Diss. ETH No. 20929
TIK-Schriftenreihe Nr. 137

Detection, Classification and Visualization of Anomalies using Generalized Entropy Metrics

A dissertation submitted to
ETH ZURICH

for the degree of
Doctor of Sciences

presented by

BERNHARD MARTIN TELLENBACH

Master of Science ETH
in Electrical Engineering and Information Technology
born June 30, 1979
citizen of Oberthal, BE, Switzerland

accepted on the recommendation of
Prof. Dr. Bernhard Plattner, examiner
Prof. Dr. Didier Sornette, co-examiner
Dr. Andreas Kind, co-examiner

2012

Abstract

Today, the Internet allows virtually anytime, anywhere access to a seemingly unlimited supply of information and services. Statistics such as the six-fold increase of U.S. online retail sales since 2000 illustrate its growing importance to the global economy, and fuel our demand for rapid, round-the-clock Internet provision. This growth has created a need for systems of control and management to regulate an increasingly complex infrastructure. Unfortunately, the prospect of making fast money from this burgeoning industry has also started to attract criminals. This has driven an increase in, and professionalization of, cyber-crime. As a result, a variety of methods have been designed with the intention of better protecting the Internet, its users and its underlying infrastructure from both accidental and malicious threats. Firewalls, which restrict network access, intrusion detection systems, which locate and prevent unauthorized access, and network monitors, which oversee the correct functioning of network infrastructures, have all been developed in order to detect and avert potential problems. These systems can be broadly defined as either reactive or proactive. The reactive approach seeks to identify specific problem patterns. It uses models learnt from theory or practice to locate common dangers as they develop. The number of patterns applied grows as each new problem is encountered. Proactive methods work differently. They start defining an idealized model of the normal behavior of a given system. Any significant deviation from this model is assumed to be an aberrance caused by an external danger. However, this assumption may turn out to be incorrect, having actually not arisen from a disruption or a malicious act. Despite considerable improvements, the development of accurate proactive detection and classification methods is still an area of intense research. This is particularly true of methods fit for high speed networks. To cope with the huge amounts of data at hand, these methods utilize highly aggregated forms of data. Volume measurements and traffic feature distributions such as the number of connections per time unit or the distribution of connection sources form their primary sources of information. Various methods have been developed to detect anomalous changes in these distributions. Among them, entropy based methods have become widely used, and demonstrate considerable success in both research and production systems. Nonetheless, there remain many challenges regarding the use of entropy.

In this thesis, we address three of these challenges. In high speed networks, packet sampling methods are widely employed to reduce the amount of traffic data measured. However, we possess no empirical data about how this affects the visibility of anomalies when using entropy or volume metrics. Another area where additional analysis is required is the value of entropy with regard to anomaly detection. A study published by Nychis *et al.* found that entropies of common traffic feature distributions correlate strongly with simple volume measurements. The authors use this to suggest that they therefore do not contribute much. However, their claims do not match the practical evidence furnished by the many successful applications of this method. The second issue is the characterization and visualization of changes in distributions. In high-speed networks, the sheer quantity of information involved makes the concise representation of changes in distributions essential. However, many of the most commonly used methods, such as the Shannon entropy, are hampered by their limited descriptive power. This stems from the fact that they capture change using a single number. Other methods, including histograms, suffer by the fact that their optimal use depends on parameters which differ across various types of change.

The third problem to consider is the way in which the detection and classification capabilities of entropy-based anomaly detectors can be improved. Existing systems do show good detection rates. They can even, to an extent, successfully classify the largest anomalies. However, there remains scope to refine their performance, specifically when dealing with small to medium sized anomalies. Furthermore, studies on distributed denial of service and port scan anomalies from malware point out that parameterized entropies such as the Tsallis entropy might be superior to non-parameterized entropies. However, how these preliminary results can be linked to arbitrary types of anomalies, as well as appropriate detection and classification systems, remains underexplored.

In this work we make the following contributions. We analyze the robustness of entropy in the presence of packet sampling. Based on traffic traces from the outbreak of the Blaster and Witty worm, we find that entropy is not only robust but, depending on the traffic mix, might even lead to an improvement in the location of anomalies for sampling rates of up to 1:10,000. Next, we analyze whether the entropy of various traffic feature distributions provides valuable information for anomaly detection. We refute the findings of previous work, which reported a supposedly strong correlation between different feature entropies. Our core contribution is the *Traffic Entropy Spectrum (TES)*, a method for the compact characterization and visualization of traffic feature distributions. We also propose a refined version of the TES, which hones its capabilities with regard to anomaly classification. To demonstrate the descriptive power of the TES, we use traffic data containing real anomalies. Finally, we build the Entropy telescope, a detection and classification system based on the TES. We provide a comprehensive evaluation using three different detection methods, and one classification method. Our evaluation, based on a rich set of artificial anomalies combined with real traffic data, shows that the refined TES outperforms the classical Shannon entropy by up to 20% in detection accuracy and by up to 27% in classification accuracy.

Kurzfassung

Das heutige Internet ermöglicht jederzeit und praktisch überall Zugriff auf eine schier endlos erscheinende Menge an Informationen und Dienstleistungen. Zahlen wie die Versechsfachung des via Internet erzielten Umsatzes des US Einzelhandels seit 2000 weisen deutlich auf dessen zunehmende Bedeutung für die Weltwirtschaft aber auch auf die damit verbundene wachsende Abhängigkeit hin. Neben erhöhte Anforderungen an das Management und Überwachung aufgrund der zunehmenden Komplexität der Infrastruktur, führte dies insbesondere auch zu einer Zunahme und Professionalisierung der Cyber-Kriminalität. In den letzten Jahren wurden deshalb verschiedenste Methoden entwickelt, um das Internet, seine Teilnehmer und die zugrunde liegende Infrastruktur besser vor mutwilligen aber auch unbeabsichtigten Störungen und Bedrohungen zu schützen. Dazu gehören Systeme wie Firewalls zur Beschränkung des Netzwerkzugriffs, Systeme zur Erkennung und Verhinderung eines unerlaubten Eindringens oder auch Systeme zur reinen Überwachung des korrekten Funktionierens einer Netzwerkinfrastruktur. Zur Erkennung und Vermeidung von Störungen und Bedrohungen gibt es grundsätzlich zwei Ansätze: Erstens, der auf Mustererkennung basierte reaktive Ansatz, der die Erkennung von in der Theorie oder Praxis bekannten Bedrohungen ermöglicht. Und zweitens, der proaktive Ansatz, der auf der Annahme basiert, dass jegliche Abweichung von einem spezifizierten normalen Verhalten eines Systems auf eine Bedrohung oder Störung hindeutet. In einer Analyse der Abweichung kann sich dann aber durchaus herausstellen, dass es sich weder um eine Bedrohung noch um eine Störung gehandelt hat.

Trotz einiger viel versprechender Ansätze ist eine präzise Erkennung und Klassifizierung mit proaktiven Methoden noch immer ein Gebiet intensiver Forschung. Dies gilt insbesondere auch für Methoden, die für den Einsatz in Hochgeschwindigkeitsnetzen geeignet sind. Um den riesigen Datenmengen

Herr zu werden, basieren die meisten dieser Methoden auf hochaggregierten Informationen. Dazu gehören primär Volumen- oder Verteilungsinformationen wie z.B. die Anzahl Verbindungen pro Zeit oder die Verteilung der Quell- und Zieladressen oder auch der Verbindungsdauer von Verbindungen. Eine Klasse von Methoden, die sowohl in der Forschung als auch in der Industrie mit Erfolg eingesetzt wird, identifiziert ungewöhnliche Veränderungen mit Hilfe der aus den Verteilungen der Quell- und Zieladressen und der Quell- und Zielports der beobachteten Verbindungen berechneten Entropiewerte. Trotz dieses Erfolgs gibt es aber noch viele offene Fragen und Herausforderungen.

In dieser Doktorarbeit adressieren wir drei dieser offenen Fragen und Herausforderungen. Die erste Herausforderung betrifft die Analyse der Auswirkungen von unvollständigen Messdaten. In Hochgeschwindigkeitsnetzen wird zur Reduktion der Systemlast oft nur ein Teil der effektiv über das Netzwerk fliessenden Datenpackete für eine Messung berücksichtigt. Bei zufälliger Wahl der gemessenen Datenpakete ist somit die Chance gross, dass die Zahl der nicht erfassten Verbindungen für Verbindungen, die nur aus wenigen Datenpaketen bestehen, grösser ist als für Verbindungen mit vielen Datenpaketen. Bis anhin ist unklar, wie sich dies bei der Verwendung von Entropie als Metrik auf die Sichtbarkeit von Anomalien auswirkt. Unklarheit besteht auch beim Nutzen von Entropie-basierten Metriken im Hinblick auf die Erkennung von Anomalien. Eine von Nychis *et al.* publizierte Studie stellte hierzu fest, dass Entropie kaum mehr Informationen liefert, als bereits in einfachen Volumenmessungen enthalten ist. Die bisherigen Erfolge mit Entropiemetriken stehen allerdings im Widerspruch dazu. Eine zweite Herausforderung stellt die Erfassung und Visualisierung von Veränderungen in Verteilungen dar. In Hochgeschwindigkeitsnetzen ist eine kompakte und mit Fokus auf Veränderung informative Erfassung und Darstellung von Verteilungen aufgrund der schieren Menge von Informationen von grosser Relevanz. Bisher verwendete Verfahren haben entweder nur eine beschränkte Beschreibungskraft, weil sie, wie die Shannon-Entropie, die Veränderung mittels einer einzigen Zahl beschreiben. Oder deren optimalen Erfassung hängt wie beim Histogramm primär von Parametern ab, die von der Veränderung selbst abhängig sind. Die dritte Herausforderung betrifft die Verbesserung der Erkennungs- und Klassifizierungsleistung von entropiebasierten Anomalie Detektoren. Existierende Systeme zeigen bei massiven Anomalien gute Detektions- und teilweise auch Klassifikationsleistungen. Für kleinere Anomalien ist ihre Leistung hingegen wenig erforscht. Studien zu Distributed Denial of Service Anomalien und Portscans von Malware weisen zudem auf die Überlegenheit von parametrisierten Entropien wie der Tsallis Entropie hin. Eine Ausweitung auf beliebige

Anomalien sowie die Frage nach passenden Detektions- und Klassifikationssystemen bleibt aber unbeantwortet.

In dieser Arbeit machen wir die folgenden Beiträge: Wir analysieren die Robustheit der Entropie beim Einsatz von Messstrategien, die für die Generierung der Verbindungsinformationen im Durchschnitt nur jedes n-te Paket berücksichtigen. Basierend auf dem Ausbruch des Blaster und Witty Wurms zeigen wir, dass Entropiemetriken robust sind und je nach Verkehrsmix und Anomalie sich deren Sichtbarkeit bis zu Abtastraten von 1:10,000 sogar verbessern kann. Ein weiterer Beitrag ist eine Analyse der Relevanz der Entropie von verschiedenen Verbindungsmerkmalen in Bezug auf die Anomaliedetektion. Wir widerlegen dabei eine Studie, die eine starke Korrelation zwischen verschiedenen Entropie- und Volumenmerkmalen fand. Unser wichtigster Beitrag jedoch ist die Entwicklung des *Traffic Entropy Spectrum (TES)*, eine auf der Tsallis Entropy basierende Methode zur kompakten Charakterisierung und Visualisierung von Verteilungen von Verbindungsmerkmalen. Wir ergänzen diesen Beitrag durch eine Verfeinerung des TES im Hinblick auf die Klassifizierung von Anomalien. Zur Demonstration der Beschreibungskraft des TES verwenden wir Verbindungsdaten mit echten Anomalien. Schliesslich bauen wir das Entropie-Teleskop, ein auf dem TES basierendes System zur Erkennung und Klassifizierung von Anomalien und liefern eine umfangreiche Evaluation basierend auf drei verschiedenen Detektionsmethoden und einer Klassifikationsmethode. Die Auswertung mit einer grossen Zahl an künstlichen Anomalien kombiniert mit realen Verkehrsdaten zeigt, dass der verfeinerte TES Ansatz der klassischen Shannon-Entropie bei der Detektion um bis zu 20% und bei der Klassifikation um bis zu 27% überlegen ist.

Table of Contents

Abstract **iii**

Kurzfassung **vii**

Table of Contents **xi**

List of Figures **xv**

List of Tables **xxi**

1 Introduction **1**

1.1 High-Speed Networks 4
1.2 Anomaly Detection and Classification in High-Speed Networks 5
1.2.1 Packet Sampling 6
1.2.2 Aggregation of Packet Data into Flows 6
1.2.3 Traffic Feature Distributions 9
1.2.4 Summarization of feature distributions 10
1.2.5 Challenges 12
1.3 Claims and Contributions 13
1.4 SWITCH Network 15
1.5 Thesis Overview 18

2 Related Work **21**

2.1 Anomaly: A Definition 21
2.2 Anomaly Types 23

2.2.1 Anomaly Types: Taxonomy 24
2.2.2 Prevalence of Anomaly Types 27
2.3 Anomaly Detection Methods 30
2.3.1 Counter and Distribution-Based Methods 31
2.3.2 Entropy-Based Anomaly Detection Methods 33
2.4 Anomaly Classification 33
2.4.1 Anomaly Signature Based Methods 35
2.4.2 Unsupervised Learning Based Methods 37
2.5 Applications of Generalized Entropy in Other Fields 38
2.6 Evaluation of Anomaly Detection Systems 39
2.6.1 Data Sets 41
2.6.2 Evaluation Metrics 45

3 Impact of Packet Sampling 51
3.1 Introduction 52
3.2 Methodology 55
3.2.1 Dataset 55
3.2.2 Reconstructing Packet Traces 57
3.2.3 Packet Sampling 58
3.2.4 Feature Set 59
3.2.5 Baseline 61
3.2.6 Anomaly Visibility 63
3.3 Impact of Sampling on Entropy and Volume Metrics 63
3.3.1 Anomaly Size 66
3.3.2 Anomaly Intensity 68
3.4 Impact of the Traffic Mix 70
3.4.1 Results 70
3.4.2 Discussion 72
3.5 Conclusion 77

4 Analysis of Feature Correlation 81
4.1 Introduction 81
4.2 Methodology 83
4.2.1 Correlation Metrics 84
4.2.2 Data Set 84

4.3 Results 85
4.4 Discussion 89
4.5 Conclusion 89

5 Traffic Entropy Spectrum 93
5.1 Introduction 93
5.2 Shannon and Tsallis Entropy 95
5.2.1 Terms and Definitions 96
5.2.2 The parameter q 97
5.3 The Traffic Entropy Spectrum 98
5.3.1 Selection of the q-Vector 100
5.3.2 Visualizing Anomalies by Normalization 101
5.4 The Refined Traffic Entropy Spectrum: TES_p 102
5.4.1 Inter-Region Dependency 103
5.4.2 Inter-Region Dependency: Relevance 105
5.4.3 The TES_p 106
5.5 Spectrum Patterns 108
5.6 Evaluation 109
5.6.1 Feature Set 109
5.6.2 Set of Anomalies 110
5.6.3 Compiling the TES 111
5.6.4 Data Analysis 112
5.7 Results 113
5.7.1 Spotting the anomalies 113
5.7.2 Spectrum Patterns 116
5.7.3 The TES in action: Anomaly drill-down 121
5.8 Conclusion 126

6 Entropy Telescope 129
6.1 Introduction 130
6.2 Entropy Telescope 131
6.2.1 Wide Angle: Using Generalized Entropy 132
6.2.2 Zooming in: Separating Activity Regions 132
6.2.3 Image processing: Anomaly Detection 132
6.2.4 Scene Analysis: Classifying Anomaly Patterns . . . 136

6.3 Methodology . . . 138
6.3.1 Data Set . . . 138
6.3.2 Entropy Features . . . 138
6.3.3 Anomaly Models and Injection . . . 139
6.4 Evaluation . . . 144
6.4.1 Detection . . . 144
6.4.2 Classification . . . 149
6.4.3 Prevalence of Anomalies in Real Backbone Traffic . 153
6.5 Conclusion . . . 155

7 Conclusions 157
7.1 Review of Contributions . . . 157
7.2 Critical Assessment . . . 159
7.3 Future Work . . . 160
7.4 Publications . . . 162

A Appendix 165
A.1 NetFlow Collection and Processing Infrastructure . . . 165
A.1.1 Overview . . . 166
A.1.2 Costs . . . 168
A.1.3 Data Volume . . . 169
A.1.4 Processing Power and Memory Requirements . . . 169
A.1.5 Contributions . . . 170
A.2 NetFlow Tools . . . 170
A.2.1 NetflowVX Library . . . 171
A.2.2 NetFlow Processing Framework . . . 173
A.2.3 Data Analysis and Processing Tools for MATLAB . 176

Acronyms 185

Acknowledgments 211

Curriculum Vitae 215

List of Figures

1.1 Evolution of Cybercrime - A figure prepared by the well-known independent security researcher Jart Armin, shown in [1] 3

1.2 Two-way communication between two hosts: Example of aggregation of packet data into flows. Note that NetFlow flow records contain relative time stamps based on epoch, not absolute ones as in this figure. 8

1.3 Screenshot of our distribution analysis tool showing a traffic feature distribution for the traffic feature *source port*. The port number is indicated on the x-axis, with the number of flows with this source port noted on the y-axis in log-scale. Note that only TCP traffic flowing into our network is considered. In the 300-second time slot starting at 16.03.2004 07:45, port 80 is the port with the most flows (marked) closely followed by port 135. 10

1.4 The SWITCH backbone and its various points of presence. . 16

2.1 When discussing anomaly classification, the term *anomaly* is mainly used to refer to a specific set of anomaly types or classes. 23

2.2 Extended version of Plonka and Barford's network anomaly taxonomy [2]. Extensions are nodes illustrated in dark gray and all dashed lines. 26

3.1 Blaster flow trace: Plots of selected metrics for flows (packets) with direction IN and protocol TCP. The plots show results for the traces both with- and without the Blaster anomaly. 64

3.2 Blaster flow trace: Impact of sampling on time series of selected metrics for flows (packets) with direction IN and protocol TCP. Note that in Figure 3.2(a) and 3.2(b) the y-axis is log scale. 65

3.3 Blaster flow trace: Anomaly visibility vs. sampling rates for four selected metrics for flows (packets) with direction IN and protocol TCP [packet counts (Pcnt), flow counts (Fcnt), flow destination IP entropy (DstIP4e), and packet destination IP entropy (p-DstIP4e)]. The plot shows the mean and 95% confidence interval over 12 sampling runs for the first interval after the Blaster outbreak around 17:00 UTC. 67

3.4 Blaster flow trace: Relative distance from the baseline for flow counts and flow destination IP address entropy for flows with direction IN and protocol TCP across increasing sampling rates and different intensities. 68

3.5 Relative difference of flow count, unique destination IP address count, flow destination IP address entropy and flow destination port entropy metric on our four border routers vs. date and time (UTC). No sampling. 71

3.6 Relative difference of flow count and flow destination IP address entropy metrics on routers 2 and 4 vs. date and time (UTC) during the outbreak of the Blaster worm. 72

3.7 Relative difference of destination port count on the four border routers vs. date and time (UTC) during the Witty worm outbreak. 73

5.1 Example of a Traffic Entropy Spectrum (TES) of the Autonomous System activity in the incoming traffic around a DDoS attack in 2007. The plot shows how the Tsallis entropy values for different values of the parameter q change over time. During the day, entropy is higher than during the night. On the weekend (01/09, 02/09), the difference between day and night is smaller. On the y-axis, we see the set of q-values for which the Tsallis entropy values are plotted as colored rectangles versus the time on the x-axis. Hence, a rectangle (x,y,c) color-encodes the (normalized) Tsallis entropy for $q = y$ and time $T = x$. The color scale used ranges from black for the minimum S_q to white for the maximum S_q. 99

5.2 Impact of changes to different regions of the distribution. Bottom: Baseline distribution. Center: Visualization of the baseline distribution (at $T = 0$) which is then iteratively transformed into the distribution shown at $T = MAX$ for low, medium and high activity regions. We call the distribution at $T = MAX$ the target distribution because it is the one we design the transformations to stop at after a certain number of iterations. Top: Resulting TES when altering the distribution in the respective region from the baseline to the target distribution in multiple, evenly sized, steps. 100

5.3 A change in the high-activity region of a distribution should only affect the TES for positive q values (strongest for $q > 1$). Therefore, a plot of the relative difference between the TES before and after the change should show no difference for negative q. However, this is not true for the relative difference plot on the right even though the only difference in the underlying distributions is the activity of the most active element. As shown in the plot on the left, its activity is 1.5 times the activity of the original distribution. The original distribution corresponds to the activity distribution of the destination ports of the TCP traffic flowing into the SWITCH network between 09:45 and 10:00 on the 31st of July 2008. 104

5.4 Destination port activity distributions (top) and selected regions for TES_p (bottom). On the x-axis all ports are ordered by rank, i.e. with increasing activity to the right. 107

5.5 Left: Two activity distributions differing only in the activity of the most active element: activity in the modified distribution is 1.5 times that of the original distribution. The original distribution corresponds to the activity distribution of the destination ports of the TCP traffic flowing into the SWITCH network between 09:45 and 10:00 on the 31st of July 2008. Right: Relative difference of both, the TES and the TES_p of the original distribution and the modified distribution. 108

5.6 Reflector DDoS attack. 111

5.7 Screenshot of the Traffic Entropy Spectrum Visualization & Anomaly Detection Tool. The plot at the top displays time series data of the selected time series: the Tsallis entropy for $q = 1.0$ for the Autonomous System traffic feature. The plot at the bottom displays the corresponding TES. 113

5.8 TES snapshots from the DDoS 1 and DDoS2 114

5.9 TES snapshots from the Witty and Blaster anomaly 115

5.10 3D TES for incoming SrcPorts before and during refl. DDoS attack for $q = -2 \ldots 2$. Diagonal axis: date (10 days), vertical axis: normalized entropy. Transparent layers: Minimum and maximum of normalized entropy at normal week days. . . . 118

5.11 Log-log plot of the source port activity distribution before and during the DDoS 2 attack. The source ports, sorted according to their activity, are on the y-axis. 120

6.1 Entropy Telescope building blocks. 131

6.2 The number of flows per 5min bin of our baseline trace with injected anomalies of intensity 75K and 200K. 138

6.3 ROC curves for different feature sets and detection methods: (a) Anomalies of intensity 50K and 75K, Kalman filter method. (b) Anomalies of intensity 50K and 75K, PCA method (c) Anomalies of intensity 100K and 200K, PCA method. 146

6.4 ROC curves for anomalies with small intensities (50K and 75K) and PCA detection method. 148

6.5 Base anomaly classification matrix. The plots show which injected base anomaly types (y-axis) were classified as which types (x-axis) with what probability (color code). Models were trained using anomalies of ALL intensities. Classification is performed on anomalies with intensity <200K. . . 151

6.6 Comparison of anomaly patterns for SHN_+ and TES_{95}. Evidently, SHN_+ misses crucial information captured by TES_{95}. 152

6.7 Fisher's LDA plots of SHN_+ versus TES_{95}. 153

6.8 Detection and Classification results for a 34-day flow trace collected by one of the border routers of the SWITCH network in August 2008. Results are for TES_{95} with a PCA [k=36] detector. Each anomaly type is assigned a color, enabling easy comparison of its share across all four pie-charts. 154

A.1 The NetFlow collection and processing infrastructure run by the Communication Systems Group (CSG) at ETH Zurich as of June 2012. 166
A.2 Outline of the design of our NetFlow processing framework. 174
A.3 Screenshot of the traffic entropy spectrum visualization and anomaly detection tool. 178
A.4 Screenshot of the week profile and statistical parameter analysis tool. 180
A.5 Screenshot of the week profile and statistical parameter analysis tool. 181
A.6 Screenshot of the traffic feature distribution analysis and visualization tool. Histogram plot of the destination port distribution for incoming TCP traffic from all flow capturing devices. Activity (#flows) is shown on the y-axis and port numbers on the x-axis. 182
A.7 Screenshot of the traffic feature distribution analysis and visualization tool. Log-log plot of the destination port distribution for incoming TCP traffic from all flow capturing devices. Activity (#flows) is shown on the y-axis and the ports on the x-axis. Ports are ordered according to their activity. Most active port on the righ (port 80). 183

List of Tables

1.1 Median of the number of flows, packets and bytes per 15 minutes on weekdays in August 2003 and 2009 for routers 1 to 4, directions *IN* and *OUT* and transport layer protocols TCP and UDP. *IN* refers to traffic flowing into AS559 (IN: OUT→IN) and *OUT* to traffic flowing out of AS559 (OUT: IN→OUT). 18

4.1 Overview of traces used. To indicate the size of traces, we list the 75-percentile (75p) of flow counts computed in 5-minute windows. 85

4.2 Correlation of different feature entropies for traces 1-9 (see Table 4.1) as absolute values of the Spearman coefficients as a percentage. The table shows maximum, minimum, average, and standard deviation for correlation of $H(X)$ for TCP traffic. 86

4.3 Correlation of different feature entropies for traces 10a-d (TCP) as absolute values of the Spearman coefficients as a percentage. The table shows the maximum and minimum of 4 different routers for $H(X)$. 88

4.4 Correlation of different feature entropies for traces 10a-d (UDP) as absolute values of the Spearman coefficients as a percentage. The table shows the maximum and minimum of 4 different routers for $H(X)$. 88

6.1 Overview of 20 base anomaly models used. HAR/LAR denotes high/low activity region. 140

6.2 IP address sets used to customize anomaly models. The ID column corresponds to the anomaly ID in table 6.1. 143

6.3 Average classification accuracy as a percentage, for different sets of features and for different training and validation data set constraints. 150

A.1 The estimated cost of building and maintaining our NetFlow collection and processing infrastructure. 168
A.2 Amount of bzip2 compressed Cisco NetFlow data for each of the years from 2003 to 2012. 169
A.3 Performance of the NetflowVX library in terms of flows per second for different configurations and NetFlow versions for both compressed and uncompressed flow data. 172

Chapter 1

Introduction

The term *Internet* was coined in the late 1980s[1] when it referred to the emerging network infrastructure which resulted from the interlinking of the ARPANET and the NSFNET. Since then, the Internet has morphed from a mere research and communications network of around 160’000 hosts (October 1989) to a key corner stone of contemporary society with around 888 million hosts (January 2012).[2]. The Internet provides a multitude of functions. It is a digital library and a source for all sorts of (dis-)information. It allows us to communicate across continents and time zones through the use of instant messaging, social network sites and voice or video communication. Furthermore, its infrastructure facilitates the remote control of homes and factories, and even power plants and surveillance systems. The Internet forms the backbone of e-government, e-commerce and online banking. It is crucial to efficient industrial management, the smooth operation of complex public transport systems, and the supply of necessary goods and services. Almost anything we do nowadays involves the Internet in some way. It is fundamental to the maintenance of our standard of living. A March 2009 report produced by the European Commission (EC) [6] illustrates its importance to the European economy: In 2007, “...purchases and sales over electronic networks amounted to 11% of total turnover of EU companies”.

---

[1] According to Tannenbaum [3], the term Internet - as we understand it today - emerged around the time when ARPANET was interlinked with NSFNET. However, the first confirmed occurence of this term was in RFC675: Internet Transmission Control Program [4] where it is used to refer to any network based on the techniques described in RFC675.

[2] Source: Internet Systems Consortium: Internet host count history [5].

Unfortunately, this comes at a price. Our dependence on stable and reliable Internet provision makes us vulnerable to blackmail, to accidental failures, or to malicious attacks. To complicate matters further, the Internet is a distributed infrastructure managed by a large number of Internet Service Provider (ISP). ISPs operate across a variety of different business cultures and legal frameworks. Their global nature makes it difficult to address new challenges quickly and efficiently. The vulnerability created by the ever-increasing expansion of the Internet, coupled with a desire to make money, have seen the growth and professionalization of cyber-crime in recent years. On the 27th of April 2007, Estonia was hit by a large-scale cyber-attack, whose source could not be identified. On the 7th of August 2008, the Georgian military's IT systems were similarly crippled by a cyber-attack, shortly before the Russian army invaded the country. These instances are frequently cited examples of a troubling phenomenon; the threat of outright cyber war. Modern society's complete dependence on the Internet could be exploited. Both covert and open cyberspace attacks constitute a potentially devastating new way to defeat one's enemy.

The risk posed by these threats, and their imminence, is hard to guess. In [6], the EC estimates the probability "...that telecom networks will be hit by a major breakdown in the next 10 years, with a potential global economic cost of around 193 billion Euro" to be around 10% to 20%. This ominous prediction is backed up by a consideration of the historical evolution of cyber-attacks between 2000 and 2009. The growth seen in Figure 1.1 tells a rather disturbing story: Cyber-attacks have grown in both prevalence and sophistication. The earlier years were characterized by relatively few attacks, predominantly Tribal Flood Network (TFN)[3] attacks carried out by hackers for fun and fame. The past decade has seen this grow to over 7000 attacks per day. These new attacks are now chiefly motivated by money, and are carried out by advanced Botnet technologies, such as the Conficker Worm. The only positive news is that the sizes of the attacks themselves appear to have tailed off.

Indeed, there is one further cause for optimism. The large-scale attacks on Estonia and Georgia appear to finally have attracted the attention of political decision-makers. In [6], the EC proposed a new strategy to prepare Europe for action in case of a major incident. One of the EC's vows for future policy was to ensure that attack detection and response are supported by "...the

---

[3] A network of master/slave programs that coordinate with each other to launch a SYN flood against a victim machine.

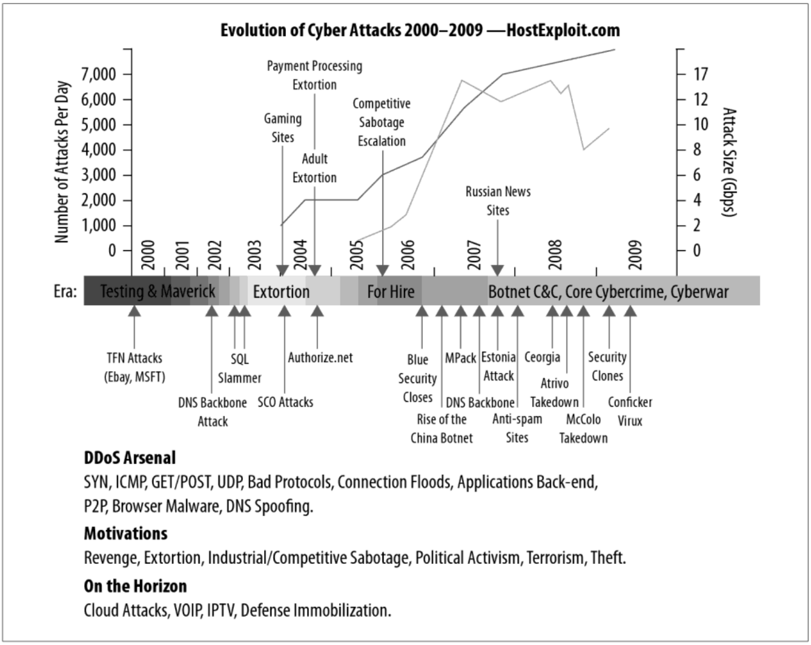


Figure 1.1: *Evolution of Cybercrime - A figure prepared by the well-known independent security researcher Jart Armin, shown in [1]*

development of a European information sharing and alert system”.

Fortunately, researchers in academia and industry have closely followed the evolution of the Internet, and its related emerging threats since its earliest days. As a result, a variety of methods have been designed with the intention of better protecting the Internet, its users and its underlying infrastructure from both accidental and malicious threats.. This has seen the creation of firewalls to restrict network access, intrusion detection systems (IDS) to locate and prevent unauthorized access, and network monitors to oversee the correct functioning of network infrastructures. Both reactive and proactive methods have been proposed as means of detecting and averting potential problems. The reactive approach seeks to identify specific problem patterns. It uses models learnt from theory or practice to locate common dangers as they develop. The number of patterns applied grows as each new problem is encountered. Proactive methods work differently. They start defining an idealized model of the “normal” behavior of a given system. Any significant deviation from this model is assumed to be an aberrance caused by an external danger. Hence, these systems are typically named *anomaly detection systems*. Confusingly, this term often refers to more than just systems which simply flag up problems. It also is often used as convenient shorthand to describe systems that also diagnose specific types of problems. In this thesis, to avoid confusion, we only employ this term where the distinction between anomaly detection and anomaly detection *and classification* systems is not relevant to analysis.

In this thesis, we concentrate on anomaly detection and classification systems tailored to operate in high-speed networks. In the remainder of this chapter, we start by explaining what kind of network we refer to when we say *high-speed network*, and the implications this has for systems designed to operate in such networks. We go on to discuss anomaly detection, and identify the most significant issues which arise from the use of these detectors. We round off this discussion by presenting our claims and contributions. Finally, we briefly introduce the network infrastructure which provides the measurement data used in this thesis before concluding with an overview of the different chapters of the thesis.

1.1 High-Speed Networks

As one would expect the term *high-speed network* refers to a network that allows to transport data in a fast and efficient way. But just how fast is fast? For traffic inspection in real-time, theoretical limits such as the bandwidth of the up and down-link(s) are merely one aspect to consider. Other aspects are e.g., network topology, sensor placement and the number of hosts communicating with each other. We therefore require a more precise definition than simply that of a network that transports data in a fast and efficient way. We propose the following alternative description:

Definition - High-speed network: *A high-speed network is a network where traffic monitoring and anomaly detection is typically performed on aggregate views of traffic data. Such networks handle traffic from thousands or millions of hosts and rely on technology and equipment expensive enough to prevent their use if not appropriate or no longer required. Network traffic data is collected at multiple locations in the network (e.g., border routers but also internal routers or switches).*

Examples of networks conforming to this definition are e.g., those of companies such as Amazon or eBay where network quality and speed is at the core of their business but also those of medium- to large-scale ISP.
Note that the methods and systems presented in this theis are not restricted to this type of network. However, in networks where it is an option to perform network monitoring and anomaly detection on full packet traces [4], our methods and systems could not make full use of the data available.

1.2 Anomaly Detection and Classification in High-Speed Networks

The main problem with a high-speed network is that it is almost impossible to inspect and analyze all of the network traffic flowing in- and out of the network. There is no hardware fast enough to inspect and search every single data packet for abnormal patterns as traditional IDS or other deep packet inspection (DPI) based systems do.[5] A DPI system typically inspects both the data and header information of each packet to search, for example, for protocol violations, malware or for traffic from a specific application. This is a CPU intensive process. The system must decompose the many protocol layers found in a network packet to identify its content.

At the same time, this large amount of data also constitutes the biggest advantage of systems operating at high-speed network level. Their traffic extends into the thousands or millions of hosts. This allows them to identify anomalies such as a sudden coordinated activity from a subset of hosts which would be invisible without a “global” view. Anomaly detection in high-speed networks exploits this advantage to focus on changes in traffic characteristics that could pose a potential threat to the stability and availability of a network. To exploit the benefit of this “birds-eye view”, the following data reduction steps are typically involved when operating at high-speed network level. The first two steps are generic and produce the basic input data for any kind of measurement. The third step is specific to most anomaly detection and classification approaches, whilst the fourth step is specific to entropy based appraches:

- Perform packet sampling to reduce the load on the capturing device

---

[4] =non-aggregate view

[5] Such hardware might be built but at a price where the cost-value ratio is too low.

- Aggregation of packet data into flow data on the capturing device
- Extraction of traffic feature distributions
- Summarization of feature distributions

We briefly discuss these steps, and a selection of their related problems and challenges before concluding with a short summary of the challenges addressed in this thesis.

1.2.1 Packet Sampling

Most of the sensors used for data collection in high-speed networks are devices whose primary task is to switch or route packets toward their destination. Any other task is considered to be of low priority, and is thus afforded only a limited share of its precious resources. In practice, this means that many operators configure them to consider only every tenth packet, every hundredth packet, or sometimes an even smaller fraction of packets, when performing network measurements or generating aggregate forms of data. Clearly, for most applications, sampling removes a significant chunk of information. The impact of this removal therefore needs to be studied in order for us to be able to identify information that is more robust to sampling, or to quantify the robustness to sampling of a given application.

1.2.2 Aggregation of Packet Data into Flows

In a next step, related (sampled) data packets are aggregated into a so called *flow*. A general definition of a flow can be found in RFC5101 [7], the IP Flow Information Export (IPFIX) Protocol Specification. According to RFC5101, a flow is a set of Internet Protocol (IP) packets sharing common properties and passing an observation point in the network during a certain time interval. Properties are the result of applying a function to the values of:

1. One or more packet header fields (e.g. destination IP address), transport header fields (e.g. destination port number), or application header fields (e.g. RTP header fields [8]).
2. One or more characteristics of the packet itself (e.g. packet length, etc.).
3. One or more of fields derived from packet treatment (e.g. next hop IP address, etc.).

With this definition, a flow can e.g. consist of all packets with the same source and destination,[6] all packets observed at a specific network interface, or a single packet between two specific applications. However, in this thesis, we use a more restrictive definition of the term *flow*. Firstly, a flow contain at least the following information:

- Source and destination of the packets (IP address and port number)
- Protocol used
- Time of the first and last packet
- The number of bytes and packets transferred from the source to the destination

Note that this requirement is in line with the common usage of this term in related work, as well as with the properties which providers of high-speed networks typically use for their flow data.[7] Also note that if packet sampling is used, the information contained in a flow is inaccurate. For example, instead of the true number of bytes and packets transferred from the corresponding source to the destination, only the bytes and packets of the packets that were sampled are reported. Secondly, we use the following definition for how packets are aggregated into flows (with respect to a single network link):

Definition - Flow: *A flow is a summary of all packets with the same 6-tuple - source IP address and port, destination IP address and port, protocol and Type of Service (ToS) field.*[8] *A flow starts with the first packet of a specific 6-tuple and ends either based on a timeout strategy or protocol specific triggers (e.g., TCP flags). A flow is an* unidirectional *flow since it accounts only for packets flowing from the source to the destination. Note that this specification complies with the traditional definition of a flow by the Cisco NetFlow [9] flow format.*

---

[6] E.g. all packets that share the same (1) source and destination IP address, (2) source and destination port and (3) the same protocol number

[7] Note that with older flow data formats such as Cisco NetFlow [9] version 5, there was no flexibility in the properties that define a flow

[8] Since 1998, this 8-bit long field has been termed Differential Service Field and consists of the 6 bit long Differentiated Services Code Point [10] and the two bit long Explicit Congestion Notification (from 2001 onwards) [11]. Previous definitions [12, 13] as well as the name *ToS field* are thus outdated and should no longer be used. See [14] for an interesting discussion of this field and its use related to flows.

A consequence of this definition is that a two-way communication results in two flows:[9] one for each direction. Figure 1.2 illustrates this with an example of a two-way communication between two hosts.

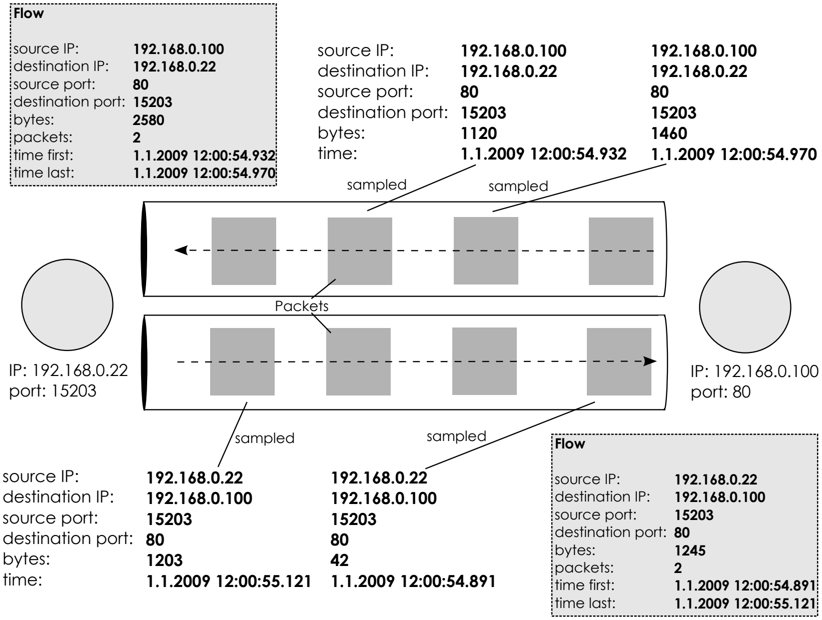

Flow
source IP: 192.168.0.100
destination IP: 192.168.0.22
source port: 80
destination port: 15203
bytes: 2580
packets: 2
time first: 1.1.2009 12:00:54.932
time last: 1.1.2009 12:00:54.970
source IP: 192.168.0.100 192.168.0.100
destination IP: 192.168.0.22 192.168.0.22
source port: 80 80
destination port: 15203 15203
bytes: 1120 1460
time: 1.1.2009 12:00:54.932 1.1.2009 12:00:54.970
sampled
sampled
Packets
IP: 192.168.0.22
port: 15203
IP: 192.168.0.100
port: 80
sampled
sampled
source IP: 192.168.0.22 192.168.0.22
destination IP: 192.168.0.100 192.168.0.100
source port: 15203 15203
destination port: 80 80
bytes: 1203 42
time: 1.1.2009 12:00:55.121 1.1.2009 12:00:54.891
Flow
source IP: 192.168.0.22
destination IP: 192.168.0.100
source port: 15203
destination port: 80
bytes: 1245
packets: 2
time first: 1.1.2009 12:00:54.891
time last: 1.1.2009 12:00:55.121

Figure 1.2: *Two-way communication between two hosts: Example of aggregation of packet data into flows. Note that NetFlow flow records contain relative time stamps based on epoch, not absolute ones as in this figure.*

If the data is collected by a device with multiple input and output interfaces, both the fields of the 6-tuple and the identifier of the ingress interface recording the packet must match. Thus, a TCP connection between two computers might be reported by two flows if the packets are routed over different paths arriving at different interfaces of the device. The same is true if data is collected by multiple sensors at different locations. The connection might be reported multiple times, either because all packets take the same path but

---

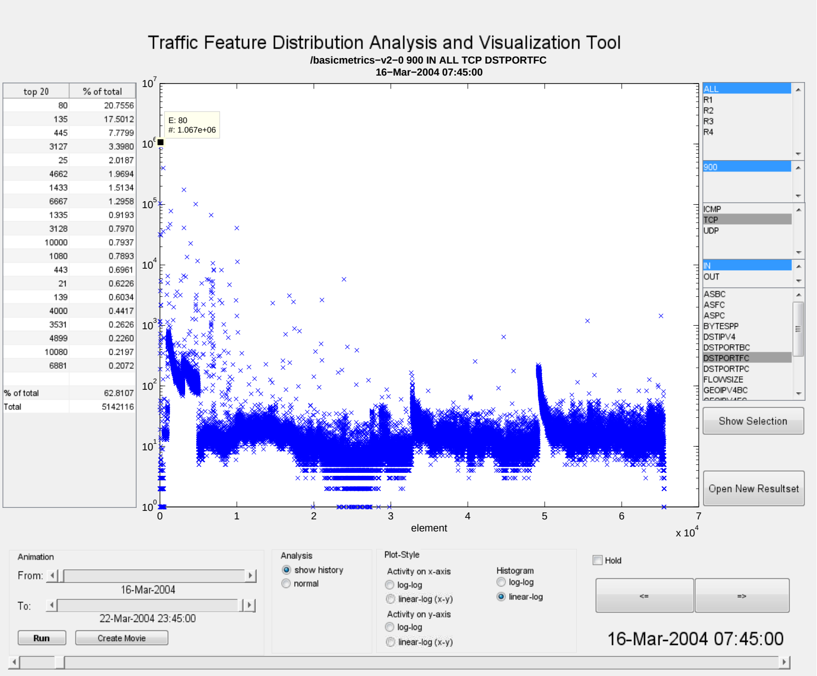
[9] Note that for IPFIX, RFC5103 [15], also proposes a way to aggregate and export flow information for both directions in one single flow. However, this is not yet widely used. Also note that if packets in one direction do not cross the same flow data collector as those in the other direction, the collector can report only an unidirectional flow.

cross multiple devices, or because different packets take different paths, or both. If an application expects a flow to represent, for example, one direction of a TCP connection as closely as possible, it is necessary to de-duplicate and merge such flows into a single flow. In the case of the border traffic of a stub network, this can be achieved by considering only flows from the border interfaces and by merging flows with the same 6-tuple.

1.2.3 Traffic Feature Distributions

The flow data obtained from the previous step is fairly detailed. Nonetheless, even more complex analysis is possible. This could involve tracking each host to see which other hosts they contact, when this is done, how much data they exchange, and even more complex behavioral analysis.

However, with traffic from up to several millions of hosts at aggregate line-speeds of up to hundreds of gigabits per seconds, this would require a considerable amount of memory and processing power. Systems that could do this on-line (in real-time) would be fairly expensive or might no even exist. As such, anomaly detection and classification systems operating in high-speed networks typically fare badly in locating "needles in the haystack" such as a single host getting infected by stealthy malware. Indeed, this is a particular strength associated with systems deployed to protect small networks or groups of host. Nonetheless, the large amount of data is also the biggest advantage of systems which operate at the high-speed network level. Dealing with traffic in the thousands and millions of hosts allows them to identify anomalies, such as a sudden coordinated activity from a subset of hosts, which would otherwise be invisible without a "global" view. Anomaly detection in high-speed networks exploits this advantage to focus on changes in traffic characteristics which might pose a potential threat to the stability and availability of a network. However, the birds-eye view provided by their huge amount of traffic allows them to identify anomalies invisible at the local scale, or problems whose extent and impact is not captured accurately enough at this scale. In the first case, this could constitute a sudden coordinated activity from a subset of hosts, where the majority of the hosts are located beyond the local scale. In the latter, this could refer to an anomaly visible at the local scale which affects not only the local network but many different other networks. Since complex analyses of traffic characteristics are typically expensive in terms of computational time and memory, the majority of anomaly detection systems in high-speed networks base their detection on an analysis of the number of flows, bytes or packets per time interval. This is complemented by an analysis of changes in *traffic feature distributions* of different flow features like source and destination port or source and destination IP address. Figure 1.3 shows a sample of such a traffic feature distribution.



Traffic Feature Distribution Analysis and Visualization Tool

/basicmetrics–v2–0 900 IN ALL TCP DSTPORTFC
16–Mar–2004 07:45:00

top 20	% of total
80	20.7556
135	17.5012
445	7.7799
3127	3.3980
25	2.0187
4662	1.9694
1433	1.5134
6667	1.2958
1335	0.9193
3128	0.7970
10000	0.7937
1080	0.7893
443	0.6961
21	0.6226
139	0.6034
4000	0.4417
3531	0.2626
4899	0.2260
10080	0.2197
6881	0.2072
% of total	62.8107
Total	5142116

E: 80
#: 1.067e+06

Animation
From: 16-Mar-2004
To: 22-Mar-2004 23:45:00

Analysis
● show history
○ normal

Plot-Style
Activity on x-axis
○ log-log
○ linear-log (x-y)
Activity on y-axis
○ log-log
○ linear-log (x-y)
Histogram
○ log-log
● linear-log

☐ Hold

16-Mar-2004 07:45:00

Figure 1.3: *Screenshot of our distribution analysis tool showing a traffic feature distribution for the traffic feature* source port. *The port number is indicated on the x-axis, with the number of flows with this source port noted on the y-axis in log-scale. Note that only TCP traffic flowing into our network is considered. In the 300-second time slot starting at 16.03.2004 07:45, port 80 is the port with the most flows (marked) closely followed by port 135.*

1.2.4 Summarization of feature distributions

Unfortunately, even traffic feature distributions are often infeasible, as several millions of data points need to be stored and compared in order to identify anomalous changes in such distributions. Hence, **a more compact representation of information about relevant changes is required**.

A prominent way of capturing important characteristics of distributions in a compact form is the use of entropy analysis. Entropy analysis (1) reduces the amount of information needed to be kept when characterizing changes in a distribution and (2) allows for a compact visualization of such changes. However, other summarization techniques such as histograms [16, 17] or Sketch data structures [18] are also used. Sketch-based approaches rely on a set of histograms where the elements are assigned to bins using a set of different hash-functions. Both histogram and sketch-based summarization enables the tuning of the amount of data to be stored and analyzed. Histogram based methods can be tuned by choosing an appropriate binning method and bin size. Sketch based approaches are tunded by selecting the type and number of hash functions and the number of bins per Sketch.

While many different forms of entropy exist, only a few have been studied in the context of anomaly detection and classification in high speed networks. The most common form is the Shannon entropy. This is not only used in research [19–21] but is employed in a variety of other contexts, including in NetReflex, a commercial anomaly detection and classification system made by Guavus [22]. However, other forms of entropy such as the Titchener T-entropy [23, 24] are also used. Nonetheless, most works agree that there are limits to entropy based detection, especially when it comes to detecting slow worms or small-scale attacks [19]. The strength of the approach lies in its broad scope [19, 23, 24]. These common claims may well hold true for entropy based approaches in general. However, studying different forms of entropy in detail might allow us to discover a form that can assist in refining the approach, and help push forward the boundaries regarding their use.

One promising form of entropy is the Tsallis entropy, a parameterized form of entropy. Two studies, one performed by Ziviani *et al.* [25] and the other performed by Shafiq *et al.* [26], provide evidence that this form of entropy might be superior to the Shannon entropy. The primary reason as to why this form of entropy might be superior to the Shannon entropy, or indeed to any other non-parameterized form of entropy, is that it can focus on changes in different regions of a distribution, depending on the parameter. As an example, let us consider the distribution of port numbers (see Figure 1.3). Changes in entropy caused by rarely used ports could potentially be obscured by more immediately noticeable change in those ports whose use is more frequent. In [25], Ziviani *et al.* investigate at which parameter value distributed denial of service attacks are best detected. Iin [26], Shafiq *et al.* do the same for port scan anomalies caused by malware. Unfortunately, the optimal choice of this parameter seems to depend either on the anomaly or the baseline traffic, or for both, since they did not report similar values for the optimal operation point. Thus, a generalization of these preliminary results towards arbitrary types of anomalies and an appropriate detection and classification systems remains unachieved.

Despite the many positive results noted with regard to the use of entropy, a recent study by Nychis *et al.* [27] questions the usefulness of the entropies of feature distributions such as source and destination IP addresses or source and destination port numbers. The study found a persistent and strong correlation between these entropies, which led them to their critical conclusion.

1.2.5 Challenges

In our thesis, we focus on the following challenges, as introduced in the previous section:

Robustness and significance of metrics:

Packet sampling methods are widely employed to reduce the amount of traffic data measured. It is therefore critical to identify anomaly detection metrics that are robust in the presence of packet sampling. Additionally, these metrics should provide valuable information in the sense that they do not show persistent and strong correlation.

Characterization and visualization of changes in distributions:

A compact characterization and visualization of changes in distributions is essential for most anomaly detection and classification methods for high speed networks. Many of the most commonly used methods, such as the Shannon entropy, are hampered by their limited descriptive power. This stems from the fact that they capture change using a single number. Other methods, including histograms, suffer by that fact that their optimal use depends on parameters which differ across various types of change

Improving entropy-based detection and classification methods:

Existing systems show good detection performance. They even, to an extent, perform successfully with regards to classification. Nonetheless, there remains room for improvement with regard to small to medium sized anomalies. Generalized forms of entropy seem to offer promise. However, a generalization of preliminary results towards arbitrary types of anomalies as well as appropriate detection and classification systems remains unachieved.

1.3 Claims and Contributions

The central claim of this thesis is that entropy is an accurate tool to both detect and characterize anomalous changes in traffic feature distributions of high-speed networks extracted at the network flow level. A detector and classifier built on the basis of generalized entropy can detect and classify network anomalies accurately and outperforms traditional volume or Shannon entropy based detectors.

We make the following research and engineering contributions to demonstrate that our claim holds:

- **A study into the robustness of entropy features with regard to packet sampling [28, 29]:** Many flow collectors make use of packet sampling to reduce their processing load. It is therefore important to know whether entropy features are robust with regard to exposing anomalies in the presence of (random) packet sampling. Our analysis [28] based on the Blaster worm anomaly showed that the Shannon entropy of feature distributions such as IP addresses or port numbers is less affected by sampling than traditional volume metrics such as byte- or flow count. We extended our study in [29] using the Blaster and Witty worm to evaluate how different traffic mixes and packet sampling affect the exposure of anomalies in volume, feature counts and entropy metrics. Based on traffic recorded by different sensors we show that the traffic mix has a significant impact on the visibility of an anomaly and might even lead to an increase in its visibility for sampling rates of up to 1:10,000. A comparison of feature counts[10] and entropy metrics reveals that they should both be used, as both have their respective strengths and weaknesses.

---

[10] E.g. the number of different ports or the number of different IP addresses per time window

- **Feature selection: A correlation analysis of various volume and entropy features [30]:** We analyze whether commonly used volume- and Shannon entropy features provide valuable, unbiased information, and refute the conclusions of previous work which suggested a supposedly strong correlation between different feature entropies.

- **A method for capturing and visualizing anomalous changes in traffic feature distributions [30, 31]:** We introduce the Traffic Entropy Spectrum (TES) to analyze changes in traffic feature distributions and demonstrate its ability to characterize the structure of anomalies using traffic traces from a large ISP ([31]). With regard to detection, we propose to use the information from the TES to derive patterns for different types of anomalies and present ideas as to how these could be used to automatically detect and classify anomalies. Furthermore, we propose a refinement of the TES that mitigates an unwanted effect, increasing the level of detail exposed [30].

- **Design and evaluation of a comprehensive anomaly detection and classification framework based on the TES [30]:** We propose a comprehensive anomaly detection and classification system called the entropy telescope and provide an extensive evaluation with three different detection methods, one classification method and a rich set of anomaly models and real backbone traffic. Our evaluation demonstrates the superiority of the refined TES approach over both TES and the more traditional approaches which employed only the Shannon method.

- **Redesign and implementation of NetFlow data processing infrastructure and libraries and tools:** At the start of this thesis, basic tools for reading, writing and processing NetFlow v5 data were available. However, many features needed for the thesis were still missing and many tasks involved running a set of different tools. Therefore, we redesigned the existing tools, contributing a significant number of new features in the process. Two of the most important features added are the flow's origin and destination Autonomous System (AS)[11] number and the origin and destination country in which the source and destination of the flow are located. Both features are derived automatically from the source and destination IP address of the flow and the flow's time stamp. The time stamp of the flow is required to identify which

---

[11] See 1.4 for a definition of AS.

lookup table to use since the mapping of IP address to AS number or country is far from static. It involves setting up an infrastructure to collect and process Border Gateway Protocol (BGP) data to generate the required lookup and developing tools to access and use them efficiently. The same had to be done for the IP address to country lookup tables provided by MaxMind® GeoIP databases [32]. We compiled everything into a comprehensive framework consisting of two major building blocks: the *NetflowVX* libraries and tools for working with NetFlow v5 and v9 data and a modular *NetFlow Processing Framework* providing a customizable abstraction from the different data formats and a simple way to assemble a processing chain using basic and custom modules. The basic modules provided include a reader capable of reading and merging two input streams, a reader and a writer for the internal flow format and a configurable filter to delete certain flows. Other contributions are a large set of largely object oriented MATLAB tools and a Flow-Level Anomaly Modeling Engine (FLAME) [33] based tool for an automated, efficient and highly parallel injection of synthetic anomalies into flow traces. The MATLAB tools are in the areas of transparent data access, statistics of time series and the detection, classification and visualization of anomalies. A more detailed description of the NetFlow data processing infrastructure and the libraries and tools can be found in the appendix of this thesis.

1.4 SWITCH Network

Our research- and engineering contributions all depend in one way or the other on the availability of network flow data from high-speed networks. Unfortunately, there are several problems (see 2.6.1) which prevent the ready availability of such data. Among them include practical issues caused by the huge amount of data to be collected and stored, legal issues, and the privacy concerns of potential data providers. We are therefore very thankful to our partner SWITCH [34] from whom we receive and archive flow data from all of its border routers since 2003.

SWITCH is the operator and also the name of the Swiss national research and education network. This network connects all Swiss universities and various research labs such as the IBM Zurich Research Laboratory or CERN to the Internet. Universities and research labs are not the only sites it links. Other sites such as VSnet, a network in the canton of Valais with many mem-

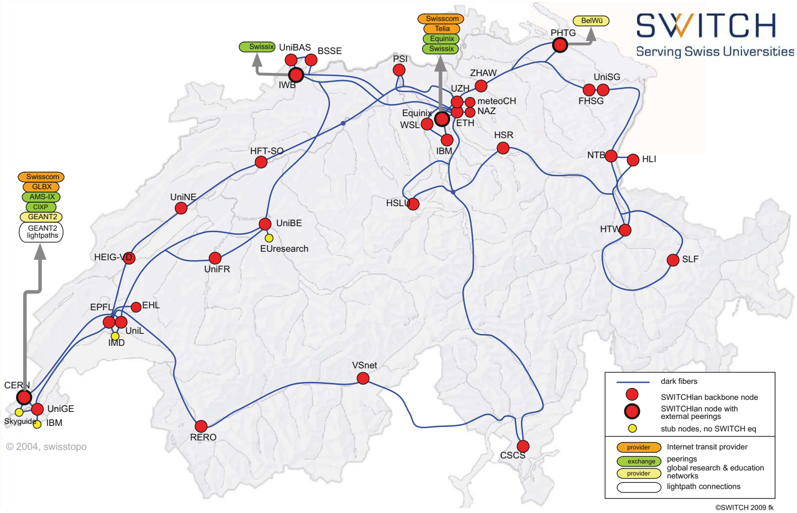


Figure 1.4: *The SWITCH backbone and its various points of presence.*

bers from the scientific community, education, culture and information, and Skyguide, the country’s air navigation service provider, are also attached to it. We believe that this makes SWITCH traffic fairly representative in the sense that it includes many different types of traffic. Traffic from residential (broadband) networks or a company network usually lacks grid computing traffic or other "‘special"’ types of traffic.

Figure 1.4 shows the SWITCH backbone and its various points of presence (customers), external peerings in Geneva, Basel and Zurich and other exchange points with research and education networks (BelWü, GARR). The border routers from which we get the flow data have the following names and locations:

- Router 1: Located in Zurich (Telia, Equinix).
- Router 2: Located in Geneva (GÉANT2, GBLX, CIXP).
- Router 3: Located in Basel (SwissIx).

- Router 4: Located in Geneva (GÉANT2[12], Swisscom, AMS-IX).
- Router 5: Located in Zurich (SwissIx, Swisscom). Added: April 2008.
- Router 6: Located in Kreuzlingen (BelWü). Added: February 2009.

These border routers export flow data in the Cisco NetFlow [35] format[13] which is neither anonymized nor sampled. We therefore receive a complete view of the traffic flowing in and out of this network[14].

The above network is also known as Autonomous System AS559. According to section three of RFC 1930 [36], an Autonomous System is a collection of connected IP routing prefixes under the control of one or more network operators that presents a common, clearly defined routing policy to the Internet. It is considered to be a connected segment of a network topology controlled by a single operations and maintenance organization presenting a consistent picture of the destinations reachable through the respective AS. Note that this refers either to destinations that are part of the respective AS or destinations located in another AS for which this AS acts as a transit.

In the case of the SWITCH network, the largest portion of network traffic originates from hosts inside AS559 to a destination outside AS559 - denoted by IN→OUT or its shortcut *OUT* - and the opposite - denoted by OUT→IN or its shortcut *IN*. Transit traffic (OUT→OUT) is limited to traffic from one research network or site (e.g., BelWü) to another research network (e.g. GÉANT2). In our thesis, we only look at network traffic ending or originating in AS559 and ignore other traffic.

It is clear that since 2003, when we started to collect this data, the network has undergone several structural and technological changes. While the number of IP addresses remained more or less constant - roughly 2.15 million in 2003 and 2.4 million in 2010, with a peak of 2.5 million in 2006 - the network load did not. Table 1.4 lists the median number of flows, bytes and packets per 15 minute bin on weekdays in August 2003 and August 2009. Results are listed for routers one to four considering the transport layer protocols UDP and TCP as well as the direction of the traffic. With the exception of router 3, most flow, byte and packet counts increase by a factor of two to 8 when comparing the data from 2003 with that 2009. With a median of around 76 million

---

[12] Backup only

[13] From 2003 to 2008, they exported NetFlow version 5 (NFv5) data, and from 2009 NetFlow version 9 (NFv9) data.

[14] We are aware of the fact that this does not hold in times when the flow table is full or the CPU load on the router is very high. Fortunately, these conditions rarely affect us, and we rarely lose information because of them.

		#Flows		#Bytes		#Packets		
		08/03	08/09	08/03	08/09	08/03	08/09	
IN	R1	9.40E+05	2.60E+06	8.80E+09	3.30E+10	2.30E+07	6.60E+07	TCP
	R2	1.70E+05	2.60E+06	4.70E+09	6.30E+10	6.60E+06	8.50E+07	
	R3	1.30E+05	2.90E+03	9.00E+08	5.60E+07	1.60E+06	8.60E+04	
	R4	6.40E+05	7.10E+05	5.90E+09	2.30E+10	9.50E+06	2.40E+07	
OUT	R1	7.80E+05	9.80E+05	2.50E+10	5.30E+10	2.80E+07	5.80E+07	
	R2	1.50E+05	7.10E+05	9.30E+09	5.30E+10	9.80E+06	5.70E+07	
	R3	4.10E+04	1.60E+03	5.10E+08	1.60E+07	9.00E+05	5.40E+04	
	R4	2.20E+05	6.50E+05	3.90E+09	1.60E+10	5.80E+06	2.30E+07	
IN	R1	4.40E+05	2.80E+06	2.30E+08	1.80E+09	1.30E+06	8.00E+06	UDP
	R2	7.90E+04	2.90E+06	5.60E+07	2.00E+09	1.90E+05	8.00E+06	
	R3	5.80E+04	1.10E+04	2.70E+07	8.60E+05	1.20E+05	1.10E+04	
	R4	5.20E+05	7.40E+05	1.70E+08	3.50E+08	9.50E+05	1.70E+06	
OUT	R1	3.30E+05	1.10E+06	1.40E+08	3.10E+09	8.80E+05	5.60E+06	
	R2	8.30E+04	2.30E+06	3.70E+07	2.30E+09	2.60E+05	6.10E+06	
	R3	1.10E+04	8.70E+03	5.90E+06	1.20E+06	4.90E+04	8.90E+03	
	R4	2.10E+05	1.00E+06	6.70E+07	1.50E+09	5.30E+05	3.50E+06	

Table 1.1: *Median of the number of flows, packets and bytes per 15 minutes on weekdays in August 2003 and 2009 for routers 1 to 4, directions* IN *and* OUT *and transport layer protocols TCP and UDP.* IN *refers to traffic flowing into AS559 (IN: OUT→IN) and* OUT *to traffic flowing out of AS559 (OUT: IN→OUT).*

flows, 10^{12} bytes[15] and $1.4 * 10^9$ packets per hour, the SWITCH network is indeed a high-speed network. This is also reflected in the size and growth rate of our flow data archive. On the 28th of October 2010, it contained 76 TiB of bzip2 compressed Cisco NetFlow data and showed a growth rate of around 400 GiB per week.

1.5 Thesis Overview

The remainder of this thesis is structured as follows. In chapter 2 we review related work in the field of anomaly detection and classification. We start by reviewing the different meanings associated with the term anomaly and define how it is used in this thesis. We then continue with a brief review of taxonomies of network anomalies and discuss related work investigating the different types of network anomalies. Next, we present related work on anomaly detection and classification in high-speed networks and conclude

---

[15] Corresponding to a *median* bandwidth usage of around 2.2 Gbps.

the chapter with a brief review of the literature concerning the evaluation of anomaly detection systems.

Chapters 3 and 4 investigate the robustness and usefulness of various entropy and volume metrics. In chapter 3, we study the impact of packet sampling on the visibility of anomalies in various entropy and volume metrics and shed some light on the role of traffic mix. In chapter 4 we continue our assessment by analyzing whether commonly used volume and entropy metrics provide valuable, unbiased information. We present and discuss the results of our analysis and discuss why our analysis refutes the findings of previous work which reported a supposedly strong correlation between different entropy metrics. Furthermore, we introduce new metrics reflecting the geographical structure of traffic and show that they add another layer of potentially useful information.

In chapter 5 we present and discuss the *Traffic Entropy Spectrum*, our method for a compact characterization and visualization of traffic feature distributions based on a parameterized form of entropy. After introducing the concept and properties of the TES, we demonstrate its descriptive power using traffic data from different massive real world anomalies.

Finally, in chapter 6 we introduce the entropy telescope, our anomaly detection and classification system. We provide evidence for our claim that a detector and classifier built around this tool can detect and classify network anomalies accurately, outperforming traditional volume or Shannon entropy based detectors.

Chapter 7 summarizes the results and contributions from our thesis and discusses directions for future work.

Chapter 2

Related Work

The field of network anomaly detection and classification is not new. A considerable amount of related work which analyzes and addresses the various aspects of the problem already exists. This chapter reviews a number of key publications which cover the field of anomaly detection and which focus on anomaly detection and classification in high-speed network.

The chapter is structured as follows. Starting from common general definitions of the term *anomaly*, we review its interpretation by the networking community and explain how it is used in this thesis. Next, we discuss related work on the identification and characterization of common anomaly types. We examine notable publications on anomaly detection methods, and subsequently focus on works concerning anomaly classification. We conclude the section with a short review of investigations into the use of generalized forms of entropy in other domains.

2.1 Anomaly: A Definition

Events as diverse as a massive DDoS attack, a network sensor reporting incorrect information, a host not following a given communication protocol, or the network bandwidth used at 12:00am being twice as high as the maximum seen in the last seven days can all be said to be some sort of anomaly. Nonetheless, despite the potential different meanings associated with the term anomaly, the following common generic definitions are used [37–40]: An anomaly is (1) a rare or infrequent event with a frequency below a certain threshold,[1] (2) an unexpected result, (3) a deviation from a normal form or rule, or (4) a state outside the usual range of variations.

In anomaly detection in network traffic, these definitions are often used interchangeably, even though there might be significant differences. A rare event might actually just be the normal behavior of a system which has not been seen often and therefore has been difficult or impossible to include in models which define normal behavior. Nassim N. Taleb coined the terms Black Swan and Grey Swan to refer to exactly this problem. His book [41] gained a lot of interest, mainly because of its crucial relevance in light of the crash of the financial markets in 2008. However, Taleb's ideas are not entirely new. The logical roots of his Black Swan theory lie in the Knightian uncertainty,[2] which refers to an immeasurable risk. In contrast to Taleb's formulation, the *dragon-kings* [43] concept claims that black swans are practically non-existent, and that the notion of black swans is largely unhelpful when studying real systems. In [43], Sornette defines dragon-kings as extreme events which cannot be classified in the same way as other events. The hypothesis, articulated in [43] and elaborated in [44], holds that dragon-kings appear as a result of a set of amplifying mechanisms which remain absent or inactive for other sets of events. This gives rise to specific properties and typologies that may be unique characteristics of dragon-kings. Sornette and Ouillon [44] explore the theory and practical application of this concept, finding significant relevance for the natural, biological and social sciences.
As a consequence, many researchers measure deviations from a *baseline*, rather than from a pre-determined "norm" [40]. Others avoid this problem by designing and evaluating detectors for a set of prevalent anomaly types identified and specified in advance. In doing so, they forfeit the ability to detect new types of anomalies. Another underappreciated issue is that deciding whether something is an *anomaly* is not a simple or objective zero-sum decision [40]. Do we require 50, 500 or 5000 clients connecting per minute to an otherwise unpopular web server to trigger an alert? From an operational point of view, the policies of the operator specify what is considered an anomaly. They may define some sort of minimum size or even decide to ignore some types of anomalies entirely because they do not pose a threat to the operator's

---

[1] Typically from 5% to less than 0.01%, depending on the application.

[2] Named after Frank Knight (1885-1972), an economist from the University of Chicago. In [42, p.19], Frank Knight writes about the need to distinguish risk and uncertainty: "But Uncertainty must be taken in a sense radically distinct from the familiar notion of Risk, from which it has never been properly separated".

infrastructure. However, since a policy-based view can typically be obtained from any detection and classification system by tuning the sensitivity of the detector and by filtering anomalies of certain classes using the labels from a classifier, such a perspective is rarely found in research.

In this thesis, we use the term *anomaly* in two ways. When discussing anomaly detection, we use the term *anomaly* to refer to deviations from a *baseline* derived from a large number of measurements. When discussing anomaly classification, the term *anomaly* is mainly used to refer to a specific set of anomaly types or classes. See Figure 2.1 for an illustration of this relationship. The reason for using the term in this way is that while it is possible to detect as of yet unknown anomalies, it is generally difficult if not impossible to classify them.[3] Note that the terms *anomaly type* and *anomaly class* are often used interchangeably. However, in the context of anomaly classification systems, the term *anomaly type* might also be used to refer to anomalies that share certain properties independent of the classification system. Meanwhile, the term *anomaly class* might be used to refer to the label attributed to an anomaly by the anomaly classification system.[4]

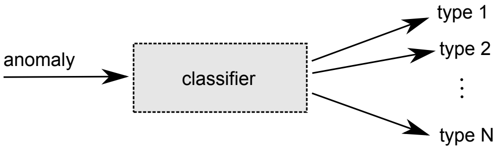


Figure 2.1: *When discussing anomaly classification, the term* anomaly *is mainly used to refer to a specific set of anomaly types or classes.*

2.2 Anomaly Types

As mentioned in the previous section, our thesis focuses on a set of well-known anomaly types, as it is generally difficult if not impossible to classify unknown anomalies. The following review of related work on taxonomies gives an overview of the well-known anomaly types we can choose from. In

---

[3] Note that approaches like unsupervised clustering might be able to find similarities between unknown anomalies and assign them to different clusters (groups). However, the main problem is then to identify what kind of anomalies these clusters are representing.

[4] Ideally, the anomaly type and anomaly class are the same. But if a classification system uses, for example, classes that include several anomaly types, there is no one-to-one relationship.

our thesis, we make use of a subset of them only: *Probing/Scanning*, *DoS*, *DDoS* and *Worms*, where the DDoS type also includes the sub-types *reflector DDoS* and *Flash Crowd*. Clearly, we could have included more anomaly types and then evaluated our system with a labeled trace of captured flow data with a few anomalies per anomaly type at fixed times only. However, we preferred to use a smaller set of anomaly types and to instead spend our time with a thorough evaluation taking the many different forms and intensities of these anomalies into account as well as their time of occurrence.[5] Another reason for not extending our set further was that the two most prominent anomaly types not included in our set are the *Outage* and *Heavy Hitter (or* α*-flow)* types. Both of theses are already relatively easy to spot using volume-based anomaly detectors only.[6]

In addition to these considerations, our choice of anomaly types was guided by the following two criteria: (1) the prevalence of anomalies of this type and (2) evidence that this type can be detected with methods applicable to high-speed networks. The review of related work on the prevalence of selected anomaly types which concludes this section shows that they all meet the first criteria. The review of related work on anomaly detection and classification shows the same for the second criteria. Note that the review of related work on the prevalence of selected anomaly types also includes the Flash Crowd anomaly type, to help shed some light on a type that is generally considered to be difficult to distinguish from anomalies of the DDoS type. Due to their similarity to the DDoS type, they are sometimes considered to be a subtype of the DDoS type.

2.2.1 Anomaly Types: Taxonomy

In [2] Plonka and Barford propose a taxonomy of network anomalies based on the system used at the University of Wisconsin to log anomalies. Their taxonomy includes the following generalc anomaly types: *Denial of Service (DoS)*, *Probing/Scanning*, *Popular Content Exchange (Flash Crowd)*, *Maintenance*, *Network (Failure)* and *Anomalous* or *Faulty Measurement* as well as *Prohibited Content Exchange (Flash Crowd)*. The taxonomy is shown in Figure 2.2 as a tree rooted at the *Anomaly* node. Note that both, *Popular Content*

---

[5] A volume anomaly might be easier to spot where the expected variance is small.

[6] Note that depending on the actual size of the anomaly, and the dynamics of the normal traffic, the same might be but is not necessarily true for anomalies of the types included in our set. It is only when doing classification, that these types might profit from additional information provided by the entropy features.

Exchange and *Prohibited Content Exchange* are considered to be of the *Flash Crowd* type. Imagine, for example, that a hacker manages to break into a protected area of a webserver, removing its security measures and sharing its private information in a manner that makes it publicly available to a wide audience. It would be impossible to distinguish this attack from a *Flash Crowd* caused by *Popular Content Exchange*[7] without first knowing that the content had been shared in this malicious way.

When we ignore the gray nodes and dashed lines, the tree expands to five general anomaly types and continues to more specific causes and characteristics that further define and differentiate the anomalies. Other works subdivide these broad anomaly types into a set of more specific types. The taxonomy provided by Hussain *et al.* [45] splits denial of service anomalies into single-source, multi-source and reflector anomalies. These anomalies are also known as DoS, DDoS and reflector DDoS. Even more detailed taxonomies also exist, although they are rarely used with systems for high-speed networks. This is chiefly due to the fact that the collecting and processing of the required information does not scale. An example of such a taxonomy is presented in [46]. The authors define a large set of parameters, including the attack dynamics, the impact of the attack, and the victim type, in order to distinguish various types of DDoS anomalies.

Other important anomaly types that do not have their own node in Plonka and Barford's taxonomy include alpha flows (α-flows), Worms and botnet activity and anomalous activities from Peer-to-Peer (P2P) networks. α-*flows* are a small number of flows that have a very large quantity of packets or bytes or that represent an unusually high rate of point to point byte transfer. Note that α-*flows* are sometimes also called *Heavy Hitters*. Examples of publications using these types include [47–49] for α-*flows*, [19,48–50] for *Worms*, [51,52] for botnet activity and [27] for anomalous P2P network activity. Figure 2.2 shows the possible integration of these types into Plonka and Barford's taxonomy. α-*flows* could be inserted as an additional node to the Anomalous Measurement type while worms could be seen as new form of abuse arising from a combination of *Prohibited Content Exchange* and *Probing/Scanning*. Botnet activity is more difficult to integrate as it includes activity such as *Probing/Scanning*, *Prohibited Content Exchange*, etc. To differentiate between anomalies in real and virtual networks (e.g. P2P), the Network node might be replaced with one for each virtual network type.

---

[7] As an example, when breaking news published on a news site attracts a massive number of visitors in a short time frame.

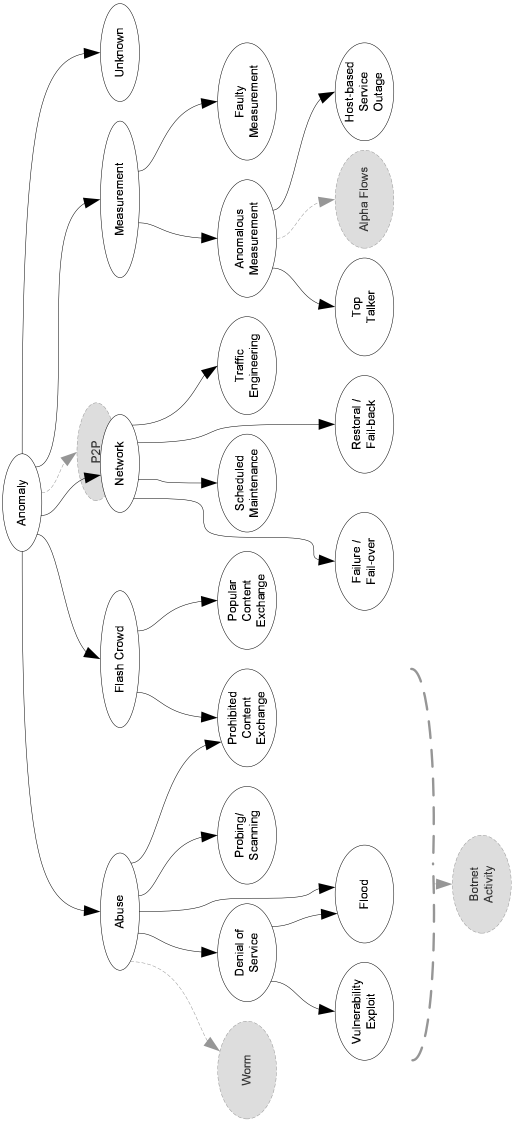


Figure 2.2: *Extended version of Plonka and Barford's network anomaly taxonomy [2]. Extensions are nodes illustrated in dark gray and all dashed lines.*

2.2.2 Prevalence of Anomaly Types

Probing/Scanning and Worms:

The root causes of probing and scan traffic are manifold. They might originate from attackers looking for a target to attack and compromise. However, they can also arise from productive, non-vindictive activity, including monitoring services such as pingdom,[8] or web crawlers from search engines like Google.

Another source of scan traffic is network service worms.[9] In contrast to viruses and other types of malware, worms are completely self-contained. They can modify, copy, execute and propagate themselves without user intervention. Whenever a worm tries to propagate and infect another system, it starts to scan for appropriate targets. If it finds a vulnerable target, it quickly attacks the target and copies itself to it. Since the whole scanning and spreading process is fully automated, a worm can rapidly infect most, if not all, vulnerable and reachable hosts. Consequently, a worm causes distinctive patterns of scan traffic. Its volume and spatial distribution are markedly different from the scan traffic caused by other attackers looking for a target to attack and compromise, or from monitoring activity and other forms of productive scanning activity.

In [54], Yegneswaran *et al.* study the prevalence and distribution of different categories of anomaly. Based on a large set of firewall logs from more than 1,600 networks, they analyzed the quantity and variety of their data before extrapolating the results to the Internet as a whole. They investigated different kinds of worm activity along with four categories of port scans.[10] They identified a significant estimated prevalence of 25 billion anomalies per day. Furthermore, their in-depth investigation revealed that (1) worm traffic is visible long after the original release of the worm and (2) that the sources responsible for the anomalies are spread uniformly across Autonomous Systems, whereas a significant share of them can be attributed to a relatively small number of sources exhibiting a correlated on/off behavior. Similar findings about the prevalence of scanners are presented in [55] where Allman *et al.* studied the behavior of scanners between 1995 and 2007.

While the findings of [54] regarding the persistence of worm traffic are

---

[8] `https://www.pingdom.com`: A service to monitor, for example, the uptime and response time of web servers.

[9] See [53] for a more detailed definition of different types of worms and other types of malware.

[10] Horizontal, vertical, coordinated and stealth port scans

confirmed by several sources (e.g. [56, 57]), many believed that, having seen a significant decline in the volume of network worms from 2006 (see e.g., [58, 59]), the days of worms[11] would soon be over. In September 2006, Symantec wrote in its Internet Security Threat Report [58, p.2] that:

" Since that first report,[12] much has changed. Large Internet worms targeting everything and everyone have given way to smaller, more targeted attacks focusing on fraud . . . "

However, the Conficker worm, which appeared in November 2008 [60, 61], was a wake-up call to those who thought that these kinds of worms had been rendered obsolete. In its Internet Threat Report for 2008 [61, p.7] published in April 2009, Symantec wrote that:

> Previous editions of the Symantec Internet Security Threat Report noted that there has been a decrease in the volume of network worms, partly due to a lack of easily exploitable remote vulnerabilities in default operating system components. Many network worms exploited such vulnerabilities in order to propagate. Highly successful worms such as CodeRed, Nimda, and Slammer all exploited high-severity vulnerabilities in remotely accessible services to spread. These worms prompted changes in security measures, such as the inclusion of personal firewall applications in operating systems that are turned on by default. This helped protect users from most network worms, even if the vulnerability being exploited was not immediately patched. . . . Soon after [the discovery of a high-severity vulnerability in the Microsoft® Windows® Server® Service RPC Handling component], a new worm called Downadup (also known as Conficker) emerged that exploited this vulnerability.

Hence, while worms might no longer be the biggest threat, we should keep them on our radar because (1) recent worms like the Conficker worm (2008), the Stuxnet [62][13] worm (2010) or the Morto [63] worm (2011) demonstrated that worms still pose a very realistic threat and (2) because worm traffic from past worms is still around.

---

[11] This mainly referred to self-spreading and/or fast-spreading worms.

[12] The first Symantec Internet Security Threat Report was published in 2002.

[13] Note that this worm was found to chiefly use self-generating techniques in the local area network, where it sought out and infected SCADA systems.

DoS and DDoS:

A first quantitative estimate and characterization of denial-of-service anomalies is presented in [64] and refined in [65], where Moore *et al.* analyzed a set of 22 traces of at least a week long between 2001 and 2004. The authors propose and employ an analysis technique called backscatter analysis in order to capture denial-of-service anomalies originating from attackers spoofing the source addresses of attack packets at random.[14] By monitoring the traffic to a \8 network, they can search for response packets from victims that do not have a corresponding request originating from the monitored network. Their methodology allows them to account for DoS and DDoS anomalies. [15] However, it ultimately causes them to underestimate their true number, as they do not account for anomalies caused by attackers who apply different spoofing schemes, or who employ techniques which involve no spoofing whatsoever. Moore *et al.* observed over 68,000 attacks to over 34,000 distinct victim IP addresses. In all of their traces, they found around at least 1000 anomalies per week, or roughly 125 anomalies per day. While their data does not allow them to identify whether anomalies of this type are on the rise, they do provide evidence that they are prevalent. The yearly Worldwide Infrastructure Security Reports from Arbor Networks [67–72] show similar results for the years 2005 to 2010. The evaluation of a questionnaire completed by a number[16] of Tier 1, Tier 2 and other IP network operators from around the world confirms that (1) (D)DoS attacks were among the top threats in all of these years and (2) the bandwidth consumed by the largest attacks has seen a significant increase from 10 to 100 Gigabit per second. [72] also contains some quantitative information about the number of attacks. 47% of the survey's participants experienced 1 to 10 attacks per month, 41% suffered 10 to 500 attacks per month and another 6% endured more than 500 attacks per month. Only 6% reported that they did not suffer from any DDoS attacks.

---

[14] This is considered to be the origin of the term *backscatter* referring to background radiation resulting from denial-of-service attacks using multiple spoofed addresses. According to [66], a paper co-authored by Vern Paxson, background radiation is network traffic directed to unused addresses which is either malicious traffic (backscatter, scanning or worm traffic) or benign traffic (misconfigurations).

[15] The anomaly looks like it is a DDoS anomaly since it appears to be triggered by many (spoofed) sources, but without tracing the anomaly back to its true source(s) it remains unclear whether it is a DoS or DDoS anomaly.

[16] 2005: 36, 2006: 55, 2007: 70, 2008: 66, 2009: 132, 2010: 111.

Flash Crowds

The flash crowd is an anomaly type which looks very similar to a denial of service attack at the network level. During a flash crowd, the number of machines accessing a specific resource in the network increases significantly, until reaching an abnormal level. Events that trigger this anomaly are typically high-profile events, such as the FIFA World Cup final, presidential elections or the release of a new version of a popular type of software. This suggests that flash crowds are prevalent and that distinguishing them from DDoS anomalies is important. [73] summarizes results from various studies on flash crowds including [74] where Jung *et al.* present an analysis of flash crowds and (D)DoS attacks on web servers. They pay special attention to characteristics which distinguish the two. They find that during flash crowds the client population has a significant overlap with the normal population. This stands in contrast to (D)DoS attacks. Furthermore, their results show that with flash crowds, the request per client rate is lower than usual and seems to adapt to the current performance of the server. In (D)DoS attacks the rate is stable and higher than usual. Based on these findings, [48] labels traffic emerging from topologically clustered hosts and directed to well-known service ports (e.g. port 80 for web servers) as flash crowd events. However, both this method and other promising approaches (e.g. [75, 76]) tend to typically depend on behavioral aspects which are expensive to track in high-speed networks.

2.3 Anomaly Detection Methods

Since the first attempts at detecting outliers or anomalies undertaken by Edgeworth in the 19th century [77], a vast amount of anomaly detection methods have been developed and used. An overview of the most important methods in different application domains, including intrusion detection, fraud detection, medical and public health anomaly detection, industrial damage detection, and anomaly detection in text data or sensor networks is presented in [38]. Other surveys take a narrower focus [78, 79], with [78] concentrating on the field of network intrusion detection in general and [79] dedicating a separate section to flow-based detection methods. The tutorial by Callegari [80] as well as [39] provide an overview of the most prominent methods used for detecting anomalies in network traffic, namely:

- Change-Point detection
- Kalman filter

- Principle Component analysis
- Wavelet analysis
- Markovian models
- Clustering
- Histograms
- Sketches
- Entropy

The first six items refer to techniques operating on time series or other forms of input data. They either report anomalies directly or output residual signals to be used with a simple threshold detector. The last three items, however, are not anomaly detection methods in a strict sense. They are pre-processing tools which help to summarize distribution-based features.

Feature distributions provide a more detailed view of network activity than traditional counter-based features, whilst still remaining lightweight enough to translate to large-scale and high-speed networks. This is typically not the case for methods which rely on detailed behavioral information of single hosts or groups of hosts.

We now review related work on approaches based on counter or distribution information, before switching our discussion to approaches which rely on entropy information.

2.3.1 Counter and Distribution-Based Methods

Counter Based Methods

Initially, most anomaly detection methods relied on features such as the number of forwarded packets, fragmented packets, discarded packets or bytes per time bin. These can be derived from the counters found in routers or other networked devices. In [81], Barford and Plonka present a project for a precise characterization of anomalous network traffic behavior from network flow data. They propose to look at the number of flows, packets and bytes per second. In [82], they refine their approach, proposing a wavelet analysis based detector. In their refined approach, they use both network flow and Simple Network Management Protocol (SNMP) data to extract the previous count metrics, whilst also adding new count metrics such as the average packet size. They carry out a true positive analysis using a trace from their campus network. They find that they can detect up to 95% of anomalies selected from a larger set of anomalies, where there is sufficient evidence that they are true anomalies.

In [83] Soule *et al.* make use of traffic matrices to capture the traffic volume exchanged between different points of presence. They first apply a Kalman filter to filter out the contribution of the normal traffic. The remaining signal is then inspected and analyzed for anomalies based on multiple characteristics and methods. They enact a thorough evaluation based on a combination of realistic workloads from a backbone network and synthetic anomalies. They find that the conventional generalized likelihood ratio (GLR) test [84] method performs best with a true positive rate of 100% and a false positive rate of 7%.

In [85] Lee *et al.* present a traffic collection algorithm for frequent collection of SNMP data that does not degrade (server) performance. To assess whether or not the algorithm can retain relevant information, they check how it impacts on the detection of volume anomalies. The detector used for this purpose is a threshold-based detector measuring the deviation from a mean value. A comparison of the detection results when using the original traffic collection algorithm and their new algorithm shows only some minor differences.

Distribution-Based Methods

Most approaches for anomaly detection in large scale networks rely (to some extent) on traffic-feature distributions. Some operate directly on empirical distributions, whilst others use summarization techniques such as histograms [16, 17] or Sketch data structures [18, 21, 86, 87]. Sketch-based approaches rely on a set of histograms where the elements are assigned to the bins using a set of different hash-functions. Both histogram and sketch-based summarization allows for the tuning of the amount of data to be stored and analyzed. Histogram-based methods do this by choosing an appropriate binning method and bin size, whilst sketch-based approaches select the number of hash functions and the number of bins per sketch. For the detection of abnormal changes, most methods rely either on entropy or distribution distance metrics. Prominent examples of approaches using distance metrics are [16, 88] where the authors make use of the Kullback-Leibler distance.[17]

---

[17] Note that the Kullback-Leibler distance actually corresponds to the Kullback-Leibler entropy or Rényi distance of order 1 ($\alpha = 1$) [89].

2.3.2 Entropy-Based Anomaly Detection Methods

Approaches that rely on entropy use Shannon entropy [20, 21, 48, 52, 90, 91], an approximation of Shannon entropy based on compression algorithms [19], the Titchener's entropy (T-Entropy) [23, 24] or some other generalized form of entropy [25, 26]. In [25], Ziviani *et al.* propose to use Tsallis entropy, a generalized form of entropy with parameter q, for the detection of network anomalies. By injecting DoS attacks into several traffic traces they search for the optimal q-value for detecting the injected attacks. While Ziviani *et al.* found that a q-value of around 0.9 is best for detecting DoS attacks, Shafiq *et al.* [26] found that they could optimize the detection of port scans of malware by using a q-value equal to 0.5. A different application of entropy is presented in [92], where the authors introduce an approach based on maximum entropy estimation and relative entropy. The distribution of benign traffic is estimated with respect to a set of packet classes and is used as the baseline for detecting anomalies.

2.4 Anomaly Classification

There are basically two fundamental approaches to classify anomalies. The first one is based on how an anomaly affects a system when it unravels. An analogy from medicine would be the classification of diseases based on the symptoms they cause. In practice, this might be more convenient than the second approach which classifies according to the mechanisms or processes involved. This is because symptoms are typically straightforward to observe or measure, whilst underlying mechanisms or processes are not. For example, similar symptoms might be found to actually have very different root causes when examined from a mechanism or process perspective. Examples of two network anomalies with similar symptoms, but different underlying mechanisms, are flash crowds and DDoS attacks. Classification according to mechanisms and processes is more accurate. However, as in medicine, the method actually used is ultimately decided by the question of cost and benefit. The extra analysis and measurement required to identify the mechanism or process is only undergone when deemed necessary. Unfortunately, this same approach generally does not work for network anomalies in high-speed networks. The processes and measurements (e.g. full packet traces) required for this form of analysis are either prohibitively expensive or simply not available[18].

As a consequence, even though the mechanisms and processes of most network anomalies are quite well understood, network anomaly classification in high-speed networks is almost always based on symptoms. As such, the following discussion of anomaly classification focuses on these approaches only.

However, note that while the classification itself is based on symptoms, the different categirues used in the classification are typically mechanism or process driven. Hence, the symptom based classification might be seen to wrongly class an anomaly considered from a mechanism or process perspective (e.g. DDoS instead of distributed scanning, if the scan intensity is high enough), even though this cassification would technically be correct from the symptom perspective.[19]

Broadly, there are two fundamentally different approaches to symptoms based anomaly classification. The first approach is to extract a *fingerprint* of an anomaly and to compare it to a series of other known fingerprints. The other method theoretically does not require a priori knowledge. It applies data mining and unsupervised learning technologies to identify the characteristics of a yet unknown anomaly, and to subsequently generate classes of anomalies with similar characteristics.

In the first case, a fingerprint typically consists of a combination of measurements such as “the bandwidth usage on the link to server X is 98%”, and observations such as “server X provides live TV streams”. When comparing this sample fingerprint to other fingerprints, we might find they match one with the label DDoS attack, but also one labeled “high-profile TV event (UEFA Champions League final, FIFA World Cup final ...)”. Note that the terms *pattern*, *fingerprint* and *signature* are often used synonymously to refer to some sort of description of an arbitrary item. These descriptions are then used by systems that try to find and/or classify such items. More precisely, in computer security the term pattern is typically used when talking about a description which specifies a characteristic sequence of bits and bytes in data streams (e.g. described by regular expressions). The term *fingerprint* is used to refer to a description which specifies the parameters and their values, or the required value ranges of a predefined parameter space (e.g. the parame-

---

[18] How do you measure the difference between a flash crowd and a DDoS attack on a web server when the only difference might be the intent of the user(s)?

[19] Because a specific set of symptoms has been associated (e.g. by machine learning mechanisms) with a specific mechanism or process driven class.

ters measured when doing an operating system fingerprinting of a host) and the term *signature* is used mainly when talking about the descriptions used by anti-virus software or intrusion detection systems.

In the second case, where unsupervised learning technologies are applied to identify the characteristics of a yet unknown anomaly, the concepts of patterns, signatures or fingerprints do not exist. However, note that if we do not repeat the unsupervised learning for each new classification task and instead map the attribute vectors to clusters generated earlier, these clusters could also be seen as a signature of a signature based classification system.

2.4.1 Anomaly Signature Based Methods

At a first glance, the term signature might be confusing when used in the context of anomaly classification. After all, the main advantage of anomaly detection is that it can detect both known problems and threats and as of yet unknown ones. How can we have signatures for the unknown? This obvious contradiction can be resolved in two ways. We use a system that analyzes anomalies and tries to find groups of similar anomalies by comparing them with each other and then outputting descriptions for these groups. These descriptions are then analyzed by an expert in order to identify whether they reflect a known or unknown anomaly, or whether they in fact do not represent anything useful at all. Such systems are discussed in the next subsection. The other way to resolve this contradiction is to only use anomaly signatures to identify anomalies that are already known. Anomaly classification based on signatures does exactly that.

One of the first proposals to use anomaly signatures for network anomaly detection can be found in [93]. In their paper *Fault Detection In an Ethernet Network Using Anomaly Signature Matching* Feather *et al.* propose to take the anomaly detection signals from anomaly detection done on a per feature basis[20] and to compare the resulting anomaly signal vector to a so called Fault Feature Vector. The Fault Feature Vector specifies the performance parameters and the corresponding value or value range required to match a specific fault. Performance parameters include the number of packets, the network load, the number of collisions or the number of new source addresses. The signatures used by Feather *et al.* are at least partially an example of signatures built on expert knowledge and based around and experience of common faults and their manifestations. However, since they also used a refinement strategy

---

[20] For anomaly detection, they use statistical methods (PAMS and AADS) described in [94].

after the initial definitions were tested with real data, they also depend on observations made in training data. Another example of a system that uses signatures that are to some extent constructed using expert knowledge and practical experience is [20]. In [20], Lakhina *et al.* use volume features and the Shannon entropy of the source and destination port and IP address traffic feature distributions to identify and classify anomalies. They argue that many anomaly types cause either a concentration or a dispersal of a specific traffic feature distribution and propose to use these characteristics to identify the type of anomaly in question. They first turn to an unsupervised clustering approach as it could potentially reveal as of yet unknown anomalies. In the next step, they analyze the clusters produced by the clustering algorithm finding that the resulting clusters do indeed represent different anomaly types. They assign labels to them accordingly. Finally, they propose to use these labeled clusters (i.e. signatures) for automated classification.

With the increasing popularity and sophistication of data mining methods, the automated extraction of signatures based on training data has become the primary choice of most anomaly classification systems. An example of a specialized classification system which solely handles the worm anomaly is described in [95]. An example of a signature-based anomaly classifier for anomalies found in the Border Gateway Protocol (BGP) routing update messages is presented in [96]. Here, Dou *et al.* use decision trees to learn the signatures of the impact of network anomalies such as worms and outages on BGP data.

A more recent example of decision tree based anomaly classification is [97], where Paredes-Oliva *et al.* make use of the descriptive power of these data structures to classify different types of network anomalies. To classify an anomaly, they first apply the FPmax* algorithm to mine frequent itemsets in flow data collected in a time interval of length T. These frequent itemsets are then fed into the classifier which uses the decision trees to assign one of the following types to each item set: (D)DoS, port scan, network scan, unknown[21] and no anomaly. To construct the decision tree, the authors applied the C5.0 machine learning algorithm[22] and fed it a set of labeled anomalies found in the GÉANT backbone network. Note that since their classifier can assign the label “no anomaly” to an itemset, it could basically be used for both anomaly detection and anomaly classification.

---

[21] This refers to anomalies in the training data which could not be assigned one of the other labels.

[22] The C5.0 algorithm was developed by RuleQuest Research [98] as an improvement to the C4.5 algorithm [99].

If we include the waste literature from the area of intrusion detection, we can identify many approaches which rely on the use of learning techniques to automatically generate signatures from labeled anomalies. An early example is [100], where the authors both compare and employ Support Vector Machine (SVM) based machine learning and neuronal networks to identify normal traffic, and four types of malicious activity found in the Knowledge Discovery and Data Mining competition 1999 (Knowledge Discovery and Data Mining competition 1999 (KDD99)) dataset.

2.4.2 Unsupervised Learning Based Methods

To overcome the limitations of signature based methods that can only classify anomalies whose properties we already know or have samples for, a significant number of anomaly classification systems classify anomalies using unsupervised machine learning techniques. These techniques do not require a priori knowledge on anomalies. They typically attempt to create clusters (groups) of similar items by looking at the data that is fed to the classification system. These clusters can then be analyzed by an expert to find out what kind of anomaly each of them represents (if any) and whether it is an already known or an as yet unknown anomaly. This final step is typically not required when one only wants to perform anomaly detection rather than anomaly classification. Separation of normal and abnormal data is possible if the following two assumptions about the data hold. Firstly, the amount of normal data must be far bigger than the amount of anomalous data. Second, the abnormal data must be statistically different from normal data [101]. Unsupervised learning methods do not require this expert cluster analysis process. This is the most likely reason why they are far more frequently used in anomaly detection systems (e.g. [101, 102]) than in anomaly classification systems. In actual fact, publications which employ unsupervised learning methods for anomaly classification mainly use them as a means of building the “signatures” required for the anomaly classification component. One example of such usage is presented by Lakhina *et al.* [20], which we discussed previously in the section on signature based approaches. Another example of such a usage is [103] where Filho presents a system that applies hierarchical clustering to map sets of anomalous flows extracted by an anomaly detection system to anomaly classes. More precisely, this classification algorithm works as follows. Firstly, the anomalous flow set is added to the flow sets that have already been labeled. Next, these flow sets are clustered using the following features as coordinates of the input vector: the average flow size in packets, the average packet size in bytes, the entropy of the distribution of packets, and the fraction of feature values in the full link traffic[23] that also appear in the anomalous traffic for ten different flow features.[24] The clustering process then outputs a taxonomy tree T, where each anomaly corresponds to a leaf. If the leaves in the smallest sub-tree containing all the siblings of the new anomalous flow set all possess the same label, then the new anomalous flow set is considered to have the same label too. If they have different labels, an expert must analyze the new flow set and assign a label to it. A graph-like visualization method reflecting the structure of the flows in this flow set is provided to help with this process.

2.5 Applications of Generalized Entropy in Other Fields

Generalized entropy has many applications in fields such as communication systems, physics, and biomedical engineering as well as in the broader context of complex systems.

An example from biomedical engineering can be seen in [104], where Torres *et al.* exploit the ability of multi-resolution entropies to show slight changes in a parameter of the law that governs the nonlinear dynamics of complex biomedical signals. To do so, they first apply a continuous wavelet transform to the signal and calculate the time evolution of the wavelet coefficients' Tsallis entropy in a sliding temporal window. Next, they analyze the principal component of the resulting multi-dimensional signal and apply the CUSUM [105] algorithm to identify abrupt changes in its mean.

However, since applications in fields other than anomaly detection are not the focus of this thesis, we refer the reader to Constantino Tsallis' *Introduction to Nonextensive Statistical Mechanics: Approaching a Complex World* [106] which presents and discusses a selection of paradigmatic applications in various sciences. [107] also offers a comprehensive bibliography of Tsallis entropy related publications.

---

[23] Not just the anomalous flows extracted by the detector but all of the flows seen in the interval where the anomaly was detected.

[24] Source and destination port and IP address, previous and next-hop AS numbers, source and destination AS numbers and input and output router interface.

2.6 Evaluation of Anomaly Detection Systems

As pointed out by Gates *et al.* [37], Ringberg *et al.* [108] and Sommer and Paxson [40], anomaly detection, and more specifically the evaluation of anomaly detection systems, is full of pitfalls and thus might never be fully accomplished. **Gates *et al.*** discuss nine assumptions often made in the domain of anomaly detection which they consider to be problematic:

- attacks differ from the norm
- attacks are rare
- anomalous activity is malicious
- attack-free data is available
- simulated data is representative
- network traffic is static
- false alarm rates $> 1\%$ are acceptable
- the definition of malicious is universal
- administrators can interpret anomalies

The first three assumptions are mainly an issue if the anomaly detection system is expected to report attacks instead of anomalies and if the attack to be detected by the anomaly detector might be tuned to look like normal activity. In the domain of anomaly detection in high-speed networks, these assumptions are rarely made. This is because the omnipresence of attack traffic from activity such as network or port scans is a known fact, as is the existence of benign anomalous activity such as flash crowds. This is also the reason why most approaches do exactly what the authors of [109] recommend. They begin by seeking to determine what malicious activities should be detected. Next, they check which (if any) characteristics of these activities appear anomalous. Finally, they design the system to detect them.

Another problematic assumption is that attack-free data is available. More precisely, it is problematic if *an attack* and *an anomaly* are considered to be equivalent; attack traffic such as scan traffic is typically omnipresent. This is why this assumption is often modified to “anomaly-free data is available”. Although it is probably still impossible to guarantee this for data collected in a high-speed network, more confidence can be placed in the veracity of this modified assumption. After all, an anomaly is something that deviates from the normal, whilst an attack does not necessarily have to do this. Moreover, if a system focuses on anomalies of a certain “size” - e.g. in terms of number of packets or flows involved - data which is free of such anomalies might be easier to find. Another solution to get attack-free (or anomaly-free) data would be to use simulated data. However, Gates *et al.* argue that there is evidence that an accurate simulation of real network data is probably impossible.

Another issue raised by Gates *et al.* is that authors often forget that low false positive percentages do not necessarily imply that the system is actually useful in practice. The false positive percentage does not include information on how many false positives per time interval are to be expected. If 1% is equal to five anomalies per day, an operator might find this number acceptable. However, this would prove less palatable in a system where 1% is equal to tens or hundreds of anomalies per day. Hence, a false positive rate should always be accompanied by information on the number of false positives per time interval.

Finally, the authors point out that an anomaly detection system should also provide some form of labeling component since the administrators cannot be assumed to be able to (or to have the time to) interpret anomalies. Furthermore, this component could also be used to make the system report only anomalies in which the operator is interested. This would thereby also address the problem that the definition of what is anomalous (malicious) is not universal.

Ringberg *et al.* discuss a less pessimistic view, but stress the need for simulation in evaluating anomaly detectors. Evaluation should be based on simulation, not on real data with some known anomalies in them. The main thrust of their argument is that real data typically only contains a "few instances" of a certain anomaly at a specific time of day. Simply considering a few instances makes it difficult to account for the fact that different instances of a certain type of anomaly can differ significantly (in volume, timing or other features). Furthermore, the highly dynamic background traffic also plays an important role in the detection process.

Simulation is a way of accounting for this problem. Anomalies can be parameterized with their many variants injected into background traffic that is either simulated or taken from virtually anomaly-free sections of real network traces. Unfortunately, comprehensive studies on how to parameterize anomalies are still lacking [33,46]. Nonetheless, a few studies do provide fragments of analysis which can still prove useful [48,64–66,74,81,82]. These studies discuss parameters such as the distribution of the IP addresses of both attacker(s) and victim(s), the protocol(s) or the transport layer (L4) ports used, and the timing etc. for a broad set of volume anomalies. While they do not provide statistics detailed enough to compile a rigorous outline for the parameterization of these anomalies, they do provide valuable insights which prove constructive when setting up one's own simulation models and parameters. In [33], Brauckhoff *et al.* go further, presenting three anomaly models which they derive from real anomalies found in their network traces. They also present FLAME which can be used to model and inject anomalies into flow traces.

Sommer and Paxson discuss why applying machine learning in intrusion detection[25] is harder than in other domains. They do so with focus on network-based systems reliant on algorithms which are first trained with reference input so as to learn its specifics (threshold(s), model(s), etc.), before being exposed to input for the actual detection process. The reason why they consider this area to be more difficult is the premise that anomaly detection is generally good at finding novel attacks. More precisely, Sommer and Paxson state that the main strength of machine learning lies in finding activity which bears similarity to something previously observed, but for which a precise a priori description is not held. Since novel attacks are only novel if they have not previously been seen, the authors conclude that this premise is not well aligned with the strength of machine learning.

However, this premise is not the only characteristic of anomaly detection that does not fit the requirements of machine-learning. They also list the following additional characteristics:

- a very high cost of errors
- lack of training data
- a semantic gap between results and their operational interpretation
- enormous variability in input data
- fundamental difficulties in conducting sound evaluation

In summary, Sommer and Paxson state that it is safer to only use machine learning to find something that has previously been seen, but for which a precise a priori description does not exist. Other requirements, such as sufficient training data and a sound evaluation methodology, remain problematic.

2.6.1 Data Sets

The problem of the non-availability of network traces is mainly due to privacy laws. Network traces typically contain privacy-sensitive information such as packet payload and/or IP addresses. Packet payloads can be used to build full-fledged user profiles. IP addresses are also highly sensitive. If a user's IP

---

[25] Here, intrusion detection is probably too general. What they actually mean is anomaly detection based intrusion detection.

address is known, packets or flows can be linked to the user in question. This again could be exploited to build profiles of who communicates with whom.

Anonymization of network traces could resolve these issues. However, the vast amount of methods and tools addressing network data anonymization [110–112] and/or privacy-protected network data sharing [113, 114] do not seem to foster public access to network traces. The root cause for this situation might be twofold. On the one hand, it has been shown [115–118] that it is very difficult to anonymize traces so as to prevent them from leaking valuable information. On the other hand, it is difficult to quantify the impact of anonymization on the utility of the data. Brauckhoff *et al.* [119, 120] studied this impact in regards to anomaly detection. They found that anonymization significantly degrades anomaly detection performance. As the strength of anonymization increases, more and more information relevant to the detection process is removed. Hence, using anonymized network traces is not an option if we want to assess the performance of an anomaly detector with respect to non-anonymized data.

Another problem, especially with network flow traces, is that in order to reduce the load on components[26] involved in the traffic capturing process, some sort of filtering or sampling is applied. Most network operators configure their capturing devices to apply packet sampling with rates of 1 out of 100 packets or lower. In our thesis, we investigate the impact of packet sampling on anomaly detection metrics in more detail. Our findings are similar to those presented by the authors of [121, 122].

Some notable sources of network traces are:

- The packet traces published and maintained by the MAWI [123] Working Group of the WIDE [124] project. Among other traces, they provide daily packet header traces for different, mostly trans-Pacific lines with link speeds of up to 150Mbps. Per sampling point and day, an approximately 15 minute-long trace is made available to the public. The data is anonymized using a prefix-preserving anonymization scheme.[27]
- The Simpleweb/University of Twente traffic traces data repository [125]. The repository contains a series of six packet header traces[28] from relatively small sized stub networks and a NetFlow v5 trace from August 2007 captured in a /16 university network. The IP addresses of these

---

[26] E.g. a router. The primary task of a router is to route packets, not to generate flow data.

[27] They use the tool *wide-tcpdpriv* with the options *-A50 -C4 -M99 -P99*. The tool and the parameters used can be found in the *tcpd-tools* source code package available from the MAWI Working Group Traffic Archive homepage `http://mawi.wide.ad.jp/mawi/`.

[28] One from 2002, two from 2003 and 2004 and one from 2007.

traces have been anonymized using a prefix-preserving anonymization strategy

- The NetFlow data repository of the Internet2 backbone network [126] (formerly Abilene). This repository provides flow data anonymized by zeroing the last 11 bits of any non-multicast IPv4 address.[29] Moreover, the flow collectors perform packet sampling at a rate of one out of 100 packets. Internet2 offers access to their NetFlow data on request.
- NetFlow data from the GÉANT [128] backbone network. GÉANT grants access to flow data on request. Its flow traces are not anonymized. However, since their flow collectors perform packet sampling at a rate of 1 out of 1000 packets, a lot of potentially useful information is discarded.

Unfortunately, almost all of these traces are unlabeled. Information about the anomalies they contain is not available. This is by far the biggest problem. A notable exception are the daily packet traces published and maintained by the MAWI [123] Working Group. In [86], Dewaele *et al.* made a first attempt at labeling the traces from 2001 to 2006. Firstly, they proposed and employed a sketch-based anomaly detection approach to locate anomalous traffic patterns. Following on from this, they performed a manual inspection[30] of the anomalous traffic to confirm and label it, or to discard it.

In [129], a second attempt at providing labels for this dataset is made. In this case, the anomalies are located and labeled based on the outputs of four different anomaly detectors.[31] The labels are accessible through MAWILab [130] ,[32] a database that helps researchers to evaluate their traffic anomaly detection methods.

An alternative way of addressing the problem of unlabeled traces is to take a community-based approach. In [131], Gates *et al.* discussed requirements for evaluating data and proposed a community-based approach for the labeling of released traffic traces. A similar proposal can be found in [132], where Ringberg *et al.* present WebClass, a tool to store and compare labels that have been assigned to a trace by different domain experts. Moreover, according to the MAWILab [130] homepage, MAWILab people also ask for help from

---

[29] As of August 2012, IPv6 addresses are anonymized by zeroing the last 80 bits [127].

[30] They did so by looking at the traffic features associated with the detection. Only *flows* carrying more than 1% of the total volume of the traffic are looked into by manual inspection of the traces by a network expert.

[31] Principal Component Analysis, the Gamma distribution, the Kullback-Leibler divergence and the Hough transform.

[32] `http://www.fukuda-lab.org/mawilab/`

the community. They encourage researchers to submit both their own results and their detectors. However, the proposed collaborative system designed to simplify and automate contributions has not yet been implemented.

In this thesis, we use network traces captured at the border of the SWITCH network. These traces contain unsampled and non-anonymized NetFlow data. Since this data contains sensitive information, the dataset cannot be shared with third parties, unless they visit us to work with it on-site under a non-disclosure agreement. Interested third parties can apply to the authors for such a visit.

Synthetic Datasets

As pointed out by Ringberg *et al.*, synthetic datasets and simulation tools should play an important role in the evaluation of anomaly detection systems. The DARPA data sets from 1998, 1999 and 2000 [133] - sometimes also called the Lincoln Lab data sets - are probably among the first synthetic datasets released to the research community. It might initially appear surprising that the DARPA data set from, for example 1998, still plays a role in intrusion detection research (see: [134,135]). After all, not only does it lack recent attacks, but it has also been criticized for its unrealistic traffic (see: [136]).[33] However, better options are so scarce that researchers are virtually forced to still use it. According to [137] these data have provided significant contributions to research on anomaly detection. Of 276 research studies published between 2000 and 2008, the LL data and its derivative, the KDD dataset, have been used in over 50% of these studies. Another 15% used additional attack data.

An alternative to ready-made traces are tools which can build custom traces by generating either synthetic background traffic, synthetic attack traffic, or both. Whilst a number of tools for generating such traces at packet level exist (e.g. [138–141]), Harpoon [142] and FLAME [33] are the only notable tools that do the same at flow level.[34]

In [143, p.1], the authors of Harpoon write that

> ...Harpoon is a flow-level traffic generator. It uses a set of dis-

---

[33] Both, the synthetic background traffic and the attack traffic recorded in a testbed have been criticized.

[34] Note that packet traces could be converted to flow traces by replaying them versus a (virtual) flow meter. However, the generation and conversion of packet traces does not scale for traces representing traffic in high speed networks.

> tributional parameters that can be automatically extracted from NetFlow traces to generate flows that exhibit the same statistical qualities present in measured Internet traces, including temporal and spatial characteristics. Harpoon can be used to generate representative background traffic for application or protocol testing, or for testing network switching hardware.

The newer of theses two tools, FLAME, offers additional flexibility. It can be used to generate flow traces from traffic models, but also has the capacity to take these models as a basis for the injection (or removal) of flows into existing network traces. Furthermore, FLAME comes with a set of predefined network traffic models extracted from real world network traces: DoS attacks, scans from several scan tools and spam.

In our work, we make use of FLAME to generate synthetic anomalies and to inject them into captured network traces. The captured network traces also serve a second purpose. Similar to [33, 140, 141], we use them to extract our models. However, our focus is different from those of [140, 141]. Our models reflect flow-level rather than packet-level traffic characteristics and we do not restrict ourselves to DoS attacks.

2.6.2 Evaluation Metrics

The task of an anomaly detection system is to detect anomalies. When performing this task, there are four possible outcomes when the detector decides whether or not the data under scrutiny contains an anomaly it should report:

- True Positive (TP): The data contains an anomaly and the detector reports it.
- False Positive (FP): The detector reports an anomaly but the data does not contain one.
- False Negative (FN): The detector says everything is normal but the data does contain an anomaly.
- True Negative (TN): The detector says everything is normal when everything is normal.

Hence, an ideal anomaly detection system should only produce TP and TN outcomes. To rate and compare all non-ideal anomaly detection systems, one could simply measure their False Positive Rate (FPR) and False Negative Rate (FNR). The FPR corresponds to the share of benign activities mistakenly reported as anomalous and the FNR denotes the share of anomalies missed by the detector.

Precision and recall are two related measures. Precision is defined as $\frac{\#TP}{\#TP+\#FP}$, which refers to the proportion of anomalies reported by the detector which turn out to be true anomalies. Hence, a high precision means that an operator wastes less time on following up detections where there is no anomaly, and can spend more time investigating true anomalies. Recall, on the other hand, is defined as $\frac{\#TP}{\#TP+\#FN}$ which is the share of true anomalies detected compared to the total number of anomalies an ideal anomaly detector would detect. To assess the influence of the sensitivity parameter S (e.g. the threshold of a threshold-based detector), these measures are often compiled into Precision-Recall (PR) curves. These curves are obtained by plotting the precision versus the recall value for each value of S. Based on this graph, one can then select the "best"[35] operation point for the detector.

Another means to find the "best" operation point of an anomaly detector is through using Receiver Operating Characteristic (ROC) curves [144, 145]. Instead of plotting precision versus recall, ROC curves plot the True Positive Rate (TPR) versus the FPR for different values of S. If the axes are of the same scale and if the costs for FNs and FPs are the same, the best operating point is tangent to a line with a slope of 45 degree. Note that ROC curves and PR curves are closely related. In [146] Davis and Goadrich show that the fact that the ROC curve and PR curve for a given algorithm contain the same points for any dataset leads to the theorem that a curve dominates[36] in ROC space if and only if it dominates in PR space. Moreover, Davis and Goadrich show the existence of the PR space analog to the convex hull in ROC space. The convex hull is of interest since all points on it are achievable. If we have two neighbouring points p_1 and p_2 reflecting two different classifiers[37] c_1 and c_2 with (FPR_{p_1}, TPR_{p_1}) and (FPR_{p_2}, TPR_{p_2}), then it is possible to construct a *stochastic classifier* that interpolates between them by selecting classifier c_1 with probability p and classifier c_2 with probability $(1-p)$. The resulting classifier has an expected false positive rate of $p * FPR_{p_1} + (1-p) * FPR_{p_2}$ and an expected true positive rate of $TPR_{p_1} + (1-p) * TPR_{p_2}$. When verifying whether a similar form of simple interpolation exists also in the PR space, Davis and Goadrich found that this is not the case.

---

[35] Typically based on the estimated cost of a false negative (incident handling costs, reputational damage etc.) vs. those of a false positive (e.g. cost to do forensics etc.).

[36] According to [147], one curve dominates another curve, if all other curves are beneath it or equal to it.

[37] E.g. from two detectors using the same detection algorithm but distinct parameters S, but also two detectors using distinct algorithms and the same (or different) parameter S.

Another analytical tool related[38] to ROC curves is the error diagram. In [149] Molchan introduces the error diagram as a means of measuring the success of earthquake prediction strategies. While a ROC curve plots the TPR versus the FPR, an error diagram plots the miss rate $n = \frac{\#FN}{\#TP+\#FN}$ versus the alarm rate $\tau = \frac{\#TP+\#FP}{\#TP+\#FP+\#TN+\#FN}$. In the time series context, the miss rate is the number of time slots in which an anomaly was present but no alert was raised. The alarm rate corresponds to the share of time slots for which the system reports alarms. For an ideal detector, the alarm rate would correspond to the anomaly rate and the miss rate would be zero. In contrast to ROC curves, error diagrams provide a view of system performance which is stronlgy centered on economics. Missing true alarms and investigating false alarms costs money. Hence, minimizing both of the components of the error diagram is directly related to saving money. Furthermore, in [150], Molchan *et al.* describe how the error diagrams can be used to find an optimal operation point that minimizes an arbitrary cost function $\phi(n, \tau)$. However, their approach does not seem to be able to handle cases where the cost of an alarm or an FN depends on the type of the event.

While ROC curves are a convenient tool to display key performance parameters of anomaly detectors in a clean and compact way, they should nevertheless be used with caution. The following issues have been raised in the past:

- **Facett [151]:** Detector performance and, as a result, ROC curves themselves can vary significantly across different datasets. This has two implications: (1) a direct comparison of detection performance based on ROC curves should only be carried out if the ROC curves are derived from the same dataset and (2) to draw a more general conclusion about detector superiority, one should use a series of representative data sets.
- **McHugh [136]:** Different detectors use different units of measure and follow different strategies to match detection results to the ground truth. Examples of different units of measure include different sizes of data aggregation intervals or different bases for input metrics. For example, one detector might log the number of packets seen per 5 minute interval, while another does the same but for a 4 minute interval. One detector might calculate the entropy of source IP addresses based on the number of packets per source IP address, whilst the other does the same based on the number of flows. To avoid bias, the detectors should

---

[38] According to [148], the FPR and the alarm rate τ are of similar size, and the ROC curve is almost a mirror image of the error diagram.

use the same units of measure. McHugh argued that the strategy used to match the detection results to the ground truth,[39] must be chosen with care.

- **Axelsson [152]:** Anomalies to be detected by the detector (TP or FN) are much less frequent than normal activity. Intrusion detection may require FPRs much smaller than 0.1% to be effective.
- ROC curves do not consider the notion of costs. While it might be easy to find the optimal operating point where the costs for FNs and FPs are identical and fixed,[40] identifying where they are not is far from straight-forward. Solutions mitigating this problem include cost-based modeling approaches [153], transformations of ROC curves facilitating cost-related comparison [154] or explicit computation of the expected costs of each detector operating point [155].

In this thesis, we use ROC curves for the comparison of different configurations of the anomaly detection component of our entropy telescope. More precisely, we use them to compare different anomaly detection algorithms and also different sets of input metrics. To avoid bias, we use the same unit of measure (5-minute intervals) for all measurements and we consider consecutive detections as a single event.

---

[39] E.g. do consecutive detections of an anomaly lasting multiple time intervals count as one TP or as multiple TPs?

[40] If the axes of the ROC plot have the same scale, the best operating point is tangent to a line with a slope of 45 degree.

Chapter 3

Impact of Packet Sampling

In high-speed networks, packet sampling methods are widely employed as a means of reducing the amount of traffic data measured. Sampling strategies can vary significantly depending on the traffic data measured, i.e. packet traces or flow traces, and the purpose of the measurement. A common sampling strategy to measure the application mix based on packet traces is to perform deep packet inspection on the first few[1] packets of a connection only. With respect to flow data collection, the most popular strategies are random packet sampling and the sampling of every n-th packet. Flow meters employing these strategies generate flow data based on the sampled packets only. Hence, devices such as Cisco routers first apply packet sampling before they forward the sampled packets to the flow metering part of the router. The flow metering part of the router then simply generates flows following the same strategy and flow definition (see 1.2.2) as before but based on sampled packets only.

Packet sampling has several benefits including smaller flow tables,[2] fewer loads on the flow processing device and less flow data to be exported and stored. However, there are also several drawbacks. A key problem of packet sampling is that it is an inherently lossy process. It results in an incomplete and, more significantly, biased approximation of the underlying traffic trace. For example, if we apply one of the popular sampling strategies such as *ran-*

---

[1] For the Protocol and Application Classification Engine from *ipoque*, this is e.g., 1-3 packets for unencrypted traffic and 1-20 packets for encrypted traffic [156].

[2] A flow table has one entry per active flow. It is used to store and update the information relating to this flow, such as the number of packets and bytes associated with it.

dom packet sampling or *every n-th packet*, small flows consisting of a few packets only are likely to be missed entirely. The more packets a flow contains, the higher the chance that at least one of its packets will be sampled. As a consequence, the bias of the approximation does largely depend on the underlying traffic mix.

As a consequence, sampling can also have a significant impact on the entropy or volume time series which serves as input to many anomaly detection systems in high-speed networks. To shed some light on this issue, we performed an empirical evaluation of the impact of packet sampling and traffic characteristics on various entropy and volume metrics. This chapter presents the motivation, methodology and detailed results of this study. Our most important finding with regard to anomaly detection is evidence which shows that entropy is less affected by sampling and that traffic mix characteristics can compensate or even boost anomaly visibility in sampled views up to sampling rates of 1 out of 10,000 packets.

3.1 Introduction

Traffic sampling has emerged as the dominant means of summarizing the vast amount of traffic data continuously collected for network monitoring. The most prevalent and widely-deployed method of sampling traffic is *packet sampling*, where a router inspects only a subset of packets and records its features such as source and destination IP address and port numbers, protocol, and flags. Depending on the sampling strategy, the subset is either constructed by selecting every n-th packet (one out of n sampling) or by selecting every n-th packet on average (uniform random sampling). Packet sampling is attractive because it is computationally efficient, requiring minimal state and counters, and is implemented in most high-end routers today (e.g. with NetFlow [35]). As such, many providers of high-speed networks are now using packet sampling to obtain rich views[3] of the traffic directly from their routers.

Nonetheless, whilst it is attractive because of its efficiency and availability, sampling is an inherently lossy process. It discards many packets without inspection. As such, sampled traffic is incomplete. More importantly, it offers a biased approximation of the underlying traffic trace, as small flows are likely to be missed entirely. Previous work has largely focused on an-

---

[3] Flow formats such as Cisco NetFlow or IPFIX support a rich set of flow features which can be used to compose different views of the traffic.

alyzing this bias, devising better sampling strategies [157], and recovering statistics (moments and distribution) of the underlying traffic trace using inference [158–160].

It might therefore be surprising that sampled traffic views have been used for anomaly detection with considerable success (see: [20, 161]). Since this appears counter-intuitive, it is important to understand how and why it is possible. Is it because if an anomaly is of a certain size, it distorts the measurements from sampled data enough to remain detectable? How does the size of an anomaly or the choice of metrics impact on these results? How complete are the detections revealed by these methods on sampled traffic? And what impact on traffic mix can be seen when no anomaly is present? This mix is likely to be quite different when we compare the characteristics reported from sensors placed on a link interconnecting two research networks with those reported from sensors placed on a link connecting a residential network to the Internet. In contrast to the second case, we might see little or no web traffic on the link between the research networks, as research networks typically have their own Internet uplinks. Instead, we might see a lot of large file transfers from distributed infrastructures such as compute grids or other traffic to and from services located inside of the research networks.

Unfortunately, when we started to look into this topic, there was little previous work on how sampling impacts network monitoring applications and, in particular, anomaly detection. The publications which did investigate the impact of packet sampling focused on a wide variety of different aspects. In [162] Duffield *et al.* study the problem of estimating flow distributions from sampled flow statistics. In [163], Estan and Varghese look into the accuracy of sampling with regard to accounting. Two notable studies related to the impact of sampling with respect to anomaly detection published at the same time as our work [28] are [164] and [121]. In [164], Mai *et al.* analyzed how packet sampling impacts three specific portscan detection methods, TRWSYN [165], TAPS [166] and entropy-based profiling method of [20, 167]. This work was extended to analyze the impact of other sampling schemes in [121]. Both studies conclude that packet sampling significantly degrades detection performance using these detection methods. While this is in line with our intuition, the studies do not answer a more basic question: How does packet sampling impact the *input data* to the detectors? If we look at the impact of sampling on the *input data* instead of the detection results of a specific detector, we remove any dependency from specific detection techniques.

In contrast to the impact of sampling, the impact of the traffic mix has largely been ignored so far. The only notable study covering the impact of traffic mix characteristics on anomaly detection on sampled views is [122]. Based on sampled views from routers of two large scale networks, the authors inspect the propagation of several anomalies from networks with different traffic mix characteristics. Their findings suggest that traffic mix characteristics can be an important factor. However, because they did not have the unsampled traffic traces, they lacked information about the original structure of the baseline and the anomalous traffic. It was therefore not possible to quantify the impact, nor investigate it at different sampling rates. We address these issues with an empirical evaluation of the impact of packet sampling and traffic characteristics on various volume and entropy metrics used by many detection methods [19, 20, 82, 90, 168]. Starting with flow records collected during the Blaster and Witty worm outbreak, we reconstruct the underlying packet trace and simulate packet sampling at increasing rates. We then use our knowledge of the Blaster anomaly to build a baseline of normal traffic (without Blaster or Witty), against which we can measure the anomaly size at various sampling rates. This approach allows us to evaluate the impact of packet sampling on anomaly detection without being restricted to (or biased by) a particular anomaly detection method.

As a starting point, we investigate how packet sampling impacts the three principal volume metrics; number of bytes, packets and flows per time bin. We find that anomalies that impact heavily on packet and byte volume will stand out even in sampled traffic. Whilst this is ultimately good news, byte and packet volume metrics remain highly variable which makes it very difficult or even impossible to use them to identify small and medium scale events. This is even truer of anomalies which mainly impact flow counts, such as distributed scans, or several forms of denial of service attacks. To detect these types of anomalies, detection schemes based on the number of flows per time bin would be best. However, we found that this metric is heavily influenced by sampling, limiting its usefulness with sampled flow data.

Next, we study the impact of packet sampling on entropy metrics by evaluating how effective entropy is at exposing worm-like anomalies at increasing sampling rates. Our results here are surprising: we find that while volume metrics are significantly affected by packet sampling, entropy metrics are relatively undisturbed. In particular, our data showed that the Blaster worm is heavily dwarfed by sampling when measured in flow counts but remains largely unaffected when looking at entropy metrics. Our findings underline that even though packet sampling produces imperfect traffic views for anomaly detection, there are metrics (such as entropy) that allow us to harness useful information in sampled traces. Finally, we make use of traces collected at different collection points to analyze the role of the traffic mix. We measure the size of the Blaster and Witty worm anomaly at various sampling rates and collection points and find that at some collection points, the negative impact of packet sampling is compensated by, and can even boost anomaly visibility in sampled views. For some of the traffic features and collection points, sampled views outperform unsampled views even at sampling rates of 1 out of 10,000 packets.

The remainder of this chapter is organized as follows. First, we provide an overview of our methodology. We introduce our dataset, explain how we derive both packet traces and corresponding flow data for sampled and unsampled views, discuss the set of traffic features used for our evaluation and conclude with a description of how we measured the visibility of an anomaly independent of a specific anomaly detection method. Next, Section 3.3 presents our evaluation of the impact of packet sampling which is then extended by a study of the impact of the traffic mix in Section 3.4. Finally, we conclude and outline directions for future work in Section 3.5.

3.2 Methodology

Our study is based on the following building blocks:

- A dataset containing well known anomalies.
- A method to apply packet sampling to data described by flow traces.
- A measure to quantify the impact of packet sampling and the traffic mix on an anomaly.

3.2.1 Dataset

For this study, we use two week-long extracts from our comprehensive set of NetFlow traces collected by the border routers of the Swiss Education and Research Network (SWITCH) [34]. Since the traces were collected from all[4] border gateway routers of the SWITCH backbone and since these routers do not apply any sampling technique, we have a complete view of all Internet traffic that enters and leaves the network. Furthermore, it is important to note

---

[4] In 2003 and 2004, the SWITCH Network had four border gateway routers. See 1.4 for an overview of the SWITCH network.

that the networks to which the border routers are attached are quite different: private peering with an international research network only, peering with international carriers only, or combinations of the two. Thus, the traffic mix seen by each of them differs significantly.

One problem with the collection of flow information is that even if the routers do not apply sampling, we need to be sure that non-deterministic data loss due to CPU overload, overfull flow tables or line problems is negligible. Our analysis of the CPU load, fill-level of the router flow tables and the sequence numbers of the exported NetFlow packets showed that this criteria is met (loss rates < 1%).

The first week-long extract was collected between the 8th and 15th of August 2003 and contains the well-known *Blaster* worm. The Blaster worm is one of the most extensively analyzed Internet worms. First observed on August 11, 2003, Blaster uses a TCP random scanning strategy with fixed destination and variable source port to identify potential infection targets. Specifically, the infected host tries to connect to TCP port 135 on the target machine. When trying to connect, Blaster selects either a random IP address (with a probability of around 60%) or an IP address derived from the local IP (with a probability of around 40%) and then scans a block of 20 subsequent IP addresses in the chosen network. With respect to the network under observation, Blaster reached its peak activity on August 11, 2003 between 20:00 and 21:00 UTC. In this hour, around 5500 external IP addresses scanned (and eventually attacked) up to 1.2 Mio. IP addresses in the SWITCH network.

The primary reason to use the Blaster data as basis for our measurements is that this anomaly is well understood. Moreover, it is representative of the many anomalies which are visible in the number of flows per time bin, a metric that is biased significantly by packet sampling, but hardly visible in the other volume metrics. The Blaster worm is therefore an ideal candidate for our analysis of the effect of packet sampling on anomaly detection metrics in Section 3.3.

The second week-long data set was collected between the 17th and 21st of March 2004, during the outbreak of the *Witty* worm. It is used to complement the first trace in our analysis of the impact of the traffic mix in Section 3.4. The reason for selecting the Witty worm is that its characteristics are both well-known and very different from those of the Blaster worm:

1. Witty uses UDP random scanning for target identification while Blaster uses TCP.
2. Witty infected only about 15,000 hosts while Blaster infected between

200,000 and 500,000 hosts worldwide.

3. Witty uses a fixed source port and variable destination port while Blaster uses a variable source port and fixed destination port.

3.2.2 Reconstructing Packet Traces

A prerequisite to studying the impact of packet sampling is to have unsampled packet traces. Unfortunately, packet traces from high-speed networks rarely exist. They consume hundreds of gigabytes of storage space per hour making them difficult to collect and hard to store.

As an alternative, we reconstruct packet traces from flow data. We claim that this reconstruction is fairly accurate if things such as the packet content or inter arrival times of packets are not of interest. Instead, if primary interst is focused on aggregate information such as the number of packets to port 80 in a time window of length T with T in the range of several minutes and where most flows have a duration significantly smaller than T - then the reconstruction is hghly useful. In this case, the problem of distributing the packets of a flow to the correct time windows[5] is less relevant, as there is a high chance that all packets of the flow fall into the same time window.

In our study, the aggregation interval size is equal to the maximum flow duration L of 15 minutes. We can confirm that most flows last less than one minute (more than 99% of the total number of flows). Therefore, deviations from measurements with real packet traces occur only if a flow crosses the border of an aggregation interval and its packets have to be distributed to two different intervals.

By choosing the following packet-trace reconstruction algorithm, we preserve (on average) the often assumed (see: [169], [74]) constant throughput property of flows to reduce errors in case of splitting a flow across interval boundaries:

---

[5] By “correct time window” we refer to the time window in which they would have appeared in the real packet trace.

Algorithm 1 PACKET-TRACE RECONSTRUCTION

```
for all $f$ in flowtrace do
    $packetsize = \lfloor \frac{f.bytes}{f.packets} \rfloor$
    $reminder = f.bytes - packetsize \cdot f.packets$
    packet = get_packet_from_flow(f);
    for $i = 0$ to $f.packets$ do
        packet.time = get_random_time_in_interval($f.start, f.end$);
        if $i < reminder$ then
            packet.size++;
        end if
        write(packet);
    end for
end for
```

The additional byte for *reminder* packets is due to the fact that the number of bytes of a packet is an integer, whereas the number of bytes in a flow divided by its number of packets is not necessariliy so.

Note that the constant throughput assumption is supported by [170] where the authors present empirical evidence that the constant throughput property is a good approximation of the behavior of large flows (heavy hitter, elephant flows) while still being a reasonable approximation for small ones (mice flows).

3.2.3 Packet Sampling

Having reconstructed the packet traces from our NetFlow data, the next step is to apply packet sampling to those traces. For our study, we use the following five sampling rates:

- 1 out of 10
- 1 out of 100
- 1 out of 250
- 1 out of 1000
- 1 out of 10000

With these sampling rates, we include sampling rates typically found in production or research networks such as the GÉANT [128] network with a sampling rate of 1 out of 1000 or the Abilene network (now part of the Internet2 network [171]) with a sampling rate of 1 out of 100. However, note that we use the sampling rates 1 out of 250 only in the first part, and 1 out of 10,000 only in the second part of our study. The reason for this is that we found that we needed higher sampling for the second part of the study in order to identify up to which sampling rate packet sampling can improve the visibility of anomalies in sampled traces.

The sampling method used is random probabilistic packet sampling. Therefore when sampling at a rate of p we independently select each packet with a probability of p or discard it with a probability of $1 - p$.

With the sampled traces constructed in this way, we could start to investigate the impact of packet sampling on volume and per packet feature entropies. But since we also want to investigate the impact on per flow feature entropies, we need to get the flow data corresponding to the now sampled packet trace. One way to achieve this would be to emulate the flow generation as is done by, for example, NetFlow exporting routers. Unfortunately, the process of how routers construct flows is not entirely deterministic.[6] This makes reconstruction carried out in this way problematic [9]. Therefore, instead of trying to emulate a certain router behavior for each time window, we simply aggregate all packets with the same five-tuple (source IP, destination IP, source port, destination port, protocol) into a single flow. While this might introduce some error, we believe the chance that two hosts use exactly the same ports for two or more connections within several minutes to be small. Operating systems do not reuse ports immediately but instead wait for some time before doing so. Even if they were reused within a single time window, it is not at all certain that the port would be used for a connection to the same host.

3.2.4 Feature Set

For our analysis, we compiled a set of 15 metrics, of which 11 are frequently used as input to anomaly detection systems. All of these metrics are computed on a per time window basis with a window of length T equal to 15 minutes:

- Volume metrics:
    + number of flows → **Fcnt**
    + number of packets → **Pcnt**

---

[6] Well, actually it is. However, this depends on many factors such as timeouts or the fill level of the flow table, which are impossible to simulate in retrospect without having more information than just the flow level data.

  - number of bytes → **Bcnt**
  - number of unique source IP addresses → **SrcIP4cnt**
  - number of unique destination IP addresses → **DstIP4cnt**
  - number of unique source port numbers → **SrcPcnt**
  - number of unique destination port numbers → **DstPcnt**
- Entropy metrics: The Shannon entropy of...
  - source IP address distributions of flows → **SrcIP4e**
  - destination IP address distributions of flows → **DstIP4e**
  - source port number distributions of flows → **SrcPe**
  - destination port number distributions of flows → **DstPe**
  - source IP address distributions of packets → **p-SrcIP4e**
  - destination IP address distributions of packets → **p-DstIP4e**
  - source port number distributions of packets → **p-SrcPe**
  - destination port number distributions of packets → **p-DstPe**

The four metrics rarely seen in such systems are entropy metrics based on packets. Our evaluation will provide some insight as to why not using them makes sense.

For our study of entropy metrics, we selected the most popular form of entropy: the Shannon entropy. The Shannon entropy is defined as follows for a probability distribution $P(X)$:

$$S_s(X) = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i) \tag{3.1}$$

where X is a random variable over a range of values $x_1,\ldots,x_n$ and $p(x_i) = p(X = x_i)$. Since we do not have true probability distributions, only measurements of the number of occurrences or *activity* a_i of x_i in a specific time window of length T, $p(x_i)$ needs to be replaced by the sample probability derived from the sample activity distribution as follows:

$$p(x_i) = \frac{a_i}{\sum_{j=1}^{n} a_j} \tag{3.2}$$

In our context, if we calculate, for example, the Shannon entropy of a source IP address distribution, a_i would refer to the number of occurrences of IP address x_i. Note tha,t for the sake of brevity, we do not repeat the fact that the distributions are actually sample distributions. We refer to them as probability- and activity distributions.

We calculated these metrics on a per *direction* and per *protocol* basis. We distinguish the directions *IN* and *OUT* and the protocols *TCP* and *UDP* resulting in a total number of four sets of these 15 metrics.

Direction IN refers to flows (or packets) from sources located outside of the SWITCH network to destinations inside of this network.
Direction OUT refers to flows (or packets) from sources located inside the SWITCH network to destinations outside of this network.

Because of this large number of metrics, we shall not discuss all of the results. We instead choose to focus on the metrics which reveal the most relevant or most surprising results. More specifically, when discussing results from the dataset around the Blaster worm, we focus on the TCP protocol and flows with direction IN. When discussing results from the data set around the Witty worm, we focus on the UDP protocol and flows with direction IN. The reason for this is that the worms used these protocols to scan for (or to attack) vulnerable hosts. There were almost no Blaster or Witty infected hosts inside the SWITCH network.

3.2.5 Baseline

Before we can start with our analysis of the effect of sampling on volume and entropy metrics with focus on anomaly detection, we need a method to quantify and measure the factor by which an anomaly disturbs them. In anomaly detection systems, this is typically done by comparing a predicted value, whose prediction is based on a model of the so-called baseline or "normal" behavior of a specific metric, to the true value. Unfortunately, the baseline behavior cannot be modeled entirely accurately,[7] rendering an accurate measurement of the disturbance caused by an anomaly impossible. To compensate for this inaccuracy, most detection systems define a minimum size of the distortion (threshold) required to raise an alert; *the bigger the distortion, the safer it is to assume that it is truly caused by an anomaly*.

For our study, we have the advantage that we know the Blaster and Witty worm anomaly used in our trace very well. We are therefore able to construct

---

[7] For reasons discussed in 2.2

an “ideal” baseline by removing the traffic that constitutes the anomaly.

Blaster baseline

To obtain the baseline for the Blaster worm, we removed only the initial connection attempt and left the remaining worm traffic in the trace. We did this for two reasons. Firstly, removing the initial connection attempt can be done with rather basic rules. Secondly, the share of the remaining worm traffic is less than 1‰of the total worm traffic. The rule used to remove the initial scan packet is simple: remove all TCP packets with destination port 135 and packet sizes of 40, 44, or 48. We are aware that this heuristic not only removes packets from connection attempts from the Blaster worm but also removes a share of packets of non-Blaster port scans to TCP port 135. However, before the Blaster outbreak, this number was very low[8]. Deciding not to make this distinction therefore seems reasonable. A detailed analysis of the Blaster worm with respect to the flow traces used in this study can be found in [172].

Witty baseline

To obtain the baseline for the Witty worm, we removed all UDP packets matching the following heuristic definition: a packet size between 796 and 1307 bytes, and source port of 4000. Since the Witty worm infection attempt consisted of a single packet only and the packet size is significantly different from the sizes of typical scan packets, this heuristic should be more accurate than those for the Blaster worm.

An alternative approach for determining the baselines would have been to use the average over a specific historic time period. This is similar to what anomaly detection methods with a baseline model based on past behavior do. However, our approach has two advantages: (1) it produces the most accurate “best-case” baseline model than any anomaly detection method could achieve, and (2) it is more general and is independent of the detection methods applied.

A brief assessment of how well these heuristics work is discussed at the end of the next subsection.

---

[8] Typically much less than 10,000 flows. During the outbreak, the number was in the range of 10^6 flows.

3.2.6 Anomaly Visibility

Having constructed the baselines and packet traces for different sampling rates and metrics, we can now measure how much an anomaly disturbs these metrics. We do so by calculating the absolute and relative distance or anomaly visibility between a sampled view x and $\hat{x}$, the corresponding sampled baseline:

- the absolute distance or anomaly visibility, defined as: $(x - \hat{x})$
- the relative distance or anomaly visibility, defined as: $(x - \hat{x})/\hat{x}$

By comparing the distances for different sampling rates, we can analyze if, and by how much, sampling boosts or attenuates this distance.

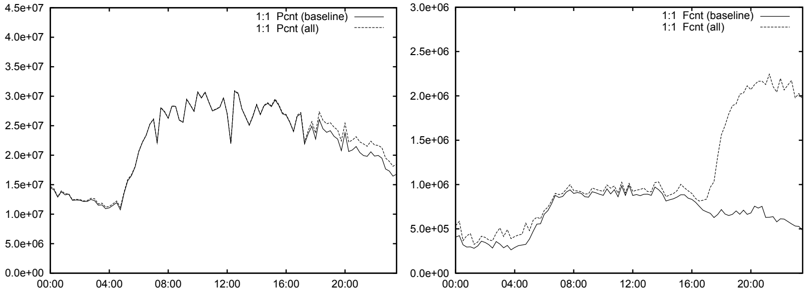
Note that while absolute distance $(x - \hat{x})$ is a result of adding the anomaly to the baseline, the distance is often not equal to the value of the metric obtained for the anomalous traffic only. Such a direct relationship requires the metric to be an additive metric. In our set of metrics, the flow, packet and byte count metrics are additive metrics. For entropy metrics, the sum of the two sample entropies from the baseline traffic and the anomalous traffic is not equal to the sample entropy from both the baseline and the anomalous traffic.

3.3 Impact of Sampling on Entropy and Volume Metrics

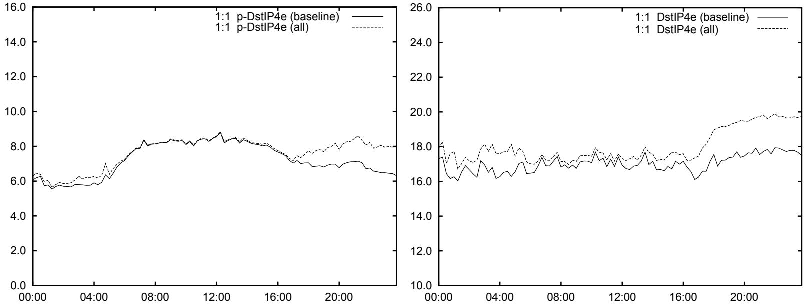
The first step in our analysis was to sample our one-week data set around the Blaster worm outbreak at four different sampling rates of 1 out of 10, 100, 250 and 1000. We then computed the time series of volume and entropy metrics as described previously.

To illustrate the following discussion on sampling effects, a selection of the most meaningful time series are depicted in Figure 3.2 and Figure 3.1: namely packet counts, flow counts, packet destination IP address entropy, and flow destination IP address entropy. Figure 3.1 shows the output for the baseline and the original traffic while Figure 3.2 displays the output for the original traffic at different sampling rates.

As expected, Figure 3.2(a) shows that packet counts are not disturbed by packet sampling. The unsampled values can simply be estimated by multiplying the sampled value by a factor of $1/p$. Likewise, byte counts (not shown)



(a) Packet counts vs. time (UTC) (b) Flow counts vs. time (UTC)

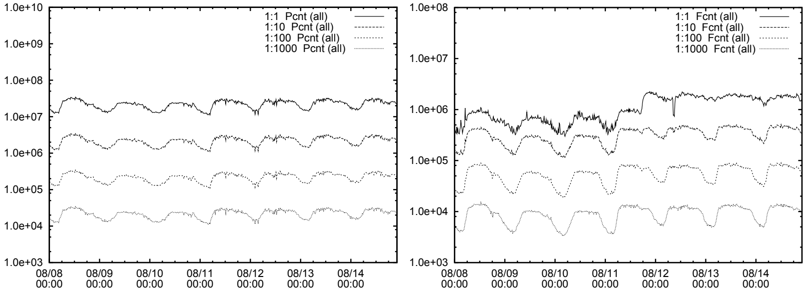


(c) Packet destination IP address entropy vs. time (UTC) (d) Flow destination IP address entropy vs. time (UTC)

Figure 3.1: *Blaster flow trace: Plots of selected metrics for flows (packets) with direction IN and protocol TCP. The plots show results for the traces both with- and without the Blaster anomaly.*

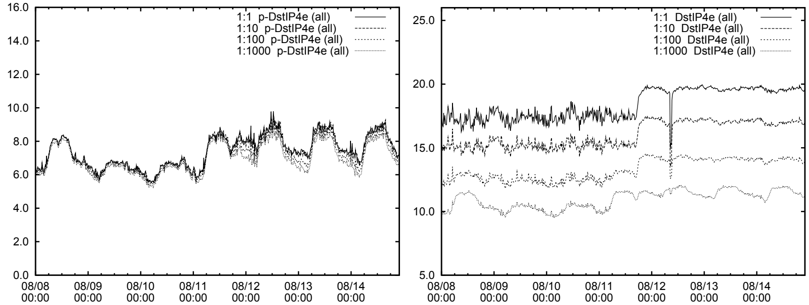
are not impacted by packet sampling. This is due to the fact that the variation of packet sizes by a factor of 100 (between 40 and 1500 Bytes) is very small compared to the overall number of Bytes ($\approx 10^{10}$) within one interval of 15 minutes. However, as depicted in Figure 3.1(a), packet counts only show a minor increase in distance before and after the Blaster outbreak. The other three metrics in Figure 3.1 indicate a more drastic and visible change. Packet (and Byte counts) might therefore not be a good choice for detecting anomalies other than volume anomalies.

On the contrary, flow counts are heavily disturbed by packet sampling, even at a sampling rate as low as 1 out of 10 (see Figure 3.2(b)). This is because flow counts are typically dominated by flows with few packets only



(a) Packet counts vs. date and time (UTC)

(b) Flow counts vs. date and time (UTC)



(c) Packet destination IP address entropy vs. date and time (UTC)

(d) Flow destination IP address entropy vs. date and time (UTC)

Figure 3.2: *Blaster flow trace: Impact of sampling on time series of selected metrics for flows (packets) with direction IN and protocol TCP. Note that in Figure 3.2(a) and 3.2(b) the y-axis is log scale.*

(e.g. scan-traffic or other background radiation). Few packets imply a smaller probability of being sampled in comparison to larger flows [159].

More interestingly, packet entropy metrics (Fig. 3.2(c)), as well as flow entropy metrics (Fig. 3.2(d)) are well preserved even at higher sampling rates. Though we see that packet sampling disturbs entropy metrics (the unsampled value cannot easily be computed from the sampled value as for byte and packet counts), the main traffic pattern is still visible in the sampled trace. This result was a crucial intellectual motivation for investigating the use of entropy metrics in more detail when the research for our study was initiated.

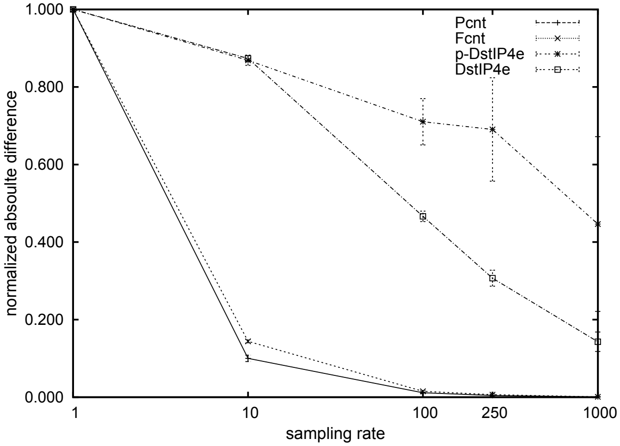
3.3.1 Anomaly Size

In Figure 3.3 we plot the sampling rate vs. the *absolute distance* (normalized to the respective value of each metric in the unsampled trace) as well as the sampling rate vs. the *relative distance* for packet counts, flow counts, flow destination IP entropy, and packet destination IP entropy. The Figure shows four curves, one for each metric under investigation, at each sampling rate for one interval. We selected the first interval after the Blaster outbreak around 17:00 UTC as a representative interval.

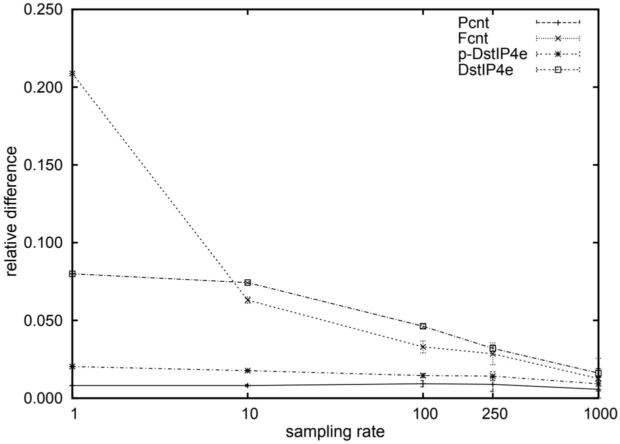
Let us consider the results for volume metrics first. For the flow count metrics the absolute as well as the relative distance decrease drastically when sampling is applied. Thus, we confirm the results of previous work, namely that flow counts, whilst effective in exposing Blaster in the unsampled data, are not a suitable metric for detecting flow-based anomalies when packet sampling is used. In contrast, packet counts are not impacted by packet sampling. Consequently, the relative difference for packet counts remains constant. However, as can be seen in Figure 3.1, the problem with packet counts is that Blaster-type anomalies, which usually represent only a very small fraction of all packets (less than 1% in our trace), are not very visible even in the unsampled data traces.

The flow and the packet entropy curves stand in sharp contrast to flow counts. The absolute as well as the relative distance decrease only very slightly, even for sampling rates as high as 1 out of 1000 for both the entropy metrics, implying that the size of the Blaster worm remains largely unaffected when viewed using entropy. For other intervals (not shown here) we find that entropy metrics can even emphasize Blaster-type anomalies in sampled views. Possible root causes as well as the root cause identified in our data are discussed in more detail in section 3.4.

To summarize, our results demonstrate that entropy-based metrics have two key benefits over volume-based metrics: (1) they are more effective at capturing the Blaster worm in unsampled traffic, even though the Blaster worm is not clearly visible in packet and byte counts and, more importantly: (2) they are impacted negligibly by sampling when compared to flow counts.



(a) Normalized absolute difference vs. sampling rate



(b) Relative difference vs. sampling rate

Figure 3.3: *Blaster flow trace: Anomaly visibility vs. sampling rates for four selected metrics for flows (packets) with direction IN and protocol TCP [packet counts (Pcnt), flow counts (Fcnt), flow destination IP entropy (DstIP4e), and packet destination IP entropy (p-DstIP4e)]. The plot shows the mean and 95% confidence interval over 12 sampling runs for the first interval after the Blaster outbreak around 17:00 UTC.*

3.3.2 Anomaly Intensity

Now that we have studied the effect of packet sampling on the Blaster anomaly within our data, we evaluate how effective entropy is at capturing Blaster-like anomalies of varying intensities. We utilize the given trace, and attenuate or amplify the strength of the Blaster anomaly accordingly. In order to amplify the Blaster anomaly, for each observed Blaster-packet we insert a second packet with the same source IP and a destination IP randomly selected from the SWITCH IP address range. To simulate an attenuated attack, we keep only 50%, 20%, and 10% of the attack packets in the packet trace.

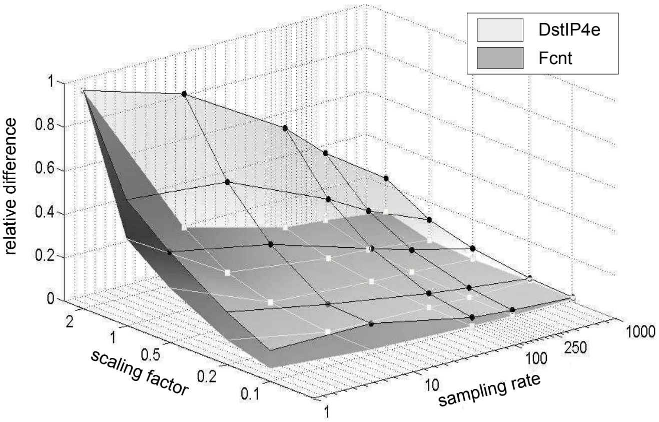


Figure 3.4: *Blaster flow trace: Relative distance from the baseline for flow counts and flow destination IP address entropy for flows with direction IN and protocol TCP across increasing sampling rates and different intensities.*

Figure 3.4 presents the relative difference for the flow counts (dark gray) and flow entropy (light gray) metrics, across increasing sampling rates and different intensities.[9] It provides considerable insight into the efficacy of flow counts and the flow destination IP address entropy in exposing the Blaster anomaly at various intensities and at various sampling rates.

As expected, the stronger the anomaly, the larger the relative difference

---

[9] For presentation purposes, we normalized each surface by the maximum size for that metric, so that the size of the anomaly for each metric falls between 0 and 1.

for both metrics. However, flow counts decrease sharply as the Blaster worm is attenuated, even with unsampled traffic. Moreover, this decrease in flow counts is even sharper as the sampling rate increases. In contrast, flow destination IP address entropy decreases remarkably slowly, both with increased sampling rate and for varying intensities of the Blaster attack.

From this figure we conclude that in the case of Blaster-like anomalies, entropy metrics are far more robust to packet sampling than simple flow count based summaries.

Furthermore, although our study focuses on the Blaster worm anomaly, we argue that our results are not specific to the Blaster worm anomaly, but are relevant for any anomaly with the following properties: (1) the anomaly causes a notable dispersion (or concentration) of one or more traffic feature distributions in the unsampled trace and (2) most distribution elements (such as an IP address in the source IP address distribution) added or modified by the anomaly are referenced by more than one flow each. The reason for property (1) is simple: if unmet, traffic feature distributions are not meaningful for this type of anomaly. The intuition behind property (2) is that entropy depends on both the number of distribution elements and their activity.[10] Thus if an anomalous flow referencing a specific distribution element is not sampled, there is a chance that other anomalous flows referencing the same element are sampled, keeping the element's influence active. In contrast, there is no such indirect impact in the case of the flow count metric. If a flow is not sampled, it no longer contributes to this metric.

Fortunately, if we look at the definition and the models of the anomalies discussed and referenced in Section 2.2, properties (1) and (2) are expected to be met by DDoS, reflector DDoS or various types of (large-scale) scanning. Furthermore, the successful application of entropy based anomaly detection in combination with sampled traces (e.g. in [20]) with many different types of anomalies support our claim.

Having said this, it is clear that how well an anomaly is visible with respect to our set of metrics also depends on the baseline traffic, as well as anomalous traffic. The next section discusses this in more detail.

---

[10] By how many flows an element is referenced, see Equation 3.1.

3.4 Impact of the Traffic Mix

In the previous section, we analyzed and compared the impact of packet sampling on various packet- and flow based feature entropies and volume metrics. Even though our results suggest that feature entropies are more resilient to sampling than volume metrics, the role of the traffic mix of the baseline traffic is still unclear. We therefore need to investigate this in more detail.

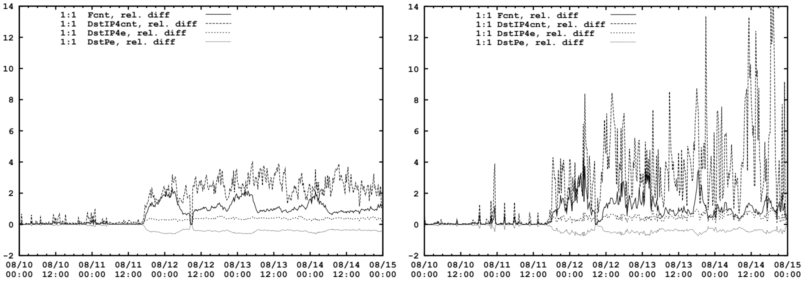
The traffic from the four routers from which we acquire our flow data appears to have quite differing characteristics, especially when comparing traffic from routers 2 and 3 to routers 1 and 4. We therefore examined the differences in exposure of the Witty and Blaster worm for each of these routers. Details on the characteristics of the traffic of the four routers can be found in our technical report [173].

Our results might appear surprising at first. Intuitively, packet sampling reduces the amount of information contained in our traces. One would expect that the visibility of an anomaly decreases with an increased sampling rate. Our evaluation tells a different story. We provide evidence that, for some of the border routers, the impact of packet sampling on anomaly visibility is contrary to the expected result. For two of the routers, the anomaly visibility is highest with unsampled data. For two other routers, it is highest with sampled data and outperforms the unsampled trace up to sampling rates of 1:10,000.

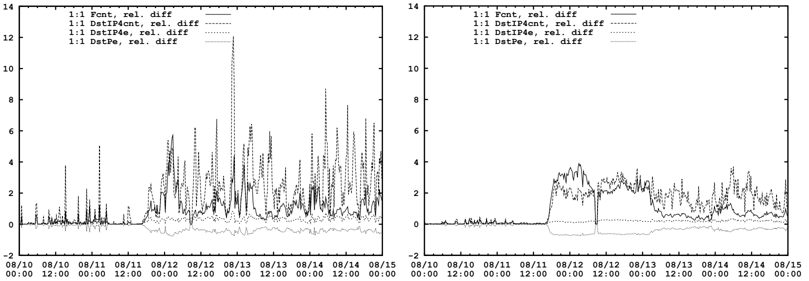
3.4.1 Results

Interestingly, each of these routers shows different visibility of the anomalies. This difference illustrates the impact of the traffic mix on the anomaly visibility. In order to uncover this, we have a closer look at the traces of router 2 and 4. Figure 3.6 shows the relative difference for flow count and flow destination IP address entropy for these selected routers.

A comparison of the flow count plots (on the left side) reveals that the anomaly visibility on router 4 is approximately twice as strong as on router 2. This indicates that router 4 has received much more Blaster flows than router 2. However, the impact of packet sampling is similar for both routers. The anomaly visibility is decreased significantly when going from unsampled traffic to traffic sampled at a rate of 1:10,000. Furthermore, the basic structure of the relative distance curves for the unsampled traffic is very well preserved by the curves for the sampled traffic. However, if we compare the



(a) Router 1 (b) Router2

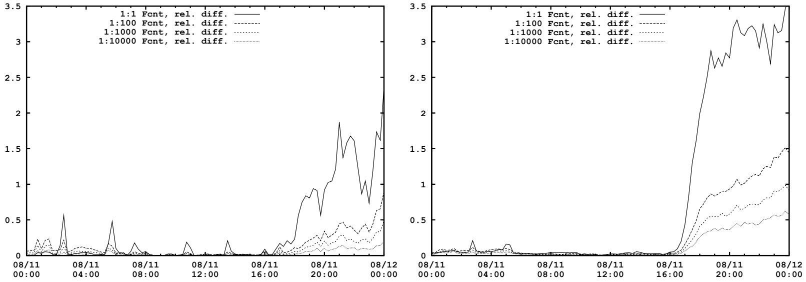


(c) Router 3 (d) Router 4

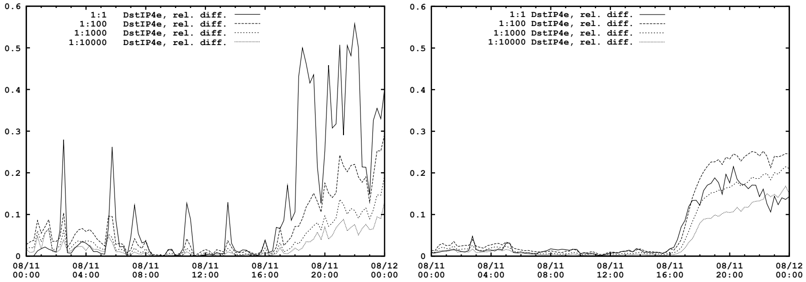
Figure 3.5: *Relative difference of flow count, unique destination IP address count, flow destination IP address entropy and flow destination port entropy metric on our four border routers vs. date and time (UTC). No sampling.*

flow destination IP address entropy metric (on the right side), we can see that the impact of sampling is totally different: Packet sampling *increases* Blaster visibility on router 4, while it *decreases* the visibility on router 2.

Another example is shown in Figure 3.7. This figure presents the anomaly visibility of the destination port count during the outbreak of the Witty worm for all four routers. A comparison of the four plots shows that the impact of sampling is similar for the traces of router 1 and 3 and for the traces of router 2 and 4. However, the impact on the traces of router 1 and 3 is completely different from the impact on the traces of router 2 and 4.



(a) Router 2 (b) Router 4



(c) Router 2 (d) Router 4

Figure 3.6: *Relative difference of flow count and flow destination IP address entropy metrics on routers 2 and 4 vs. date and time (UTC) during the outbreak of the Blaster worm.*

3.4.2 Discussion

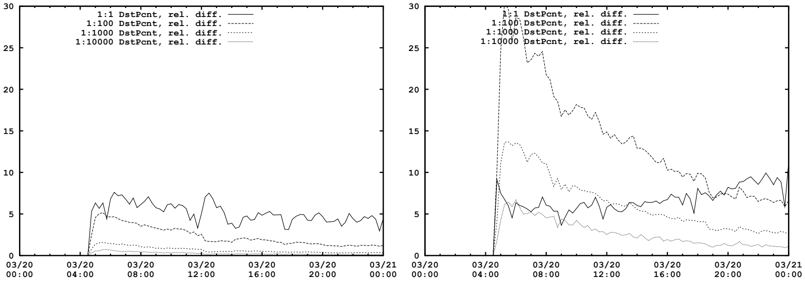
The increase in visibility for sampled traffic is significant, which may initially appear counter-intuitive. It is particularly surprising that the increase is big enough to preserve the visibility of the Blaster (Figure 3.6(d), Router 4) and Witty (Figure 3.7(b) and 3.7(d)) worm up to sampling rate of 1 out of 10,000.

What are the possible reasons that cause sampling to boost anomaly visibility? Or, in mathematical terms, what is the reason for:

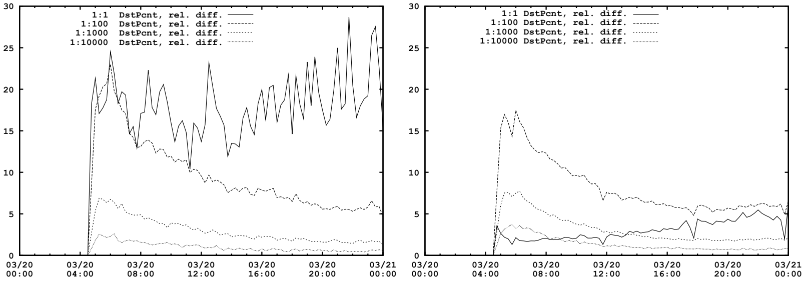
$$\frac{(x_p - \hat{x}_p)/\hat{x}_p}{(x - \hat{x})/\hat{x}} > 1 \tag{3.3}$$

for sample probability p?

To understand this effect, we first need to discuss the characteristics of



(a) Router 1 (b) Router 2



(c) Router 3 (d) Router 4

Figure 3.7: *Relative difference of destination port count on the four border routers vs. date and time (UTC) during the Witty worm outbreak.*

the baseline and anomalous traffic that determine the impact of sampling, and how they do this.

With regard to our set of metrics, the following two distributions capture the characteristics relevant to the impact of sampling; the number of packets per flow and the number of packets per distribution element.[11] Depending on the metric under scrutiny, only one of them or both matter:

Flow count: We want to decide whether the number of flows per time window of size T of the baseline or the anomalous traffic is more affected by sampling. To do so, it is sufficient to analyze the distribution of the number of packets per flow in the baseline traffic and in the anomalous traffic. For this analysis, we can plot the share of flows with less than *i* packets versus *i*. If the line of the plot for the baseline is always below the line for the anomalous

---

[11] For example, a specific source port number.

traffic, the baseline is expected to be less affected. In the opposite case, the same is true for the anomalous traffic.

If the lines cross each other at least once, it will depend on the sampling rate whether the flow count of the baseline or the anomalous traffic is more affected. For a sampling probability or p, the expected number of flows after sampling expressed as a share s of the original number of flows is

$$s = \frac{\sum_{i=1}^{N} f_i \times (1 - (1 - p))^i}{\sum_{i=1}^{N} n_i}$$

with n_i as the number of flows with i packets.

Since flow count is an additive metric, it can be shown that (3.3) is simply: $\frac{s_A}{s_B}$ with s_A and s_B as the expected share of anomalous and baseline-flows remaining after sampling.

It follows that there is no easy way to formulate a rule of thumb such as "if the average number of packets per anomalous flow is smaller than the average number of packets per baseline flow" the anomaly visibility decreases for all sampling probabilities. However, in practice we see many anomalies, such as different forms of scan-traffic and DDoS attacks, which involve flows with only one packet per flow. We also observe anomalies such as flash crowds or application-layer DDoS attacks where the number of packets per flow remains in a specific, narrow range. In the first case, we can expect the anomaly visibility to either decrease or to remain the same.[12] In the second case, we can expect to see an increase in anomaly visibility for selected sampling rates, considering that the distribution of the number of packets per flow of the baseline is typically heavy-tailed.

However, note that because of the variability introduced by sampling, if the increase or decrease of anomaly visibility is small, a single or just a few sampling runs are unlikely to provide the same trend or even exact result as the theoretical results derived from the true packet per flow distributions. This is clearly a general problem with sampling. It is theoretically possible that if we are really unlucky, we might not see an anomaly until it consists of *more* than the total number of packets minus the number of sampled packets. Hence, the exact same anomaly might be found in one case while it might be missed in another. This will be the case when using one of the popular sampling strategies such as *uniform random packet sampling* or *every n-th packet*. If possible, sampling should therefore not be used in security related applica-

---

[12] If the baseline also consists mainly of one packet flows.

tions. Unfortunately this is not always the case, as the broad application of sampling in high-speed networks suggests.

Unique count: To understand a boost in anomaly visibility for metrics capturing the number of unique elements (e.g. source port numbers) per time window of size T, it is sufficient to look at the number of packets per distribution element. However, unlike for the flow count metric, the distributions of the baseline and anomalous traffic are not independent, as the intersection of the distribution elements is typically not empty.

Let X be the set of unique elements x_i and $n_b(x_i)$ the number of baseline packets and $n_a(x_i)$ the number of anomalous packets with this element. Furthermore, let us consider the following (simplified) scenarios:

- The x_i in the anomalous traffic do not appear in the baseline traffic. Hence, if we look at the number of unique source IP addresses the anomalous traffic is originating from different source IP addresses than the traffic in the baseline. In this case, an anomaly would see an increase in its visibility if and only if the number of x_i in the baseline traffic decreases faster than the number of x_i in the anomalous traffic. If there are a lot of x_i with just one packet contributing to them in the baseline but not in the anomalous traffic, the visibility of the anomaly increases until the sampling rates reach a point where the number of x_i removed from the baseline again become smaller than the number of x_i removed from the anomalous traffic. Since today's "normal" traffic contains a significant amount of backscatter or scan traffic resulting in a large number of flows with just one or a few packets, any anomaly consisting of many flows with more than one packet per x_i should see an increase in its visibility, up to a certain sampling rate.
- The x_i in the anomalous traffic and the baseline traffic are identical. As a result, the anomaly is not visible in the unsampled traffic. However, if we apply sampling, the anomaly should become visible. This is because if we have the same set of x_i, the anomalous traffic contributes at least one packet to each x_i. If we now sample the baseline and the full traffic trace at the same sampling rate, the number of x_i in the baseline should decrease faster than the number of x_is in the full traffic traces. The x_i in the full traffic trace have more packets per x_i. Furthermore, if the baseline contains also some x_i with a considerably larger number of packets from the baseline traffic, the increase in visibility will at some point again turn into a decrease.
- A mix of disjoint and common x_i in the baseline and anomalous traffic.

> In practice, we expect most anomalies to match this kind of relationsship to the x_i in the baseline traffic. This is basically a combination of two portions of the traffic, one containing the flows related to x_i appearing in both baseline and anomalous traffic, and one consisting of the flows related to x_i appearing in either the anomalous or baseline traffic only. Whether or not the visibility of an anomaly is increased at a certain sampling rate now depends on the combined impact on the separate portions of the traffic. The visibility may be increased if we look at the portion related to the disjoint x_i only, but the decrease in visibility in the other portion more than compensates for it.

Entropy: For entropy metrics, it also matters how packets are distributed into flows. For example, if the baseline contributes a single flow with 100 packets to a specific item and the anomaly contributes the same number of packets but all from different single-packet flows, the contribution (number of flows) of the anomalous flows to this item decreases with increasing sampling rates. Accordingly, the opposite would be true, if the case were reversed.

Hence, we now have the means to explain the increases in anomaly visibilty for increasing sampling rates in Figures 3.6 and 3.7. Let us start by explaining the impact of sampling shown in Figure 3.7. First, let us consider the differences in the visibility of the anomaly in unsampled traffic. In order to get a relative difference of around 20, router 3 should see traffic of roughly 3000 ports per 15 minutes bin. This is based on the consideration that Witty used random destination ports for its attack. For router 1 it should be around 11,000 ports, for router 2 around 6000 ports and for router 4 around 16,000 ports. We can confirm these assumptions by extracting this information from our data. With this finding, and the fact that Witty used random destination ports for its attack, we can conclude that the set of ports contributed by the anomaly is largely disjoint from the set of ports contributed by the baseline. As long as the number of ports contributed by the baseline decreases faster than the ports contributed by the anomaly, we can expect an increase in anomaly visibility for the port count metric. By looking at the distribution of the number of packets per port with and without the anomaly, we found that while the baseline for routers 2 and 4 contain a significant number of ports with less than three packets per port, there were almost none of these in the Witty traffic. Most ports here were hit by around 5 packets. Consequently, until the decrease in ports is dominated by these ports, the baseline port count decreases much faster than the port count of the full packet trace. In contrast, on router 1 and 3 Witty traffic contributes a large number of ports that occur only once or twice compared to the number of ports with the same characteristics in the baseline traffic. Hence, the number of ports contributed by the Witty worm decreases faster than those of the baseline traffic.

The increases in anomaly visibility for sampling rates of up to one out of 100 packets, shown in Figure 3.6, remains harder to explain. This time, we consider not a unique count metric but an entropy metric. For entropy metrics, both the number of flows and the number of packets per distribution element are important. However, since the Blaster anomaly consists mainly of one packet flows, flows and packets can (almost) be considered the same. An inspection of the Blaster traffic seen by router 4 showed that, in contrast to other routers and the baseline traffic of router 4, it contains many IP addresses which are hit by more than one attack source. The increase can therefore be explained along the same lines as the Witty case.

3.5 Conclusion

In this section, we presented an empirical evaluation of the impact of packet sampling on anomaly detection metrics. Starting with a week-long dataset of unsampled NetFlow traces containing the Blaster worm, we asked how packet sampling impacts volume metrics such as the number of bytes, packets, and flows that have been commonly used in anomaly detection. To answer this question, we employed a unique and general methodology which treats anomalies as deviation from an idealized baseline. We used this to evaluate the fidelity of sampled traffic in exposing anomalies. Our first finding is expected. We found that packet sampling produces accurate estimates of byte and packet counts, when compared to the underlying trace. However, packet sampling produces grossly inaccurate estimates of flow counts. Indeed, the Blaster worm, which was prominent in the unsampled traffic view of flow counts, disappears entirely at higher sampling rates. This is because, as shown in previous work, small, single packet flows are missed entirely. Therefore, anomalies that impact only packet counts or byte counts are likely to be visible in sampled views, but anomalies that impact flow counts (such as the Blaster worm in our data) will not be visible.

We subsequently evaluated the effect of packet sampling on feature entropy. Surprisingly, we found that while the Blaster worm is hardly visible in flow counts of sampled traces, it remains visible in entropy metrics. While sampled traffic views are inherently incomplete and imperfect, they are not completely useless. In fact, we provided evidence that sampled traffic can be of use if analyzed using the appropriate metrics, such as entropy.

Finally, we extended our study to include the impact of the traffic mix. Starting with two week-long datasets of unsampled traffic records from four border routers, we ask how traffic mix affects anomaly metrics in combination with packet sampling. By comparing the 15 metrics for four border routers at different sampling rates, we found that the visibility of certain metrics, for example flow destination IP entropy, was even more pronounced by packet sampling up to sampling rates of one out of 10,000. We subsequently elaborated on possible root causes. In retrospect, our findings certainly were surprising even though potential explanations are manifold. However, in order to observe this effect both the anomaly and the baseline traffic must be shaped accordingly. As a result, we do not expect them to be relevant to a larger number of anomalies and real-world baseline traffic mixes. Nevertheless, our findings remind us that we cannot simplistically argue that sampling is an inherently lossy process.

Chapter 4

Analysis of Feature Correlation

Having discussed the impact of sampling and the traffic mix on various volume and entropy features in the last chapter, we continue our assessment of these features with regard to their use with anomaly detection in high-speed communication networks. We perform a detailed correlation analysis of a broad set of volume- and entropy features to analyze whether any of these features are redundant. In contrast to the analysis in [27], we did not find persistent and strong correlations in entropy features. On the contrary, we show that extending the classical feature set with features reflecting the geographical structure of the traffic adds another layer of potentially useful information. Finally, we discuss why we think that the different approaches used to build the traffic feature distributions are the real reason for the different findings. Whilst Nychis *et al.* measure the number of packets per feature instance, we measure the number of flows per feature instance.[1]

4.1 Introduction

One of the key challenges with anomaly detection systems is to minimize the size of the input vector while at the same time maximizing its informa-

---

[1] In practice, packet-based approaches are hard to find. Most works use a flow-based approach [19, 20, 31, 48, 91].

tion content. Any input that does not contribute toward a better detection performance should not be fed to the detector. Doing so helps to optimize resource usage and limits potential sources of errors and misinformation. This problem is typically addressed by a feature selection process in which either the performance of the system with different subsets of the input vector is analyzed or in which a correlation analysis on the full input vector reveals which components of the vector are likely to carry only redundant information. From these approaches, the first is likely to result in a biased selection of the input vector if the set of anomalies does not include (1) a similar number of anomalies of each type and (2) anomalies from all possible anomaly types and variations. This is particularly a problem if the detector should also detect rare or yet unknown anomalies. The second approach, the feature correlation analysis, does not suffer from this problem. It provides information about the "similarity" of each pair of components of the input vector given a series of observations of the input vector. Hence, if we feed a series of observations where most observations are not anomalous,[2] we can locate the components relevant to capturing and describing the current state of the communication network. The drawback of this approach is that not all components relevant to the current state must also be relevant to detecting an anomalous state. Some might never be affected by anomalies. However, since we cannot decide this without knowing all past, current and feature anomalies, this approach might be a better choice with regard to the problem of grey and black swans.[3]

A recent analysis of the pairwise correlation of different feature entropies by Nychis *et al.* [27] raised some concern regarding the usefulness of these features. Nychis *et al.* found that port entropy, address entropy and traffic volume (packets/s) are highly correlated. Therefore, a single feature, such as traffic volume, would already provide enough information for the reliable detection of DDoS-like events. Consequently, the use of multiple features would not provide additional information to improve the anomaly detection rate.

Motivated by our own experience in the field, which contradicts the results reported by Nychis *et al.*, we performed our own correlation analysis of traffic features. This analysis confirmed what we suspected. The analysis did not expose any persistent strong correlation between traffic features, except for one: a strong and persistent correlation of the flow size entropy and the bytes

---

[2] Which is relatively easy to do if the assumption that anomalies are typically rare holds for the data under scrutiny.

[3] The problem of detecting rare and yet unknown anomalies. See 2.1 for more details on this topic.

per packet entropy. To aid detection and especially classification of network anomalies, we therefore suggest the use of a wide range of features to capture different aspects of traffic dynamics.

4.2 Methodology

For our correlation analysis of feature entropies, we selected the following set of traffic features:

- Flow size in bytes (Fsize)
- Bytes per packet (BytesPP)
- Source and destination port (Dp,Sp)
- Source and destination IP address (Sip,Dip)
- Autonomous System (AS)
- Country code (Country)

From these features, the source and destination port and source and destination IP address features can be found in virtually all entropy based anomaly detection approaches. The flow size (in bytes or packets) and the bytes per packet are less frequently used in the entropy context but are quite popular features in anomaly detection in general [174–177].

Concerning the last two features, we do not know of any (entropy based) anomaly detection system making use of them. For now, we can merely state that these features may expose certain anomalies that might otherwise simply vanish into the background noise. More details can be found in chapters 5 and 6.

As in the previous chapter, we compute the Shannon entropy of these features as follows. Firstly, we count the number of occurrences a_i of all instances x_i of a specific traffic feature in a time window of length T. More specifically, if we take the feature *source port*, we count the number a_i of flows containing a specific source port x_i and do this for all x_i's found in the flows. Next, we calculate the Shannon entropy according to the following equation:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i), \tag{4.1}$$

$$p(x_i) = \frac{a_i}{\sum_{j=1}^{n} a_j} \tag{4.2}$$

We then repeat this procedure for all time windows and all traffic features and compare the resulting entropy time series. Note that by weighting the contribution of each instance x_i of a traffic feature with the number of flows containing it, we might introduce a correlation with the total number of flows (Fcnt) in the respective time window. To analyze this, we include this metric in our analysis.

4.2.1 Correlation Metrics

A possible correlation metric for two time series X and Y consisting of n data points is the Pearson product-moment correlation r, as used by [27]. The Pearson correlation coefficient r_{xy} is defined as

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{(n-1)\sigma_x\sigma_y}. \tag{4.3}$$

where $\overline{x}$ and $\overline{y}$ are the sample means of X and Y, and σ_x and σ_y are the sample standard deviations of X and Y. In particular, Nychis *et al.* measured Pearson correlation scores bigger than 95 for port and address distributions, where score 1 means maximum correlation. An alternative correlation metric is the Spearman's rank correlation:

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)} \tag{4.4}$$

$d_i = x_i - y_i$ is the difference between the ranks of corresponding values X_i and Y_i. Whereas Pearson only captures linear correlation, Spearman considers any correlation described by a monotone function, including linear correlation. A comparison of the two correlation metrics on our data set showed that Spearman correlation was consistently higher than Pearson correlation, hinting at considerable non-linear correlation. Therefore, we used Spearman's correlation for our analysis.

4.2.2 Data Set

To evaluate the feature correlation, we used 10 different traces summarized in Table 4.1 from the SWITCH backbone.

ID	Description	Start	Days	75p Fcnt TCP	75p Fcnt UDP
1	Blaster worm	08/01/03	22	567K	146K
2	DNS attack	02/04/04	6	919K	793K
3	Witty worm	03/16/04	6	1,095K	304K
4	Sasser worm	04/26/04	9	1,068K	276K
5	YouTube outage	08/07/06	13	544K	468K
6	Telia fiber cut	08/12/07	26	877K	921K
7	Géant anomaly	10/17/07	6	954K	1,456K
8	YouTube outage II	02/01/08	25	895K	1,404K
9	Reflector DDoS	03/31/08	14	954K	1,479K
10a	4 months (router 1)	02/29/08	120	930K	1,520K
10b	" (router 2)	"		442K	618K
10c	" (router 3)	"		206K	82K
10d	" (router 4)	"		547K	623K

Table 4.1: *Overview of traces used. To indicate the size of traces, we list the 75-percentile (75p) of flow counts computed in 5-minute windows.*

Traces 1-9 were captured on the largest exchange point (router 1) around major anomalies, such as global worm outbreaks, outages or a DDoS attack using internal hosts as reflectors. On average, roughly 50% of their duration is considered anomalous. Trace number 10 is a continuous trace over 4 months from all exchange points with no major anomaly. In total, the traces cover 247 days from 5 years.

4.3 Results

The absolute values of the Spearman coefficients as a percentage are presented in the tables 4.2, 4.3, and 4.4. A value of 100 denotes maximum correlation where on the other hand 0 means no correlation.

	Sp				Dp				AS				Sip				Dip				Country				BytesPP				Fsize			
	max	min	avg	std	max	min	avg	std	max	min	avg	std	max	min	avg	std	max	min	avg	std	max	min	avg	std	max	min	avg	std	max	min	avg	std
Fcnt	78	14	48	22	73	26	56	16	60	27	43	13	76	45	57	12	71	1	34	22	64	10	46	15	**90**	44	65	15	**90**	48	65	14
Sp	-	-	-	-	**89**	19	75	21	69	3	41	22	69	5	31	20	65	1	36	22	75	7	42	23	**80**	13	54	19	78	12	52	20
Dp	-	-	-	-	-	-	-	-	**90**	36	68	17	76	25	52	16	**81**	3	40	25	52	3	28	19	**85**	65	76	7	**86**	64	76	7
AS	-	-	-	-	-	-	-	-	-	-	-	-	**89**	35	65	17	73	6	38	22	41	1	25	11	**87**	38	75	14	**88**	33	76	16
Sip	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**90**	3	40	28	41	4	22	12	**93**	53	**81**	12	**94**	54	**83**	12
Dip	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	56	18	34	12	**90**	4	43	28	**93**	5	44	28
Country	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	51	2	25	18	54	1	24	18
BytesPP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**100**	**99**	**100**	0
Fsize	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 4.2: *Correlation of different feature entropies for traces 1-9 (see Table 4.1) as absolute values of the Spearman coefficients as a percentage. The table shows maximum, minimum, average, and standard deviation for correlation of $H(X)$ for TCP traffic.*

Table 4.2 shows correlation statistics for traces 1-9, comprising several *anomalous* intervals from a range of 5 years. Strong correlations (≥ 80) are highlighted. For each feature pair, we compute the correlation of the respective time series for each of the nine traces. Then the maximum, minimum, and average correlation is selected for each feature pair. Generally, correlation of the different features is low. For some feature pairs, correlation is high in certain traces, but low in general. This is, for instance, the case for (Sip, Dip). It has a maximum correlation of 90 but an average correlation of only 40. Only the pair (BytesPP, Fsize) has a very strong average correlation of almost 100. This is not unexpected, since the bytes per packet value contributed by a flow is calculated by dividing the flow size in bytes by the number of packets of this flow. Hence, if the number of packets of a flow correlate with its size in bytes,[4] we should also see a correlation of the Fsize and the BytesPP feature.

The next highly correlated feature pairs are (Sip, Fsize) with 83 and (Sip, BytePP) with an average correlation of 81. All other pairs have an average correlation of less than 80. Summing up, almost all of the feature pairs defy the allocation of a fixed correlation value. If we look at the minimum and maximum of the correlation values for each pair, we see that they span quite a large range of values.

Table 4.3 and 4.4 show correlation statistics for traces 10a-d. This charts the correlation between different routers during a 4-months period of relatively *normal* traffic containing no major anomaly. Table 4.3 shows the correlation for TCP traffic and Table 4.4 for UDP traffic respectively. For TCP correlation is again in general very low. The only exception is (Sp, Dp), with correlations between 96 and 98. Surprisingly, the three most correlated pairs from table 4.2 are not at all correlated in traces 10a-d, although both tables show statistics for TCP traffic. This suggests that correlation can vary significantly with time and between normal or anomalous traffic conditions. In other words, we cannot assume that a specific feature pair is correlated but neither can we assume that it is uncorrelated. For UDP, there are a number of pairs with high maximum correlations. However, this is usually not stable over all routers, as the minimum correlation is quite weak for most of them. The only pair with constant strong correlation is again (Sp, Dp). However, while (Sp, Dp) is strongly correlated in normal traffic (traces 10a-d), it is only moderately correlated in anomalous traffic (traces 1-9). The summary for the results for the 4-months trace is therefore similar to the summary for

---

[4] Intuitively, this correlation is expected. For longer flows, it is even (partially) enforced be the size limit of packets.

the 9 traces before. For almost all feature pairs, no fixed correlation value that can be attributed.

Altogether, our findings suggest that in some situations, the different feature entropies do contribute information not yet included in one or more of the other feature entropies. However, in others, they might not.

	Sp		Dp		AS		Sip		Dip		Country		BytesPP		Fsize	
	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min
Fcnt	**94**	51	**93**	38	70	46	61	26	45	7	61	19	67	36	43	5
Sp	-	-	**98**	**96**	63	28	65	38	57	36	76	43	26	4	35	7
Dp	-	-	-	-	68	19	66	32	58	32	73	34	22	3	37	7
AS	-	-	-	-	-	-	**85**	62	45	14	29	18	43	9	44	23
Sip	-	-	-	-	-	-	-	-	64	58	70	42	23	15	27	12
Dip	-	-	-	-	-	-	-	-	-	-	**93**	54	58	7	67	7
Country	-	-	-	-	-	-	-	-	-	-	-	-	54	14	56	22
BytesPP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	39	5
Fsize	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 4.3: *Correlation of different feature entropies for traces 10a-d (TCP) as absolute values of the Spearman coefficients as a percentage. The table shows the maximum and minimum of 4 different routers for $H(X)$.*

	Sp		Dp		AS		Sip		Dip		Country		BytesPP		Fsize	
	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min
Fcnt	**82**	73	**80**	64	**95**	63	**86**	7	**84**	13	**83**	14	78	13	**86**	12
Sp	-	-	**96**	**93**	79	65	79	29	70	27	78	42	64	1	76	6
Dp	-	-	-	-	78	49	79	46	72	18	64	39	63	2	74	2
AS	-	-	-	-	-	-	**89**	11	**94**	16	**84**	19	79	22	**96**	8
Sip	-	-	-	-	-	-	-	-	**89**	20	79	0	**87**	21	**91**	21
Dip	-	-	-	-	-	-	-	-	-	-	**92**	27	70	14	**94**	53
Country	-	-	-	-	-	-	-	-	-	-	-	-	47	1	**88**	8
BytesPP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	78	4
Fsize	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 4.4: *Correlation of different feature entropies for traces 10a-d (UDP) as absolute values of the Spearman coefficients as a percentage. The table shows the maximum and minimum of 4 different routers for $H(X)$.*

4.4 Discussion

Besides a strong correlation of (Sp, Dp) in normal traffic, our results do not confirm the very strong correlation between src/dst port and IP address entropies in normal and anomalous traffic found by Nychis *et al.* [27]. In our results, the pairwise correlations tend to be weaker but also quite variable. We think that these differences can largely be explained by the way the a_i (number of occurrences of item i) are calculated for equation (4.2). Nychis *et al.* compute a_i by counting the number of *packets* containing element i whereas we count the number of *flows*. Clearly, the number of packets is highly correlated with overall traffic volume, whereas a high volume file transfer is usually summarized in a single flow. Thus, by computing the a_i using packet counts, one introduces a potentially strong correlation with traffic volume. This consideration might also have been the reason why most approaches to entropy based anomaly detection [19,20,31,48,91] choose to use the number of flows to calculate the a_i instead of the number of packets.

Another interesting observation can be made when looking at the results which Nychis *et al.* found regarding their flow size distribution feature. This feature is based on a distribution constructed on a per flow and not a per packet basis. Its correlation with their packet-based features is very weak. While this might solely be due to the fact that these features are indeed more or less independent, it might also come from the fact that they are constructed differently: one on a per flow and the other on a per packet basis.

4.5 Conclusion

We revisited the results of Nychis *et al.* [27] regarding a persistent and strong correlation between traffic feature entropies. We did this by performing an extensive correlation analysis of traffic feature entropies on a large data set containing traffic from a diverse set of customers. In contrast to Nychis *et al.*, we did not find a strong, persistent correlation between traffic feature entropies. Our analysis did not expose *strong pairwise correlations* which are invariant over time, different routers, and normal/anomalous traffic conditions. We argued that the differences between our results and the findings of Nychis *et al.* can largely be explained by the way the distributions are constructed, by either flow or packet based approach. Our results suggest that if we use the flow based method to construct the distributions from which the entropy is calculated, the pairwise correlation of the selected traffic features tends to be weaker than when using the packet based method. However, this also varies quite significantly for traffic traces from different times, collected at different locations, or containing mainly normal traffic or anomalous traffic. Hence, if we drop one of these features, we might lose relevant information to detect and classify an anomaly in some but not all situations. To avoid this, we make use of *all* of these features in our entropy telescope.

Chapter 5

Traffic Entropy Spectrum

In the previous two chapters, we presented and discussed empirical evidence for the first two claims made by this thesis. Firstly, that entropy is robust to packet sampling techniques often used with measurement infrastructures in high speed networks. Secondly, that volume and entropy features provide largely independent information. In this chapter, we present and discuss the *Traffic Entropy Spectrum (TES)*, a method for a compact characterization and visualization of traffic feature distributions based on a parameterized form of entropy; the Tsallis entropy. After introducing the concept behind the TES, and its properties, we demonstrate its descriptive power using traffic data from different real world anomalies.

5.1 Introduction

Fast and accurate detection of network traffic anomalies is a key factor in providing a reliable and stable network infrastructure. In recent years, a wide variety of advanced methods and tools have been developed to improve existing alerting and visualization systems. Some of these methods and tools focus on analyzing anomalies based on volume metrics, such as traffic volume, connection count or packet count [82]. Others look at changes in traffic feature distributions [178] such as IP address or flow size distributions, or apply methods involving the analysis of content or the behavior of each host or group of hosts [50]. However, content inspection or storing state information on a per host basis is usually limited to small and medium-scale networks.

Most approaches designed for large-scale networks thererfore have two things in common. Firstly, they reduce the amount of input data by looking at flow-level information only (e.g. Cisco NetFlow [35] or IPFIX [179]). Secondly, they use on-the-fly methods that do not rely on a large amount of stored state information. As a consequence, using the history of traffic feature distributions to detect relevant changes over time is not feasible since they can consist of millions of data points. A related problem arises when one wants to visualize the evolution of these distributions; a compact form containing information about relevant changes is required.

A prominent way of capturing important characteristics of distributions in a compact form is the use of entropy analysis. Entropy analysis (1) reduces the amount of information needed to be kept for detecting distributional changes and (2) allows for a compact visualization of such changes. A popular form of entropy is Shannon entropy. Its success when first used with an anomaly detection system might be the main reason why most entropy based systems adopted this form of entropy [19–21].

However, the good detection and, to a lesser extent, classification performance of these systems is mainly reached with regard to massive anomalies only. For us, a massive anomaly is an anomalie which is well-visible in one or multiple volume time series[1]. There is some empirical evidence which suggests that parameterized forms of entropy such as the Tsallis entropy might be superior to Shannon entropy. In [25], Ziviani *et al.* [25] investigate at which value of the parameter q of the Tsallis entropy DDoS attacks are detected best. In [26], Shafiq *et al.* do the same for port scan anomalies from malware. They then use this optimal q value with their detection system. Unfortunately, the optimal value of q seems to depend somehow on the anomaly, or the baseline traffic, or both, since they did not report similar values for the optimal q. Therefore, a generalization of these preliminary results towards arbitrary types of anomalies as well as appropriate detection and classification systems remains unachieved.

To address these issues, we propose a method integrating generalized entropy metrics in a new and more general way. More precisely, we make the following contributions:

- We define the TES for capturing and visualizing relevant changes in traffic feature distributions requiring little or no tuning to specific attacks.
- We demonstrate that the TES can be used for both anomaly detection

---

[1] Flows, packets or bytes per time bin

and classification as well as for visualization of their characteristics.
- We confirm the findings of [25, 26] for a broader set of anomalies.

Furthermore, we propose to add Autonomous System (AS) entropy to the set of commonly used traffic features. We provide evidence that it is a valuable addition.

The remainder of this chapter is organized as follows. In Section 5.2, we start with a review of the Tsallis entropy, introduce important terms and definitions and discuss the advantage of the Tsallis entropy over the Shannon entropy. Next, we introduce the TES and explain how it is used to capture and visualize distributional changes. Section 5.4 proposes a refinement of the TES, addressing a problem that makes characterization and classification of anomalies difficult under certain conditions. In Section 5.5, we discuss the concept of *Spectrum Patterns* and outline how such patterns could be used to classify anomalies. We continue with Section 5.6, discussing important aspects of our evaluation before presenting our results in Section 5.7. Finally, we conclude this chapter with Section 5.8.

5.2 Shannon and Tsallis Entropy

The Shannon entropy [180]

$$S_s(X) = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i) \tag{5.1}$$

can be seen as a *logarithm moment* as it is just the expectation of the logarithm of the measure (with a minus sign to get a positive quantity). Given that different *moments* reveal different clues into distribution, it is clear that using other generalized entropies may reveal different aspects of the data. Two such generalized entropies relying on *moments* different from the *log-moment* are the Rényi and Tsallis entropies, the latter being an expansion of the former. A comprehensive introduction to entropy in general, and to the Tsallis entropy more specifically, can be found in Constantino Tsallis' book *Introduction to Nonextensive Statistical Mechanics: Approaching a Complex World* [106].

The Tsallis entropy is defined as follows [181]: Let X be a random variable over the range of values $x_1, \ldots, x_n$ and $p(x_i) = p(X = x_i)$. Then, the

Tsallis entropy $S_q(X)$ is equal to

$$S_q(X) = \frac{1}{q-1}\Big(1 - \sum_{i=1}^{n} p(x_i)^q\Big) \tag{5.2}$$

$$p(x_i) = \frac{a_i}{\sum_{j=1}^{n} a_j} \tag{5.3}$$

where q is a parameter specific to the Tsallis entropy and a_i is the number of occurrences or *activity* of x_i in a time window of length T. In our context, the x_i are the feature elements, e.g. specific IP addresses or port numbers. Note that only elements occurring at least once contribute to the entropy S_q of a specific time window.

For q equal to 0 and 1, the Tsallis entropy has a special meaning. For $q \longrightarrow 1$, S_q recovers the Shannon entropy (up to a multiplicative constant). And for $q = 0$, it corresponds to $n - 1$, the number of unique feature elements minus one.

5.2.1 Terms and Definitions

Before we take a closer look at the meaning of the parameter q, we summarize important terms and definitions used in the reminder of this thesis:

- *system*: A (set of) network(s) described by an ensemble of network flows
- *feature*: Any flow property that takes on different values and whose characterization using a distribution is potentially useful. Flow properties used in this thesis are: source and destination IP address, source and destination port number, origin and destination AS, origin and destination country code, flow size and average bytes per packet.
- *(feature) element i*: A specific instance of a feature (e.g., source IP address `10.0.0.1`)
- *activity a_i*: The number of occurrences of element i within a time window of size T.
- *feature distribution*: The sample probability distribution $P[I = i] = p(x_i)$ of e.g., the feature *source port* with $p(x_i)$ as in Equation (5.3). Note that $p(x_i)$ can also be interpreted as *relative activity* of element i. These feature distributions serve as input for the Tsallis entropy calculation.

5.2.2 The parameter q

In the literature, q is referred to as a measure for the non-extensitivity of a property of the system of interest. In physics, an extensive property is a property that is additive for independent, non-interacting subsystems. It is directly proportional to the “size” of the systems. Examples of such properties are the mass and volume of systems. In contrast, an intensive property does not depend on the “size” of the system; it is scale invariant. Density is a good example of such a property. With respect to entropy, if we measure the entropy of a system consisting of two subsystems described by the random variables X and Y, the entropy of an extensive system is expected to satisfy Equation (5.4), whereas a non-extensive system should satisfy Equation (5.5).

$$S(X,Y) = S(X) + S(Y) \tag{5.4}$$

$$S_q(X,Y) = S(X) + S(Y) + (1-q) \cdot S_q(X) \cdot S_q(Y) \tag{5.5}$$

However, *we do not use Tsallis entropy in an information-theoretic sense* but rather in an operational sense. We use it as a metric measuring whether a distribution is concentrated or dispersed. The main difference to approaches which use Shannon entropy in the same manner is that Tsallis entropy can concentrate on different regions of the distribution.

In use, the Tsallis entropy offers many possible choices for q. Each q reveals different aspects of distributions used to characterize the system under study. First, it is essential to stress that both $q = 0$ and $q = 1$ have a special meaning. For $q = 0$, we get $n-1$, the number of elements in the feature distribution minus one. For $q = 1$, the Tsallis entropy corresponds to the Shannon entropy. This correspondence can be derived by applying l’Hôpital’s rule to (5.2) for $q \longrightarrow 1$. For the interpretation of the other q values, let us consider the following example. In a time window of size T we observed that IP address A was the source of 1000 connections and IP address B was the source of 10 connections. In total, we observed 2000 connections. If we choose $q = 2$, the contribution of IP address A to the sum S_q is $p_A^2 = 0.25$ and that of IP address B $p_B^2 = 0.000025$. If, on the other hand, we choose $q = -2$, the contributions are $p_A^{-2} = 4$ and $p_B^{-2} = 40000$. Whereas the contribution of A was clearly dominant with $q = 2$, the contribution of B is dominant with $q = -2$.

Hence, for a q other than 0 or 1, we see that (5.2) puts more emphasis on those elements which show high (low) activity for $q > 1$ ($q < 1$). Hence, by adapting q, we are able to highlight anomalies that

1. increase or decrease the activity of elements with little or no activity for $q < 1$,
2. affect the activity of a large share of elements for q around 1,
3. increase or decrease the activity of a elements with high activity for $q > 1$.

In other words, it is possible to focus, for instance, on IP addresses that we see often, occasionally, or rarely in a specific time interval. The main advantages of this filter-like property are (1) that changes which only affect parts of the distribution are more pronounced and (2) that there is more detailed information for the classification of different anomalies.

5.3 The Traffic Entropy Spectrum

To leverage the full capabilities of Tsallis entropy, we introduce a new characterization and visualization method called the Traffic Entropy Spectrum (TES). The TES is a three axis plot that plots the entropy value over time (first axis) and for several values of q (second axis). For convenient 2D presentation, the third axis (showing the normalized entropy values) can be mapped on to a color range. Hence, the TES illustrates the temporal dynamics of feature distributions in various regions of activity, ranging from very low activity elements for negative q to high activity elements for $q > 1$. Figure 5.1 shows a sample of such a Traffic Entropy Spectrum.

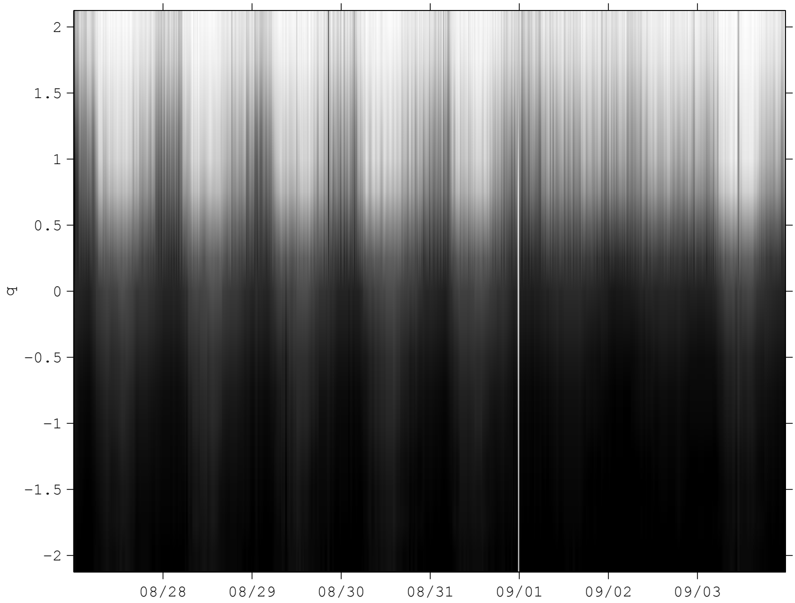


Figure 5.1: *Example of a Traffic Entropy Spectrum (TES) of the Autonomous System activity in the incoming traffic around a DDoS attack in 2007. The plot shows how the Tsallis entropy values for different values of the parameter q change over time. During the day, entropy is higher than during the night. On the weekend (01/09, 02/09), the difference between day and night is smaller. On the y-axis, we see the set of q-values for which the Tsallis entropy values are plotted as colored rectangles versus the time on the x-axis. Hence, a rectangle* (x, y, c) *color-encodes the (normalized) Tsallis entropy for* $q = y$ *and time* $T = x$*. The color scale used ranges from black for the minimum* S_q *to white for the maximum* S_q*.*

5.3.1 Selection of the q-Vector

However, what values should be used for the parameter q? And do they need to be tuned to the characteristics of the network traffic at a specific sensor? By experimenting with traces from different sensors and from different years (2003 to 2008) which showed largely differing traffic characteristics, we found that the selection $q = -2, -1.75, \dots, 1.75, 2$ gives sufficient information to detect network anomalies in all of these traces. Large values $q > 2$ or smaller values $q < -2$ did not provide notable gains. We consider this conclusion as strong empirical evidence towards the case that the parameters of the TES require little or no tuning to different traffic characteristics.

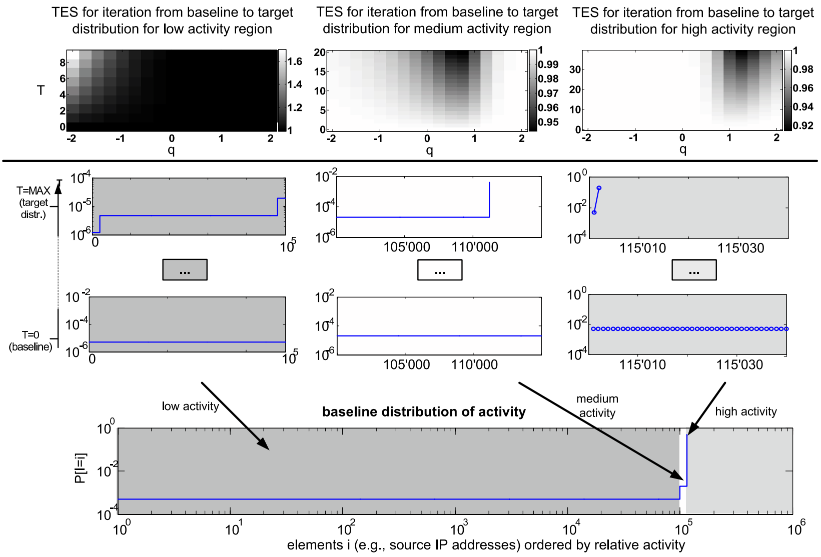


Figure 5.2: *Impact of changes to different regions of the distribution. Bottom: Baseline distribution. Center: Visualization of the baseline distribution (at $T = 0$) which is then iteratively transformed into the distribution shown at $T = MAX$ for low, medium and high activity regions. We call the distribution at $T = MAX$ the target distribution because it is the one we design the transformations to stop at after a certain number of iterations. Top: Resulting TES when altering the distribution in the respective region from the baseline to the target distribution in multiple, evenly sized, steps.*

To illustrate the impact of the parameter q in the TES, we make use of an artificial feature distribution $P[I = i]$ of elements i (see Figure 5.2) where we identify exactly three different regions. Each region contains elements that show *low*, *medium*, or *high activity*. Note that for simplicity, all elements in a region have the same absolute activity. We first look at the impact of modifications that are (1) limited to one of those regions and (2) that do not affect the total contribution of this region to $\sum p_i = 1$. To see how the TES reacts to such changes, we iteratively transform the distribution of each region, starting from the baseline distributions in time slot $T = 0$ until it looks like the distribution at $T = MAX$. We call the distribution at $T = MAX$ the target distribution since it is the one we design the transformations to obtain after a certain number of iterations. Figure 5.2 shows both the baseline (at $T = 0$) and the target distribution (at $T = MAX$) for each region.

For each of the iterations from the baseline to the target distribution, we calculate the entropy for the different values of q which we then divide by the corresponding entropy value of the baseline ($T = 0$). Hence, a value less than one denotes a decrease and a value greater than one an increase in entropy compared to the baseline. The topmost plots in Figure 5.2 display the relative increase or decrease on the way from the baseline to the target distribution for all of the three different regions. Inspecting the TES for the different modifications reveals that they behave as expected:

- high activity: reducing the number of elements decreases entropy for $q > 1$
- medium activity: reducing the number of elements decreases entropy for $-1 < q < 1$
- low activity: reducing the activity of some elements increases entropy for $q < -1$

5.3.2 Visualizing Anomalies by Normalization

To compensate for the large absolute difference of the entropies for different q's, we can apply different normalization methods:

- Global normalization using the maximum *max* and minimum *min* entropy value for a given q during a training period as follows:

$$S_{norm,q} = \frac{S_q - min}{max - min}$$

  This maps all entropy values to the range [0,1]. Figure 5.1 shows an example of a TES plotted using this method.

- Normalization using the maximum and minimum entropy for a given q during a training period, for instance from just before the anomaly under scrutiny. Here, we map entropy values between the minimum and maximum of the training day to [0,1]. Other values are either above 1 or below 0. Figure 5.6(a) shows an example of a TES plotted using this method.
- Normalization using the inter-quartile range: For this type of normalization, we calculate the first quartile Q_1 and the third quartile Q_3 of a given set of training data points. The first (third) quartile is defined as the value that cuts off the lowest 25% (75%) of these data points. The interquartile range or IQR is a measure of statistical dispersion. It is defined as $IQR = Q_3 - Q_1$. The IQR can be used to detect outliers by defining a normal range of values $[Q_1 - k \cdot IQR, Q_3 + k \cdot IQR]$ for some constant k. Everything above (below) this range is then colored in red (blue).

The TES based on global normalization is used to identify dominating changes. If such a dominating change is present, it stands out at the cost of a decreased visibility of non-dominating changes. The second or third normalization method is used to assess whether changes stay within the variations of the training day. Using these normalization methods, it is easy to develop a simple anomaly detector. Values going below the minimum or above the maximum of the training day expose only the anomalous parts of the TES. Even though this detection procedure is very straightforward, our evaluation shows that this simple method is already sufficient for detecting and classifying critical anomalies in network traces.

5.4 The Refined Traffic Entropy Spectrum: TES_p

To directly infer the state of a specific activity region, it would be most useful if the different S_q were largely independent from each other. Then, an increase or decrease of one or multiple S_q from two different time intervals would imply a change of activity pattern in the respective region. For instance, a significant change of the S_q for $q > 1$ would imply a change in the high activity region. Unfortunately, this is usually not true if one compares traffic feature distributions from real network traffic traces. To understand this, we must appreciate the problem of inter-region dependency and why this problem affects applications of the TES to traffic feature distributions from real network traffic traces.

5.4.1 Inter-Region Dependency

In Section 5.3.1, we looked at the impact of modifications that meet the following criteria:

- The modifications are limited to one of the three activity regions: low, medium or high.
- The modifications do not affect the overall contribution of a region

In other words, if, for example, the activity a_i of an element i in the high activity region is increased by Δa_i, the activity a_j of another element j[2] in the same region must be decreased by Δa_i such that the total activity $\sum_{j=1}^{n} a_j$ remains constant.

However, what now happens if a modification does affect the overall contribution of a region? Figure 5.3 and 5.5 illustrate this based on the following scenario. Let us assume that we want to detect changes in the activity of TCP port numbers in the interval T_n from those in the next interval T_{n+1}. Let us further assume that the activity of the top port increases by a factor of 1.5 from interval T_n to interval T_{n+1}. The activity of the other ports remains the same. The left hand plot in Figure 5.3 shows the activity plots for this scenario. The original distribution at interval T_n corresponds to the activity distribution of the destination ports of the TCP traffic flowing into the SWITCH network in the time from 09:45 to 10:00 on the 31st of July 2008. The distribution in interval T_{n+1} is a modified version of this distribution, as specified before. Note that the ports are sorted according to their activity. Port 80 is the port with the highest activity in both intervals.

If we now calculate the Tsallis entropy for the different q-values for both intervals and then compare them to those obtained for the second interval, we get the relative difference in Tsallis entropy shown on the right hand side in Figure 5.3. The relative difference of the two Tsallis entropies of the intervals T_n and T_{n+1} is defined as follows:

$$\frac{S_q(X_{\mathsf{T}_{\mathsf{n+1}}}) - S_q(X_{\mathsf{T}_{\mathsf{n}}})}{S_q(X_{\mathsf{T}_{\mathsf{n}}})} \tag{5.6}$$

Figure 5.3 illustrates the inter-region dependency quite well. It shows that this change has a strong impact on both, the entropy values for positive AND

---

[2] Or the sum of the activities of other elements in this region.

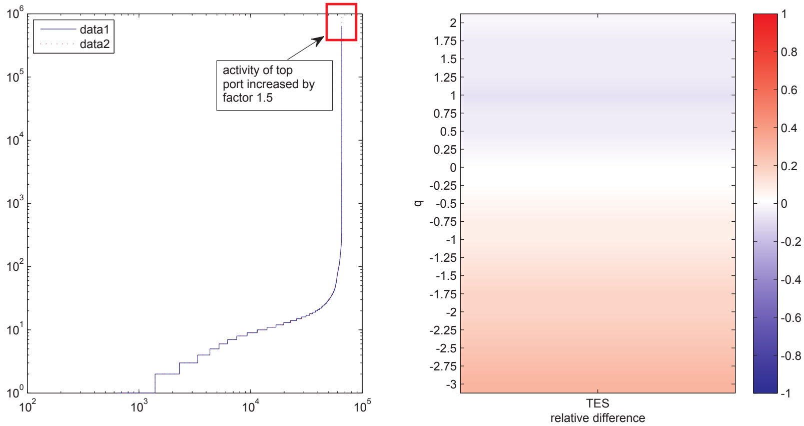


Figure 5.3: *A change in the high-activity region of a distribution should only affect the TES for positive q values (strongest for* $q > 1$*). Therefore, a plot of the relative difference between the TES before and after the change should show no difference for negative q. However, this is not true for the relative difference plot on the right even though the only difference in the underlying distributions is the activity of the most active element. As shown in the plot on the left, its activity is 1.5 times the activity of the original distribution. The original distribution corresponds to the activity distribution of the destination ports of the TCP traffic flowing into the SWITCH network between 09:45 and 10:00 on the 31st of July 2008.*

negative q-values. It does not just decrease the entropy for q-values stressing changes in high activity region. Why does this happen? Why do we see a result that reports a change in multiple regions even though we only modified the activity of a port in the high activity region?

The reason for this is that the sample probabilities $p(x_i)$ of the elements i contributing to S_q are computed by dividing their activity a_i by the total activity $\sum_{j=1}^{n} a_j$. In our example, the increased overall activity - caused by a host in the high activity region - led to a decrease in the sample probabilities which in turn led to a significant increase in entropy for $q < -0.5$.

A similar result is obtained when the overall activity is decreased. In this case the overall normalization factor $\sum_{j=1}^{n} a_j$ is smaller and the sample probabilities get bigger, even though the activity of the element might not have changed. With respect to entropy, this implies that the entropy for negative q-values decreases compared to the entropy in the first interval.

An important insight gained from this is that if our goal is to find abnormal changes in activity distributions based on the TES, the size of the *normal* fluctuations of the total activity dictates the change required to start considering a q-entropy to be abnormal. The problem with this is that, depending on the actual activity distributions, it might result in bounds dominated by the overall changes in one activity region only. For example, if most changes to the overall activity are due to changes in the high activity region then the minimum change required to start considering a Tsallis entropy value to be abnormal might reflect reality quite well if $q \geq 1$. However, it would be far too pessimistic to accurately describe other q-values.

To decide whether or not this problem is relevant when we apply the TES to traffic feature distributions from real network traffic traces, we briefly discuss two important characteristics of network traffic related to this problem.

5.4.2 Inter-Region Dependency: Relevance

Based on insights into characteristics of network traffic gained from both, our own NetFlow data and literature, we claim that the inter-region dependency is indeed a problem which should be addressed. Our claim is based on the following observations:

- **Compensation of activity changes is rare:** It is unlikely that the decrease in activity of some elements is compensated by an increase in activity of other elements in the same region. We can take an anomaly as an example. A typical impact of an anomaly is that the activity of a single element, and of multiple elements, is increased or decreased while the activity of other elements in the corresponding region does not change significantly.
- **Heavy-tailedness of traffic feature distributions:** Most distributions show some sort of heavy-tailedness. Heavy-tailed distributions amplify the problem related to the impact of change in the overall activity as it means that there are some elements which clearly “dominate” these changes. In our example in Section 5.4.1, the element with the highest activity (port 80) accounts for more than 15% of the total activity. As a consequence, if its activity doubles, the probability p_i of all other elements is decreased by more than 10%

5.4.3 The TES_p

The pruned Traffic Entropy Spectrum (TES_p) mitigates unwanted normalization effects by computing a *pruned* entropy in a two-step approach. For each time interval, we start by calculating the TES consisting of the entropy values S_q for a set of q-values. We then zoom in on the elements most responsible for p percentage of the value of S_q, for a given q. In the second step, we calculate the pruned entropy for the selected elements only, denoted by $S_{q,p}$. With this procedure, we make sure that the changes of elements i that contribute almost nothing to the sum $\sum_{i=1}^{n} p(x_i)^q$ have no impact on the final $S_{q,p}$, neither through direct contribution nor through normalization.

More formally, let the original distribution of activities be $A = \{a_i, \ldots, a_n\}$. Then we first compute $S_q(A)$ as defined by 5.2. Now let $C = c_i$ be the set of element contributions, that is $c_i = (a_i / \sum_{j=1}^{n} a_j)^q$. Then we let C' be the sorted version of C such that $c'_j \geq c'_{j+1}$ and store the mapping of indices between C and C' in a table ϕ. Thus, if c_k is mapped to element c'_l, we have $\phi(k) = l$. Let $\sigma(x)$ be the partial entropy computed by summing up all contributions of C' up to element x, that is $\sigma(x) := \sum_{j=1}^{x} c'_j$. Further, let $\hat{x}$ be the smallest index x for which $\sigma(x) \geq p/100 \cdot S_q(A)$ holds. From this we construct the set of selected activities $A' = \cup_{j=1}^{\hat{x}} a_{\phi^{-1}(j)}$. Finally, the pruned entropy is computed by $S_{q,p} := S_q(A')$.

The pruned TES_p, is now simply the values of $S_{q,p}$ for the given set of q-values. It can therefore be plotted in the same way as the original TES. Note that the original TES corresponds to TES_{100}.

Figure 5.4 illustrates the effect of TES_p. The top figure shows a destination port activity distribution with the ports on the x-axis ordered by ascending activity. That is, the leftmost port with index 1 is the rarest and the rightmost port is the top port (port 80 in this case). The activity of a port is plotted on the y-axis (i) during an anomaly and during normal activity.

For both distributions there is one plot below, which shows the selected elements for different values of q and p. At a specific coordinate (x,q) there is a grey dot where element x was selected for the pruned entropy $S_{q,p}$. For instance, looking at the regions for the anomalous port distribution, we see that for $q = -3$ and $p = 80$, only about 10,000 ports on the left (i.e., the low activity ports) are selected. Looking at the regions for the normal port distribution, we see that $q = -3$ kept the low activity region in focus even though there are now around 28,000 low activity ports. Similar observations can be made for other q and p-values, with smaller p values tending to capture

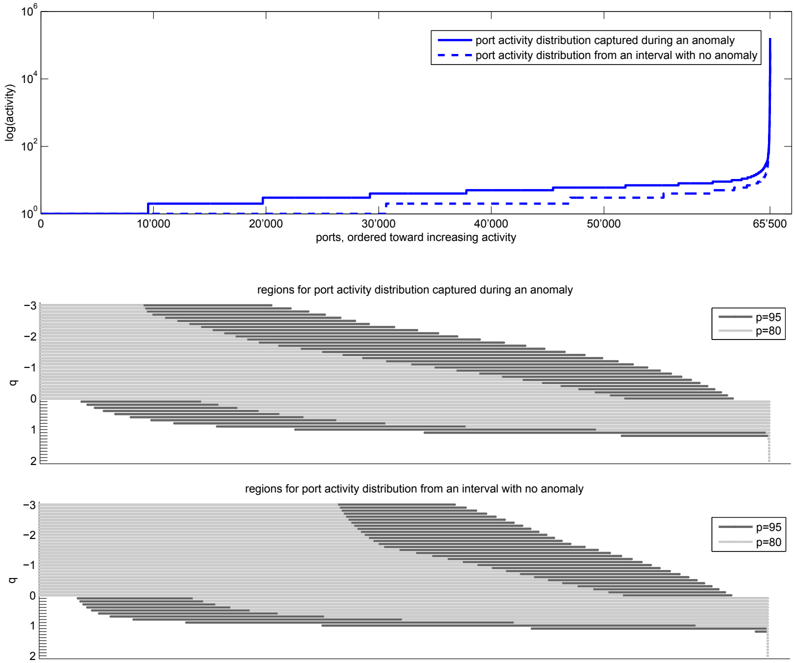


Figure 5.4: *Destination port activity distributions (top) and selected regions for TES_p (bottom). On the x-axis all ports are ordered by rank, i.e. with increasing activity to the right.*

the different activity levels more tightly at the cost of probably being too tight: $q = -3, p = 80$ fails to select the full range of low activity ports in the normal port activity distribution.

To check whether this selection process is indeed able to remove the inter-region dependency, let us turn back to the example used when we introduced the inter-region dependency problem. However, this time we compute the relative difference plot for the TES_p instead of the TES. The result is shown in Figure 5.5: the inter-region dependency which led to significant changes in the entropies capturing changes in the low activity region (see Figure 5.5) is no longer visible. The only change affects entropies capturing changes in the high activity region. Additional examples of comparisons of the TES and TES_p can be found in Section 5.6.

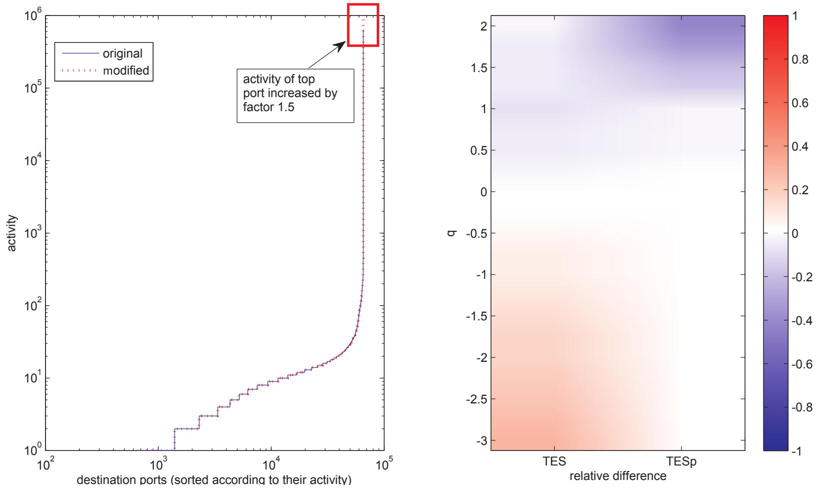


Figure 5.5: *Left: Two activity distributions differing only in the activity of the most active element: activity in the modified distribution is 1.5 times that of the original distribution. The original distribution corresponds to the activity distribution of the destination ports of the TCP traffic flowing into the SWITCH network between 09:45 and 10:00 on the 31st of July 2008. Right: Relative difference of both, the TES and the* TES_p *of the original distribution and the modified distribution.*

5.5 Spectrum Patterns

Malicious attacks often exhibit very specific traffic characteristics that induce changes in feature distributions known to be heavy-tailed. In particular, the set of involved values per feature (IP addresses or ports) is often found to be either very small or very large. In a DDoS attack, for instance, the victim is usually a single entity, such as a host or a router. The attacking hosts, on the other hand, are large in numbers, especially if source addresses are spoofed. Similarly, if a specific service is targeted by an attack, a single destination port is used, whereas source ports are usually selected randomly. In general, the specific selection of victims or services leads to *concentration* on a feature and, in turn, to a change in the high activity domain. In contrast, random feature selection results in *dispersion* and impacts the low activity domain (e.g. spoofed IP addresses only occur once in the trace). Knowing this, it is possible to profile an attack based on the affected activity regions for each feature.

When describing these patterns we use a shorthand notation representing the state of S_q with respect to some upper- and lower threshold $Th_{q,upper}$ and $Th_{q,lower}$. Note that the thresholds do not have to be constant but might depend on time and other factors in a more general case:

$$c_q = \begin{cases} \text{‘1’} & \text{if } S_q \geq Th_{q,upper} \\ \text{‘-1’} & \text{if } S_q \leq Th_{q,lower} \\ \text{‘0’} & \text{else (normal conditions)} \end{cases}$$

By a *Spectrum Pattern* we denote the consecutive c_qs for a representative set of values of q. This set might include all of the q values used in the Traffic Entropy Spectrum but might also be sampled to simplify the patterns at the cost of coarse graining the result. A modified version of the concept of the Spectrum Pattern is used later in this thesis when we integrate the TES into a fully automated anomaly detection and classification system. There, we do not sample the set of values of q to get a more compact form of the Spectrum Pattern but rather aggregate the S_q[3] of multiple q values.

5.6 Evaluation

To get an idea whether or not the TES is a suitable tool to capture the characteristics of anomalies, we check its descriptive power with respect to a set of anomalies whose characteristics and time of occurrence in our traces is known. Since we started out with the original TES and refined it only later, the results published in [31] do still contain significant inter-region dependencies. To show the impact of the TES_p - especially how it simplifies the interpretation of the TES -, we put the original results side-by-side with the results obtained with the TES_p.

5.6.1 Feature Set

For our evaluation, we analyzed the TES for the following set of traffic feature distributions:

- source IP address (SrcIP) and destination IP address (DstIP)

---

[3] Or more precisely, the anomaly scores related to the S_q values.

- source port number (SrcPort) and destination port number (DstPort)
- AS number

Note that since we observe the traffic to and from the SWITCH AS, the AS traffic feature is the origin AS for incoming traffic and the destination AS for outgoing traffic.

Hence, for each of these features the corresponding TES had to be built. We did this on a per protocol basis for the TCP, UDP, ICMP and OTHERS[4] protocols.

5.6.2 Set of Anomalies

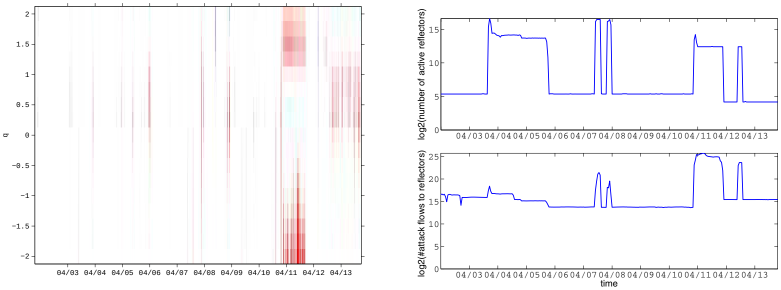
Our evaluation focuses on the following set of well-known anomalies:[5]

- **Refl. DDoS:** A reflector DDoS attack involving 30,000 reflectors within the SWITCH network, used to attack a web server. Two weeks of traffic were analyzed including some preliminary scanning activity (April 2008). Figure 5.6(a) shows the TES for incoming DstIPs. The attack is clearly visible around 04/11 and lasts for almost one day. Figure 5.6(b) shows the effective activity of the reflectors during a two-week period. The sustained activity on 04/04 and 04/05 without attack flows suggests that attackers are scanning the network for potential reflectors.
- **DDoS 1:** A short 10 minute DDoS attack on a router and a host with 8 million spoofed source addresses (Sept. 2007). DstPort is TCP 80. Figure 5.8(a) plots the TES for incoming AS numbers. The attack is clearly visible for $q < 0$ on the 09/01. Although the covered period is 8 days, the attack is visible with an excellent signal to noise ratio and *no false alarms*. Note that for Shannon entropy ($q = 1$) the peak is insignificant.
- **DDoS 2:** A long 13 hour DDoS attack on a host with 5 million spoofed source addresses (Dec. 2007/Jan. 2008). DstPort is TCP 80.
- **Blaster Worm:** Massive global worm outbreak caused by random selection/infection of new hosts, exploiting a RPC vulnerability on TCP DstPort 135 (Aug. 2003).
- **Witty Worm:** Fast spreading worm exploiting a vulnerability in ISS network security products. Uses UDP SrcPort 4000 and random DstPort (March 2004).

---

[4] Includes traffic for all protocols except TCP, UDP and ICMP.

[5] They are well-known either because we or other researchers have studied them in detail.



(a) TES of DstIP addresses for flows into our network during the reflector attack. Alerts are shown in red (resp. blue) above (below) threshold of a normal training day.

(b) The effective number of active reflectors (top) and the effective number of attack flows toward (candidate) reflectors in our network (bottom)

Figure 5.6: *Reflector DDoS attack.*

5.6.3 Compiling the TES

Since we know the characteristics and time of occurrence of the aforementioned anomalies, compiling the TES for the corresponding portions of our network trace archive[6] is all we need to do.

For this purpose, we extended our NetFlow processing framework[7] with a module that compiles the TES (and also the TES_p) in a two stage approach. Firstly, the module reads incoming flows, determining the interval to which they belong in order to update the traffic feature distributions of this interval. Next, if no more flows are expected to arrive for an interval, it calculates the Tsallis entropy for the different q-values according to the procedure described in 5.3 and 5.4.3. Note that with our selection of qs (see 5.3.1), we need to do this for a set of 17 values per interval and traffic feature distribution. The results are then written to a Comma Separated Values (CSV) file in form legible to a human observer. More precisely, the results are written to either one or multiple files, depending on whether or not the tool is configured to compile the TES for different lengths of the aggregation interval simultaneously. Simultaneous computation is possible, if the lengths of the aggregation intervals are a multiple of the smallest interval. For this evaluation, we configured the tool to output results for interval lengths of 5, 10 and 15 minutes.

---

[6] A general description of the network traces and the network in which they are captured can be found in Section 1.4.

[7] A brief description of this framework can be found in Section A.2.2.

Another feature that we make use of in our evaluation is that it can also output the actual activity distributions. This can be done either in binary, which saves space, or in conventional written form.

5.6.4 Data Analysis

To now gain an idea whether or not the TES is a suitable tool to capture the characteristics of anomalies, we analyzed the TES information produced in the previous step as follows. Firstly, we browsed through the more than 600 TES[8] to check whether we can spot anomalies simply by looking at them. We then derived the spectrum patterns from what we found and checked (1) whether they are different for different anomalies and (2) whether they capture the known key characteristics of our anomalies.

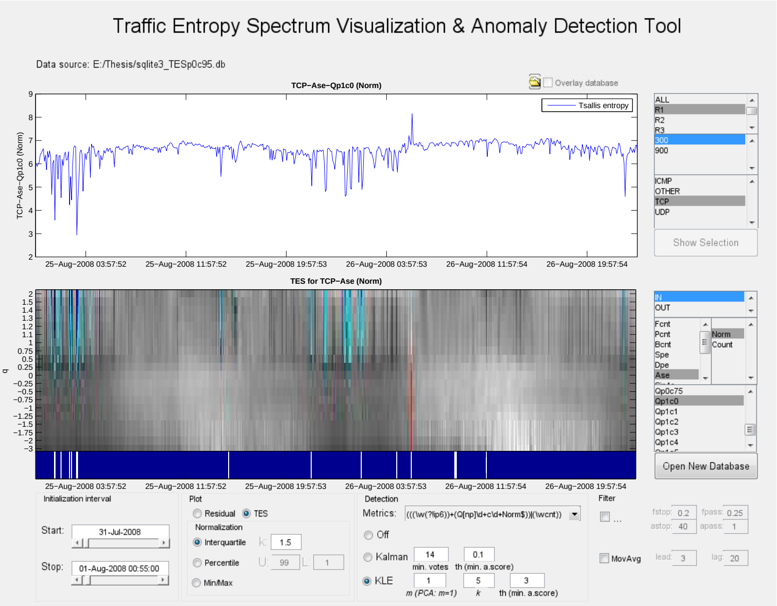
To speed up the process of browsing through the output of our NetFlow processing framework module, we implemented a tool called the *Traffic Entropy Spectrum Visualization & Anomaly Detection Tool*. Two of the most important features for our analysis were the GUI-based hierarchical selection of the TES[9] and controls for the selection and configuration of different normalization methods (see 5.3.2).

Figure 5.7 shows a screenshot of this tool. The plot at the top displays time series data of the selected time series: the Tsallis entropy for $q = 1.0$ for the AS traffic feature. The plot at the bottom displays the corresponding TES. More details on this tool can be found in Section A.2.3.

---

[8] One TES for each of the 5 traffic features for all of the 4 routers for both, incoming and outgoing traffic for all of the five anomalies and three time interval sizes.

[9] Flow exporter {ALL, router 1 to 4} → protocol {TCP, UDP, ICMP, OTHER} → direction of the traffic {IN, OUT} → traffic feature.



Traffic Entropy Spectrum Visualization & Anomaly Detection Tool

Data source: E:/Thesis/sqlite3_TESp0c95.db

Initialization interval
Start: 31-Jul-2008
Stop: 01-Aug-2008 00:55:00

Plot
○ Residual ● TES
Normalization
● Interquartile K: 1.5
○ Percentile U: 99 L: 1
○ Min/Max

Detection
Metrics: (((\w(?!ip6))+(Q[np]\d+c\d+Norm$))|(\wcnt))
○ Off
○ Kalman 14 min. votes 0.1 th (min. a.score)
● KLE 1 m (PCA: m=1) 5 k 3 th (min. a.score)

Filter
☐ … fstop: 0.2 fpass: 0.25 astop: 40 apass: 1
☐ MovAvg lead: 3 lag: 20

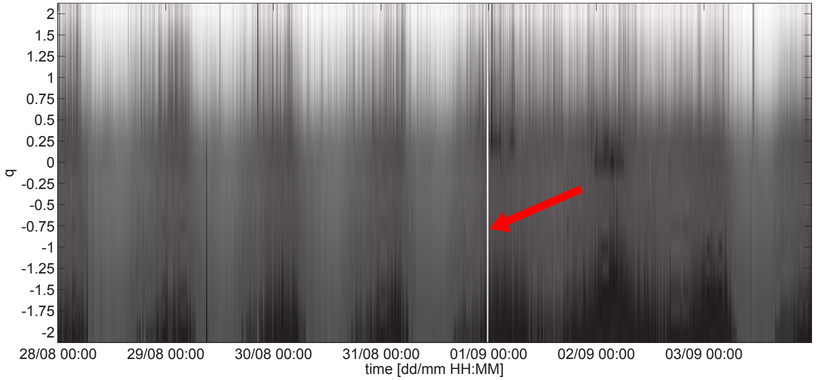
Figure 5.7: *Screenshot of the Traffic Entropy Spectrum Visualization & Anomaly Detection Tool. The plot at the top displays time series data of the selected time series: the Tsallis entropy for* $q = 1.0$ *for the Autonomous System traffic feature. The plot at the bottom displays the corresponding TES.*

5.7 Results

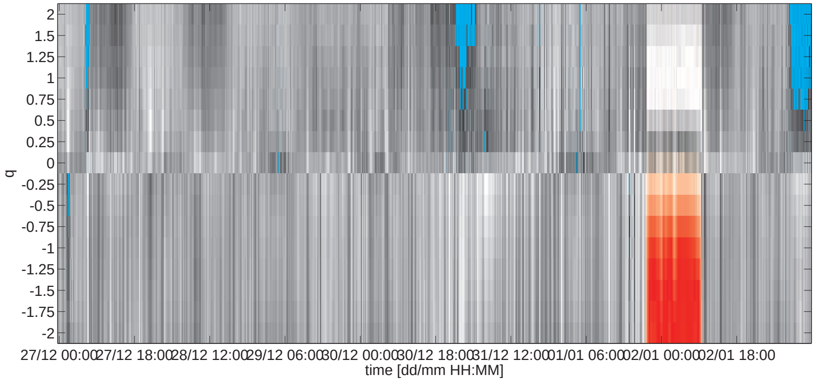
5.7.1 Spotting the anomalies

Our analysis of the 600 TES confirmed what we expected. We could easily spot all of the anomalies. Figures 5.8(b), 5.8(a),5.9(a),5.9(b) and 5.6(a) show a sample TES for each of the five anomalies. In all of these figures, the respective anomaly is clearly visible.

Note that we also checked how the different lengths of the aggregation intervals (5, 10 and 15 minutes) impact what we see in the TES. Here, our observations can be summarized as follows. While the results using the 15 minutes interval are much smoother, shorter intervals are better suited to pointing out anomalies that last only tens of seconds or a few minutes.

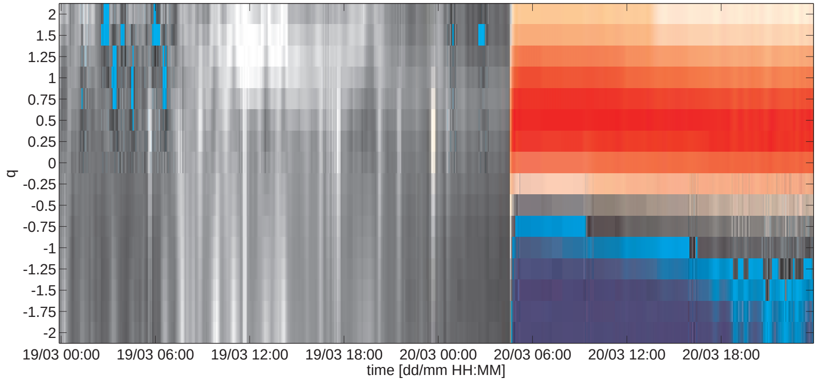


(a) TES of origin ASs in the incoming traffic during the DDoS 1 attack represented with global normalization.

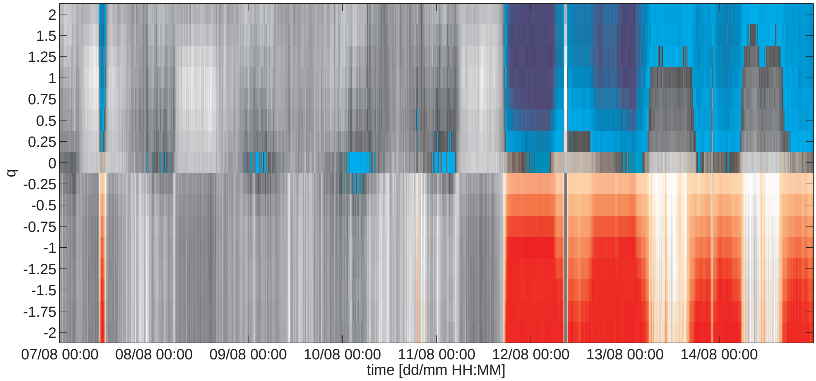


(b) TES of SrcPort numbers for flows into our network during the DDoS 2 attack. The TES is normalized using the inter-quartile range approach. Areas in red (resp. blue) represent locations where the TES is above (resp. below) the threshold.

Figure 5.8: *TES snapshots from the DDoS 1 and DDoS2*



(a) TES of DstPort numbers for flows into our network during the Witty worm outbreak. The TES is normalized using the inter-quartile range approach. Areas in red (resp. blue) represent locations where the TES is above (resp. below) the threshold.



(b) TES of DstPort numbers for flows into our network during the Blaster worm outbreak. The TES is normalized using the inter-quartile range approach. Areas in red (resp. blue) represent locations where the TES is above (resp. below) the threshold.

Figure 5.9: *TES snapshots from the Witty and Blaster anomaly*

5.7.2 Spectrum Patterns

In this section we analyze the spectrum patterns exhibited by the attacks described above. The following table shows the spectrum patterns for the five anomalies, the five traffic features[10] and for both the incoming and outgoing traffic. Furthermore, we also added the spectrum patterns produced by the modified version of the TES, the TES_p. These will be used to show that interpreting the spectrum patterns produced by the TES_p is easier than those produced by the TES.

			SrcIP					DstIP					SrcPort					DstPort				
			-2	½	0	½	+2	-2	½	0	½	+2	-2	½	0	½	+2	-2	½	0	½	+2
Refl. DDoS	IN	TES	+	+	0	-	-	+	0	0	0	+	-	-	0	+	0	+	+	0	-	-
		TES$_p$	0	0	0	-	-	0	0	0	+	+	+	-	0	+	0	0	0	0	-	-
	OUT	TES	+	0	0	+	0	+	+	0	-	-	+	+	0	-	-	-	-	0	+	0
		TES$_p$	0	0	0	+	+	0	0	0	-	-	0	0	0	-	-	+	-	0	+	
DDoS 1	IN	TES	+	+	+	+	0	+	+	0	-	-	+	+	0	-	-	0	0	0	-	-
		TES$_p$	+	+	+	+	0	0	0	0	-	-	0	0	0	-	+	0	0	0	-	-
	OUT	TES	+	+	0	-	-	+	+	+	+		0	0	0	-	-	+	+	0	-	0
		TES$_p$	0	0	0	-	-	+	+	+	+		0	0	0	0	-	0	0	0	-	+
DDoS 2	IN	TES	+	+	+	+	0	+	+	0	-	-	+	+	0	+	+	+	+	0	-	-
		TES$_p$	+	+	+	+	0	0	0	0	-	-	-	-	0	+	+	0	0	0	-	-
	OUT	TES	0	0	0	+	0	0	0	0	0	0	0	+	0	0	0	0	0	0	-	-
		TES$_p$	0	0	0	+	0	0	0	0	0	0	0	+	0	0	0	0	0	0	-	-
Blaster W.	IN	TES	+	+	+	-	0	+	+	+	+	0	+	+	0	-	0	+	+	-	-	-
		TES$_p$	+	+	+	-	0	+	+	+	+	0	+	+	0	0	0	0	0	0	-	-
	OUT	TES	+	+	0	0	0	+	+	+	+	0	+	+	0	-	0	+	+	0	-	-
		TES$_p$	+	+	0	0	0	+	+	+	+	0	0	0	0	-	+	0	0	0	-	-
Witty W.	IN	TES	0	0	0	-	-	+	+	+	+	0	+	+	0	-	-	+	+	+	+	0
		TES$_p$	0	0	0	-	-	+	+	+	+	0	0	0	0	-	-	+	+	+	+	0
	OUT	TES	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0
		TES$_p$	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0

Note that the spectrum patterns are expressed in both color and written form. The `-` sign and the color *blue* denote a significant decrease, no color and the `0` sign denote no change and red and the `+` sign stand for a significant increase of the entropy of the corresponding traffic feature and activity region.

Let us now have a closer look at the spectrum patterns of the five anomalies. Note that we do not discuss all of the spectrum patterns in detail. Instead, we select a number of illustrative examples to show how the TES (and the TES_p) capture the characteristics of our five anomalies.

Refl DDoS: The servers used as reflectors in the Refl. DDoS attack appear in the incoming destination IP addresses as requests from the real attackers.

---

[10] Note that our traffic is recorded at a single stub AS. Consequently, source AS are shown for incoming and destination AS for outgoing traffic, respectively.

The targets of this attack were mainly existing servers which would respond to incoming requests. Many of these machines could therefore already be found in the medium to high activity region prior to the attack. The attack led to more machines being part of the high activity region, but also to less diversity in the activity of these hosts. Both of these effects led to a significant increase in the high activity region ($+2$). Furthermore, we can see that the TES also seems to report an increase in the destination IP address entropy for negative qs. However, the Refl. DDoS attack hardly contributed to the activity of the IP addresses attributed to this region. The reason for this observation is that the attack led to a considerable increase in total activity. Hence, the inter-region dependency[11] led to an increase in the entropy of the low activity region. If we compare the spectrum pattern of the TES and the TES_p, we can see that the TES_p does not suffer from this effect. It reports the change in the medium to high activity region only.

The victim, being a single high activity host, had a contrary influence on the outgoing DstIPs and AS. Because the attackers spoofed the source IP address of the requests sent to the reflectors to match the address of the victim, the reflectors sent their replies to the victim instead of back to the attackers. This turned the IP address of the victim into one of the most active IP addresses[12] seen in the outgoing traffic. Consequently, the high-activity region becomes more concentrated. This is reflected by the decrease in the entropy of this region. As in the case of the destination IP address spectrum pattern for the incoming traffic, the change in the total activity is again large enough for the inter-region dependency effect to trigger a significant increase of the entropy in the low activity region. Note that the TES_p once more does not suffer from this problem.

A similar effect can be observed in the spectrum pattern for the DstPorts for incoming traffic. Here, the concentration is related to the attack traffic being sent to destination port 80.

In contrast, the spectrum patterns for the SrcPorts look quite different. The reason for this is that the attackers used more or less randomly distributed source ports in their requests to the reflectors. As a consequence, the more or less uniform distribution of the source ports in the anomalous traffic now dominates the distribution of the rare ports. This leads to an increase in the

---

[11] Remember that the activities of the elements are normalized by the total activity to get the sample probabilities for entropy calculation. A significant increase in the total activity decreases the sample probabilities of the elements in the low activity region, even though their activity did not change.

[12] Most of the time, it was the top IP address.

entropy for the low activity region, at least in the case of the TES_p. In the TES, we can only see an increase in the entropy of the upper medium activity region. Here, a few source ports which were seen significantly more often in the attack traffic caused the upper medium activity region to become more uniform too.

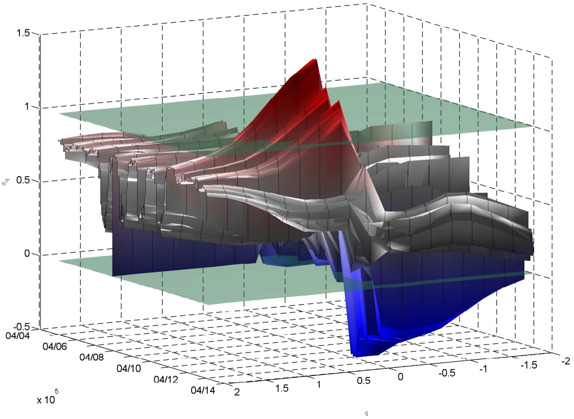


Figure 5.10: *3D TES for incoming SrcPorts before and during refl. DDoS attack for $q = -2 \ldots 2$. Diagonal axis: date (10 days), vertical axis: normalized entropy. Transparent layers: Minimum and maximum of normalized entropy at normal week days.*

In case of the TES, the modification to the low activity region is again hidden by the inter-region dependency. The increase in total activity, mainly attributed to the medium activity region, more than compensates for the change inflicted by the actual changes in activity in this region. Figure 5.7.2 nicely illustrates the observed pattern (`-0+0`) for the TES. Note that the patterns are symmetric with respect to the diagonal. That is, changes in incoming SrcIP/SrcPort columns are reflected in outgoing DstIP/DstPort columns and vice versa. This indicates that the reflectors actually managed to reply to most requests (no egress filter was in place).

DDoS 1: The main difference between the Refl. DDoS and the ordinary DDoS attacks is that the former uses real hosts (the reflectors), whereas the latter uses massively spoofed source IP addresses. For both attacks, the incoming SrcIP TES was affected over a wide range (`++++0`), including the SrcIP count ($q = 0$). It is important to note that in this case, there is no difference between the TES and the TES_p. The reason for this is that, in this case, the increase in the total activity is not mainly attributed to the medium or high activity region, but also to the low activity region. A lot of IP addresses with just a few occurrences are responsible for a large share of the change in the total activity.

DDoS 2: With respect to the spectrum patterns for the incoming traffic, the DDoS 2 anomaly is quite similar to the DDoS 1 anomaly. The main differences lie in the spectrum pattern for the SrcPort TES and those for the outgoing traffic. To understand what happens in the case of the SrcPort TES, we have to look at the actual activity distribution of the source ports before and during the attack. Figure 5.7.2 shows the two activity distributions in a log-log plot. According to the TES, we see an increase in the entropy in both the high and low activity regions. According to the TES_p, this is not true. If we now check the actual activity distributions, we see that the TES_p is right. The medium to high activity region becomes more “uniform”, meaning that it contains more source ports with a similar activity level. Meanwhile, the low activity region becomes more concentrated, since it contains fewer ports with a similar activity level; the activity of the ports in the activity distribution during the attack increases much faster for ports with low activity. Again, the TES captures the change in the medium to high activity region as we would expect it to. Unfortunately, since the major part of the change in the total activity is again attributed to the medium to high activity region, the inter-region dependency prevents us from seeing the expected reaction for the low activity region.

The patterns for the outgoing traffic are different because the victim did not send a noticeable number of responses. However, there is a quite notable change in the TES for the AS traffic feature. A closer look at the actual distribution revealed that between the 29th of December 2008 and the 2nd of January 2009, the outgoing traffic concentrated on fewer AS than before and after. One possible explanation for this is that during this time, most people spent their time on something other than browsing the web.

Blaster and Witty: In both the Blaster and the Witty worm destination addresses for the spreading of attack traffic were generated randomly, in much the same way as sources were spoofed during the DDoS attacks. In fact, the pattern exhibited by incoming worm DstIPs is exactly the same as the pattern for incoming DDoS SrcIPs. The pattern produced by random feature selection (`++++0`) is also visible in incoming DstPort for the Witty worm. On the other hand, the pattern specific to feature concentration (`++0-`) is for

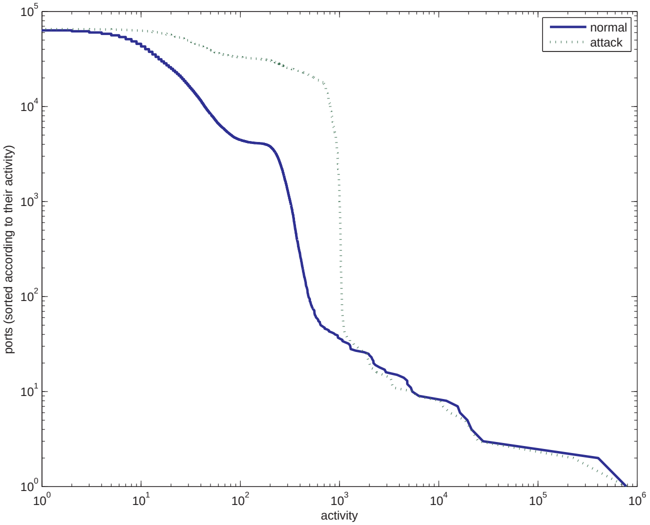


Figure 5.11: *Log-log plot of the source port activity distribution before and during the DDoS 2 attack. The source ports, sorted according to their activity, are on the y-axis.*

instance visible in incoming Witty SrcPort (fixed to UDP 4000), incoming refl. DDoS DstPort (fixed to TCP 80) or incoming DstIPs for DDoS 1 and 2. Moreover, if we compare the TES with the TES_p, we can again see that the TES_p removes inter-region dependencies if the change of the total activity is mainly attributed to one of the activity regions only (e.g. Blaster, IN, SrcPort or Witty, IN, SrcPort).

Random feature selection can have a different impact on ports than on IP addresses. Whereas incoming DstPort for Witty demonstrates the typical pattern, the one for incoming SrcPorts of the refl. DDoS looks quite different (`-0+0`). Random selection of IP addresses leads to many addresses with very low activity because the range of potential addresses is big. For ports, the range is limited to 65,535 values. Thus, if intensive random port scanning is performed, all ports are repeatedly revisited and become frequent, basically replacing the low activity area. This is what happened in the refl. DDoS case. We conclude that for ports, the strength (volume) of the attack plays a crucial role. For low volume attacks, the random port pattern looks like the random

IP pattern. However, increasing attack volume shifts the pattern toward `-0+0`.

Summing up, we see that fundamental distribution changes such as concentration or dispersion of features are well reflected by different TES patterns and can therefore be used to infer underlying traffic structure. In future work, we will consider the effect of attack volume as well as additional patterns, such as the distribution of flow sizes and durations. The ultimate goal is to develop a comprehensive and diverse set of TES patterns, suitable to accurately detect and classify network anomalies. For this, we need to do a more in-depth evaluation to prove that the improved detection sensitivity does not operate with a high ratio of false positives. Because our preliminary results suggest that TES is very robust (e.g. 8 days without a false alarm in 5.8(a)) even when using our trivial detection approach, we are positive that this will not be the case.

5.7.3 The TES in action: Anomaly drill-down

Until now, we used the TES only with anomalies for which we knew the characteristics and location in the trace quite well. To check whether the TES, or more precisely, the TES_p with $p = 95$, can pinpoint and characterize other anomalies in our trace, we selected four arbitrary intervals by looking at just one TES_p per anomaly to be selected. From these TES_p we then selected the time bins that looked suspicious. Note that we used the TES_p related to TCP traffic only. Using the following drill-down procedure, we then either confirm or reject our hypothesis that the interval contains an anomaly matching the characteristics hinted at by the TES_p. We apply the same procedure for all of the drill-down work done in the context of our thesis.

- Inspect the TES_p to identify those TES_p showing abnormal activity.
- Determine which regions of the TES_p (which q-values) are affected.
- Check the affected regions and determine whether the change in this region hints at a concentration (increase in entropy) or dispersion (decrease in entropy).
- Use the information from the previous step to guide our analysis of the actual traffic feature distributions. The result of this analysis should be a series of specific values of traffic features which flows involved in the attack should match: e.g. one could find that incoming flows should match destination port 80 and target a single IP address which does not seem to respond to most flows.
- If the type of the anomaly cannot yet be identified, filter flows that do

not match the identified values or patterns to get the candidate flows and analyze the candidate flows using frequent itemset mining.

To illustrate this process, we take one of the four anomalies and discuss how we performed the drill down. For the remaining anomalies, we provide the results only.

26.08.2012 at 06:00: The anomaly used for our discussion of the drill-down process is one occurring on the 26th of August 2008 at around 06:00. Using our *Traffic Entropy Spectrum Visualization & Anomaly Detection Tool,* we browsed through the various TES_p to determine those showing abnormal activity for this time bin. First, we inspected the TES_p for the incoming traffic.

- **Source ports:** The TES_p shows a decrease in the entropies of the medium activity region. This is quite unusual since it hints at anomalous traffic from just a small set of source ports and a moderate number of attack flows.
- **Destination ports:** From our first observation, we would have expected the TES_p to show an increase in the entropies for the low activity activity region. After all, a TCP connection typically has the source port at the initiator side of the connection selected by the operating system. Seeing concentration for both the source and the destination port is therefore rather unlikely. Nonetheless, the TES_p exposed a decrease in the entropies of the high-activity region.
- **Autonomous Systems:** Our subsequent inspection of the TES_p for the AS traffic feature shed some light on whether or not the anomaly involves traffic from many or just a few AS. A very pronounced increase of the entropies in the low and medium-activity region hinted at an attack involving quite a lot of AS from which we do not see much traffic under normal circumstances. Hence, the attack traffic might either come from hosts all around the world or from someone spoofing the source IP addresses used.
- **Source IP address:** Here, we see an impact on the TES_p similar to that with the AS traffic feature. We see an increase in the entropies of the low-activity region. However, this time the increase also extends to q values up to $q = 1.25$.
- **Destination IP address:** In this TES_p, we found a sharp decrease in the entropies of the high-activity region hinting at an attack on a single IP address only. If it were not for the strange behavior of the TES_p with respect to the source ports, we now would have said that the anomaly

might be because of a DDoS attack on a single port and host located inside the SWITCH network.
- **Country code:** The TES_p for the country code traffic feature was disturbed only slightly for the medium activity region. Here, we saw an increase in the entropies of the medium-activity region. This caused us to believe that the attack probably did not involve random source IP addresses. As we will see later on, we were wrong.
- **Flow size:** In this TES_p, we observed a slight decrease in the entropies in the upper medium activity region. However, this decrease was not as pronounced as, for example, the changes in the TES_p for the AS or the destination ports. Nevertheless, the change led us to believe that the flows causing this anomaly are probably all of the same size or of just a few different sizes.
- **Bytes per packet:** The TES_p for the average bytes per packet supported our hypothesis on the size of the flows. It showed a decrease in the entropies of the high-activity region.

Next, we inspected the TES_p for the outgoing traffic. In these TES_p, we found the same as for the incoming traffic but with source and destination switched. Hence, the location of the source of the anomaly remains unclear. To shed some light on this question, we looked at the flow count metric and saw that the number of incoming flows jumped from 1.3 in the preceding interval to 1.8 million flows in the anomalous interval. In contrast, the same metric for outgoing flows showed an increase from 1 million flows to only 1.3 million flows. Based on this observation, our guess was that the difference in the increase of those metrics is attributed to the victim(s) being unable to answer all of the incoming attack traffic. Note that fluctuations in the flow count metric for incoming flows of about 300,000 flows happen quite often. This still makes it difficult to spot an increase of around 500,000 flows simply by looking at a plot of the flow count metric.

Hence, what we have now is a guess about the type of the anomaly we see in this interval. If it were not for the strange behavior of the TES_p for the source port traffic feature, our first guess would be a DDoS anomaly with a single victim located inside the SWITCH network. Furthermore, since the victim appears to send replies back to the attackers, the victim is likely to be a server meant to answer such requests. We therefore assume that the destination port used by the attacker is either the ports used by a web server (80 or 443) or one of the other popular service ports such as 25, used for mail servers or 22, often used for remote access to compute infrastructures.

To confirm or reject our guesses, we consulted the full traffic feature distributions for those metrics. With the help of our *Traffic Feature Distribution Analysis and Visualization Tool* (see A.2.3 for details), and our guess from the information provided by the TES_p, we simply had to check whether our guess was reflected in the full traffic feature distributions. For this analysis, we looked at the distributions for the incoming traffic only.

- **Source ports:** Comparing the distribution of subsequent intervals, we found a significant absolute increase in the number of flows with source ports 1,024 and 3,072. If the anomaly involved other ports, the distortions to the distribution were not big enough. The number of flows to the other top 100 ports showed no abnormal behavior. This confirmed our guess, that the anomaly involved just a few source ports.
- **Destination ports:** In this distribution, we found a clear increase in the number of flows with port 22 as their target. While the share of flows with this port as their destination around 6 o'clock in the morning is around 1%, it suddenly reached a share of around 14%. With this share, it reached the second rank (after port 80) in the ranking of the most active ports. Again, if the anomaly involved other ports, the distortion of the distribution is not big enough to stand out.
- **Autonomous Systems:** Here, we could verify that our guess based on the TES_p was right: we could observe that the number of distinct AS responsible for less than 10 flows increased significantly, from roughly 4000 to 8000. Furthermore, the number of flows from the top AS was comparable to those from previous intervals.
- **Source IP address:** Comparing the distribution of subsequent intervals, we found a significant absolute increase in the number of IP addresses occurring in less than 10 flows. From a total of 100,000 IP addresses matching these criteria in intervals prior to the anomaly, this number increased to around 400,000 IP addresses in the anomalous interval.
- **Destination IP address:** In this distribution, we observed a significant increase in the number of flows directed to the top IP address.[13] Furthermore, the increase in the number of flows to the top IP address matched the increase expected from the increase in the flow count metric.
- **Country code:** A closer look at the distribution of the flows with re-

---

[13] From the distribution we do not know whether this is the same top IP address as in the previous interval since we do not store or show this information.

spect to their country of origin did not reveal anything particularly noteworthy. The only thing which is slightly different is the total number of countries represented in this distribution: 218. This number is typically a bit lower (around 210). However, not finding anything that stands out is also itself a finding. It tells us that the flows contributing to the anomaly might still come from spoofed IP addresses. If they were, say, originating from a botnet, the change in the country distribution would reflect the distribution of the bots with respect to countries. This distribution is likely to be quite focused on those countries with a lot of vulnerable hosts.

- **Flow size:** This distribution showed a clear absolute increase in the number of flows with a size of 48 bytes. The increase matched the increase expected from the increase in the flow count metric.
- **Bytes per packet:** This distribution confirms what we see in the distribution of the flow sizes.

Having analyzed the distributions, we already have quite a lot of evidence that our guesses based on the TES_p are correct. Moreover, we now have some numbers which are likely to characterize the anomaly under scrutiny:

- Flow size: 48 bytes
- Source ports: 1,024 and 3,072
- Destination ports: 22
- Victim: 1 (inside the SWITCH network)
- Attackers: Unknown (IP spoofing)
- Countries: Unknown (IP spoofing)
- Autonomous Systems involved: Unknown (IP spoofing)

We could now go and look at the traffic directly. To do this, we would extract and inspect flows with different combinations of the above characteristics. However, in this case we first browsed the Internet for similar observations, since the source port behavior is really quite strange. After some searching, we found a thread on the WebHosting TALK®portal from 2001 [182] describing this pattern and relating it to the DDoS attack tool *juno* whose code can be found here [183]. The reason for the strange port-related behavior is that the code selecting the source port is broken. At least, this is what we assume, as the code makes use of the *random()* function but comes to reside in only port 1024 or 3072. Since this tool is only generating flows with the characteristics identified and no others, we can be almost sure[14] that

---

[14] There is always the possibility that someone wanted the attack to look like it is generated by the *juno* tool. However, it is impossible to tell this based on flow data alone.

we have found the true root cause for this anomaly.

The results of the remaining three anomalies are as follows:

- **03.08.2012 at 02:45**: The SrcIP and SrcPort TES_p for incoming TCP traffic expose abnormal activity. The SrcIP TES_p showed an abnormal increase in the entropies for $q < 1$ and the SrcPort TES_p for $q < 0$. While we do not know the root cause of the anomaly, we found that the abnormal TES_p is caused by roughly 150,000 hosts sending one or two packets to port 6,501 of a host in the SWITCH network. There was not a single reply to these packets.
- **17.08.2012 at 16:45**: The SrcIP, DstIP, AS and DstPort TES_p for incoming TCP traffic expose abnormal activity hinting at a scan from a small number of sources to a large number of destinations scanning a small number of ports only. Our investigation confirmed the characteristics hinted at by the TES_p. The anomaly is caused by incoming TCP traffic from a single source to roughly 700,000 destinations in the SWITCH network. Almost all flows were directed to port 1433 and consisted of a single packet with a size of 46 bytes. Note that port 1,433 is typically used for remote access to Microsoft SQL Servers for which several remotely-exploitable vulnerabilities have been documented. The massive scan could therefore have been triggered by some malware such as the Gaobot family[15] of worms.
- **29.08.2012 at 18:25**: Here, the only TES_p exposing an abnormal activity is the DstIP TES_p for incoming TCP traffic. The other TES_p and the count metrics all looked normal. Hence, our only clue to find out more about this anomaly was that it involved flows to IP addresses inside the SWITCH network showing low activity only. Unfortunately, our search did not reveal anything conclusive. We assume that the number of flows involved in this anomaly was simply too small. However, we cannot prove this assumption.

5.8 Conclusion

The characterization and visualization of changes in feature distributions involves the analysis and storage of millions of data points. To overcome this constraint, we propose a new method called Traffic Entropy Spectrum. Using

---

[15] This family of worms include exploit code for several remotely-exploitable Microsoft SQL Server vulnerabilities.

a series of anomalies whose characteristics are well-known to us, we evaluate whether the TES is a suitable tool for capturing changes in traffic feature distributions. Our evaluation provides evidence that the TES is indeed a suitable tool for this purpose. However, our evaluation also exposes a weakness of the TES: the inter-region dependency. We address this problem with a modified version of the TES, the TES_p, and show that the TES_p captures changes in a more effective and efficient way. Furthermore, we demonstrate that we can capture changes introduced by different types of anomalies using just a few Tsallis entropy values. Our method does not require adaptation of its parameters even though the network and the underlying traffic feature distributions change significantly. On the detection side, we propose to use the information from the TES (or TES_p) to derive patterns for different types of anomalies. To characterize anomalies in a compact form, we introduce the concept of spectrum patterns and provide evidence that spectrum patterns can indeed be used to characterize and distinguish anomalies. However, while the results of our evaluation are promising, a more general statement about whether or not the TES is a suitable tool for anomaly detection and classification would be too optimistic. For such a statement, the set of anomalies used in our evaluation is simply too small and specific. Moreover, visual inspection might not be a suitable anomaly detection approach: inspecting the various TES is not convenient in practice. Hence, we need to integrate the concept of the TES and the spectrum patterns in a full-fledged anomaly detection and classification approach.

Chapter 6

Entropy Telescope

In the previous chapter, we presented evidence that supported our claim that generalized entropy is an accurate tool to characterize anomalous changes in traffic feature distributions of high-speed networks extracted at the network flow level. We introduced the Traffic Entropy Spectrum (TES) and its refined version TES_p and demonstrated its ability to characterize the structure of anomalies using traffic traces from the border routers of the SWITCH network. While these results derived from visual inspection supported by different coloring schemes are clearly promising, we lack an anomaly detection system which integrates the TES in a fully automated way.

In this chapter, we propose a comprehensive anomaly detection and classification system called the entropy telescope. We also provide evidence for our claim that a detector and classifier built around this tool can detect and classify network anomalies accurately, outperforming traditional volume or Shannon entropy based detectors.

Existing systems show good detection, and reasonable classification performance with regard to massive anomalies. Nonetheless, there remains room for improvement with regard to small to medium sized anomalies. Our extensive evaluation uses three different detection methods, one classification method, a rich set of anomaly models and real backbone traffic. It shows that the TES successfully addresses this challenge by (1) detecting small to medium sized anomalies by up to 20% more accuracy and (2) by improving classification accuracy by up to 27%.

6.1 Introduction

The attractiveness of entropy metrics stems from their ability to condense an entire feature distribution into a single number, whilst simultaneously retaining important information about the overall state of distribution. This makes it possible to detect the concentration and dispersal of feature distributions typical for certain types of attacks, such as DDoS attacks or worm outbreaks.

Compared to simply detecting an anomalous state, it is significantly harder to *classify* an on-going anomaly and identify its root cause. Attempts to combine changes in multiple features in order to establish anomaly patterns are very promising (e.g. [20]), but the accurate automatic classification of anomalies is still a major challenge, especially where anomaly sizes and affected host populations vary. The TES method introduced in the previous chapter is able to focus on specific areas of distributions; for instance, on heavy-hitters, or on rare elements. It thus retains the general advantages of entropy metrics, but also provides additional information about the nature of the changes, which help in distinguishing anomalies. Specifically, the TES evaluates the Tsallis entropy of the traffic feature distribution aggregated over intervals of length T for different values of its characteristic parameter q. Our evaluation in chapter 5 showed how the TES could be used to visually match occurring patterns against known patterns to identify different types of anomalies. We provided evidence for the descriptive power of the so called Spectrum Patterns based on a selection of real anomalies. However, the suitability of TES for large-scale automatic detection and classification has not been evaluated.

In this chapter, we build and extensively evaluate a complete anomaly detection and classification system, which we call the *entropy telescope*. The entropy telescope integrates several components, such as the TES, SVM based pattern-matching, and several detection approaches such as the Kalman filter [83], PCA [48], and KLE [91] (see Figure 6.1).

We rigorously evaluated the entropy telescope with a combination of simulated and real background traffic. As we outlined in Section 2.6, we share the concerns regarding AD evaluation practice expressed in [108] and avoid ground truth identification by manual labeling. Instead, we developed a rich set of diverse flow-level anomaly models inspired by real anomalies. These models allow to vary parameters and to abstract from a specific instance of an anomaly to a broader class, e.g. DDoS attacks of a certain type. Using FLAME [33], it is possible to inject our anomalies to arbitrary trace files. Ease of reproduction and fair comparison of methods are crucial for scientific

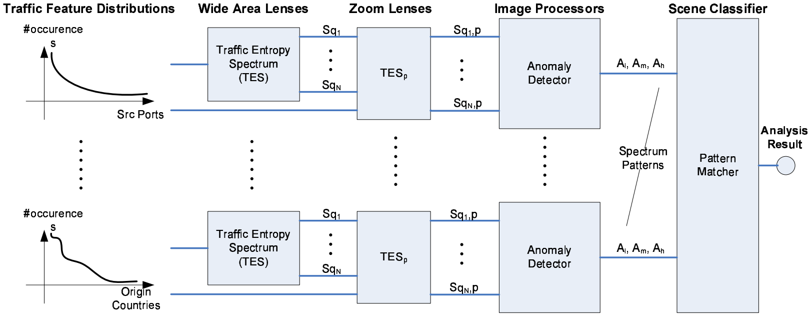


Figure 6.1: *Entropy Telescope building blocks.*

progress. For these reasons, and to foster further research in this direction, we have made the set of anomaly models designed for this study publicly available [184]. Furthermore, we provide access (on request) to the labeled time series data along with a MATLAB toolset to process them. When evaluating the entropy telescope, we found that when switching from Shannon to the refined TES approach, the PCA method detects small to medium sized anomalies up to 20% more accurately. The classification accuracy is improved by up to 19% when switching from Shannon-only to TES and by another 8% when switching from TES to the refined TES approach. Finally, to complement the evaluation with injected anomalies, we ran the entropy telescope on a 34 days trace from a backbone network and reported on the prevalence of traffic anomalies. The most prevalent anomalies found in this trace were different types of scanning (69%-84%) and reflector DDoS attacks (15%-29%).

The remainder of this chapter is organized as follows. In Section 6.3 we describe our data set, the traffic features we use, and the anomaly models we designed. In Section 6.2, we describe the different components of the entropy telescope in detail before evaluating the detection and classification accuracy of several techniques in Section 6.4. Section 6.5 concludes the chapter.

6.2 Entropy Telescope

In this section we describe the entropy telescope, which consists of Wide Angle Lenses (6.2.1), Zoom Lenses (6.2.2), Image Processors (6.2.3) and a Scene Classifier (6.2.4). Figure 6.1 gives an overview of the different components. The Wide Angle Lenses capture the big picture in order to inform the Zoom Lenses which region they should focus on. The Image Processors then take the signals from the zoom lenses and check them for anomalies. If the composed image is considered anomalous, it is condensed into a so-called Spectrum Pattern and fed to the Scene Analyzer for identification.

6.2.1 Wide Angle: Using Generalized Entropy

The task of the Wide Angle Lenses is to calculate the TES from the different traffic feature distributions fed to the detector. As we discussed in chapter 5, the TES suffers from inter-region dependency. Changes in one region might affect other regions. The reason for this problem is the “global” focus of the TES. The calculation of region-specific entropies encompasses total activity in all of the regions. To mitigate this problem, the TES are handed over to the Zoom Lenses.

6.2.2 Zooming in: Separating Activity Regions

The task of the Zoom Lenses is to now use the TES from the Wide Angle Lenses to determine, which elements (e.g. which IP addresses) contribute most to the respective entropy value, for each of the Tsallis enropies. The Zoom Lens then recalculates the TES using only those elements. In doing so, the Zoom Lens shifts from a global focus and zooms in on the elements most relevant for the Tsallis entropies for the different values of q. The only parameter of the Zoom Lens is the cut-off condition p. It defines the percentage of the original entropy value that must be reached for the Zoom Lens to stop adding elements to the set used when re-calculating the TES. In our evaluation, we experiment with the following values for p: 80, 95, and 99. A detailed description of the calculation of the TES_p and the role of the parameter p can be found in chapter 5.

6.2.3 Image processing: Anomaly Detection

In this section we describe how anomaly detection is performed on the various entropy signals for different metrics and q-values. Specifically, we use 20 different values for q:

$$q \in \{-3\} \cup \{-2, -1.75, \ldots, 1.75, 2\}.$$

Experiments with traces from different years and containing different known and unknown anomalies suggested that including bigger or smaller values is of limited use: S_2 is already very much dominated by the biggest heavy-hitter and S_{-3} by the rarest elements, respectively. With 8 feature entropies [1], 3 volume metrics[2], and two directions, this yields a total number of $2*(3+8*20) = 326$ different metrics for TES_p. Note that this component is not limited to working with the full set of metrics. It can also be used with other sets like the Shannon classic (SHN_C) or the Shannon extended (SHN_+) set used in our evaluation (see 6.3.2). These sets contain only a subset of the aforementioned metrics. Shannon classic (SHN_C) consists of $2*(3+4) = 14$ metrics, and Shannon extended (SHN_+) of $2*(3+8) = 22$ metrics.

The computational overhead is already dominated by the generation of element distributions. Whether we compute a single entropy value or draw multiple values from a distribution does not make a big difference in terms of running time or memory consumption.

From the list of available statistical anomaly detection methods, including wavelet transformation [82], Kalman filter [83], Principal Component Analysis (PCA) [20], and Karhunen-Loeve Expansion (KLE) [91], we selected the Kalman filter due to its simplicity as well as the PCA and the KLE method because they reflect the most advanced methods currently available.

- **The Kalman filter** models normal traffic as a measurement-corrected AR(1) auto-regressive process plus zero-mean Gaussian noise. The difference between this model and the actually measured value is the residual, a zero-mean signal without the diurnal patterns found in original time series. We calculate this residual for all input time series separately.

- **The Principal Component Analysis (PCA)** condenses the information of all input time series to a single output time series reflecting how closely the current input matches the model built from some other input. The output signal reflecting the difference between the model and the actually measured values is the residual. PCA has a parameter k determining how many of the components are used for modeling the normal activity. We discuss the impact of k in our evaluation section.

---

[1] Source and destination port number, source and destination IP address, AS number, country code, flow size in bytes and bytes per packet.

[2] Flow, packet, and byte count.

- **The Karhunen-Loeve Expansion (KLE)** is based on the Karhunen-Loeve Transform and an extension of the PCA method which accounts for temporal correlation in the data. The output signal (or residual) reflects the difference between the model and the actually measured values. The only important difference is that KLE has an additional parameter m stating how many time bins should be included when accounting for temporal correlations.

Our goal is not the optimization of the detection step, but rather to demonstrate that the extended set of Tsallis entropy values improves the detection accuracy using existing methods.

On the residual(s) we detect anomalies using a quartile-based approach. The first quartile Q_1 of a sample of values corresponds to the 25th percentile and is defined as the value that cuts off the lowest 25% of values. That is, one fourth of the values are smaller than Q_1. Similarly, Q_2 (the median) and Q_3 are defined as the 50th and 75th percentile, respectively. The interquartile range IQR is a measure of statistical dispersion and is defined by $IQR = Q_3 - Q_1$. The IQR can be used to detect outliers by defining a normal range of values $[Q_1 - k \cdot IQR, Q_3 + k \cdot IQR]$ for some constant k. We choose $k = 1$ and define the normalized anomaly score $A(x)$ for a residual value x by the ratio of the distance of x from the normal band and the size of the normal band, which is $3IQR$:

$$A(x) := \begin{cases} \frac{x-(Q_3+IQR)}{3IQR} & \text{if} \quad x > Q_3 + IQR \\ \frac{x-(Q_1-IQR)}{3IQR} & \text{if} \quad x < Q_1 - IQR \\ 0 & \text{else (signal is normal)} \end{cases} \tag{6.1}$$

For each output time series, we compute the anomaly score and call it a *vote* if the signal is exceeding a threshold t, that is, $|A(x)| > t$.

In the case of PCA and KLE, we have only one output time series. As a consequence, one vote is enough to trigger an anomaly and the threshold t is the main parameter to tune the sensitivity of a specific detector[3]. However, in the case of the Kalman filter, we have one residual per input time series. Detection is done using a two parameter approach. Firstly, we do the same as in the case of PCA and KLE for each of the output time series. We put

---

[3] Note that there are other tuning parameters such as the parameters k for PCA and k and m for KLE as described before.

a threshold t on all of the anomaly scores A(X) of their residuals. Next, we perform the detection by setting a minimum number v of votes required to trigger an *alarm* for the current time interval. In practice, determining good values for the threshold t and votes v is done by measuring the performance of the detector for different combinations of t and v. Ideally, this is done using training data containing a representative set of anomalies. The same holds for determining k for PCA and k and m for KLE or any other anomaly detection system having one or more tuning parameters. In summary, we need to sweep the following tuning parameters to fully assess the performance of the different algorithms:

- **Kalman**: Threshold t and number of minimum votes v.
- **PCA**: Threshold t and the number k of components used for modeling the normal activity.
- **KLE**: Threshold t, the number k of components and the number m of time bins used for modeling the normal activity.

Note that all of the three approaches require training data for two reasons: (1) for defining a conservative normal band to derive the normalized anomaly score A(X) and (2) to get training data for training the models used by the Kalman, PCA and KLE methods. While the first training problem is easy to solve, the second one is more difficult. The reason for this is that our IQR based normalization is based on the first and third quartiles, which do not depend on the 25% smallest and biggest values in the data. It is therefore not affected by outliers. Unfortunately, to solve the second training problem, we need all of the data points. To ensure that the training data actually reflects normal behavior, we selected it based on manual analysis of the time series using box plots and raw time series plots. While there remains an uncertainty over whether our selection of training data really is clean and representative, we mitigated this by confirming our findings using different training samples. However, we can not omit this problem entirely when working with real traces containing millions of flows per hour.

In our evaluation, we focus on those configurations showing the “best” performance for a specific method. We are aware of the fact that different sets of anomalies and/or other background traffic characteristics might result in a different choice for these values. Worse, they might result in a different rating for the different methods. However, we believe that our comparison is fair for two reasons: (1) the selection of the “best” parameters is based on a large set of different anomaly types and intensities. This removes any potential bias caused by certain anomalies appearing more frequently than others, as is typically the case with real world traces. And (2), our traffic trace used as background traffic originates from a large stub AS with fairly complex and dynamic traffic mix characteristics.

6.2.4 Scene Analysis: Classifying Anomaly Patterns

The basic idea behind the scene analysis component is the notion of *Spectrum Patterns* introduced in the previous chapter and in [31]. The assumption underlying our anomaly classification is that each anomaly class leaves a characteristic and (to some extent) invariant footprint in different features and activity regions. As a consequence, the input to this component must be one signal per count or feature entropy. While the input signals could be the original time series signals of these features, we want to avoid this for two reasons. Firstly, removing trend and daily patterns from the signals is difficult but has to be done for most supervised pattern recognition approaches. Secondly, we are not interested in the exact amplitude of the signals. Rather we seek a conservative estimate as to whether they are abnormal and, if so, by how much.

An obvious choice for the input of the classification component is therefore the output of the Kalman detector, as it outputs a conservative anomaly score per input time series. To reduce the volume of data provided by this detection component, we aggregate anomaly scores in three buckets corresponding to the low/medium/high activity regions. We do so by calculating the weighted sum of the scores for all q-values in a region. The low activity region is defined by $q \leq -1$, medium by $-1 < q < 1$, and high by $q \geq 1$. That is, we calculate three values for each metric, measuring the abnormality of the specific region, denoted by A_l (low), A_m (medium), and A_h (high). While different weights might be used to tune our classification approach in future work, we found that the simplest choice of setting all weights to one is enough to achieve a classification accuracy of around 85 percent.

Next, the Scene Analyzer scans the values A_l, A_m, and A_h of each traffic feature and decides whether they signal an increase, decrease or no change of entropy of the corresponding regions. This transformation can be summarized as follows:

$$C_i := \begin{cases} \text{`1'} & \text{if } A_i \geq \text{upper threshold} \\ \text{`0'} & \text{otherwise} \\ \text{`-1'} & \text{if } A_i \leq \text{lower threshold} \end{cases}$$

An example of such a pattern is shown in Figure 6.6 of the evaluation section. For the upper and lower threshold, we use the values 0.5 and -0.5 respectively. A value of $A_i = 0.5$ is obtained, if each metric contributing to A_i exceeds its 75th percentile value by around $1.2 * IQR$[4]. Another situation resulting in $A_i = 0.5$ is when one of the metrics contributing to A_i has an anomaly score of 0.5 and all others an anomaly score of zero. From (6.1) it follows that for an anomaly score of 0.5, the metric exceeds the 75th percentile value by $2.5 * IQR$. Note that a deviation of $1.5 * IQR$ is typically attributed to *mild outliers* while a deviation of at least $3 * IQR$ is attributed to *extreme outliers*.

The main reason for transforming the continuous values A_l, A_m, and A_h of each traffic feature into discrete (tri-state) values is to avoid the pitfall of over-fitting our classifier to specific amplitudes. Despite the good results produced by this approach, we need to investigate the impact of this quantization in more detail. However, not using quantization should mainly improve the classification quality in cases where the input signals are not well-behaved, in the sense that the IQR is not meaningful for separating normal and abnormal values. An example of such a signal is where a signal has a more or less bi-modal distribution of its values during normal activity.

Lastly, the Scene Analyzer feeds the discretized spectrum pattern to a support vector machine (SVM) trained with different training sets, discussed in the evaluation section. Our Scene Analyzer makes use of the LIBSVM [185], a popular SVM with very good performance and a wide range of available interfaces. For each of the different training sets, we followed the basic strategy outlined in [186]. First, we split the full dataset into three parts containing approximately the same amount of anomalies of each anomaly type and size. Next, we take two parts of the split for training and one part for validation. By doing this, we get three different training and validation set combinations. On the training set, we then perform a grid search and 3-fold cross-validation to identify the best parameters for the SVM's RBF kernel. The classification result reported in the evaluation section is the average classification accuracy obtained from the three training and validation set combinations. Note that the output of the SVM - the label of the anomaly - is at the same time the final result and output of our Entropy telescope.

---

[4] With 5 metrics as in the high activity region, we get $A_h = 0.5$ if all metrics have an anomaly score of 0.1. It follows from (6.1) that an anomaly score of 0.1 is the same as exceeding the 75th percentile by $1.2 * IQR$.

6.3 Methodology

6.3.1 Data Set

For our evaluation, we again use NetFlow data captured from SWITCH [34]. For our analysis of the prevalence of real-world anomalies, we use a period of 34 days from 07/31/2008 until 09/02/2008 (see Sec. 6.4.3). For evaluating the entropy telescope with injected anomalies, we use one week of the month-long trace from 08/09/2008 0:00am to 08/15/2008 11:59pm.

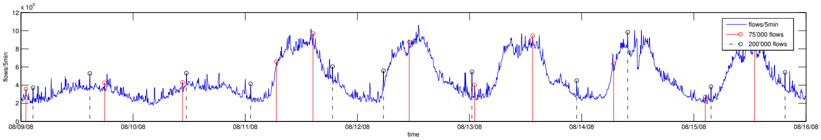


Figure 6.2: *The number of flows per 5min bin of our baseline trace with injected anomalies of intensity 75K and 200K.*

6.3.2 Entropy Features

In addition to packet, flow, and byte count, we compute the entropy of different traffic feature distributions. We define the following basic set of traffic features:

- ***Shannon classic*** **(**SHN_C**)**: The Shannon entropy of the source/destination port and the source/destination IP address distribution.
- ***Shannon+*** **(**SHN_+**)**: The same traffic features as in SHN_C but extended with the Shannon entropy of the following additional feature distributions:
  - AS distribution
  - country code distribution
  - average packet size per flow distribution
  - flow size distribution
- ***Tsallis sets*** **(**TES_p**)**: Based on the same feature distributions as SHN_+.

For AS numbers and country codes, the distribution is always computed from external addresses only, as we have data from a single stub AS.

To justify the selection of these features, we performed a detailed analysis of whether it is necessary and/or useful to use all of the 7 (11) features in SHN_C (SHN_+). This analysis can be found in chapter 4 and [30] where we discuss this issue based on a comprehensive pairwise correlation analysis. Our results suggest that different feature entropies *do indeed provide useful information*.

6.3.3 Anomaly Models and Injection

To evaluate the accuracy and sensitivity of the anomaly detector and the anomaly classifier component, we injected artificial anomalies into one week of real background traffic using FLAME [33]. This approach has two main advantages. First, it provides well-defined ground truth independent of an expert labeling the events. Second, it is able to inject the same type of anomaly in different scales, with different parameters, and at different offsets. Thus, the evaluation is not biased by the very set of anomalies accidentally present in a collected trace [108]. However, for background traffic, we chose to use real instead of simulated traffic to get more realistic results. The main problem with real background traffic is that it potentially contains anomalies for which we do not know the ground truth. Therefore, we first inspected the background traces for existing anomalies by searching for heavy outliers in each traffic feature using a robust statistical outlier definition [187] based on the interquartile range. Where obvious anomalies were found, we labeled the traces accordingly and did not consider the corresponding time bins for injection and validation. To mitigate the effect of smaller anomalies still present in the trace, we injected each anomaly at different random locations.

Previous work argues that concentrated activity on few elements (e.g. the victim of a DDoS attack) leads to a decrease in entropy, while dispersed activity (e.g. the spoofed source addresses of the same DDoS attack) leads to an increase in entropy [20, 25, 31]. However, this is not necessarily true. The precise effect on the entropy metric depends on whether the element set involved in a change was already present in the traffic before (intrinsic event) or not (extrinsic event). Therefore, we explicitly consider, for instance, sets of active and inactive IP addresses.

The 20 base anomalies listed in Table 6.1 are variations of DDoS attacks, worm outbreaks, scans and P2P outages. Each combination of base anomaly and intensity was injected in at least 42 different (random) timeslots.

ID	Anomaly Type	Description	SRC/DST, variation of IPs
	Reflector DDoS I	DDoS with few sources but medium intensity from each source	Attacker: OUT, Victim: OUT, Reflectors: IN
1		Reflector IPs in LAR	
2		Reflector IPs in HAR	
	Reflector DDoS II	DDoS with few sources but medium intensity from each source	Attacker: OUT, Victim: IN, Reflectors: OUT
		Matches other similar attacks such as coordinated password-guessing	
3		Attacker IPs in LAR	
4		Attacker IPs in HAR, Victims in LAR	
5		Attacker IPs in HAR, Victims in HAR	
6	DDoS I	Botnet DDoS 1 (SYN flood)	Victim: IN, Attacker: OUT
7	DDoS II	Flash Crowd / Botnet DDoS 2 (HTTP GET requests)	Victim: IN, Attacker: OUT
8	DDoS III	DDoS with spoofed sources (SYN flood)	Victim: IN, Attacker: OUT
9	Worm I	Worm Outbreak (Blaster)	Victims: mainly IN, Attacker: mainly OUT
10	Worm II	Worm Outbreak (Witty)	Victims: mainly IN, Attacker: mainly OUT
11	P2P	P2P Supernode outage (distributed scanning event)	Mix of external/internal addresses
	Scan I	Scanning from single host outside	Victim: IN, Attacker: OUT
12		All ports on single victim	
13		All ports on subnet (hosts in LAR)	
14		Selected ports on subnet (hosts in LAR)	
	Scan II	Scanning from a botnet (˜2000 hosts in LAR)	Victim: IN, Attacker: OUT
15		All ports on single victim	
16		All ports on subnet (hosts in LAR)	
17		Selected ports on subnet (hosts in LAR)	
	Scan III	Scanning from single host inside	Victim: OUT, Attacker: IN
18		All ports on single victim	
19		Selected ports on subnet and random IPs	
20	DoS	DoS (1 to 1), HTTP GET requests	Attacker: OUT, Victim: IN

Table 6.1: *Overview of 20 base anomaly models used. HAR/LAR denotes high/low activity region.*

For each injection, the flow parameters, such as the source/destination IP address or the source/destination port are drawn from the feature distribution defined by the models. Furthermore, depending on the base anomaly model, the feature distributions for some of the flow parameters were modified according to the schemes described below. As a consequence, each injected anomaly is uniquely parameterized. For more details, we refer to the model description files for FLAME which we make available on [184]. In total, we injected 8064 anomalies into our baseline trace. Or more precisely, we injected 42 anomalies in each of the 192 copies of our baseline trace.

Anomaly Intensity

Each base anomaly is injected with various intensities, defined by the number of injected flows per 5min. Chosen intensities are 50,000 (50K), 75,000 (75K), 100,000 (100K), 200,000 (200K), 500,000 (500K), and one million (1M) flows. Note that the actual number can vary a bit since the injection decision is probabilistic. The motivation for this choice is that the intensities should be (1) realistic and (2) small enough that for most of them the anomaly is invisible when using simple metrics, such as flow count only. We verified these criteria by analyzing the intensities of a set of well-known anomalies and checked that most intensity values are hard to spot when considering the variability and the average number of flows per 5min bin contained in our traffic traces. We illustrate this with Figure 6.2 showing a plot of the number of flows per 5min bin of our baseline trace into which we injected several anomalies of intensities 75K and 200K. While the anomalies of intensity 75K do not cause a significant change in the flow count signal, those of intensity 200K start to become visible. However, most of the time they do not stand out clearly, instead vanishing in the normal variability of the flow count signal.

IP addresses

As our traffic traces are collected from a stub AS, we distinguish addresses from the internal address space (IN) and external addresses (OUT). In our anomaly models, the victims are located inside our stub AS, except for the case of the reflector DDoS I and Scan III model. We observed that the characteristics of the traffic flowing into the network show a higher variability than those of the traffic leaving our network. Hence, if we place the victims inside our AS, and if the anomalous traffic to the victim(s) is more pronounced than the response traffic, the more pronounced share would be part of the traffic with higher variability and therefore be more difficult to isolate. Previous work, as well as intuition, confirms this imbalance for most anomalies. Most victims of scans do not reply because the scan is blocked by a firewall, and victims of a DDoS attack do not reply to (all) requests because they have crashed or are simply too busy to serve all requests.

Another important aspect is whether the hosts acting as the source and/or destination of normal and abnormal traffic are mostly from disjoint sets of hosts or not, and whether they target hosts that show rather high or low activity.

To take this into account, we defined and constructed various IP address sets based on an analysis of the persistence and activity of IP addresses in our baseline trace and draw IP addresses from many combinations of activity regions and set sizes. The source and destination IP addresses for one instance of an anomaly of the base anomaly types described in Table 6.1 are then determined as follows. For each flow, the source and destination IP address are drawn from a set of IP addresses assigned to this anomaly. If multiple sets are assigned, only one is used for a specific anomaly instance. But in total, all sets are used the same number of times.

The sets used for our evaluation are the following:

- **IP:** A single fixed IP measured from real attacks.
- **IP-LA / IP-HA:** An IP with low/high activity.
- **IPS:** IPs from all activity ranges.
- **IPS-HA:** IPs with high activity.
- **IPS-LA:** IPs with low activity.
- **IPS-Pxx:** IPs with activity on port xx.
- **IPS-Pxx-HA:** IPs with high activity on port xx.
- **IPS-Pxx-LA:** IPs with low activity on port xx.
- **IPS-RAND:** Randomly chosen IPs. They might or might not be present in the base trace.

An IP address shows low activity (LA) if it occurs on a more or less regular basis but is not the source/destination of a significant number of flows (typically less than 10 flows per protocol and 5 minutes). An IP address showing high activity (HA) is one that occurs on a regular basis and is the source/destination of a significant number of flows (typically more than 100 flows per protocol and 5 minutes). To indicate the size of the sets, we append the number of IP addresses to the set name. Also, the prefixes INT and

EXT denote whether IP addresses were chosen from the internal or external address range. For instance, the set INT-IPS-HA-5000 contains 5000 IP addresses randomly chosen from highly active internal addresses. Likewise, the set EXT-IPS-RAND-2.5MIO contains 2.5 million random addresses from the external range. Table 6.2 shows which sets were used for which anomaly type.

ID	Attacker IPs	Victim IPs	Reflector IPs
1	EXT-IP	EXT-IP	INT-IPS-P80-LA-{500,2000}, INT-IPS-P80-5000
2	EXT-IP	EXT-IP	INT-IPS-HA-{500,2000,5000}, INT-IPS-P25-HA-{500,2000}
3	EXT-IP	INT-IP-{LA/HA}	EXT-IPS-P25-LA-2000, EXT-IPS-LA-500
4	EXT-IP	INT-IP-LA	EXT-IPS-P25-HA-500, EXT-IPS-HA-{2000, 5000}
5	EXT-IP	INT-IP-HA	EXT-IPS-P25-HA-500, EXT-IPS-HA-{2000, 5000}
6	EXT-IPS-LA-{5000,10000}	INT-IP-HA	n/a
7	EXT-IPS-LA-{5000,10000}	INT-IP-HA	n/a
8	EXT-IPS-RAND-2.5MIO	INT-IP-HA	n/a
9	EXT-IPS-RAND-2.5MIO	INT-IPS-RAND-0.5MIO	n/a
10	EXT-IPS-RAND-2.5MIO	INT-IPS-RAND-0.5MIO	n/a
11	INT-IPS-1000	EXT-IPS-20	n/a
12	EXT-IP	INT-IP-LA	n/a
13	EXT-IP	INT-IP-LA-1200	n/a
14	EXT-IP	INT-IP-LA-1200	n/a
15	EXT-IPS-LA-2000	INT-IP-LA	n/a
16	EXT-IPS-LA-2000	INT-IP-LA-1200	n/a
17	EXT-IPS-LA-2000	INT-IP-LA-1200	n/a
18	INT-IP	EXT-IP	n/a
19	INT-IP	EXT-IPS-2000	n/a
20	EXT-IP	INT-IP	n/a

Table 6.2: *IP address sets used to customize anomaly models. The ID column corresponds to the anomaly ID in table 6.1.*

Ports

For application specific attacks and worm outbreaks exploiting vulnerabilities, we selected fixed ports. For instance the HTTP GET requests used in DDoS attacks are targeted at port 80. Otherwise we assign random ports (i.i.d.) from these sets: all ports, ports above/below 1024, selected set of application ports, and a dynamic port range (1024-4999).

Packet Sizes

Depending on the attack, we modeled different stages of the 3-way TCP handshake with different response probabilities from $\{0.0001, 0.02, 0.05, 0.2, 0.8\}$. For HTTP requests and flash crowds, we modeled a percentage of delivered web pages of size 0.5KB and 20KB, distributed over several packets. For worm attacks we used characteristic packet sizes known from studies of the Blaster [50] and Witty [188,189] worm. For the reflector DDoS, we measured the actual flow and packet size distributions during a real attack found in our traffic traces and used these distributions for modeling.

6.4 Evaluation

In this Section, we evaluate the entropy telescope using the feature sets SHN_C, SHN_+, TES, $TES_{99.9}$, TES_{99}, TES_{95}, and TES_{80}. We show that the biggest improvement in detection accuracy can be achieved when switching from Shannon entropy based feature sets to the TES set. The novel TES_p makes another significant step in classification accuracy and optimizes detection for some anomaly categories.

After discussing our detection and classification results thoroughly, we conclude the section with an analysis of anomaly prevalence in a 34-days trace of real traffic.

6.4.1 Detection

In Section 6.2.3 we defined a metric as anomalous, denoted by a *vote*, if its anomaly score is bigger than a threshold t, i.e., $|A(x)| > t$. Moreover, for an anomaly alarm to be raised in a time slot, a number of v votes need to be present. For the PCA and KLE method, v is equal to one since they have only one output time series. Naturally, high thresholds for t - and in the case of the Kalman filter also for v - will lead to low true/false positives while low thresholds lead to high true/false positives. The preferred operation point, however, has a high true positive (TP) and a low false positive (FP) rate. To assess detector performance, we use ROC curves [190, 191] that plot the TP rate versus the FP rate for a range of threshold values. In our case, we vary t between 0 and 100. Note that for ease of readability, we plot the ROC curves using a logarithmic scale for the FP axis and display the results for FP rates of 0.4% to 10%. With our time bins of 5 minutes, this corresponds to roughly one false positive per day for an FP rate of 0.04% to one false positive per 50 minutes if the FP rate is 10%.

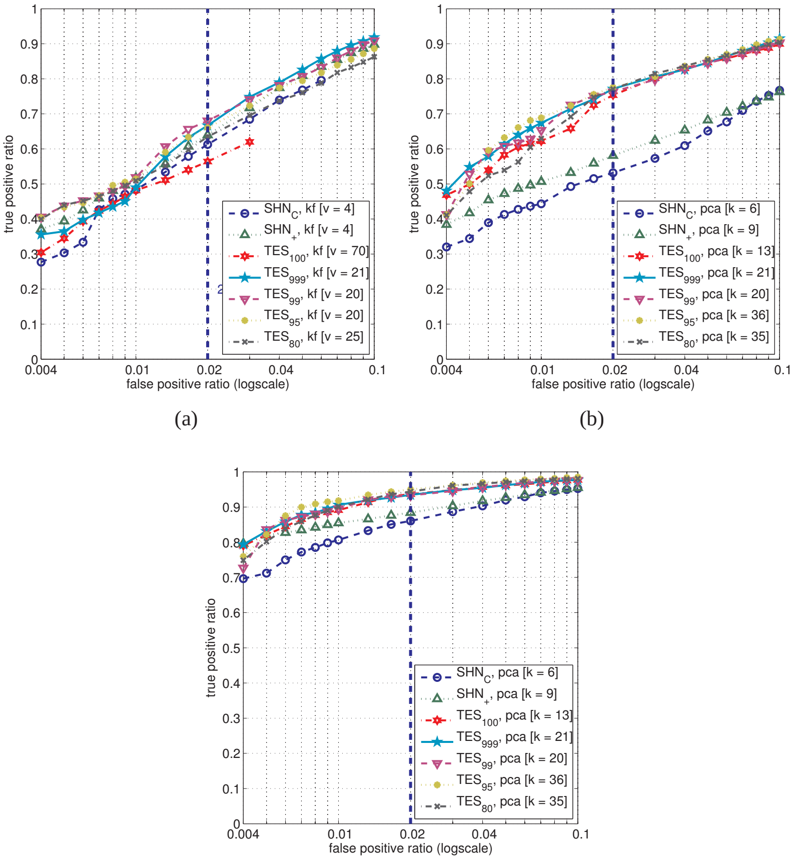
Issues with KLE

The following discussion focuses on the evaluation results for the Kalman and the PCA methods only. The reason for this is that our results for KLE are somewhat ambivalent. For intensities larger than 100K, KLE shows a worse performance than PCA for all feature sets. The same holds for the feature sets TES or TES_p and anomalies of intensity up to 100K. However, for SHN_C and SHN_+ and anomalies of intensity up to 100K, we see an improvement in detection quality of up to 15%. While the improvement for SHN_C is consistent with the finding in [91], we are not quite sure about the root cause for the results with other feature sets. More research is required to better understand the performance of the KLE method with different feature sets, anomalies and network characteristics.

Shannon versus TES feature sets

Figure 6.3 shows the ROC plots for the Kalman and PCA method for intensities 50K and 75K as well as the PCA method with 100K and 200K. The plots show the detection accuracy for the best configuration of different detectors for the feature sets. To find the best configurations, we performed an extensive parameter sweep for both the Kalman and PCA detector. For PCA, the parameter is the number of components k used to build the model of normal activity. For Kalman, the parameter is the number of votes v required to trigger an alert. Doing these sweeps, we found that while the detection accuracy changes quickly for the feature sets SHN_C and SHN_+, there is a clear peak for one specific value of k. In contrast, this is not true for TES or TES_p. After reaching the optimal detection accuracy, it remained at a comparable level for a wide range of k values. One interpretation of this is that the additional time series in the TES feature sets make the detectors more robust with regard to the selection of the parameter k.

From the plots in Figure 6.3 we can see that a switch to TES improves the detection accuracy for PCA by up to 20%. However, for the Kalman filter approach, the gain is rather small and lies around 5% for TES feature sets other than TES or TES_{80}. It seems that while the TES adds features carrying valuable information, it also adds noise with which the simple per feature detection and voting scheme of the Kalman detector does not cope well.



(a)

(b)

(c)

Figure 6.3: *ROC curves for different feature sets and detection methods: (a) Anomalies of intensity 50K and 75K, Kalman filter method. (b) Anomalies of intensity 50K and 75K, PCA method (c) Anomalies of intensity 100K and 200K, PCA method.*

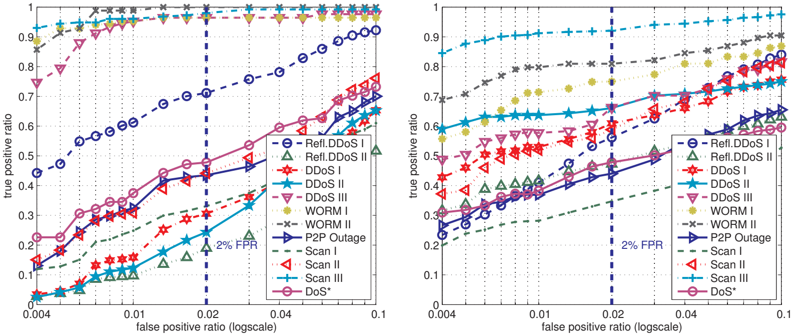
Unlike PCA, our Kalman detector does not make use of inter-feature relations. That this is the main reason for the poorer performance is supported by the Kalman filter's very poor performance for TES but significantly improved performance for TES_p. As explained in section 6.2, the features reflecting the high and low activity area can be strongly correlated in TES, but are not correlated in TES_p. As a comparison of the different plots in Figure 6.4 shows, the improvement in detection accuracy can also be confirmed when looking at the detection accuracy per anomaly type. Switching from SHN_C or SHN_+ to TES improves detection accuracy for most types for FP rates of 0.6% (=1 alert per 14 hours) and above.

SHN_C versus SHN_+

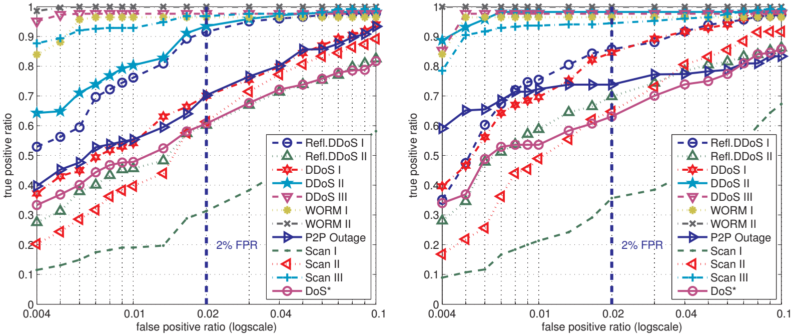
Another observation we can make based on Figure 6.3 is that our extension of the traditional feature set SHN_C to SHN_+ improves detection results by up to 10%. This, as well as the increase from $k = 6$ to $k = 9$ components required to achieve the best detection accuracy with PCA, confirms that the features added to SHN_C carry relevant information. Nevertheless, as can be seen in Figure 6.4, the better overall detection accuracy comes with a decrease for the anomaly types Worms I&II, DDoS III and Scan III while most of the other types show an increase in detection accuracy.

Kalman versus PCA

However, the most surprising result is exposed when comparing the performance of the different detection methods for the feature set SHN_C and SHN_+ in Figures 6.3(a) and 6.3(b): The Kalman filter method detects anomalies up to 10% more accurately than the PCA method. Considering that PCA has been used with the feature set SHN_C in the past, this is an interesting finding. However, since it only holds for anomalies with intensities less than 100K, PCA might still be the best choice for SHN_C in general. The effect disappears when the feature set is extended to TES_p. There, we found that the PCA method provided consistently better results than the Kalman filter method.



(a) SHN_C,k=6 (b) SHN_+, k=9



(c) TES_{100}, k=13 (d) TES_{95}, k=36

Figure 6.4: *ROC curves for anomalies with small intensities (50K and 75K) and PCA detection method.*

Relations between parameters *k*

A final observation from Figure 6.3 is that the optimal *k* value for both Kalman and PCA increases when switching from Shannon to TES. The increase is even of comparable size. However, this is not true for the TES_p. This is because the Kalman method does not make use of inter-feature relationships such as the correlations between high and low activity regions in the TES.

In summary, the shift from Shannon-based feature sets to TES-based sets can improve detection accuracy up to 20%. The reason why a shift from TES to a refined version of the TES leads to only minor improvements might be the fact that the main difference between TES and TES_p is the decorrelation of the HIGH/LOW intensity parts of the distribution. Intuitively, we are not concerned whether an anomaly is seen in two (correlated) metrics or just one (uncorrelated) metric for detection. For PCA and KLE, which account for correlations between metrics, this makes no big difference. We believe that the minor gains are most likely due to the better signal to noise ratio for anomalies affecting the low activity region only. In the TES, such anomalies could be concealed by large (but not yet anomalous) changes in the overall activity.

6.4.2 Classification

It is important that detection and classification rely on models that are robust with respect to varying intensities. That is, if we train an SVM with DDoS models of a certain intensity, we do not want to miss the same attacks simply because the real attack size differs slightly from the training size. Therefore, we trained the SVM with different intensities and evaluated the models on varying intensities. We always trained all of the 20 base models from Table 6.1. For measuring classification accuracy, we counted the percentage of anomaly instances that were assigned to the correct base model. Thus, if anomaly #16 was classified as anomaly #17, this is considered incorrect, even though both belong to the same base anomaly type (Scan II). *For assessing classification quality we assumed a perfect detector.* That is, the true anomalous intervals are considered for classification. In a real environment, classification would only be run on those instances that were detected by an anomaly detector in the first place. The consequence of this is that the difference between classification accuracy using Shannon entropy or Tsallis entropy would be even bigger in practice because a detector based on Shannon entropy would feed more false positives to the classifier.

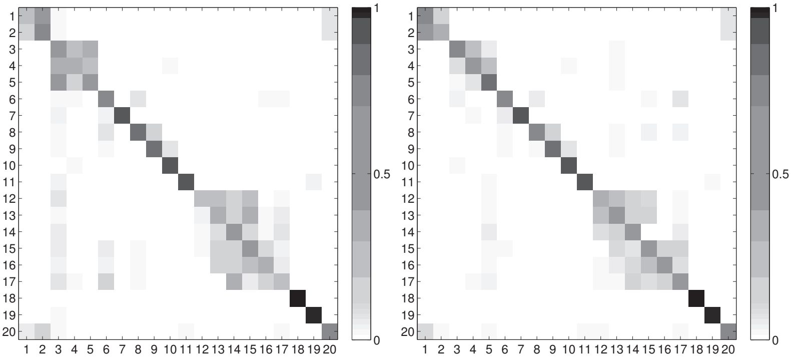
Table 6.3 summarizes the classification accuracies for different anomaly intensities and feature sets. The columns labeled with arrows (⇒) show the performance difference between the feature sets on the left and right side. The use of SHN_+ over SHN_C yields a gain in classification accuracy of between 7.14% and 14.21% across all intensities. Using TES (TES_{100}) gives an additional gain of 7.84% to 9.38% for small intensities in the top three rows. For training and classification with bigger anomalies, the gain is generally smaller. Although accuracy with TES is already quite high, the introduc-

Train	Evaluation	SHN_C	$\Rightarrow$	SHN_+	$\Rightarrow$	TES_{100}	$\Rightarrow$	$\mathrm{TES}_{99.9}$	TES_{95}	TES_{80}
50K	50K	55.13	10.42	65.55	9.38	74.93	7.14	80.58	82.07	80.95
	>50K	54.73	8.78	63.51	7.84	71.35	7.16	74.26	78.51	77.35
200K	<200K	49.38	8.04	57.42	9.13	66.54	8.43	72.82	74.98	73.83
	200K	66.07	14.06	80.13	2.16	82.29	5.21	86.53	87.50	87.72
	>200K	64.69	14.21	78.91	1.49	80.39	4.09	80.95	84.49	84.34
ALL	<200k	60.91	7.14	68.06	8.85	76.91	7.04	80.95	83.95	86.46
	200K	68.30	11.68	79.99	3.65	83.63	4.17	85.27	87.80	87.80
	>200K	67.49	13.36	80.84	1.75	82.59	3.50	83.15	86.09	82.96

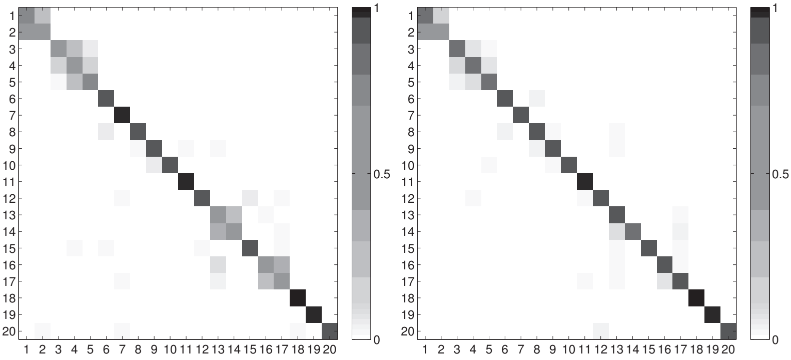
Table 6.3: *Average classification accuracy as a percentage, for different sets of features and for different training and validation data set constraints.*

tion of the pruned TES_{95} adds another 5.8% on average. While choices of $p = 99.9$ and $p = 80$ also improve over TES, $p = 95$ works best in our setting. The average aggregated gain of TES_{95} over SHN_C is 22.3%, leading to an average classification accuracy of 83.17%. The improvement is generally bigger for low-intensity anomalies.

SHN_C and SHN_+ often misclassified anomalies of types 3-5 and 13-18. As expected, the classification accuracy with regard to sub-types of the broader anomaly types increases when switching from SHN to TES feature sets. This is expected since TES provides a more detailed view on the changes in a distribution. For a broad classification, these details are clearly less important.



(a) SHN$_C$ (b) SHN$_+$



(c) TES$_{100}$ (d) TES$_{95}$

Figure 6.5: *Base anomaly classification matrix. The plots show which injected base anomaly types (y-axis) were classified as which types (x-axis) with what probability (color code). Models were trained using anomalies of ALL intensities. Classification is performed on anomalies with intensity <200K.*

We illustrate the learned SHN_+ and TES_{95} anomaly patterns for base anomaly #10 (Worm II) in Fig. 6.6. Grey areas indicate metrics within normal range, red areas (+) represent metrics with a positive anomaly (upper threshold was exceeded), and blue areas (-) show negative anomalies. Hatched areas indicate information not available in SHN_+ patterns. For TES_{95}, each feature is represented by the three regions low, medium, and high activity, whereas for SHN_+, only a single value ($q = 1$) is available. In both directions, SHN_+ misses important information about changes in various features, including IP addresses. For direction IN→OUT, the bytes per packet (BytesPP) distribution shows that, while SHN_+ detects a decrease in entropy, TES_{95} has more detailed information about the change. In particular, it shows that the decrease occurred in the high activity region, while in the medium region there was actually an increase of entropy.

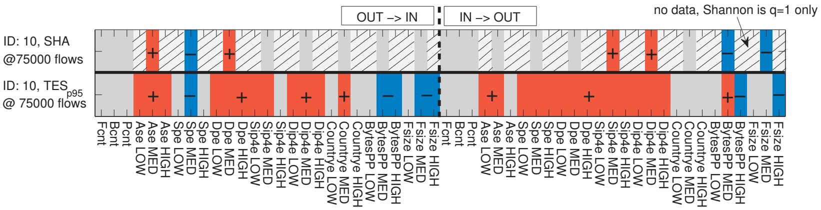
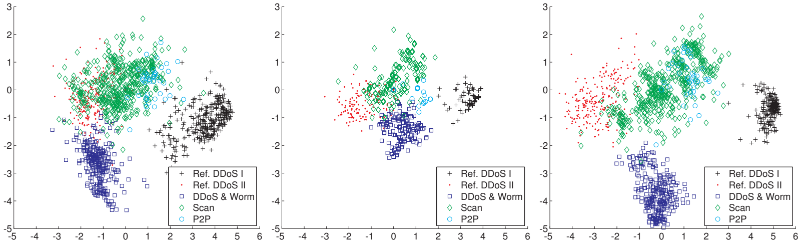


Figure 6.6: *Comparison of anomaly patterns for SHN_+ and TES_{95}. Evidently, SHN_+ misses crucial information captured by TES_{95}.*

To give a graphical intuition of cluster centers and boundaries for different anomaly types, we show Fisher's LDA (Linear discriminant analysis) in Figure 6.7. LDA is typically used in machine learning to find a linear combination of features which characterize or separate two or more classes of objects. The resulting combination may be used as a linear classifier or, more commonly, for dimensionality reduction before later classification. The plots show that for intensity 50K, Shannon yields no clear clusters, whereas TES_{95} is capable of separating *Ref. DDoS 1* from *DDoS + Worm* and *Scans*. With intensity 200K, the situation improves for both sets of metrics, but clusters are still better distinguished for TES_{95}.



(a) TES_{95} with 50K (b) SHN_{+} with 200K (c) TES_{95} with 200K

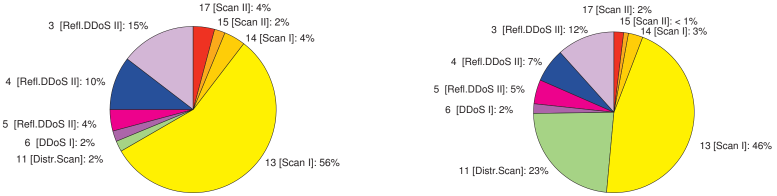
Figure 6.7: *Fisher's LDA plots of* SHN_{+} *versus* TES_{95}.

6.4.3 Prevalence of Anomalies in Real Backbone Traffic

The final stage of our evaluation is to report and discuss the results from applying our entropy telescope to a 34-day flow trace collected by one of the border routers of the SWITCH network in August 2008.

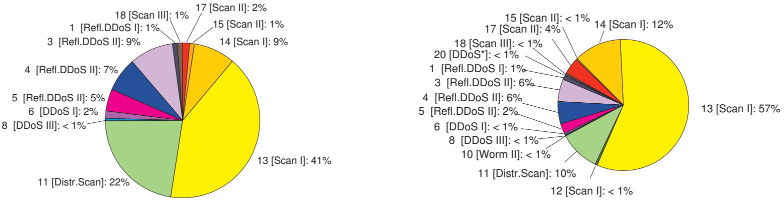
Figure 6.8 shows four pie charts representing the detected anomalies for different detection thresholds. From subfigure (a) to (d), the detection threshold is lowered successively, resulting in alert rates of 0.5% for (a), 1% for (b), 3% for (c), and 10% for (d). An alert rate of 0.5% means that 1 in 200 timeslots of 5 minutes is considered anomalous, i.e. one anomaly is reported every 16.7 hours. A high alert rate of 10% as in subfigure (d) results in one alert every 50 minutes and is certainly not desirable for daily operations. It is only shown to give an idea of the behavior of the classifier for very low thresholds. This is interesting since we expected a larger number of false positives for this setting and were interested to see whether this leads to classifications of anomalies as events that are presumably not present in our trace: worm outbreaks.

For all thresholds, scans are predominant, accounting for roughly 2/3 to 3/4 of all anomalies. This result is consistent with the fact that scanning has become omnipresent in today's networks [55] and is often not even considered to be of special interest anymore. Among scans, type 13 (scan of a subnet from a single host) has by far the biggest share. Type 11 (distributed scanning) increases from 2% to 23% when going from (a) to (b). The relatively high threshold in (a) was most likely not sensitive enough to detect the distributed n-to-m scanning modeled with type 11. Therefore, it is only reported with lower thresholds as in (b) to (d). Regarding worm activity, no alerts were



(a) Alert rate: 0.5% (=1 alert every 16.7 hours)
Total alerts: 49 (out of 9792 time bins)

(b) Alert rate: 1% (=1 alert every 8.3 hours)
Total alerts: 103 (out of 9792 time bins)



(c) Alert rate: 3% (=1 alert every 2.8 hours)
Total alerts: 196 (out of 9792 time bins)

(d) Alert rate: 10% (=1 alert every 50 minutes)
Total alerts: 979 (out of 9792 time bins)

Figure 6.8: *Detection and Classification results for a 34-day flow trace collected by one of the border routers of the SWITCH network in August 2008. Results are for TES_{95} with a PCA [k=36] detector. Each anomaly type is assigned a color, enabling easy comparison of its share across all four pie-charts.*

triggered and the network operator is also not aware of any incidents. There is only one worm alert in subfigure (d), which we consider to be a false-positive.

DDoS-type anomalies have a share between 23% and 31% for (a) to (c). Translated into number of incidents, this means between 15 and 45 DDoS events for the measured period of one month. Note that these events may also contain flash crowd events, as these are generally very hard to distinguish from DDoS attacks. Or in the case of the type Refl.DDoS II, massive coordinated password guessing attacks. It is difficult to compare these figures to external numbers, primarily due to the difficulty of quantifying global DDoS activity. Furthermore, it is not clear how global numbers are broken down to an individual network for comparison. Moore *et al.* estimate 2,000-3,000 global DDoS attacks per week already for 2001-2004 [65]. VeriSign, drawing from different sources, estimates between 1000 and 10,000 DDoS attacks per day in 2008 [192]. The CSI computer crime and security survey 2008 [193] states that of the 522 respondents, 21% were affected by DoS attacks in 2008. Of course, the reported incidents are only those that had enough impact to be recognized by operations.

Considering that our traces contain traffic from around 40 individual organizations, we think our numbers are realistic. That is, for a medium alert rate, we expect around 1 DDoS alert per day. However, note that the entropy telescope cannot classify anoamlies it does not know. The classifier labeles such anomalies with the anomaly which "‘matches best"’. Therefore, if we an anomaly type occuring often in the observed time period

6.5 Conclusion

In this chapter, we improved network anomaly classification by introducing the pruned TES (Traffic Entropy Spectrum) feature set, which uses the non-extensive Tsallis entropy to focus on specific regions of feature distributions. We built an integrated anomaly detection and classification system called the *entropy telescope* and compared the performance of different well-known detectors, such as the Kalman filter, PCA, and KLE. We extensively evaluated the entropy telescope with a rich set of artificial anomalies and real backbone traffic. We showed that using the pruned TES instead of traditional Shannon only approaches improves detection accuracy by up to 20% and classification accuracy by 22.3% on average[5]. In particular, the pruned TES is much more sensitive for small anomalies and established anomaly patterns are very robust with respect to varying anomaly intensities. A run of the entropy telescope on one month of backbone traffic shows that the most prevalent anomalies are different types of scanning (69%-84%) and reflector DDoS attacks (15%-29%).

---

[5] Minimum was 19.6%, maximum was 27%

Chapter 7

Conclusions

In this chapter, we conclude our work on the detection, classification and visualization of anomalies using generalized entropy metrics. We first offer a review of the main contributions made in this thesis. Next, we mention possible shortcomings and weaknesses of our work. Finally, we identify and discuss open research issues in the field of anomaly detection and classification that deserve further attention.

7.1 Review of Contributions

In this thesis we presented three core contributions discussed below.

A study of the robustness of entropy features with regard to packet sampling

The first part of our work was devoted to the problem of how packet sampling impacts on the visibility of anomalies with respect to both count and entropy metrics. Starting from NetFlow data generated based on unsampled packet streams, we simulated the impact of packet sampling at various sampling rates. To get flow traces from sampled packet traces, we first reconstruct packet traces from our flow traces and then sample them prior to feeding them to a (virtual) flow generator. We then argued that possible bias in this method is practically non-existent since our count and entropy metrics are aggregate metrics with a granularity in the range of minutes. Using this methodology, we then generated various sampled views from the Blaster and Witty worm anomaly. We compared measurements obtained from the trace where the respective anomaly had been removed with those from the trace including them. This revealed that entropy metrics are more robust than count metrics. Moreover, we found that under certain circumstances, sampling can even increase the visibility of an anomaly. We discussed situations where this could happen. One case where packet sampling increased the visibility of the anomaly up to a sampling rate of one out of 10,000 was when the baseline traffic contributes many elements (e.g. an IP address) with a support of just one or two packets, but where the anomaly mostly contributes elements with a much larger support. However, since this effect requires specific baseline and anomaly traffic characteristics, in practice, its relevance is probably small.

A method for capturing and visualizing anomalous changes in traffic feature distributions

In the second part of our work, we introduced the Traffic Entropy Spectrum as means of analyzing and visualizing changes in traffic feature distributions. We analyzed its properties using both artificial and real traffic feature distributions. Moreover, we found and discussed a shortcoming of the TES which hinders the interpretation of the TES in certain situations. We called this problem the inter-region dependency since a change in one region of the TES, i.e. the high-activity region, can have a strong influence on what we see in other regions of the TES, i.e. the low-activity region. We then presented a modification to the TES called the TES_p that allowed us to mitigate the inter-region dependency problem. Next, we introduced the concept of spectrum patterns, which enabled us to capture the impact of an anomaly on the various TES under scrutiny in a compact form. An anomaly classifier could use these patterns to identify different types of anomalies. Our evaluation of both the descriptive power of the spectrum patterns and the capability of the TES to expose anomalies in the traffic traces provided evidence that both of these tools can do what we expected them to do. However, since our evaluation was performed with just a few well-known anomalies, we concluded that we need to perform a more in-depth evaluation.

Design and evaluation of a comprehensive anomaly detection and classification framework based on the TES

In the third part of our work we focused on a thorough evaluation of our work. Since visual inspection is not a suitable anomaly detection approach for more in-depth evaluation, we designed a comprehensive anomaly detection and classification framework which integrates the concepts of the TES and spectrum patterns. We then used this framework to perform an extensive evaluation of the TES. We made use of three different detection methods, one classification method and a rich set of anomaly models injected into real backbone traffic. Our evaluation demonstrated the superiority of the refined TES (TES_p) approach over TES and the classical Shannon-only approaches with respect to both, anomaly detection and classification.

7.2 Critical Assessment

The goal of this thesis was to find answers to two broad questions. Are entropies a useful tool in the context of anomaly detection? And, can generalized entropy metrics help to improve on results obtained when using non-parameterized forms of entropy only? In the following discussion, we assess the extent to which we answered these questions.

We addressed the first question in the first part. We showed that entropy metrics are more robust to sampling than traditional flow, byte, or packet count metrics. However, our evaluation was based on data from only one network and two specific anomalies. In an attempt to compensate, we scaled one of the anomalies and considered flow data collected at different measurement points in our network. Nevertheless, a more extensive evaluation would help to show whether this finding is applicable in other settings. The only reason we did not follow up on these results was because another team of researchers published similar results for their setting at the same time as we published our work.

In the second part, we addressed whether or not generalized entropy metrics are a useful tool to use for anomaly detection and classification. While we did show that the concept of the TES does indeed have the potential to achieve this, our evaluation has several limitations. First, it is largely based on synthetic anomalies injected into real background traffic. This is now considered mandatory for any sound evaluation of anomaly detection and classification systems. However, it is unclear whether or not the different variations of these anomalies could truly be observed in the wild, even though we extracted most of the basic parameters for these anomalies from real traffic traces. And while we did look into the performance of the TES with respect to well-known anomalies in our trace, we considered only a small set of anomalies. Moreover, although our drill-down work checks whether what the TES detects is truly an anomaly, we cannot provide any information about what the TES does not expose. For this, multiple labeled traces with a large and diverse set of real world anomalies would be required. However, these limitations are probably impossible to overcome. Furthermore, our set of anomalies is far from complete. There are so many different shapes of, for example, something as simple as a DDoS anomaly, that it is hard to model them all. Nevertheless, our set of anomalies was relatively extensive when compared to rival evaluations.

7.3 Future Work

Flow-based anomaly detection is a research field which has attracted quite a lot of attention recently. The reasons for this are manifold. The number of networked devices and network bandwidths are still growing, with seemingly no end in sight. To cope with this growth, we cannot rely on tools that require a finely-grained view on the data. Such tools are likely to be far too expensive for widespread use. Flow data, however, provides an abstraction which proves good enough for things such as accounting or capacity planning. Increased attention also arises from the fact that anomaly detection based on flow data is a challenging research field. Many research questions remain unanswered. Two issues that should receive attention from the research community are discussed in the following.

The first issue is the non-availability of recent, standardized data sets for the evaluation of anomaly detection and classification systems. Without such data sets for networks of various sizes and purposes (e.g. backbone networks and different residential, company or military networks), a comparison of the many different anomaly detection systems is very hard. A comparison with respect to the traffic in one network only does not guarantee that results would be the same for another network. However, this truly is a challenging topic. Creating traces using, for example, a combination of deep packet inspection and manual labeling is not enough. Real world traces often contain many anomalies of the same type and similar sizes[1] which prejudices evaluation. To overcome this problem, researchers should develop a diverse set of anomaly models accessible to the research community which can then be used to inject such anomalies into real or artificial background traffic. Without such stan-

---

[1] E.g. in terms of the number of flows, packets or bytes involved.

dardized models, it is difficult to compare, for example, the detection quality related to DDoS attacks. Whilst some might use a model corresponding to the *juno* attack tool,nothers might model the characteristics of the *Ion Cannon* tool.

The second issue we want to discuss is the problem of the non-existence of anomaly-free traffic traces. Apart from in networks not connected to the Internet or traces collected in testbeds, anomaly-free traffic traces do not exist. Problems arise with the very definition of the term *anomaly*. Is backscatter traffic an anomaly? Is it an anomaly if we see more of it than usual? And how much is “more”? The same holds for things like regular password guessing attacks on remotely accessible machines. Do we consider them “normal” or “abnormal”? In order to get sound and comparable results, these questions should be addressed in a standardized way, possibly through policies which define what we consider anomalous, and what not. Until now, this question has largely been left unanswered.

7.4 Publications

The work presented in this thesis is based on the following publications:

P01 **Accurate network anomaly classification with generalized entropy metrics**
Bernhard Tellenbach, Martin Burkhart, Dominik Schatzmann, David Gugelmann and Didier Sornette
Computer Networks: The International Journal of Computer and Telecommunications Networking, 2011

P02 **Beyond Shannon: Characterizing Internet Traffic with Generalized Entropy Metrics**
Bernhard Tellenbach, Martin Burkhart, Didier Sornette and Thomas Maillart
Passive and Active Measurement Conference (PAM), 2009

P03 **Impact of Traffic Mix and Packet Sampling on Anomaly Visibility**
Bernhard Tellenbach, Daniela Brauckhoff and Martin May
IEEE International Conference on Internet Monitoring and Protection (ICIMP), 2008

P04 **Impact of Packet Sampling on Anomaly Detection Metrics**
Daniela Brauckhoff, Bernhard Tellenbach, Arno Wagner, Anukool Lakhina, Martin May
ACM Internet Measurement Conference (IMC), 2006

In addition, these publications were co-authored during this thesis:

P05 **Peeling Away Timing Error in NetFlow Data**
Brian Trammell, Bernhard Tellenbach, Dominik Schatzmann and Martin Burkhart
Passive and Active Measurement conference (PAM), 2011

P06 **Rating Autonomous Systems**
Laurent Zimmerli, Bernhard Tellenbach, Arno Wagner and Bernhard Plattner
IEEE International Conference on Internet Monitoring and Protection (ICIMP), 2009

P07 **0-Day Patch - Exposing Vendors (In)security Performance**
Stefan Frei, Bernhard Tellenbach
BlackHat Europe, 2008

P08 **The NoAH Project - Poster**
Spyros Antonatos, Daniela Brauckhoff, Bernhard Tellenbach, Asia Slowinska
TERENA Networking Conference (TNC), 2007

Appendix A

Appendix

In this chapter, we briefly describe our contributions to the NetFlow collection and processing infrastructure and the various software tools developed in the context of this thesis.

A.1 NetFlow Collection and Processing Infrastructure

When we started our thesis, a fully operational NetFlow collection and processing infrastructure, built in 2002/2003, was already in place. A detailed description can be found in [194]. However, in the process of the thesis, we had to update both its hardware and software several times in order to keep up with the functional, performance and space requirements involved in managing and processing the NetFlow data. We first provide a brief overview of the NetFlow collection and processing infrastructure put in place in June 2012. We subsequently discuss the hardware costs and manpower requirements, the past and current NetFlow data volume to be handled and some key figures with regard to processing power and memory. Finally, we conclude with a summary of our main contributions to the construction and maintenance of this infrastructure.

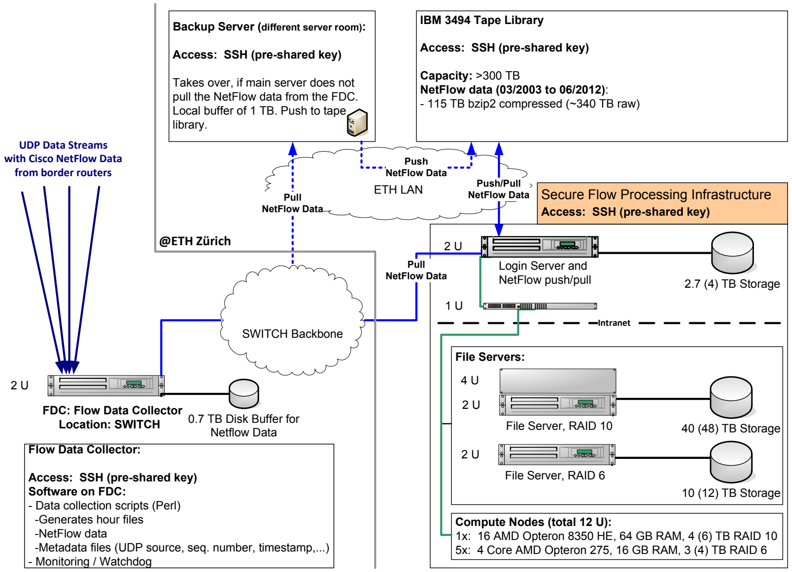

Backup Server (different server room):
Access: SSH (pre-shared key)
Takes over, if main server does not pull the NetFlow data from the FDC. Local buffer of 1 TB. Push to tape library.
IBM 3494 Tape Library
Access: SSH (pre-shared key)
Capacity: >300 TB
NetFlow data (03/2003 to 06/2012):
- 115 TB bzip2 compressed (~340 TB raw)
UDP Data Streams with Cisco NetFlow Data from border routers
Push NetFlow Data
ETH LAN
Pull NetFlow Data
Push/Pull NetFlow Data
Secure Flow Processing Infrastructure
Access: SSH (pre-shared key)
@ETH Zürich
2 U
Login Server and NetFlow push/pull
2.7 (4) TB Storage
Pull NetFlow Data
1 U
Intranet
SWITCH Backbone
File Servers:
4 U
2 U
File Server, RAID 10
40 (48) TB Storage
2 U
File Server, RAID 6
10 (12) TB Storage
2 U
FDC: Flow Data Collector
Location: SWITCH
0.7 TB Disk Buffer for Netflow Data
Flow Data Collector:
Access: SSH (pre-shared key)
Software on FDC:
- Data collection scripts (Perl)
-Generates hour files
-NetFlow data
-Metadata files (UDP source, seq. number, timestamp,...)
- Monitoring / Watchdog
Compute Nodes (total 12 U):
1x: 16 AMD Opteron 8350 HE, 64 GB RAM, 4 (6) TB RAID 10
5x: 4 Core AMD Opteron 275, 16 GB RAM, 3 (4) TB RAID 6

Figure A.1: *The NetFlow collection and processing infrastructure run by the Communication Systems Group (CSG) at ETH Zurich as of June 2012.*

A.1.1 Overview

Figure A.1 shows the setup of the NetFlow collection and processing infrastructure run by the Communication Systems Group (CSG) at ETH Zurich in June 2012. It consists of the following hardware- and software components:

- **Flow Data Collector (FDC):** This component is responsible for collecting and buffering incoming NetFlow data streams. It typically needs to be placed in a data center of the data provider. The FDC runs software to capture the data and store it in files. For this task, we use a script written in Perl that awaits UDP packets on a specified port and writes the packet content to a file named:
  `<PORT>_<FILE_COUNTER>_<TIMESTAMP>.dat`.
  The script also generates the metadata files named:
  `<PORT>_<FILE_COUNTER>_<TIMESTAMP>.stat`
  containing the source IP address of each data packet received and information such as the timestamp and sequence number, among other

things. Since the script creates new files every hour and we receive data on two different ports, we get a set of four files per hour. The FDC buffers the generated files (e.g. hour files), compresses the data files and makes them available for download. If the FDC experiences performance problems for any reason, compression is performed only after downloading the data from the FDC. However, this approximately triples the network load.

- **Secure Flow Processing Infrastructure:** This component is used for three things:
  - Downloading the flow data from the FDC and pushing it to the tape library.
  - Collecting and maintaining repositories of data needed to enrich flows with meta information such as the country and AS of the source and destination of the flow.
  - Performing analysis on the incoming or archived flow data to keep, for example, a database with information on the files on the tape library which covers things like size and whether they are missing, corrupt or up to date.
  - Custom flow data processing.

  For research or non out-of-the-box analysis, a flexible software framework is required to parse and work with the data. A near real-time analysis (processing one hour of flow data in less than an hour) of flow data can only be achieved through massive parallelization using multiple processing nodes.
- **Backup Server:** If the Secure Flow Processing Infrastructure suffers downtime (e.g. cooling system or power system failure), it is safer to have a backup machine in a different location to make sure the buffer on the FDC does not fill up. This is especially true if, as in our case, there is no 24/7 admin for this infrastructure.
- **Disk Array / Tape Library:** If the NetFlow data should be available for more than a few days or months, a large disk array or tape library is required to store the huge amount of data. We implemented a combination of Network Attached Storage devices to store data currently used for research and a tape library for long term storage of the NetFlow data.

A major issue with NetFlow data is that it contains privacy sensitive data. As such, securing the infrastructure that handles this data is crucial to prevent privacy violations. To achieve this, the Flow Processing Infrastructure is used exclusively for flow data processing and provides a tight access control at both the technical and the management level. Furthermore, the infrastructure further limits the attack surface by providing a single access method and by allowing access from specific locations only. More precisely, the infrastructure can only be accessed via a Secure Shell (SSH) connection using pre-shared key (PPK) authentication and access is restricted to hosts in the ETH (or partner) networks.

A.1.2 Costs

The estimated costs involved in building and maintaining our NetFlow collection and processing infrastructure are shown in Table A.1.2.

Hardware	**approx. costs in Swiss Francs**
Secure Flow Processing Infrastructure: • 1x 16 AMD Opteron 8350 HE, 64 GBytes RAM, 6 TBytes RAID-10 • 5x 4 AMD Opteron 275, 16 GBytes RAM, 4 TBytes RAID-6 • 2x extensible storage servers: currently 50 TBytes of net disk space	105,000
Flow Data Collector	3000
Backup Server	2500
(Tape Library)	n/a
Manpower	**person months**
Administration and support: • User management • Management of local repositories of 3rd party data (GeoIP, Routeviews,...) • Hardware/Software acquisition and updates • Incident handling and support	1-4 (per year)
NetFlow data processing: Initial overhead	3-6
NetFlow Tools development	36

Table A.1: *The estimated cost of building and maintaining our NetFlow collection and processing infrastructure.*

According to this table, two of the most significant cost factors are the initial overheads needed to build up the knowledge and to develop the software base to process and work with the NetFlow data. While there are many free and non-free tools for standard tasks such as monitoring or anomaly detection, trying out new ideas and performing new kinds of measurements typically require the development of custom software. Section A.2 briefly discusses some of the software and tools developed for this thesis.

A.1.3 Data Volume

Until the end of 2012, our archive of bzip2 compressed Cisco NetFlow will most likely break the 120 TBytes barrier. Table A.1.3 shows the amount of data for each of the years from 2003 to 2012.

2003[1]	**2004**	**2005**	**2006**	**2007**	**2008**	**2009**	**2010**	**2011**	**2012**[2]
3.0	6.0	5.1	6.1	9.8	12.6	16.9	20.1	20.3	20.3

[1] Starting from March 2003.

[2] Based on a linear projection based on data collected until the end of August 2012.

Table A.2: *Amount of bzip2 compressed Cisco NetFlow data for each of the years from 2003 to 2012.*

One major challenge with this huge amount of data is that with the current storage servers (with a net capacity of roughly 40 TiB), we can hardly store the last two years of NetFlow data. At a first glance, this might seem convenient, but if we consider that (1) different researchers might need to look at different parts of the data set for their research and that (2) they do not just need the raw data but also require a large amount of analysis results or preprocessed data optimized for doing things like forensics,[1] this soon becomes a challenge problem. As a consequence, most things have to be done on demand: only results and data from often used events (e.g. from special events such as major attacks or network failures) are kept on the storage servers.

A.1.4 Processing Power and Memory Requirements

Decompression of an hour of NetFlow data from 2012 typically takes between 20 and 40 minutes (night/day) using a single core of our AMD Opteron 275 systems. bzip2 decompresses our data at approximately 2 MB/s. If the data

---

[1] The problem with data formats optimized for fast search for specific characteristics is that most of these formats require two to ten times the storage space of the raw NetFlow data

is enriched with additional information such as the country of origin of the flow or the origin AS, it takes up to 30% longer. As a consequence, if the data comes from our tape library with approximately 20 MB/s expected transfer rate, we can decompress and parse 10 hours of data in approximately 5 hours if at least 10 CPUs are available. More CPUs are required if more than a simple parsing of the data is required. Memory is typically the limiting factor regarding analysis that requires tracking, for example, per host information, or more complex behavioral patterns over longer time periods. With up to *200 million flows per hour* and up to *several millions of hosts (IP addresses) per hour*, such an analysis is difficult to perform. Since memory requirements depend heavily on the task to be performed, it must be analyzed on a task-by-task basis.

A.1.5 Contributions

- Design, and setup of a new secure flow processing infrastructure based on off-the-shelf rack components: compute nodes and Network Attached Storage (NAS) devices).
- Administration and management of infrastructure.
- Design and implementation of tools to build and automatically update data repositories required to do time dependant IP address to country and IP address to AS lookups.
- Design and implementation of tools to automaticaly index the NetFlow files stored on the tape library and to check whether newly downloaded *.stat* files are corrupt.
- Modifications to the flow data collector (hardware and software) to meet the requirements for data compression and extended flow data buffering in case of any problem with the secure flow processing infrastructure.

A.2 NetFlow Tools

In this section, we briefly present the NetFlow tools developed in the context of this thesis and discuss our contributions to them.

A.2.1 NetflowVX Library

The NetflowVX library is a library providing NetFlow v5 and v9 parsing capabilities as well as some specialized data structures like a custom hash table and linked list. Note that the spezialized data structures are only there because the NetflowVX library is basically a NetFlow v9 extended, a refactored and well-documented version of a NetFlow processing library written by Arno Wagner [194]. The library is tailored to parse the NetFlow data as it is stored by our capturing infrastructure: raw NetFlow Protocol Data Unit (PDU) stored in *.dat* files and the corresponding metadata (such as the IP of the device that sent the PDU) in *.stat* files. Furthermore, it can be used to enrich the flow data with the origin and destination AS number. To do so, the library looks at the source and destination IP address of a flow and resolves the corresponding AS system number using the AS information repository maintained on the compute cluster (see A.1.1). The NetflowVX library was first developed by David Benninger and Loris Siegenthaler under the direction of the author of this thesis. We later added the parsing of the *.stat* files and the enrichment with AS information since they were not included in the initial version. The library exists in two versions: a version written in C and a version written in C++. However, the C version was left in its initial state since no new tools were written in C once the C++ version became available.

Performance

Performance is an important criterion for a NetFlow parsing library intended for parsing huge amounts of flow data. A parsing library should at least be able to parse data in near realtime. Hence, it should process an hour of data in less than an hour. Table A.2.1 shows the results from performance measurements in terms of flows per second for three different configurations and with either NetFlow v5 or v9 data in both, compressed and uncompressed form as input. The three different configurations of the library are:

- **(no options)**[2]: PDU parsing from *.dat* files.
- **.stat**: PDU parsing from *.dat* files and PDU meta-info parsing from *.stat* files.

---

[2] Note that this may not produce the expected results if the *.dat* files contain PDUs from different NetFlow export devices. The reason for this is that the template identifiers of the templates used to parse the v9 data can only be unique per NetFlow export device. Without knowing from which device a PDU originates, the library assumes that all PDUs come from the same device.

- **AS+.stat**: PDU parsing from *.dat* files, PDU meta-info parsing from *.stat* files and enrichment of flow data with origin and destination AS numbers.

The measurements are carried out using a simple program that makes use of the C++ version of the NetFlowVX library to iterate over all flows contained in a single NetFlow data file. All measurements are performed using a single core of an AMD Opteron 275 on a system with 16 GBytes of RAM and 4 TBytes of Redundant Array of Independent Disks, originally Redundant Array of Inexpensive Disks (RAID) six disk storage. Note that each result reflects the average performance from a series of five measurements with five different file sets. The relative standard deviation of the measurement series is lower than 5% for all of the results.

	performance in flows per second		
	(no options)	.stat	AS + .stat
NetFlow v5, bzip2	1.94E+05	1.71E+05	7.80E+04
NetFlow v5, uncompressed	2.03E+06	1.51E+06	5.98E+05
NetFlow v9, bzip2	-	1.78E+05	7.91E+04
NetFlow v9, uncompressed	-	9.96E+05	1.57E+05

Table A.3: *Performance of the NetflowVX library in terms of flows per second for different configurations and NetFlow versions for both compressed and uncompressed flow data.*

As the results in Table A.2.1 show, the worst performance of roughly 280 million flows per hour[3], is obtained for the *AS + .stat* configuration for compressed NetFlow v5 data. Hence, our library easily meets the near real-time processing requirement: it can handle the up to 200 million flows per hour that we see in our flow data in less than an hour. In order to process higher flow rates in the not-too-distant future, the load can be distributed to multiple cores by, for example, distributing the incoming flow data to multiple reader processes. Or we could try to optimize the code that does the enrichment of the flow data with AS numbers; without AS numbers, the library can process up to 615 flows per hour. What is somewhat unexpected is that the performance for compressed NetFlow v5 data is slightly worse than those for compressed NetFlow v9 data. After all, NetFlow v5 data is much simpler to parse than NetFlow v9 data. Only when comparing the performances for uncompressed input data do we get the expected result. This suggests that

---

[3] 78,000 flows/sec * 3600 sec = 280 million flows

our sample of files containing NetFlow v9 data can be decompressed faster than those containing NetFlow v5 data. Whether or not this might be true in general was not investigated further.

A.2.2 NetFlow Processing Framework

The NetFlow Processing Framework is a C++ framework forming an additional abstraction layer to the NetflowVX library. The framework provides the basis for writing modules that perform different processing tasks such as NetFlow filtering according to user-defined rules which can then be re-used by others once they have been implemented. These modules can then be put together to form a chain of modules where the first module is expected to generate some sort of data. This data is then handed over to the next module's by calling the next modules *process* method. A module checks whether it understands the data that is passed to it by checking the type information encoded in the envelope used to pass data from module to module. Even though the chain can hand over arbitrary data, it is typically a list of flows which is passed from module to module. Figure A.2 shows a sample of such a chain on the module layer.

However, the type of flows in the list might depend on the actual processing job: there is an internal flow format (*FlowCompact*) which is optimized for space but contains only selected fields of the flow records and a format which contains all information (*NetFlow internal*) in Figure A.2, including the information in the PDU header. Modules must be aware of the internal flow format and are usually designed to operate with one specific format only. An input/output (IO) abstraction layer allows the extension of the framework to handle new input flow formats such as IPFIX without changing anything, except to write a new specialized Flow IO class. Currently, there are two specialized Flow IO classes: one for the *FlowCompact* format and one for the *NetFlow internal* format. Both can read NetFlow data from the disk using the NetflowVX library, but can also read and write files with flows containing serialized versions of the flows in the internal flow format. This architecture is displayed on the IO layer in Figure A.2.

After passing the data to the next module, the module has to wait until the *process* method returns, since we do not use threads[4] in our modules. The reason for not using threads is that they are not strictly needed, and we worked

---

[4] With one exception: the parallel reader module can read two flow data streams in parallel while it waits for the next module in the chain to return from its *process* method.

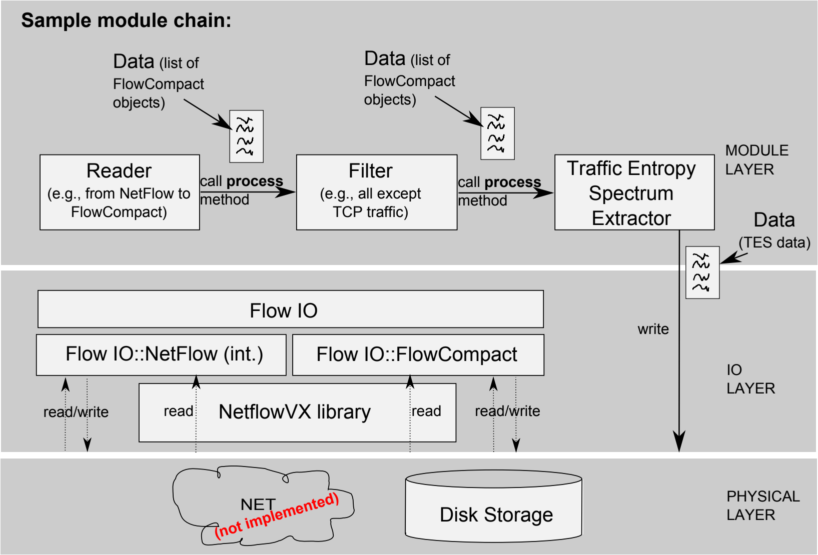


Figure A.2: *Outline of the design of our NetFlow processing framework.*

to the principle that software should be kept as simple as possible. Most of our use cases are offline NetFlow data processing where we can parallelize the processing of the data of the time interval (e.g. two weeks) to be processed by dividing it into multiple pieces (e.g. one day per processor core) which we then process on different cluster nodes.

While we designed and implemented most of the NetFlow Processing Framework, fruitful discussions and comments from Dominik Schatzmann as well as his Interface lookup module helped to turn it into a tool that was used by both students and staff members for their work with NetFlow data. The framework now contains the following modules:

- **Reader modules**: One single- and one multi-threaded module for reading and forwarding flow data in an arbitrary internal flow format using a Flow IO class.
- **Writer module**: A module to serialize flow data to disk using the internal flow data format.
- **Basic metrics module**: A module to extract count, volume, and entropy time series and traffic feature distributions on a per protocol (TCP,

UDP, ICMP, other) and direction basis for different time bin sizes.

- **Interface lookup module**: A module that is intended to filter flows that are reported by multiple flow exporters. The module does this by identifying those interfaces on the flow exporters that are either up- or downlink interfaces to or from the network under supervision and filters flows from other interfaces. Unfortunately, this is not enough since a flow entering the network under supervision and leaving it again (transit traffic) would still be reported twice. Therefore, the module works only if we look at traffic originating in or destined to the network under supervision. Hence, subsequent modules should filter any transit traffic (and internal traffic) that might be present.
- **Filter module**: A module that performs flow filtering based on rules specified in a configuration file. It allows for basic AND and OR filtering rules on values or value ranges of one or multiple flow features.
- **Timing analysis module**: A module to output information on NetFlow time stamps. This module is related to [195] where we describe a problem with the NetFlow v9 data format resulting in inaccurate NetFlow time stamps.
- **FlowToText module**: A module that outputs the flows in human readable form to the standard output.

Note that this list is incomplete. The modules contributed by Dominik Schatzmann and several students are not listed here.

Performance

Since the NetFlow Processing Framework is a framework but not an application by itself, we do not provide performance measurements for it. The performance of an application built using this framework heavily depends on the number and type of modules (and filter settings) used as well as on the input flow format and the internal flow format.

One example of such an application is the application to extract the information to build the TES. It is built based on a chain of the following modules: the parallel reader module, the interface lookup module and the basic metrics module. Experience from many hours of data processing showed that this application can process up to 200 million flows in less than an hour on an AMD Opteron 275 system with 16 GBytes RAM and 4 TBytes of RAID six disk storage to write the extracted information to. Hence, it is capable of processing our flow data in near real-time.

A.2.3 Data Analysis and Processing Tools for MATLAB

The data analysis and processing tools for MATLAB are tools used to post-process the count, volume, and entropy time series as well as the traffic feature distributions output by the basic metrics module (see A.2.2). The toolset is implemented in the MATLAB language in an object oriented way.[5] Our MATLAB tools consist of three GUI-based applications and a large set of helper classes, MATLAB functions and scripts to automate data post-processing tasks.

Before we discuss our three interactive Graphical User Interface (GUI) applications designed with the help of the GUI Design Environment (GUIDE), we first provide an overview of the building blocks of our toolset. Note that while most building blocks might be useful to other researchers working with time series data, the first building block is too specific to our use case:

- **Basic metrics data processing**: This building block consists of multiple post-processing tools for the data produced by the basic metrics module (see A.2.2). Among them are three GUI-based tools to manually inspect and analyze this data and a set of classes dedicated to the management and handling of the entropy telescope evaluation process (data management, detection runs with parameter sweeps etc.). The three GUI-based tools are discussed in more detail later in this section.
- **Data access**: A set of classes and MATLAB functions that provide transparent and optionally cached access to table and column-oriented time series data sources that meet the following requirements: (1) Time series data is organized in rows. Each row contains the measurement data for a specific point in time. (2) All tables contain a column named *time* consisting of time stamps in Unix seconds. An abstract data access class implements data source independent parts of the data access task and defines methods to be implemented by its specializations. The toolset comes with two implementations of the abstract data access class: one for data stored in SQLite databases and one for data stored in CSV files.[6] In the case of CSV files, the “tables” are the different CSV files in a directory and the data source is the directory. Given a data source identifier string and data access credentials, the data access

---

[5] Note that until release R2008a, only very few MATLAB code released on the web was object oriented. Many MATLAB users did not even know that it is possible to write object oriented MATLAB code. With release R2008a, MATLAB added many new features for defining classes of objects much like with other object oriented languages.

[6] Actually, the separator can be provided as parameter. It does not have to be a comma.

class establishes a connection to the data source and allows column-name and time interval based access to this data. This is either done using the data source selection GUI that comes with this component or by manually constructing the data access object with the required parameters. Adding other data sources (e.g. MySQL databases) is simple: One just has to provide an appropriate implementation of the abstract data access class and register it with the data source selection GUI. If an application needs to access the same data multiple times and if the time series accessed with this data access object have the same time stamp vector, they should use the provided proxy class.[7]

- **Data profiling**: This building block provides classes to construct mean, percentile and/or standard deviation based week profiles of time series data. To do so, they take time series as input and build stacked distributions for all weekdays and time bins by assigning, for example, data points from Mondays between 11:15 and 11:20 to a distribution "Monday, 11:15 to 11:20", etc. These profiles can be used to determine the expected (or "normal") value ranges for each day and time bin from a statistical point of view.
- **Date/Time tools**: MATLAB functions to convert Unix seconds to MATLAB time and vice versa.
- **Anomaly detectors**: This building block provides an abstract anomaly detector class for anomaly detection on time series and three implementations of this class. The first implements an algorithm based on the Karhunen-Loeve Expansion (KLE)[8] method, the second implements a Kalman filter based method and the third a Haar wavelet based method. The abstract anomaly detector class provides a detector-independent interface so that applications using it can switch detectors without changing the code.
- **Support Vector Machine**: This building block provides an easy to use MATLAB interface to LibSVM [185], a library implementing a support vector machine. It implements and automates tasks such as n-fold cross-validation and the plotting of classification matrices as for example, Figure 6.5 in chapter 6.
- **Utilities**: Some MATLAB functions to help with plotting, figure export and hierarchical sorting of vectors of structs.[9] Plotting functions

---

[7] FixedIntervalTimeSeriesCache

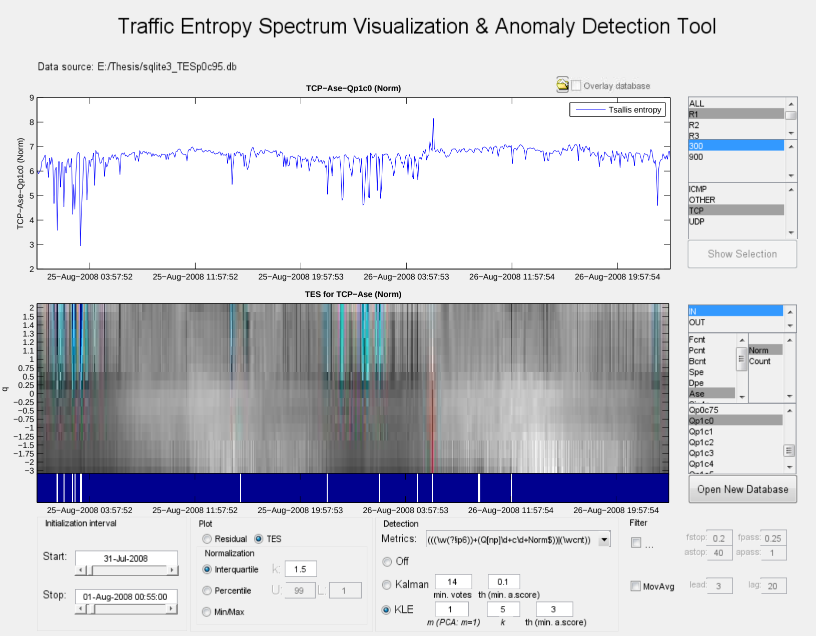
[8] And therewith also the Principal Component Analysis (PCA)

[9] E.g. start with sorting the vector according to field X, then sort entries with identical X values according to field Y etc.

include e.g. a function to plot labeled pie-charts of the kind shown in Figure 6.8 in chapter 6 or ROC plots of the same kind as those displayed in Figure 6.4.

Traffic Entropy Spectrum Visualization & Anomaly Detection Tool

The TES visualization and anomaly detection tool can be used to browse and process the raw count and entropy time series produced by the Basic Metrics Module of the NetFlow Processing Framework. Using the data access classes, it reads this data from either CSV files or from a SQLite database. Figure A.3 shows a screenshot of the graphical user interface of this tool.



Data source: E:/Thesis/sqlite3_TESp0c95.db

Initialization interval

Start: 31-Jul-2008

Stop: 01-Aug-2008 00:55:00

Plot

○ Residual ● TES

Normalization

● Interquartile K: 1.5

○ Percentile U: 99 L: 1

○ Min/Max

Detection

Metrics: (((\w(?!ip6))+(Q[np]\d+c\d+Norm$))|(\wcnt))

○ Off

○ Kalman 14 min. votes 0.1 th (min. a.score)

● KLE 1 *m (PCA: m=1)* 5 *k* 3 th (min. a.score)

Filter

☐ … fstop: 0.2 fpass: 0.25 astop: 40 apass: 1

☐ MovAvg lead: 3 lag: 20

Figure A.3: *Screenshot of the traffic entropy spectrum visualization and anomaly detection tool.*

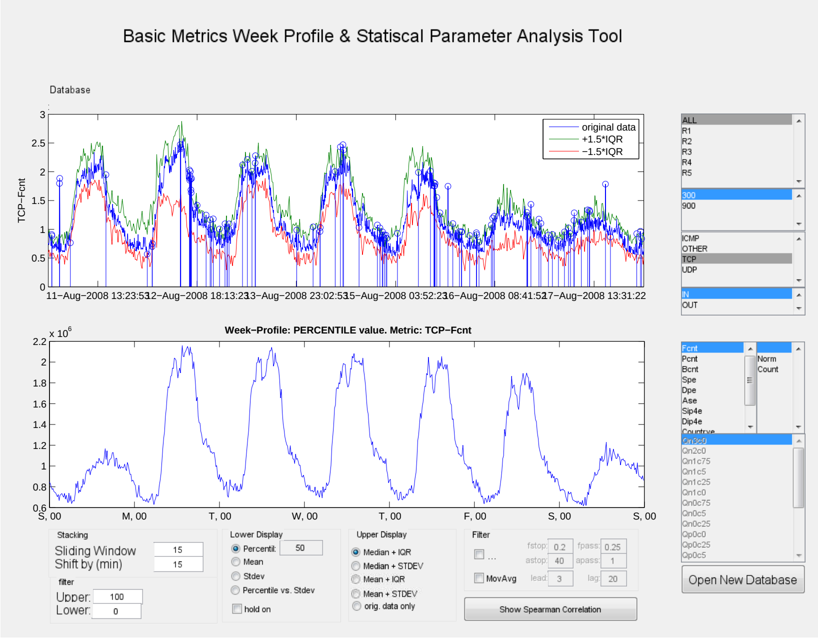
With the selectors on the right, one can select the time series to be displayed in the upper plot window. Choosing a time series consists of selecting:

1. An export device, or any aggregates (e.g. ALL for all export devices), for which the Basic Metrics module outputs measurements.
2. The direction of the traffic. In our setup: IN or OUT.
3. The protocol or any aggregates output by the Basic Metrics module. For example, the OTHER “protocol” includes all traffic except TCP, UDP and ICMP traffic.
4. The count or entropy metric.
5. For entropy metrics, the q value and the type of the metric (TES, TES_p).

The lower plot window displays the locations of anomalies and, for entropy metrics, also the TES. Which anomaly detector is used with which metrics and training data is configured using the knobs and dials at the bottom. Finally, various coloring schemes make focusing on different aspects of the data easier. For example, a black-white scheme with the global maximum marking the top end and the global minimum marking the bottom end of the color scale exposes, at a single glance, whether or not the data contains extreme anomalies. Figure 5.8(a) shows a plot using this coloring scheme.

Week Profile & Statistical Parameter Analysis Tool

From a data input and data selection point of view, the week profile and statistical parameter analysis tool is basically identical to the TES visualization and analysis tool. However, that is the only thing they have in common. As can be seen in Figure A.4, the upper plot window shows the selected time series, for example the number of flows per time bin. Furthermore, it displays the upper and lower thresholds calculated from data points in a sliding window of size n time bins and marks data points above or below these thresholds. If the data spans multiple weeks, the threshold is calculated using the data in the same sliding window, for example Monday 12:00, from all of those weeks. For threshold calculation, the tool supports different methods such as taking a multiple of the standard deviation or a multiple of the interquartile range. It also comes with support for percentile based pre-filtering and other filters such as a moving average filter. Note that this tool expects the data used to build the profiles to contain little to no anomalies. Otherwise, one might get a profile as shown in Figure A.5. Here, several anomalies in one of the weeks lead to “anomalous thresholds”. If the data contains a few outliers only, pre-filtering with a percentile-based filter might fix this issue.



Basic Metrics Week Profile & Statiscal Parameter Analysis Tool

Stacking
Sliding Window 15
Shift by (min) 15

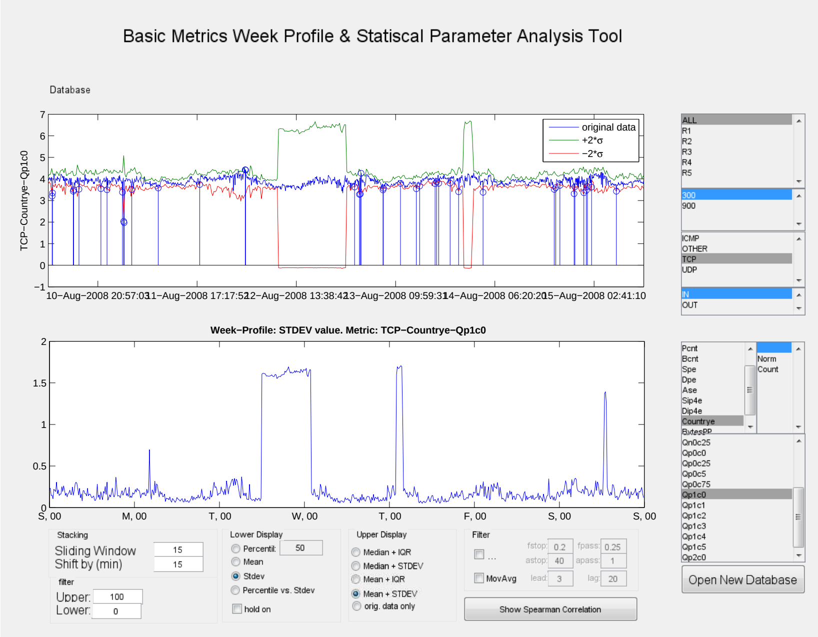
filter
Upper: 100
Lower: 0

Lower Display
● Percentil: 50

Upper Display
● Median + IQR

Filter
fstop: 0.2 fpass: 0.25
astop: 40 apass: 1
lead: 3 lag: 20

Figure A.4: *Screenshot of the week profile and statistical parameter analysis tool.*



Stacking
Sliding Window 15
Shift by (min) 15
filter
Upper: 100
Lower: 0

Lower Display
○ Percentil: 50
○ Mean
● Stdev
○ Percentile vs. Stdev
☐ hold on

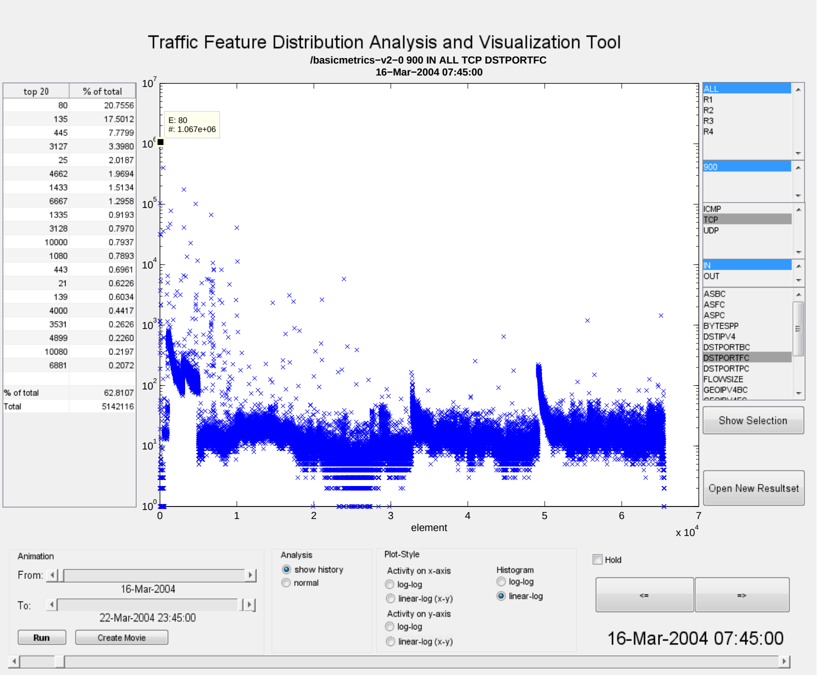
Upper Display
○ Median + IQR
○ Median + STDEV
○ Mean + IQR
● Mean + STDEV
○ orig. data only

Filter
☐ ...
fstop: 0.2 fpass: 0.25
astop: 40 apass: 1
☐ MovAvg lead: 3 lag: 20

Figure A.5: *Screenshot of the week profile and statistical parameter analysis tool.*

Traffic Feature Distribution Analysis and Visualization Tool

The last of the tree tools is the traffic feature distribution analysis and visualization tool. The purpose of this tool is to make the raw distribution information output by the Basic Metrics module browsable. Figure A.6 shows a screenshot of the graphical user interface of this tool.



top 20	% of total
80	20.7556
135	17.5012
445	7.7799
3127	3.3980
25	2.0187
4662	1.9694
1433	1.5134
6667	1.2958
1335	0.9193
3128	0.7970
10000	0.7937
1080	0.7893
443	0.6961
21	0.6226
139	0.6034
4000	0.4417
3531	0.2626
4899	0.2260
10080	0.2197
6881	0.2072
% of total	62.8107
Total	5142116

From: 16-Mar-2004
To: 22-Mar-2004 23:45:00

Analysis
● show history
○ normal

Plot-Style
Activity on x-axis
○ log-log
○ linear-log (x-y)
Activity on y-axis
○ log-log
○ linear-log (x-y)
Histogram
○ log-log
● linear-log

☐ Hold

16-Mar-2004 07:45:00

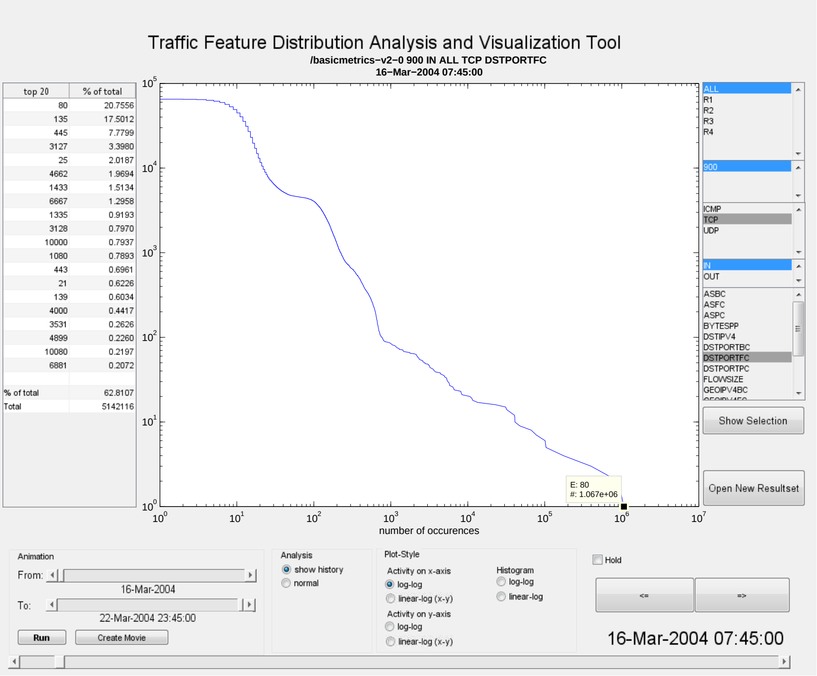
Figure A.6: *Screenshot of the traffic feature distribution analysis and visualization tool. Histogram plot of the destination port distribution for incoming TCP traffic from all flow capturing devices. Activity (#flows) is shown on the y-axis and port numbers on the x-axis.*

To display a distribution, the following three things must be selected: an export device, a protocol and the traffic feature[10] for which the distribution should be plotted. The tool offers various ways to plot and inspect the full

---

[10] Note that the Basic Metrics module outputs distributions for three different activity measures: the number of flows (FC), bytes (BC) or packets (PC) per item and time bin.

traffic feature distribution data. For example, log-log (see Figure A.7) or linear-log activity plots or histograms (see Figure A.6). Furthermore, the tool reports the 20 most active items along with how much they contribute to the overall activity in the form of a list. Finally, we can visually compare distributions by overlaying them or by creating a movie replaying how they evolve over time.


Traffic Feature Distribution Analysis and Visualization Tool

/basicmetrics–v2–0 900 IN ALL TCP DSTPORTFC

16–Mar–2004 07:45:00

top 20	% of total
80	20.7556
135	17.5012
445	7.7799
3127	3.3980
25	2.0187
4662	1.9694
1433	1.5134
6667	1.2958
1335	0.9193
3128	0.7970
10000	0.7937
1080	0.7893
443	0.6961
21	0.6226
139	0.6034
4000	0.4417
3531	0.2626
4899	0.2260
10080	0.2197
6881	0.2072
% of total	62.8107
Total	5142116

E: 80
#: 1.067e+06

Animation
From: 16-Mar-2004
To: 22-Mar-2004 23:45:00

Analysis
● show history
○ normal

Plot-Style
Activity on x-axis
● log-log
○ linear-log (x-y)
Activity on y-axis
○ log-log
○ linear-log (x-y)

Histogram
○ log-log
○ linear-log

☐ Hold

16-Mar-2004 07:45:00

Figure A.7: *Screenshot of the traffic feature distribution analysis and visualization tool. Log-log plot of the destination port distribution for incoming TCP traffic from all flow capturing devices. Activity (#flows) is shown on the y-axis and the ports on the x-axis. Ports are ordered according to their activity. Most active port on the righ (port 80).*

Acronyms

AS	Autonomous System
BGP	Border Gateway Protocol
CSV	Comma Separated Values
FDC	Flow Data Collector
FLAME	Flow-Level Anomaly Modeling Engine
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPR	False Positive Rate
GUI	Graphical User Interface
GUIDE	GUI Design Environment
IPFIX	IP Flow Information Export
ISP	Internet Service Provider
KDD99	Knowledge Discovery and Data Mining competition 1999
KLE	Karhunen-Loeve Expansion
NAS	Network Attached Storage
PCA	Principal Component Analysis
PDU	Protocol Data Unit

PR	Precision-Recall
RAID	Redundant Array of Independent Disks, originally Redundant Array of Inexpensive Disks
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
TES	Traffic Entropy Spectrum
TN	True Negative
ToS	Type of Service
TP	True Positive
TPR	True Positive Rate

Bibliography

[1] Jeffrey Carr. *Inside Cyber Warfare: Mapping the Cyber Underworld*. O'Reilly Media, Inc., 2009.

[2] David Plonka and Paul Barford. Network anomaly confirmation, diagnosis and remediation. In *Allerton'09: Proceedings of the 47th annual Allerton conference on Communication, control, and computing*, pages 128–135, Piscataway, NJ, USA, 2009. IEEE Press.

[3] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall PTR, 3 edition, March 1996.

[4] V. Cerf, Y. Dalal, and C. Sunshine. Specification of Internet Transmission Control Program. RFC 675, December 1974.

[5] Internet systems consortium: Internet host count history. `http://www.isc.org/solutions/survey/history`, July 2012.

[6] European Comission. Commission acts to protect Europe from cyber-attacks and disruptions, MEMO/09/141. `http://europa.eu/rapid/pressReleasesAction.do?reference=IP/09/494`, March 2009.

[7] B. Claise. Specification of the IP Flow Information eXport (IPFIX) protocol for the exchange of ip traffic flow information. RFC 5101, January 2008.

[8] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (Standard), July 2003. Updated by RFCs 5506, 5761, 6051, 6222.

[9] Cisco Netflow. *White Paper: NetFlow Services and Applications*.

[10] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474 (Proposed Standard), December 1998. Updated by RFCs 3168, 3260.

[11] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (Proposed Standard), September 2001. Updated by RFCs 4301, 6040.

[12] J. Postel. Internet Protocol. RFC 791 (Standard), September 1981. Updated by RFC 1349.

[13] P. Almquist. Type of Service in the Internet Protocol Suite. RFC 1349 (Proposed Standard), July 1992. Obsoleted by RFC 2474.

[14] Michael Patterson. ToS, DSCP and NetFlow.... What the DiffServ? (parts 1 to 5). `http://www.plixer.com/blog/netflow/tos-dscp-and-netflow-what-the-diffserv-part-1/`, July 2009.

[15] B. Trammell and E. Boschi. Bidirectional Flow Export Using IP Flow Information Export (IPFIX). RFC 5103 (Proposed Standard), January 2008.

[16] Marc Stoecklin. Anomaly detection by finding feature distribution outliers. In *CoNEXT '06: Proceedings of the 2006 ACM CoNEXT conference*, pages 1–2, New York, NY, USA, 2006. ACM.

[17] Marc Stoecklin, Jean-Yves Le Boudec, and Andreas Kind. A two-layered anomaly detection technique based on multi-modal flow behavior models. In *PAM'08: Proceedings of the 9th international conference on Passive and active network measurement*, pages 212–221. Springer, 2008.

[18] Balachander Krishnamurthy, Subhabrata Sen, Yin Zhang, and Yan Chen. Sketch-based change detection: methods, evaluation, and applications. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 234–247, New York, NY, USA, 2003. ACM.

[19] Arno Wagner and Bernhard Plattner. Entropy based worm and anomaly detection in fast IP networks. In *14th IEEE International Workshops on*

Enabling Technologies: Infrastructures for Collaborative Enterprises (WET ICE), 2005.

[20] Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining anomalies using traffic feature distributions. In *ACM SIGCOMM*, 2005.

[21] Xin Li, Fang Bian, Mark Crovella, Christophe Diot, Ramesh Govindan, Gianluca Iannaccone, and Anukool Lakhina. Detection and identification of network anomalies using sketch subspaces. In *Internet Measurement Conference (IMC)*, pages 147–152, Rio de Janeriro, Brazil, 2006. ACM.

[22] Guavus. `http://www.guavus.com`.

[23] U. Speidel, R. Eimann, and N. Brownlee. Detecting network events via T-entropy. In *6th International Conference on Information, Communications Signal Processing*, pages 1–5, December 2007.

[24] Raimund E. A. Eimann. *Network Event Detection with Entropy Measures*. PhD thesis, University of Auckland, 2008.

[25] Artur Ziviani, Marcelo L. Monsores, Paulo S. S. Rodrigues, and Antonio Tadeu A. Gomes. Network anomaly detection using nonextensive entropy. *IEEE Communications Letters*, 11(12), 2007.

[26] M. Zubair Shafiq, Syed Ali Khayam, and Muddassar Farooq. Improving accuracy of immune-inspired malware detectors by using intelligent features. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 119–126, New York, NY, USA, 2008. ACM.

[27] George Nychis, Vyas Sekar, David G. Andersen, Hyong Kim, and Hui Zhang. An empirical evaluation of entropy-based traffic anomaly detection. In *ACM SIGCOMM conference on Internet measurement (IMC)*, 2008.

[28] Daniela Brauckhoff, Bernhard Tellenbach, Arno Wagner, Martin May, and Anukool Lakhina. Impact of packet sampling on anomaly detection metrics. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 159–164, New York, NY, USA, 2006. ACM.

[29] Bernhard Tellenbach, Daniela Brauckhoff, and Martin May. Impact of traffic mix and packet sampling on anomaly visibility. In *ICIMP '08: Proceedings of the 2008 Third International Conference on Internet Monitoring and Protection*, pages 31–36, Washington, DC, USA, 2008. IEEE Computer Society.

[30] Bernhard Tellenbach, Martin Burkhart, Dominik Schatzmann, David Gugelmann, and Didier Sornette. Accurate network anomaly classification with generalized entropy metrics. *Computer Networks*, 55(15):3485 – 3502, 2011.

[31] Bernhard Tellenbach, Martin Burkhart, Didier Sornette, and Thomas Maillart. Beyond Shannon: Characterizing Internet traffic with generalized entropy metrics. In *10th Passive and Active Measurement Conference (PAM)*, April 2009.

[32] MaxMind® GeoIP Geolocation Technology. `http://www.maxmind.com/`.

[33] Daniela Brauckhoff, Arno Wagner, and Martin May. Flame: a flow-level anomaly modeling engine. In *Proceedings of the conference on Cyber security experimentation and test*, CSET'08, pages 1:1–1:6, Berkeley, CA, USA, 2008. USENIX Association.

[34] SWITCH. The Swiss education and research network. `http://www.switch.ch`.

[35] Cisco Systems Inc. Netflow services solutions guide. `http://www.cisco.com`.

[36] J. Hawkinson and T. Bates. Guidelines for creation, selection, and registration of an Autonomous System (AS). RFC 1930 (Best Current Practice), March 1996.

[37] Carrie Gates and Carol Taylor. Challenging the anomaly detection paradigm: a provocative discussion. In *NSPW '06: Proceedings of the 2006 workshop on New Security Paradigms*, pages 21–29, New York, NY, USA, 2007. ACM.

[38] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):1–58, 2009.

[39] M. Thottan, G. Liu, and C. Ji. *Algorithms for Next Generation Networks*, chapter Anomaly Detection Approaches for Communication Networks, pages 239–261. Springer, 2010.

[40] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *IEEE Symposium on Security and Privacy*, pages 305 –316, 16-19 2010.

[41] Nassim Nicholas Taleb. *The Black Swan: Second Edition: The Impact of the Highly Improbable*. Random House Trade Paperbacks, 2010.

[42] F.H. Knight. *Risk, Uncertainty and Profit*. Series of reprints of scarce tracts in economic and political science. Houghton Mifflin Company, 1921.

[43] Didier Sornette. Dragon-Kings, Black Swans and the Prediction of Crises. *International Journal of Terraspace Science and Engineering*, 2(1):1–18, July 2009.

[44] Didier Sornette and Guy Ouillon. Dragon-kings: Mechanisms, statistical methods and empirical evidence. *European Physical Journal Special Topics*, 205, 2012.

[45] Alefiya Hussain, John Heidemann, and Christos Papadopoulos. A framework for classifying denial of service attacks. In *SIGCOMM ’03: Proceedings of the 2003 conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 99–110, New York, NY, USA, 2003. ACM.

[46] Jelena Mirkovic and Peter Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *SIGCOMM Computer Communication Review*, 34(2):39–53, 2004.

[47] Shriram Sarvotham, Rudolf Riedi, and Richard Baraniuk. Connection-level analysis and modeling of network traffic. In *IMW ’01: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pages 99–103, New York, NY, USA, 2001. ACM.

[48] Anukool Lakhina, Mark Crovella, and Christiphe Diot. Characterization of network-wide anomalies in traffic flows. In *IMC ’04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 201–206, New York, NY, USA, 2004. ACM.

[49] Georgios Androulidakis, Vassilis Chatzigiannakis, and Symeon Papavassiliou. Network anomaly detection and classification via opportunistic sampling. *The Magazine of Global Internetworking*, 23(1):6–12, 2009.

[50] T. Dübendorfer and B. Plattner. Host behaviour based early detection of worm outbreaks in internet backbones. In *14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WET ICE)*, pages 166–171, 2005.

[51] James R. Binkley. An algorithm for anomaly-based botnet detection. In *Proceedings of USENIX Workshop on Steps to Reducing Unwanted Traffic (SRUTI)*, pages 43–48, 2006.

[52] M. Celenk, T. Conley, J. Willis, and J. Graham. Anomaly detection and visualization using Fisher Discriminant clustering of network entropy. In *3rd International Conference on Digital Information Management ICDIM*, pages 216 –220, 2008.

[53] P. Mell, K. Karen, and J. Nusbaum. *Guide to Malware Prevention and Handling: Recommendations of the National Institute of Standards and Technology*. U. S. National Institute of Standards and Technology, 2005.

[54] Vinod Yegneswaran, Paul Barford, and Johannes Ullrich. Internet intrusions: global characteristics and prevalence. *SIGMETRICS Perform. Eval. Rev.*, 31(1):138–147, 2003.

[55] M. Allman, V. Paxson, and J. Terrell. A brief history of scanning. In *ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2007.

[56] Eric Wustrow, Manish Karir, Michael Bailey, Farnam Jahanian, and Geoff Huston. Internet background radiation revisited. In *Proceedings of the 10th annual conference on Internet measurement*, IMC ’10, pages 62–74, New York, NY, USA, 2010. ACM.

[57] Michael Bailey, Evan Cooke, Farnam Jahanian, David Watson, and Jose Nazario. The Blaster worm: Then and now. *IEEE Security and Privacy*, 3:26–31, July 2005.

[58] Symantec. Symantec Internet Security Threat Report, Trends for January to June 2006, 2006.

[59] Symantec. Symantec Internet Security Threat Report, Trends for 2007, 2008.

[60] Microsoft. Microsoft security intelligence report, vol.7. `http://www.microsoft.com/security/sir/archive/default.aspx`, November 2009.

[61] Symantec. Symantec Internet Security Threat Report, Trends for 2008, 2009.

[62] Aleksandr Matrosov, Eugene Rodionov, David Harley, and Juraj Malcho. Stuxnet under the microscope. `http://eset.ru/.company/.viruslab/analytics/doc/Stuxnet_Under_the_Microscope.pdf`, October 2010.

[63] Tomer Bitton. Morto post mortem: Dissecting a worm. `http://blog.imperva.com/2011/09/morto-post-mortem-a-worm-deep-dive.html`, September 2011.

[64] David Moore, Geoffrey Voelker, and Stefan Savage. Inferring internet denial-of-service activity. In *In Proceedings of the 10th Usenix Security Symposium*, pages 9–22, 2001.

[65] David Moore, Colleen Shannon, Douglas J. Brown, Geoffrey M. Voelker, and Stefan Savage. Inferring Internet denial-of-service activity. *ACM Trans. Comput. Syst.*, 24:115–139, May 2006.

[66] Ruoming Pang, Vinod Yegneswaran, Paul Barford, Vern Paxson, and Larry Peterson. Characteristics of Internet background radiation. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, IMC ’04, pages 27–40, New York, NY, USA, 2004. ACM.

[67] Worldwide Infrastructure Security Report I. `http://www.arbornetworks.com/report`, 2005.

[68] Worldwide Infrastructure Security Report II. `http://www.arbornetworks.com/report`, 2006.

[69] Worldwide Infrastructure Security Report III. `http://www.arbornetworks.com/report`, 2007.

[70] Worldwide Infrastructure Security Report IV. `http://www.arbornetworks.com/report`, 2008.

[71] Worldwide Infrastructure Security Report V. `http://www.arbornetworks.com/report`, 2009.

[72] Worldwide Infrastructure Security Report VI. `http://www.arbornetworks.com/report`, 2010.

[73] Norihiko Yoshida. *Content Delivery Networks*, chapter Dynamic CDN Against Flash Crowds, pages 275–296. Lecture Notes Electrical Engineering. Springer, 2008.

[74] Jaeyeon Jung, Balachander Krishnamurthy, and Michael Rabinovich. Flash crowds and denial of service attacks: characterization and implications for CDNs and web sites. In *WWW ’02: Proceedings of the 11th international conference on World Wide Web*, pages 293–304, New York, NY, USA, 2002. ACM.

[75] Ke Li, Wanlei Zhou, Ping Li, Jing Hai, and Jianwen Liu. Distinguishing DDoS attacks from Flash Crowds using probability metrics. In *NSS ’09: Proceedings of the 2009 Third International Conference on Network and System Security*, pages 9–17, Washington, DC, USA, 2009. IEEE Computer Society.

[76] G. Oikonomou and J. Mirkovic. Modeling human behavior for defense against Flash-Crowd attacks. In *Communications, 2009. ICC ’09. IEEE International Conference on*, pages 1 –6, June 2009.

[77] F. Y. Edgeworth. On discordant observations. *Philosophical Magazine Series 5*, 23(143):364 – 375, April 1887.

[78] Animesh Patcha and Jung-Min Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 51(12):3448 – 3470, 2007.

[79] Juan M. Estevez-Tapiador, Pedro Garcia-Teodoro, and Jesus E. Diaz-Verdejo. Anomaly detection methods in wired networks: a survey and taxonomy. *Computer Communications*, 27(16):1569 – 1584, 2004.

[80] Christian Callegari. Statistical approaches for network anomaly detection. In *ICIMP*, 2009.

[81] Paul Barford and David Plonka. Characteristics of network traffic flow anomalies. In *Proceedings of the 1st ACM SIGCOMM Workshop on*

Internet Measurement, IMW ’01, pages 69–73, New York, NY, USA, 2001. ACM.

[82] Paul Barford, Jeffery Kline, David Plonka, and Amos Ron. A signal analysis of network traffic anomalies. In *IMW’02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, pages 71 – 82, New York,NY,USA, 2002. ACM.

[83] Augustin Soule, Kavé Salamatian, and Nina Taft. Combining filtering and statistical methods for anomaly detection. In *IMC ’05: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 31–31, Berkeley, CA, USA, 2005. USENIX Association.

[84] Douglas M. Hawkins, Peihua Qiu, and Chang Wook Kang. The changepoint model for statistical process control. *Journal of quality technology*, 35(4):355 – 366, 2003.

[85] Dong Cheul Lee, Byungjoo Park, Ki Eung Kim, and Jae Jin Lee. Fast traffic anomalies detection using SNMP MIB correlation analysis. In *ICACT’09: Proceedings of the 11th international conference on Advanced Communication Technology*, pages 166–170, Piscataway, NJ, USA, 2009. IEEE Press.

[86] Guillaume Dewaele, Kensuke Fukuda, Pierre Borgnat, Patrice Abry, and Kenjiro Cho. Extracting hidden anomalies using sketch and non gaussian multiresolution statistical detection procedures. In *Proceedings of the 2007 workshop on Large scale attack defense*, LSAD ’07, pages 145–152, New York, NY, USA, 2007. ACM.

[87] Xenofontas Dimitropoulos, Marc Stoecklin, Paul Hurley, and Andreas Kind. The eternal sunshine of the sketch data structure. *Computer Networks*, 52(17):3248–3257, 2008.

[88] Xiuyao Song, Mingxi Wu, Christopher Jermaine, and Sanjay Ranka. Statistical change detection for multi-dimensional data. In *KDD ’07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 667–676, New York, NY, USA, 2007. ACM.

[89] M.S. Alencar and F.M. Assis. A relation between the rényi distance of order α and the variational distance. In *Telecommunications Symposium*, volume 1, pages 242–244, August 1998.

[90] Anukool Lakhina, Marc Crovella, and Christoph Diot. Diagnosing network-wide traffic anomalies. In *ACM SIGCOMM*, Portland, August 2004.

[91] D. Brauckhoff, K. Salamatian, and M. May. Applying PCA for traffic anomaly detection: Problems and solutions. In *INFOCOM*, 2009.

[92] Yu Gu, Andrew McCallum, and Don Towsley. Detecting anomalies in network traffic using maximum entropy estimation. In *IMC'05*, pages 1–6, New York,NY,USA, 2005. ACM.

[93] Frank Feather, Dan Siewiorek, and Roy Maxion. Fault detection in an Ethernet network using anomaly signature matching. In *Conference proceedings on Communications architectures, protocols and applications*, SIGCOMM '93, pages 279–288, New York, NY, USA, 1993. ACM.

[94] Frank Edward Feather. *Fault detection in an Ethernet network via anomaly detectors*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1992. UMI Order No. GAX92-24199.

[95] John Mark Agosta, Carlos Diuk-Wasser, Jaideep Chandrashekar, and Carl Livadas. An adaptive anomaly detector for worm detection. In *Proceedings of the 2nd USENIX workshop on Tackling computer systems problems with machine learning techniques*, SYSML'07, pages 3:1–3:6, Berkeley, CA, USA, 2007. USENIX Association.

[96] Dejing Dou, Jun Li, Han Qin, and Shiwoong Kim. Understanding and utilizing the hierarchy of abnormal BGP events. In *SIAM International Conference on Data Mining*, pages 457–462, 2007.

[97] Ignasi Paredes-Oliva, Ismael Castell-Uroz, Pere Barlet-Ros, Xenofontas Dimitropoulos, and Josep Sole-Pareta. Practical anomaly detection based on classifying frequent traffic patterns. In *IEEE Global Internet Symposium*, 2012.

[98] RuleQuest Research . `http://www.rulequest.com/`.

[99] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[100] Srinivas Mukkamala and Andrew H. Sung. Identifying significant features for network forensic analysis using artificial intelligent techniques. *Intl. Journal of Digital Evidence*, 1:2003, 2003.

[101] Cheng-Fa Tsai and Chia-Chen Yen. Unsupervised anomaly detection using HDG-clustering algorithm. In *Neural Information Processing*, pages 356–365. Springer-Verlag, Berlin, Heidelberg, 2008.

[102] Y. Yasami, S. Khorsandi, S.P. Mozaffari, and A. Jalalian. An unsupervised network anomaly detection approach by k-means clustering & id3 algorithms. In *Proceedings of the 13th IEEE Symposium on Computers and Communications (ISCC 2008)*, pages 398–403, July 2008.

[103] Fernando J. S. Filho. *Unsupervised Diagnosis of Network Traffic Anomalies*. PhD thesis, Université Pierre et Marie CURIE, 2010.

[104] M. E. Torres, M. M. Añino, and G. Schlotthauer. Automatic detection of slight parameter changes associated to complex biomedical signals using multiresolution q-entropy. *Medical Engineering & Physics*, 25(10):859 – 867, December 2003.

[105] E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[106] C. Tsallis. *Introduction to Nonextensive Statistical Mechanics: Approaching a Complex World*. Springer, 2009.

[107] Nonextensive statistical mechanics and thermodynamics: Bibliography. `http://tsallis.cat.cbpf.br/biblio.htm`, regularly updated 2010.

[108] Haakon Ringberg, Matthew Roughan, and Jennifer Rexford. The need for simulation in evaluating anomaly detectors. *SIGCOMM Comput. Commun. Rev.*, 38(1):55–59, 2008.

[109] Wenbin Guo and Shuguang Cui. Fast convergence with q-expectation in em-based blind iterative detection. In *Signals, Systems and Computers, 2006. ACSSC '06. Fortieth Asilomar Conference on*, pages 458–462, October 2006.

[110] Jun Xu, Jinliang Fan, M.H. Ammar, and S.B. Moon. Prefix-preserving IP address anonymization: measurement-based security evaluation and

a new cryptography-based scheme. In *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*, pages 280 – 289, November 2002.

[111] Jinliang Fan, Jun Xu, Mostafa H. Ammar, and Sue B. Moon. Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme. *Computer Networks*, 46(2):253–272, October 2004.

[112] Ramaswamy Ramaswamy and Tilman Wolf. High-speed prefix-preserving IP address anonymization for passive measurement systems. *Networking, IEEE/ACM Transactions on*, 15(1):26–39, February 2007.

[113] E. Kenneally and K. Claffy. An Internet Data Sharing Framework For Balancing Privacy and Utility. In *Engaging Data: First International Forum on the Application and Management of Personal Electronic Information*. MIT, October 2009.

[114] Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas Dimitropoulos. SEPIA: privacy-preserving aggregation of multi-domain network events and statistics. In *Proceedings of the 19th USENIX conference on Security*, USENIX Security'10, pages 15–15, Berkeley, CA, USA, 2010. USENIX Association.

[115] Ruoming Pang, Mark Allman, Vern Paxson, and Jason Lee. The devil and packet trace anonymization. *SIGCOMM Computer Communication Review*, 36(1):29–38, January 2006.

[116] S. Coull, C. Wright, F. Monrose, M. Collins, and M. Reiter. Playing devil's advocate: Inferring sensitive information from anonymized network traces. *Proceedings of the Network and Distributed System Security Symposium*, 2007.

[117] Bruno Ribeiro, Weifeng Chen, Gerome Miklau, and Don Towsley. Analyzing privacy in enterprise packet trace anonymization. In *Proceedings of the 15th Network and Distributed Systems Security Symposium*, 2008.

[118] Martin Burkhart, Dominik Schatzmann, Brian Trammell, Elisa Boschi, and Bernhard Plattner. The role of network trace anonymization under attack. *SIGCOMM Comput. Commun. Rev.*, 40(1):5–11, January 2010.

[119] Martin Burkhart, Daniela Brauckhoff, Martin May, and Elisa Boschi. The risk-utility tradeoff for IP address truncation. In *ACM workshop on Network data anonymization (NDA)*, 2008.

[120] Martin Burkhart, Daniela Brauckhoff, and Martin May. On the utility of anonymized flow traces for anomaly detection. In *19th ITC Specialist Seminar on Network Usage and Traffic (ITC SS 19)*, Berlin, Germany, 2008.

[121] Jianning Mai, Chen-Nee Chuah, Ashwin Sridharan, Tao Ye, and Hui Zang. Is sampled data sufficient for anomaly detection? In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 165–176, New York, NY, USA, 2006. ACM.

[122] Augustin Soule, Haakon Ringberg, Fernando Silveira, Jennifer Rexford, and Christophe Diot. Detectability of traffic anomalies in two adjacent networks. In *Passive and Active Measurement Conference (PAM)*, Louvain-la-Neuve, Belgium, 2007.

[123] MAWI working group traffic archive. `http://mawi.wide.ad.jp/mawi/`.

[124] WIDE project. `http://www.wide.ad.jp/`.

[125] Rafael Ramos Regis Barbosa, Ramin Sadre, Aiko Pras, and Remco Meent van de. Simpleweb/University of Twente traffic traces data repository. Technical Report TR-CTI, Centre for Telematics and Information Technology, University of Twente, Enschede, April 2010.

[126] Internet2 observatory data collections. `http://www.internet2.edu/observatory/archive/data-collections.html`.

[127] Interim Internet2 IPv6 Netflow Anonymization Policy, v1.0. `http://www.internet2.edu/policies/ipv6-mask.html`, 2010.

[128] GÉANT Network. `http://www.geant.net`.

[129] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. MAWILab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *Proceedings of the 2010 ACM Conference on Emerging Networking Experiments and Technology*, CoNEXT '10, pages 8:1–8:12, New York, NY, USA, 2010. ACM.

[130] MAWILab. `http://www.fukuda-lab.org/mawilab/`.

[131] Carrie Gates, Carol Taylor, and Matt Bishop. Dependable security: testing network intrusion detection systems. In *Proceedings of the 3rd workshop on Hot Topics in System Dependability*, HotDep'07, Berkeley, CA, USA, 2007. USENIX Association.

[132] Haakon Ringberg, Augustin Soule, and Jennifer Rexford. WebClass: adding rigor to manual labeling of traffic anomalies. *SIGCOMM Computer Communication Review*, 38(1):35–38, January 2008.

[133] R.P. Lippmann, D.J. Fried, I. Graf, J.W. Haines, K.R. Kendall, D. McClung, D. Weber, S.E. Webster, D. Wyschogrod, R.K. Cunningham, and M.A. Zissman. Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation. In *DARPA Information Survivability Conference and Exposition*, volume 2, pages 12–26, 2000.

[134] Alok Sharma and Sunil Pranit Lal. Tanimoto based similarity measure for intrusion detection system. *Journal of Information Security*, 2(4):195–201, 2011.

[135] S.P. Kozaitis and W. Petsuwan. Improved anomaly detection using block-matching denoising. *Computer Communications*, 35(7):875 – 884, 2012.

[136] John McHugh. Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information System Security*, 3(4):262–294, November 2000.

[137] Mahbod Tavallaee, Natalia Stakhanova, and Ali Akbar Ghorbani. Toward credible evaluation of anomaly-based intrusion detection methods. *Transactions on Systems, Man, and Cybernetics*, 40(5):516–524, September 2010.

[138] Paul Barford and Mark Crovella. Generating representative web workloads for network and server performance evaluation. *SIGMETRICS Perform. Eval. Rev.*, 26(1):151–160, June 1998.

[139] Darren Mutz, Giovanni Vigna, and Richard Kemmerer. An experience developing an IDS stimulator for the black-box testing of network intrusion detection systems. In *Proceedings of the 19th Annual*

Computer Security Applications Conference, ACSAC ’03, pages 374–, Washington, DC, USA, 2003. IEEE Computer Society.

[140] Joel Sommers, Vinod Yegneswaran, and Paul Barford. A framework for malicious workload generation. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, IMC ’04, pages 82–87, New York, NY, USA, 2004. ACM.

[141] Jelena Mirkovic, Songjie Wei, Alefiya Hussain, Brett Wilson, Roshan Thomas, and Stephen Schwab. DDoS benchmarks and experimentation workbench for the DETER testbed. In *TridentCom*, 2007.

[142] Joel Sommers and Paul Barford. Self-configuring network traffic generation. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, IMC ’04, pages 68–81, New York, NY, USA, 2004. ACM.

[143] Joel Sommer. Harpoon - a flow-level traffic generator. `http://cs.colgate.edu/~jsommers/harpoon/harpoon_manual.pdf`, 2005.

[144] John A. Swets. The relative operating characteristic in psychology: A technique for isolating effects of response bias finds wide use in the study of perception and cognition. *Science*, 182(4116):990–1000, December 1973.

[145] J.P. Egan. *Signal detection theory and ROC-analysis*. Academic Press series in cognition and perception. Academic Press, 1975.

[146] Jesse Davis and Mark Goadrich. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning*, ICML ’06, pages 233–240, New York, NY, USA, 2006. ACM.

[147] Foster J. Provost, Tom Fawcett, and Ron Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML ’98, pages 445–453, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[148] Robert Shcherbakov, Donald L. Turcotte, John B. Rundle, Kristy F. Tiampo, and James R. Holliday. Forecasting the locations of future large earthquakes: An analysis and verification. *Pure and Applied Geophysics*, 167(6-7):743–749, 2010.

[149] G. M. Molchan. Strategies in strong earthquake prediction. *Physics of the Earth and Planetary Interiors*, 61:84–98, 1990.

[150] G. Molchan and V. Keilis-Borok. Earthquake prediction: probabilistic aspect. *Geophysical Journal International*, 173:1012–1017, June 2008.

[151] Tom Fawcett. ROC graphs: Notes and practical considerations for researchers. Technical Report HPL-2003-4, HP Labs Tech Report, 2003.

[152] Stefan Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *ACM Trans. Inf. Syst. Secur.*, 3(3):186–205, August 2000.

[153] Salvatore J. Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip K. Chan. Cost-based modeling for fraud and intrusion detection: Results from the JAM project. In *In Proceedings of the DARPA Information Survivability Conference and Exposition*, pages 130–144. IEEE Computer Press, 2000.

[154] Niall M. Adams and David J. Hand. Comparing classifiers when the misallocation costs are uncertain. *Pattern Recognition*, pages 1139–1147, 1999.

[155] John E. Gaffney Jr and Jacob W. Ulvila. Evaluation of intrusion detectors: A decision theory approach. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, SP '01, pages 50–61, Washington, DC, USA, 2001. IEEE Computer Society.

[156] Klaus Mochalski and Hendrik Schulze. Deep packet inspection: Technology, applications and net neutrality. `http://www.ipoque.com/sites/default/files/mediafiles/documents/white-paper-deep-packet-inspection.pdf`, 2009.

[157] Baek-Young Choi, Jaesung Park, and Zhi-Li Zhang. Adaptive random sampling for traffic load measurement. In *IEEE International Conference on Communications (ICC)*, volume 3, pages 1552 – 1556, May 2003.

[158] Nick Duffield, Carsten Lund, and Mikkel Thorup. Properties and prediction of flow statistics from sampled packet streams. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 159–171, New York, NY, USA, 2002. ACM.

[159] Nick Duffield, Carsten Lund, and Mikkel Thorup. Estimating flow distributions from sampled flow statistics. *IEEE/ACM Transactions on Networking*, 13(5):933–946, 2005.

[160] Nicolas Hohn and Darryl Veitch. Inverting sampled traffic. *IEEE/ACM Trans. Netw.*, 14(1):68–80, 2006.

[161] Myung-Sup Kim, Hun-Jeong Kong, Seong-Cheol Hong, Seung-Hwa Chung, and J.W. Hong. A flow-based method for abnormal network traffic detection. *Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP*, 1:599 –612 Vol.1, April 2004.

[162] Nick Duffield, Carsten Lund, and Mikkel Thorup. Estimating flow distributions from sampled flow statistics. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM ’03, pages 325–336, New York, NY, USA, 2003. ACM.

[163] Cristian Estan and George Varghese. New directions in traffic measurement and accounting. *SIGCOMM Computer Communication Review*, 32(4):323–336, August 2002.

[164] Jianning Mai, A. Sridharan, Chen-Nee Chuah, Hui Zang, and Tao Ye. Impact of packet sampling on portscan detection. *IEEE Journal on Selected Areas in Communications*, 24(12):2285 –2298, December 2006.

[165] Jaeyeon Jung, V. Paxson, A.W. Berger, and H. Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *IEEE Symposium on Security and Privacy*, pages 211 – 225, May 2004.

[166] Avinash Sridharan, Tao Ye, and Supratik Bhattacharyya. Connectionless port scan detection on the backbone. *Proceedings of the 25th IEEE International Performance Computing and Communications Conference (IPCCC)*, April 2006.

[167] Kuai Xu, Zhi-Li Zhang, and Supratik Bhattacharyya. Profiling Internet backbone traffic: behavior models and applications. In *ACM SIGCOMM*, pages 169–180, New York, NY, USA, 2005. ACM.

[168] Jake D. Brutlag. Aberrant behavior detection in time series for network monitoring. In *LISA ’00: Proceedings of the 14th USENIX conference on System administration*, pages 139–146, Berkeley, CA, USA, 2000. USENIX Association.

[169] Cristian Estan and George Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems*, 21(3):270–313, 2003.

[170] Jörg Wallerich, Holger Dreger, Anja Feldmann, Balachander Krishnamurthy, and Walter Willinger. A methodology for studying persistency aspects of Internet flows. *SIGCOMM Computer Communication Review*, 35(2):23–36, 2005.

[171] Internet2 network. `http://www.internet2.edu/network/`.

[172] Thomas Dübendorfer, Arno Wagner, Theus Hossmann, and Bernhard Plattner. Flow-level traffic analysis of the blaster and sobig worm outbreaks in an internet backbone. In *Intrusion and Malware Detection and Vulnerability Assessment*, volume 3548 of *Lecture Notes in Computer Science*, pages 103–122. Springer, 2005.

[173] Bernhard Tellenbach, Daniela Brauckhoff, and Martin May. Impact of traffic mix and packet sampling on anomaly visibility. Technical Report 275, Computer Engineering and Networks Laboratory, ETH Zurich, April 2007.

[174] Laura Feinstein, Dan Schnackenberg, Ravindra Balupari, and Darrell Kindred. DDoS tolerant networks. In *DISCEX (2)*, pages 73–75, 2003.

[175] Qi (George) Zhao, Abhishek Kumar, Jia Wang, and Jun (Jim) Xu. Data streaming algorithms for accurate and efficient measurement of traffic and flow matrices. *SIGMETRICS Performance Evaluation Review*, 33(1):350–361, June 2005.

[176] Yu Zhang and Binxing Fang. A novel approach to scan detection on the backbone. In *Information Technology: New Generations, 2009. ITNG ’09. Sixth International Conference on*, pages 16 –21, April 2009.

[177] A. Kind, M.P. Stoecklin, and X. Dimitropoulos. Histogram-based traffic anomaly detection. *Network and Service Management, IEEE Transactions on*, 6(2):110 –121, June 2009.

[178] Antoine Scherrer, Nicolas Larrieu, Philippe Owezarski, Pierre Borgnat, and Patrice Abry. Non-Gaussian and long memory statistical characterizations for Internet traffic with anomalies. *IEEE Transactions on Dependable and Secure Computing*, 4(1):56–70, 2007.

[179] IETF Working Group. Internet Protocol Flow Information eXport (IPFIX). `http://tools.ietf.org/wg/ipfix/`.

[180] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[181] Constantino Tsallis. Possible generalization of Boltzmann-Gibbs statistics. *Journal of Statistical Physics*, 52, 1988.

[182] node9. A lethal Denial of service attack. `http://www.webhostingtalk.com/showthread.php?t=12230`, 2001.

[183] juno, a DDoS attack tool. `http://packetstormsecurity.org/DoS/juno.c`.

[184] Bernhard Tellenbach. Collection of FLAME anomaly models. `http://people.ee.ethz.ch/\~{}betellen/AnomalyModels`, 2010.

[185] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[186] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2010.

[187] Harry Frank and Steven G. Althoen. *Statistics: Concepts and Applications*. Cambridge University Press, 1994.

[188] Colleen Shannon and David Moore. The spread of the Witty worm. *IEEE Security and Privacy*, 2(4):46–50, 2004.

[189] Alberto Dainotti, Antonio Pescaplé, and Giorgio Ventre. Worm traffic analysis and characterization. *IEEE ICC*, 2007.

[190] W. W. Peterson, T. G. Birdsall, and W. C. Fox. The theory of signal detectibility. *Transactions of the IRE Professional Group in Information Theory (PGIT)*, 2-4:171–212, 1954.

[191] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, June 2006.

[192] VeriSign. Distributed Denial of Service (DDoS) Attacks: Latest Motivations and Methods. `http://www.verisign.com/static/idefense\_ddos\_verisign\_20080908.pdf`, 2008.

[193] Robert Richardson. CSI computer crime and security survey 2008. *Computer Security Institute*, 2008.

[194] Arno Wagner. *Entropy-based worm detection for fast IP networks*. PhD thesis, ETH Zurich, 2008.

[195] Brian Trammell, Bernhard Tellenbach, Dominik Schatzmann, and Martin Burkhart. Peeling away timing error in NetFlow data. In *Proceedings of the 12th international conference on Passive and active measurement*, PAM'11, pages 194–203. Springer, 2011.

Acknowledgments

I would like to gratefully and sincerely thank Prof. Dr. Bernhard Plattner for his support, understanding, patience, and most importantly, the opportunity and freedom to work on different research topics and to contribute to different research projects. I greatly appreciated his support for keeping in touch with the non-academic world by encouraging his PhD students to supervise semester or master thesis in the industry or by allowing part time work as a consultant or teacher. I would also like to thank Prof. Dr. Didier Sornette (MTEC ETH) for the lively and inspiring discussions without which this thesis would not have been possible. I thank Thomas Dübendorfer (Google) for introducing me to scientific research and for his continuing support both in and outside ETH Zurich. I am grateful to my supervisors Martin May, Thomas Dübendorfer and Arno Wagner for hours of fruitful discussions as well as for enabling and facilitating contacts with other researchers and research groups. I am also grateful to Anukool Lakhina (Guavus), Thomas Maillart (MTEC ETH), Marc Stöcklin (IBM Research) and my co-workers and friends Daniela Brauckhoff, Martin Burkhart, Eduard Glatz, Dominik Schatzmann and Brian Tramell for interesting discussions and collaboration on anomaly detection or NetFlow data collection and processing issues. Special thanks go to SWITCH for providing us with a constant stream of NetFlow data since 2003 and to Simon Leinen and Peter Haag SWITC for sharing their insight and knowledge about NetFlow data, network security and network operations. I thank my colleagues at ETH Zurich (in alphabetical order) Marcel Baur, Ehud Ben Porat, Matthias Bossardt, Gergely Csucs, Wenping Den, Xenofontas Dimitropoulos, Bernhard Distl, David Gugelmann, Simon Heimlicher, Theus Hossmann, Merkourios Karaliopoulos, Ariane Keller, Franck Legendre, Vincent Lenders, Wolfgang Mühlbauer, Andreea Picu, Gabriel Popa, Ilias Raftopoulos, Thrasyvoulos Spyropoulos, Mario Strasser, Rudolf

Strijkers, Sacha Trifunovic and Jinyao Yan. Special thanks go to Rainer Baumann for his support and guidance on organizational, teaching and administrative issues and to Lukas Ruf for providing both the LaTeX template for this thesis and an opportunity to work for Consecom AG where we solved problems and challenges in the non-academic world. I am grateful to Hans-Jörg Brundiers, Monica Fricker, Damian Friedli, Beat Futterknecht, Caterina Sposato and Thomas Steingruber for their top level administrative and technical support.

I thank the students that I supervised during my time at ETH for their contribution to our NetFlow data processing framework, to building blocks of the entropy telescope and to various other projects in and outside ETH (in reverse temporal order, annotated with thesis title and institution if not ETH): David Gugelmann (Multi-sensor security monitoring, Evaluating and improving TES II), Selim Akyol, Edon Berisha (Software for identifying Application Classes in NetFlow Data), Andreas Müller (@Open Systems: Event Correlation Engine), Tonio Gsell (Evaluating and improving the TES), Cécile Lüssi (@Open Systems: Signature-based Extrusion Detection), Laurent Zimmerli (@Open Systems: Rating Autonomous Systems), Noé Lutz, (@Google: Automatic cryptographic key extraction from arbitrary binaries and decryption of binary input), Gabriel Mueller (Towards Application/Anomaly Detection and Modeling using Network Traces), Raphael Rotondari and Sascha Waser (Software Package for Modular NetFlow v5/v9 Data Analysis), Stefan Guidon (Visualization Toolkit for Long Term Time Series Analysis), Patrik Bichsel (Traffic Theft and Misuse Protection for Anonymous Credentials), Daniel Koller (Towards Application Detection and Modelling using Network Traces), Dominik Schatzmann (Towards Analyzing network traffic from the SWITCH network from 2003 until today), Pascal Gamper (Towards Towards Automated Exploit Signature Generation using Honeypots), Loris Siegenthaler and David Benninger (Software Package for NetFlow v9 Data Analysis), Marcel Marghitola (Towards Exploit Signature Generation using Honeypots II), Janet Malibago (@Open Systems: Traffic Anomaly Detection on a VPN Gateway), Stefan Keller (Entropy-Plugin and Visualisation Tool), Dominik Langenegger and Patrik Bichsel (Towards Exploit Signature Generation using Honeypots), Nicoletta De Maio (Attack Detection and Automated Attack Signature Generation using Honeypots), Oliver Keiser and Philipp Sommer (BlueLocator II), Rashid Waraich (Automatic Attack Signature Generation).

With regard to semester- and master thesis outside ETH, I would like to thank Esther Gelle and Pascal Aebli from *ABB*, Thomas Dübendorfer and

Nils Provos from *Google*, Jan Camenisch, Thomas Gross and Dieter Sommer from *IBM*, and Christoph Göldi, Stefan Lampart, David Schweikert and Roel Vandewall from *Open Systems* for the opportunity to supervise inspiring and challenging student theses in a non-academic institution.

Finally, and most importantly, I would like to thank my fiancée Anna Bähler. Her love, encouragement, support and quiet patience are undeniably the bedrock upon which the past years of my life have been built. I would also like to thank my parents Ursula and Hansruedi for their unwavering trust, faith and unconditional support. I thank the family of Anna, in particular Adrian, Fred, Marianne and Tamara for their friendship and for many relaxing and inspiring moments. And last but not least, I would like to thank all those who have accompanied and helped me in my quest in one way or the other and beg them for forgiveness for not having mentioned them explicitly.

Curriculum Vitae

Bernhard Tellenbach was born in Bern, Switzerland on June 30, 1979.

Educational Curriculum Vitae

2005 – 2012	**Doctor of Science** Swiss Federal Institute of Technology (ETH) Zurich Advisor: Prof. Dr. Bernhard Plattner
2005 – 2011	**Swiss Teaching Certificate for secondary and tertiary level education** Swiss Federal Institute of technology (ETH) Zurich
1999 – 2005	**Master of Science in Electrical Engineering and Information Technology** Swiss Federal Institute of technology (ETH) Zurich
1996 – 1999	**Grammar school** Gymnasium Bern-Kirchenfeld, Switzerland

Working and Teaching Experience

2011 – present	**Associate Professor** Zurich University of Applied Sciences (ZHAW), Winterthur
7 – 9, 2008	**Internship at Microsoft Research, Cambridge, UK**

2007 – Present	**Freelance Security Consultant**
2006 – 2011	**Associate Professor** HSR University of Applied Sciences, Rapperswil
2005 – 2010	**Teaching Assistant at ETH Zurich** Swiss Federal Institute of Technology (ETH) Zurich
1998 – 2002	**Product Management Assistant** ASCOM Autelca AG, Gümligen