
<td>L4 source port</td>
</tr>
<tr>
<td>Field 2 Type</td>
<td>11</td>
<td>4</td>
<td>L4 destination port</td>
</tr>
<tr>
<td>Field 2 Length</td>
<td>22</td>
<td>1</td>
<td>L4 protocol</td>
</tr>
<tr>
<td>Field 3 Type</td>
<td>4</td>
<td>1</td>
<td>Number of packets in flow</td>
</tr>
<tr>
<td>Field 3 Length</td>
<td>2</td>
<td>1</td>
<td>Number of bytes in flow</td>
</tr>
<tr>
<td>Field 4 Type</td>
<td>2</td>
<td>1</td>
<td>System uptime at start of flow</td>
</tr>
<tr>
<td>Field 4 Length</td>
<td>21</td>
<td>1</td>
<td>System uptime at end of flow</td>
</tr>
<tr>
<td>Field 5 Type</td>
<td>6</td>
<td>2</td>
<td>TCP flags</td>
</tr>
<tr>
<td>Field 5 Length</td>
<td>22</td>
<td>1</td>
<td>TCP flags</td>
</tr>
</tbody>
</table>

Table A.2: NetFlow v9 data template used by FLAME.
<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set ID</td>
<td>2</td>
</tr>
<tr>
<td>Length</td>
<td>48</td>
</tr>
<tr>
<td>Template ID</td>
<td>2222</td>
</tr>
<tr>
<td>Field Count</td>
<td>10</td>
</tr>
<tr>
<td>sourceIPv4Address</td>
<td>8</td>
</tr>
<tr>
<td>Field Length</td>
<td>4</td>
</tr>
<tr>
<td>destinationIPv4Address</td>
<td>12</td>
</tr>
<tr>
<td>Field Length</td>
<td>4</td>
</tr>
<tr>
<td>sourceTransportPort</td>
<td>7</td>
</tr>
<tr>
<td>Field Length</td>
<td>2</td>
</tr>
<tr>
<td>destinationTransportPort</td>
<td>11</td>
</tr>
<tr>
<td>Field Length</td>
<td>2</td>
</tr>
<tr>
<td>protocolIdentifier</td>
<td>4</td>
</tr>
<tr>
<td>Field Length</td>
<td>1</td>
</tr>
<tr>
<td>inPacketDeltaCount</td>
<td>2</td>
</tr>
<tr>
<td>Field Length</td>
<td>8</td>
</tr>
<tr>
<td>inOctetDeltaCount</td>
<td>1</td>
</tr>
<tr>
<td>Field Length</td>
<td>8</td>
</tr>
<tr>
<td>flowStartMilliseconds</td>
<td>152</td>
</tr>
<tr>
<td>Field Length</td>
<td>8</td>
</tr>
<tr>
<td>flowEndMilliseconds</td>
<td>153</td>
</tr>
<tr>
<td>Field Length</td>
<td>8</td>
</tr>
<tr>
<td>tcpControlBits</td>
<td>6</td>
</tr>
<tr>
<td>Field Length</td>
<td>1</td>
</tr>
</tbody>
</table>

Table A.3: IPFIX data template used by FLAME.
Bibliography


[Cisb] Cisco. Netflow services and applications - white paper.


[GTB07] Carrie Gates, Carol Taylor, and Matt Bishop. Dependable security: testing network intrusion de-


168 Bibliography

(SIGCOMM conference on Internet measurement, pages 147–152, New York, NY, USA, 2006. ACM.


[LFG+00] Richard Lippmann, David Fried, Isaac Graf, Joshua Haines, Kristopher Kendall, David McClung, Dan Weber, Seth Webster, Dan Wyschogrod, Robert Cunningham, and Marc Zissman. Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. In DARPA Information Survivability Con-


[SYB04] Joel Sommers, Vinod Yegneswaran, and Paul Barford. A framework for malicious workload genera-


Acknowledgments

I thank Prof. Bernhard Plattner for enabling and supporting this thesis and for providing the freedom to explore a variety of interesting research fields to his PhD students. I am grateful to Matthias Bossardt, Lukas Ruf, and Thomas Diübendorfer for sparkling my interest in scientific research. I thank my supervisors Kave Salamatian, Martin May, Xenofontas Dimitropoulos, and Arno Wagner for inspiring discussions and interesting collaborations. I thank Andreas Kind (IBM Research) for being the co-advisor of this thesis, Marc Stöcklin (IBM Research) for interesting discussions on histogram-based anomaly detectors, Prof. Didier Sornette (MTEC ETH) for enjoyable discussions on complex systems, and Anukool Lakhina (Guavus) for introducing me to the exciting topic of anomaly detection. I would also like to thank SWITCH for providing the access to their NetFlow data as this thesis would not have been possible without it. I am also grateful to Simon Leinen (SWITCH) and Peter Haag (SWITCH) for their support with the NetFlow data. I thank the Zurich Information Security Center (ZISC) for funding this thesis and providing helpful feedback.

I would like to thank my colleagues at ETH Zurich, in particular my office mates Bernhard Tellenbach and Stefan Frei and (in alphabetical order) Andreea Picu, Ariane Keller, Bernhard Distl, Brian Trammell, Dominik Schatzmann, Eduard Glatz, Elisa Boschi, Franck Legendre, Gabriel Popa, Georgios Parissidis, Ilias
Raftopoulos, Jinyao Yan, Marcel Baur, Mario Strasser, Martin Burkhart, Merkourios Karaliopoulos, Rainer Baumann, Rudolf Strijkers, Simon Heimlicher, Thomas Maillart, Theus Hossmann, Thrasyvoulos Spyropoulos, Udi Ben-Porat, Ulrich Fiedler, Vincent Lenders, Wenping Deng, and Wolfgang Mühlbauer. I further thank Caterina Sposato, Monica Fricker, Beat Futterknecht, Thomas Steingruber, and Damian Friedli for providing frictionless administrative and technical support, and Lukas Ruf for providing the \LaTeX{} template for this thesis.

I thank the students that I supervised during their semester and master theses for their contributions to FLAME and other projects, namely (in reversed temporal order and annotated with the thesis title) Evelyn Leuenberger (Extending FLAME: A framework for Anomaly Injection), Dominik Übersax (FLAME: A Flow-level Anomaly Modeling Engine), Jonathan Gysel (Analysis and Modeling of Internet Epidemics), Niklaus Frey (Localization of Known Anomaly Types in NetFlow Traces), Fabian Wanner (Anomaly Analysis using Host-behavior Clustering), Jonathan Gysel (Anomaly Visibility and Detectability in Sampled Traffic), Martin Schreiber (Flow-level Traffic Generation), Janneth Malibago (Monitoring and Debugging ISP Topologies), and Rashid Waraich (Attack Signature Generation: A Survey).

Last but not least, I would like to thank my parents and sister, and my partner for their unconditional support and all others that influenced my research and that were not explicitly mentioned here.