Diss. ETH No. 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 TELLENBACH

Master of Science ETH
born 30.06.1979
citizen of Switzerland

accepted on the recommendation of
Prof. Dr. B. Plattner, examiner
Prof. Dr. D. Sornette, co-examiner
Dr. A. Kind, co-examiner

2013

Abstract

Today's Internet allows virtually anytime, anywhere access to a seemingly endless amounts of information and services. Figures such as the six-fold increase of U.S. online retail sales since 2000 document its growing importance to global economy and stresses our demand for a healthy and always-on Internet. Unfortunately, in addition to the need for management and measurement techniques to handle the increasing complexity of the infrastructure, the prospect of making money started to attract criminals resulting in an increase and professionalization of cyber-crime. As a consequence, various methods to better protect the Internet, its participants and the underlying infrastructure from accidental and malicious disruptions and threats have been developed. They resulted in systems such as firewalls to restrict network access, intrusion detection systems to detect and prevent unauthorized access or network monitors to supervise the correct functioning of a network infrastructure. To detect and prevent disruptions and threats, both reactive and proactive methods have been proposed. The reactive approach uses the concept of patterns to identify problems known from theory or practice. Hence, the set of patterns grows with the number of known problems. Proactive methods take a different approach by defining some sort of model of the normal behavior of a system. Any significant deviation from this model is then assumed to be an anomaly caused by a disruption or threat. However, the anomaly may turn out to mean neither a disruption nor a malicious act. Despite considerable advancements, the development of accurate proactive detection and classification methods is still an area of intense research. This is particularly true for methods for high speed networks. To cope with the huge amounts of data, they make use of highly aggregated forms of data. Volume measurements and traffic feature distributions such as the number of connections per time unit or the distribution of the sources of connections are their primary source of information. To

detect anomalous changes in those distributions, various methods have been developed. Among them, entropy based methods are widely used and show a considerable success in both research and production systems. But despite this success, there are still a lot of 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. Yet there is no empirical data about how this affects the visibility of anomalies when using entropy or volume metrics. Another point where additional insight 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 are highly correlated with simple volume measurements and would therefore not contribute much. However, this contradicts experience from many successful applications. The second challenge is the characterization and visualization of changes in distributions. In high-speed networks a compact and informative representation and visualization of changes in distributions is essential simply due to the sheer quantity of information. Widely used methods have either a limited descriptive power because they capture the change with a single number such as the Shannon entropy or their optimal use depends on parameters that are different for different kind of changes as in the case of histograms.

The third challenge is improving the detection and classification performance of entropy-based anomaly detectors. While existing systems show good detection and partially even classification performance with regard to massive anomalies, there is room for improvement 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 parametrized entropies. However, a generalization of these preliminary results to arbitrary types of anomalies as well as appropriate detection and classification systems are yet missing.

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 visibility of an anomaly for sampling rates up to 1:10’000. Next, we analyze whether the entropy of various traffic feature distributions provides valuable information for anomaly detection and refute findings of previous work reporting a supposedly strong correlation between different feature entropies. Our core contribution is the *Traffic Entropy Spectrum (TES)*,

a method for a compact characterization and visualization of traffic feature distributions along with a refinement of the TES 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 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 Mengen 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 Funktionieren 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 Histogram primär von Parametern ab, die von der Veränderung selbst abhängig sind. Die dritte Herausforderung betrifft die Verbesserung der Erkennungs- und Klassifizierungsleistung von entropiebasierten Anomaliedetektoren. 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 im 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 8
1.2.4 Summarization of feature distributions 9
1.2.5 Challenges 12
1.3 Claims and Contributions 12
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 28
2.3 Anomaly Detection Methods in High-Speed Networks . . . 31
2.3.1 Counter- and Distribution-Based Methods 32
2.3.2 Entropy-Based Anomaly Detection Methods 34
2.4 Anomaly Classification 34
2.4.1 Anomaly Signature Based Methods 36
2.4.2 Unsupervised Learning Based Methods 38
2.5 Applications of Generalized Entropy in Other Fields 39
2.6 Evaluation of Anomaly Detection Systems 40
2.6.1 Data Sets 42
2.6.2 Evaluation Metrics 46

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 65
3.3.2 Anomaly Intensity 68
3.4 Impact of the Traffic Mix 70
3.4.1 Results 70
3.4.2 Discussion 71
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 86
4.4 Discussion 88
4.5 Conclusion 89

5 Traffic Entropy Spectrum (TES) 91
5.1 Introduction 91
5.2 Shannon and Tsallis Entropy 93
5.2.1 Terms and Definitions 94
5.2.2 The parameter q 95
5.3 The Traffic Entropy Spectrum 96
5.3.1 Selection of the q-Vector 96
5.3.2 Visualizing Anomalies by Normalization 99
5.4 The Refined Traffic Entropy Spectrum: TES_p 100
5.4.1 Inter-Region Dependency 100
5.4.2 Inter-Region Dependency: Relevance 102
5.4.3 The TES_p 103
5.5 Spectrum Patterns 105
5.6 Evaluation 106
5.6.1 Feature Set 107
5.6.2 Set of Anomalies 107
5.6.3 Compiling the TES 108
5.6.4 Data Analysis 109
5.7 Results 110
5.7.1 Spoting the anomalies 110
5.7.2 Spectrum Patterns 113
5.7.3 The TES in action: Anomaly drill-down 118
5.8 Conclusion 124

6 Entropy Telescope 127
6.1 Introduction 128
6.2 Entropy Telescope 129
6.2.1 Wide Angle: Using Generalized Entropy 130
6.2.2 Zooming in: Separating Activity Regions 130
6.2.3 Image processing: Anomaly Detection 130
6.2.4 Scene Analysis: Classifying Anomaly Patterns . . . 134

6.3 Methodology 136
6.3.1 Data Set 136
6.3.2 Entropy Features 136
6.3.3 Anomaly Models and Injection 137
6.4 Evaluation 143
6.4.1 Detection 143
6.4.2 Classification 147
6.4.3 Prevalence of Anomalies in Real Backbone Traffic . 151
6.5 Conclusion 153

7 Conclusions **155**
7.1 Review of Contributions 155
7.2 Critical Assessment 157
7.3 Future Work 158
7.4 Publications 160

Acknowledgments **163**

Abbrevations and Glossary **167**

Curriculum Vitae **191**

A Appendix **195**
A.1 NetFlow Collection and Processing Infrastructure 195
A.1.1 Overview 196
A.1.2 Costs 198
A.1.3 Data Volume 199
A.1.4 Processing Power and Memory Requirements 199
A.1.5 Contributions 200
A.2 NetFlow Tools 200
A.2.1 NetflowVX Library 200
A.2.2 NetFlow Processing Framework 203
A.2.3 Data Analysis and Processing Tools for Matlab . . . 206

List of Figures

1.1 Evolution of Cybercrime - A figure prepared by the well-known and independent securiy researcher Jart Armin and shown in [26] 3

1.2 Two-way communication between two hosts: Example of aggregation of packet data into flows. 8

1.3 Screenshot of our distribution analysis tool showing a traffic feature distribution for the traffic feature *source port* with the port number on the x-axis and the number of flows with this source port on the y-axis in log-scale. Note that only Transmission Control Protocol (TCP) traffic flowing into our network is considered. In the 300 seconds 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 wiht its various points of presence . 15

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 the network anomaly taxonomy from Plonka and Barford [130]. Extensions are nodes in dark gray and all dashed lines. 27

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

3.2 Blaster flow trace: Impact of sampling on timeseries 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 (Traffic Entropy Spectrum (TES)): 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 $T = x$. This TES is the TES of the autonomous system activity in the incoming traffic around a DDoS attack in 2007. The color scale used ranges from black for the minimum S_q to white for the maximum S_q. 97

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, even sized, steps. 98

5.3 Left: Two activity distributions differing only in the activity of the most active element: in the modified distribution, it's activity is 1.5 times than in the original distribution. The original distribution 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. Right: Relative difference of the TES of the original distribution and the modified distribution. 101

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. 104

5.5 Left: Two activity distributions differing only in the activity of the most active element: in the modified distribution, it's activity is 1.5 times than in the original distribution. The original distribution 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. Right: Relative difference of both, the TES and the TES_p of the original distribution and the modified distribution. 105

5.6 Reflector DDoS attack. 108

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. 110

5.8 TES snapshots from the DDoS 1 and DDoS2 111

5.9 TES snapshots from the Witty and Blaster anomaly 112

5.10 3D TES for incoming SrcPorts before and during refl. DDoS attack for $q = -2...2$. Diagonal axis: date (10 days), vertical axis: normalized entropy. Transparent layers: MIN and MAX at normal week days 115
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. 117

6.1 Entropy Telescope building blocks. 129
6.2 The number of flows per 5min bin of our baseline trace with injected anomalies of intensities 75K and 200K. 136
6.3 Receiver Operating Characteristic (ROC) curves for different feature sets and detection methods: (a) Anomalies of intensity 50K and 75K, Kalman filter (kf) method. (b) Anomalies of intensity 50K and 75K, PCA method (c) Anomalies of intensity 100K and 200K, PCA method. 144
6.4 ROC curves for anomalies with small intensities (50K and 75K) and PCA detection method. 146
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. . . 149
6.6 Comparison of anomaly patterns for SHN_+ and TES_{95}. Obviously, SHN_+ misses crucial information captured by TES_{95}. 150
6.7 Fisher's LDA plots of SHN_+ versus TES_{95}. 150
6.8 Detection and Classification results for a 34-days 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 asigned a color to easily compare its share accross all four pie-charts. 151

A.1 The NetFlow collection and processing infrastructure run by the Communication Systems Group (CSG) at ETH Zurich as of June 2012. 196
A.2 Outline of the design of our NetFlow processing framework. 204
A.3 Screenshot of the traffic entropy spectrum visualization and anomaly detection tool. 208

A.4 Screenshot of the week profile and statistical parameter analysis tool. 209
A.5 Screenshot of the week profile and statistical parameter analysis tool. 210
A.6 Screenshot of the week profile and statistical parameter analysis tool. 211
A.7 Screenshot of the week profile and statistical parameter analysis tool. 212

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) . 17

4.1 Correlation of different feature entropies for traces 1-9 (see Table 4.2) as absolute values of the Spearman coefficients in percent. The table shows maximum, minimum, average, and standard deviation for correlation of $H(X)$ for TCP traffic. . 85

4.2 Overview of traces used. To indicate the size of traces, we list the 75-percentile (75p) of flow counts computed in 5-minute windows. 87

4.3 Correlation of different feature entropies for traces 10a-d (TCP) as absolute values of the Spearman coefficients in percent. 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 in percent. 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 means high/low activity region. 141

6.2 IP address sets used to customize anomaly models. The ID column corresponds to the anomaly ID in table 6.1. 142

6.3 Average classification accuracy in percent for different sets of features and for different training and validation data set constraints. 148

A.1 The estimate costs to build and maintain our NetFlow collection and processing infrastructure. 198
A.2 The estimate costs to build and maintain our NetFlow collection and processing infrastructure. 199
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. 202

Chapter 1

Introduction

Since the late 1980s, when the term Internet was first used[1] to refer to an emerging network infrastructure resulting from the interlinking of the ARPANET and the NSFNet, the Internet morphed from a mere research and communications network of around 160'000 hosts (October 1989) to a corner stone of todays society with around 888 million hosts (January 2012).[2]. The Internet is a digital library, a source for all sorts of (dis-)information, an enabler for keeping in touch independent of time and place using instant messaging, social network sites and voice- or video communication. Furthermore, it's infrastructure provides the basis for remote controlling our houses, factories or even (nuclear) power plants or for surveillance systems, e-government, e-commerce and online banking but also for an efficient management and operation of complex public transport systems and the supply of necessary goods and services: Almost anything we do nowadays involves in one way or the other the Internet and is therefore the basis for our standard of living. A report from the European Commission from March 2009 [53] gives an idea on its importance to the European economy: In 2007, the *"...purchases and sales over electronic networks amounted to 11% of total turnover of EU companies"*.

[1] According to Tannenbaum [168], the term Internet - as we understand it today - emerged around the time when ARPANET was interlinked with NSFNET. However, the first attested occurance of this term was in RFC675: Internet Transmission Control Program [28] where it is used to refer to any network based on the techniques described in RFC675.

[2] Source for the number of hosts is *Internet Systems Consortium: Internet host count history* [78]

Unfortunately, this comes at a price: Our dependence on a stable and reliable Internet makes us vulnerable to blackmail and accidental failures or malicious attacks on (parts of) this vital infrastructure. To complicate matters further, the Internet is a distributed infrastructure run by a large number of Internet service providers (Internet Service Provider (ISP)). The different business cultures and the different legal frameworks in the home countries of these ISPs makes it difficult to address new challenges arising from the increasing complexity and size of the Internet quickly and efficiently. In the past years, all of these aspects and the prospect of making money contributed toward the attraction and professionalization of cyber-crime. But it did not stop there. On the 27th of April 2007, Estonia was hit by a large-scale cyber-attack whose source could not be identified. And on the 7th of August 2008, the IT systems of the Georgian military had been crippled by a cyber-attack shortly before the Russion military invaded the country. Both are often named examples when discussing the threat of cyber-war: the exploitation of our dependency on the Internet to secretly or openly attack an enemy in cyberspace.

The risk we are exposed to and how imminent these threats are is difficult to guess. In [53], the European Commission 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 around 10% to 20%. And what does a look back at the evolution of cyber attacks from 2000 to 2009 tell us? The evolution shown in Figure 1.1 tells quite a disturbing story: Both, the number of cyber attacks per day and the motivation and the level of sophisticaion of cyber attacks increased. From a few Tribal Flood Networks (TFN)[3] attacks motivated by testing and mavericking to over 7000 attacks per day with money as primary motivation and advanced Botnet technologies (e.g., the Conficker Worm) to carry out their attacks. The only positive news is that the size of the attacks do not seem to grow anymore but rather show a sideway trend.

But maybe there is more positive news: The large-scale attacks on Estonia and Georgia seemed to finally have attracted some attention on the political level. In [53], the European Commission proposed a new strategy to prepare Europe to act in case of major disruptions or attacks. Among others, it lists attack detection and response that should be supported by *"...the development of a European information sharing and alert system"*.

Fortunately, researchers in academia and industry have always followed

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

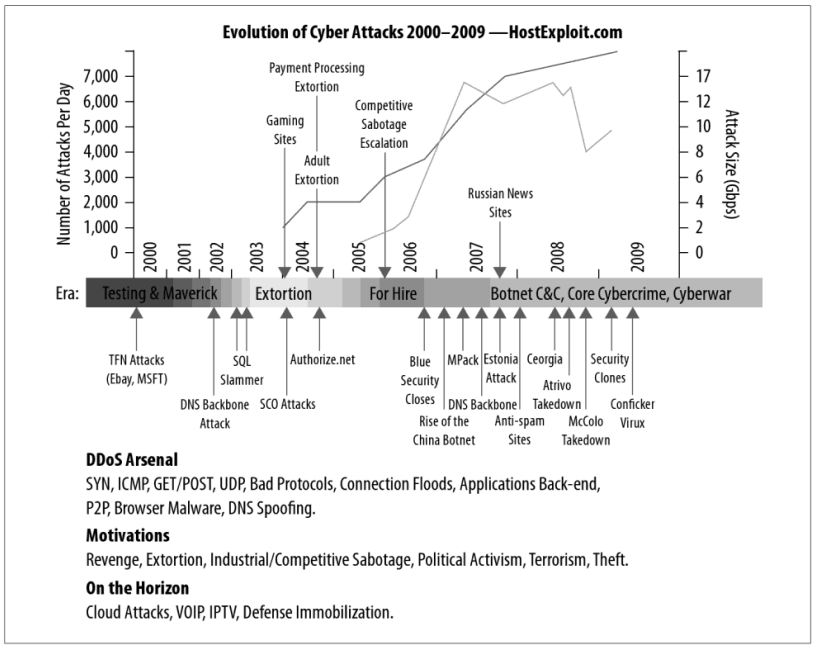


Figure 1.1: *Evolution of Cybercrime - A figure prepared by the well-known and independent securiy researcher Jart Armin and shown in [26]*

the evolution of the Internet as well as the related emerging threats very closely. As a consequence, various methods to better protect the Internet, its participants and the underlying infrastructure from accidental and malicious disruptions and threats have been developed. Among them are firewalls to restrict network access, intrusion detection and prevention systems to detect and prevent unauthorized access and network monitors to supervise the correct functioning of a network infrastructure. To detect and prevent disruptions and threats, both reactive and proactive methods have been proposed. Reactive methods identify problems by comparing an observation with a set of patterns of problems known from theory or derived from observations in the wild. Thus, the set of patterns grows with the number of known problems. Proactive methods take a different approach. They specify a model of the normal behavior of a system and report significant deviations from this model. The underlying assumption of this approach is that such a deviation

is anomalous and typically caused by accidental and malicious disruptions and threats. That's why such a system is typically called *anomaly detection system*. Unfortunately, for the sake of brevity, this term is typically not only used to refer to systems that report whether there was an anomaly or not but also for systems that report its type. Note that we adopt this usage only in cases where it is irrelevant whether or not a system is an anomaly detection system or an *anomaly detection and classification system.*

In this thesis, we concentrate on the class of anomaly detection and classification systems tailored to operate in high-speed networks. In the reminder of this chapter we start with explaining what kind of network we refer to when we say "high-speed network" and implications for systems designed to operate in such networks. We then discuss anomaly detection and what identify a series of challenges related to those detectors and present our claims and contributions. Finally, we briefly introduce the network infrastructure providing the measurement data for this thesis and conclude with an overview over the different chapters of this thesis.

1.1 High-Speed Networks

Intuitively, the term *high-speed network* refers to a network that allows to transport data in a fast and efficient way. But how fast is fast? When we want e.g., to inspect data flowing in or out of a specific network in real-time, theoretical limits such as the bandwidth of its up- and down-link(s) are just one side of the coin. A high theoretical speed limit does not necessarily mean that it is hard to keep up because there is a lot of data to inspect. If just a small share of the available bandwidth is used, there is not much to inspect. Thus, we need a more precise definition than just those of a network that allows to transport data in a fast and efficient way:

Definition - High-speed network: *A high-speed network is faster than most networks, handles the traffic of thousands or millions of hosts and relies on technology and equipment expensive enough to prevent their use if not required or appropriate.*

Examples of networks conforming to this definition are e.g., those of companies such as amazon.com or eBay where network quality and speed is at the core of their business but also those of medium- to large-scale Internet Service Providers (ISP).

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 Intrusion Detection Systems (Intrusion Detection System (IDS)) or other Deep Packet Inspection based systems (Deep Packet Inspection (DPI)) do[4]. A DPI system typically inspects both, the data and header information of each packet to look e.g., for protocol violations, malware or for traffic from a specific application. This is a time consuming process since it needs to decompose the possibly many protocol layers found in a network packet to understand the content of the packet.

At the same time, the large amount of data is the biggest advantage of systems operating at the high-speed network level: They see traffic of thousands or millions of hosts. This allows them to identify anomalies such as a sudden coordinated activity from a subset of hosts invisible without a “global” view. Anomaly detection in high-speed networks exploits this advantage to focus on changes in traffic characteristics that pose a potential threat to the stability and availability of a network.

To exploit the benefit of the birds-eye view, the following data reduction steps are typically involved when operating at the high-speed network level. The first two steps are generic and produce the basic input data for any kind of measurements. The third step is specific to most anomaly detection and classification approaches and the fourth step is specific to the class of entropy based appraches:

- Perform packet sampling to reduce the load on the capturing device
- 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, selceted related problems and challenges and conclude with a briefe summary of the challenges addressed in this thesis.

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

1.2.1 Packet Sampling

Most of the sensors used for data collection in high-speed networks are routers whose primary task is to route packets toward their destination. Any other task is therefore considered to be of low priority and thus gets only a limited share of its precious resources. In practice this means that many operators configure their routers to consider only every tenth, hundredth or even less packet when performing network measurements or generating aggregate forms of data: **many operators apply packet sampling**. Clearly, for most applications, the sampling removes a significant share of information. Thus, the impact of this removal needs to be studied to e.g., identify information that is more robust to sampling or to quantify 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 quite generic definition of a flow can be found in RFC5101 [34], the IP Flow Information Export (IPFIX) Protocol Specification. According to RFC5101, a flow is a set of IP packets shareing 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 [142]).
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[5], 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*. First, a flow contains at least the following information:

- source and destination of the packets (IP address and port number)
- protocol used

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

- time of the first and last packet
- the number of bytes and packets transfered 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 what properties providers of high-speed networks typically use for their flow data[6]. Also note that if packet sampling is used, the information contained in a flow is inaccurate: Instead of e.g., the true number of bytes and packets transfered from the corresponding source to the destination, only the bytes and packets of the packets that were sampled, are reported.

Seconde, 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*[7]*. 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 [32] flow format.*

A consequence of this definition is that a two-way communication results in two flows[8]: one for each direction. Figure 1.2 illustrates this with an example of a two-way communication between two hosts.

If the data is collected by a device with multiple input- and output interfaces, not only the fields of the 6-tuple have to match but also the identifier of the ingress interface recording the packet. Thus, a TCP connection between

[6]Note that with older flow data formats such as Cisco NetFlow [32] version 5, there was no flexibility in the properties that define a flow

[7]Since 1998, this 8-bit long field is named Differential Service Field and consists of the 6 bit long Differentiated Services Code Point [119] and the two bit long Explicit Congestion Notification (starting from 2001 only) citerfc3168. Previous definitions [6, 131] as well as the name “ToS field” are thus outdated and should no longer be used. See [128] for an interesting discussion of this field and its use related to flows.

[8]Note that for IPFIX, RFC5103 [177], 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.

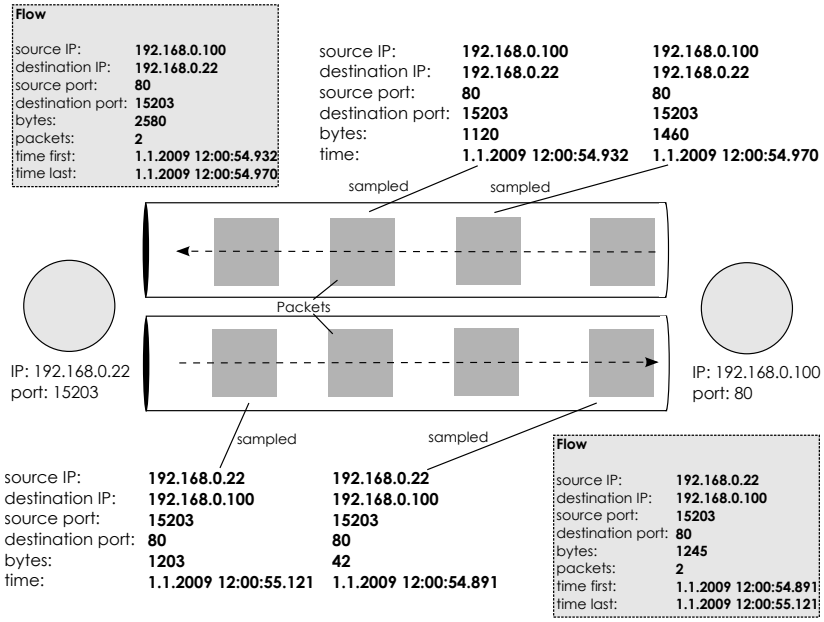


Figure 1.2: *Two-way communication between two hosts: Example of aggregation of packet data into flows.*

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 cross multiple devices or because different packets take different paths (or both). If an application expects a flow to represent e.g., one direction of a TCP connection as close 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 still quite detailed: Complex analyses such as e.g., tracking for each host which other host it contacts and

when it does this and how much data they exchange, or even more complex behavioral analysis are still possible. However, systems that could do this online (at line-speed, in real-time) would either be too expensive or do not (yet) exist. As a consequence of this anomaly detection and classification systems operating in high-speed networks are typically bad at finding needles in the haystack. This is rather the task and strength of systems deployed to protect small networks or groups of host. At the same time, the large amount of data is the biggest advantage of systems operating at the high-speed network level: They see traffic of thousands or millions of hosts. This allows them to identify anomalies such as a sudden coordinated activity from a subset of hosts invisible without a "global" view. Anomaly detection in high-speed networks exploits this advantage to focus on changes in traffic characteristics that pose a potential threat to the stability and availability of a network. However, the birds-eye view of huge amount of traffic from thousands or millions of hosts allowing them to identify anomalies invisible at the local scope or whose extent and impact is not captured well enough at this scope. In the first case, this could e.g., be a sudden coordinated activity from a subset of hosts with the majority of the hosts located outside of the local scope. In the second case, this could e.g., be an anomaly visible at the local scope affecting not only the local network but many different other networks. Since complex analyses of traffic characteristics are typically expensive with regard to compute time or memory or both, the solution adopted by the majority of anomaly detection systems for high-speed networks is to base their detection on an analysis of the number of flows, bytes or packets per time interval 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.

1.2.4 Summarization of feature distributions

Unfortunately, even traffic feature distributions are often not feasible considering that several millions of data points need to be stored and compared to identify anomalous changes in such distributions. Hence, **a more compact representation 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 characterizing changes in

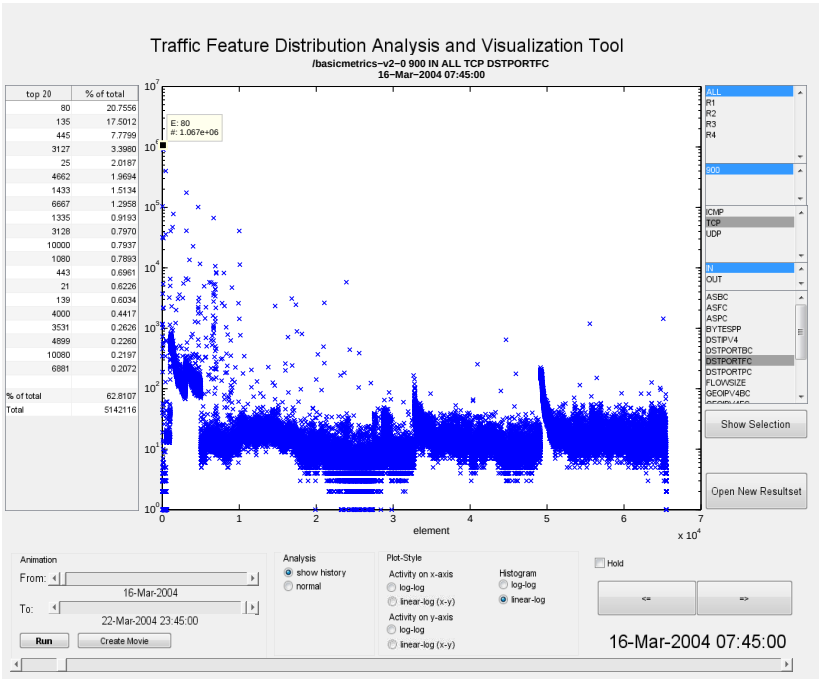


Figure 1.3: *Screenshot of our distribution analysis tool showing a traffic feature distribution for the traffic feature* source port *with the port number on the x-axis and the number of flows with this source port on the y-axis in log-scale. Note that only TCP traffic flowing into our network is considered.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

a distribution and (2) it allows for a compact visualization of such changes. But also other summarization techniques such as histograms [159, 160] or Sketch data structures [93] are 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 allows for 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 and sketch-based approaches by selecting the number of hash functions and the number of bins per Sketch.

While there exist many different forms of entropy, only a few of them have been studied in the context of anomaliy detection and classification in high speed networks. The most common form among them is the Shannon entropy which is not only used in research [96, 99, 185] but also by e.g., NetReflex, a commercial anomaly detection and classification system from Guavus [69]. But also other forms of entropy such as the Titchener T-entropy [49, 157] have been used. However, most works point out that there are limits to entropy based detection, especially when it comes to detect "'slow worms or small-scale attacks"'[9]: it's strength is rather in its generality [49, 157, 185]. While this might be true for entropy-based aproaches in general, by studying different forms of entropy we might discover a form that can assist in pushing the boundaries.

One promising form of entropy is the Tsallis entropy, a parametrized form of entropy. Two studies, one performed by Ziviani *et al.* [196] and the other performed by Shafiq *et al.* [143], provide evidence that this form of entropy might be superior to Shannon entropy. The primary reason why this form of entropy might be superiour to Shannon entropy or other non-parameterized forms of entropy is that depending on the parameter, it allows to focus on changes in diffrent regions of a distribution. If we look e.g., at the distribution of port numbers (see Figure 1.3), changes in entropy caused by ports that are rarely used might be burried in the noise caused by changes in the use of ports that are frequently used. In [196], Ziviani *et al.* study for which value of its parameter Distributed Denial of Service attacks are best detected. And in [143], 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 both since they did not report similar values for the optimal operation point. Thus, a generalization of these preliminary results to arbitrary types of anomalies and an appropriate detection and classification systems are yet missing.

But despite the many positive results on the use of entropy, a recent study by Nychis et al. [121] questions the usefulnes of the entropies of the feature distributions like those of the source and destination IP addresses or the source and destination port numbers because they found a persistent and strong correlation between these entropies.

[9] Wagner and Plattner, Entropy Based Worm and Anomaly Detection in Fast IP Networks ([185])

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 samping. 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. Current methods have either a limited descriptive power because they capture the change with a single number such as the Shannon entropy or their optimal use depends on parameters that are different for different kind of changes as in the case of histograms.

Improving entropy-based detection and classification methods:

Despite the fact that existing systems show good detection and partially even classification performance, there is room for improvement with small to medium sized anomalies. Generalized forms of entropy seem to be promising but a generalization of the preliminary results to arbitrary types of anomalies as well as appropriate detection and classification systems is yet missing.

1.3 Claims and Contributions

The central claim of this thesis is that entropy is an accurate tool to not only detect, but also 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 on the robustness of entropy features with regard to packet sampling ([18, 172]):** 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 [18] 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 [172] 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 large impact on the visibility of an anomaly and might even lead to an increase in its visibility up to sampling rates of 1:10́000. A comparison of feature counts[10] and entropy metrics reveals that they should both be used because they both have their pros and cons.

- **Feature selection: A correlation analysis of various volume- and entropy features ([173]):** We analyze whether commonly used volume- and Shannon entropy features provide valuable and largely independent information and refute findings of previous work reporting a supposedly strong correlation between different feature entropies.

- **A method for capturing and visualizing anomalous changes in traffic feature distributions ([173, 174]):** We introduce the 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 ([174]). On the detection side, we propose to use the information from the TES to derive patterns for different types of anomalies and present ideas how we could use them to automatically detect and classify anomalies. Furthermore, we propose a refinement of the TES that mitigates an unwanted effect to increase the level of detail exposed [173].

- **Design and evaluation of a comprehensive anomaly detection and classification framework based on the TES ([173]):** We propose

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

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 TES and the classical Shannon-only approaches.

- **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 this 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. Two of the most important features added are the flow's origin and destination autonomous system number and the origin and desitnation country in which the source and destination of the flow are located. Both features are derrived 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 the lookup table to use since the mapping of IP address to Autonomous System (AS) number or country is far from being static. It involves both, 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 [67]. 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. Basic modules provided are e.g., a reader capable of reading and merging two input streams, a reader and a writer for the internal flow format or a configurable filter to delete certain flows. Other contributions are a large set of (mostly) object oriented Matlab tools and a Flow-Level Anomaly Modeling Engine (FLAME) [19] based tool for an automated, efficient and massively parallel injection of synthetic anomalies into flow traces. The Matlab tools are in the domains 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) preventing that such data is readily available. Among them are practical issues related to the huge amount of data to be collected and stored as well as legal issues and privacy concerns of potential data providers. We are therefore very thankful to our partner SWITCH [163] from which we get and archive flow data from all of its border routers since 2003.

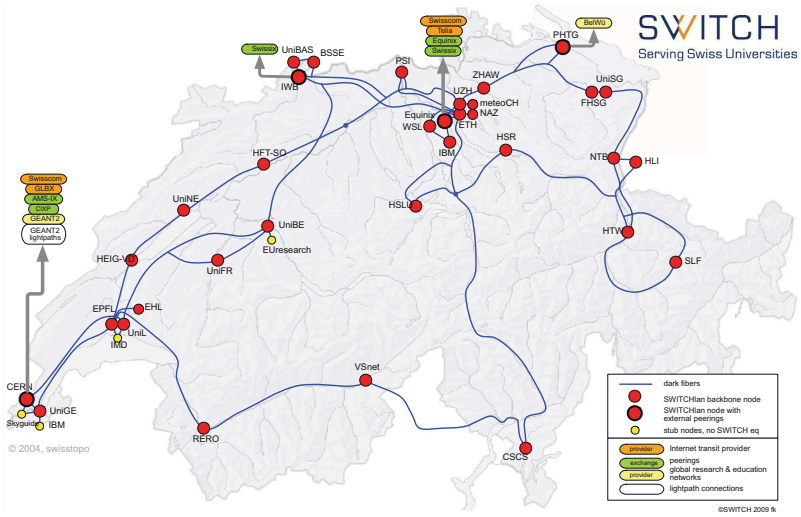


Figure 1.4: *The SWITCH backbone wiht its various points of presence*

SWITCH is the operator and also the name of the Swiss national re-

search 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. But universities and research labs are not the only sites it interconnects. Other sites like e.g., VSnet, a network in the canton of Valais with many members from the domains of science, education, culture and information or Skyguide, the countryś air navigation service provider are also attached to it. Figure 1.4 shows the SWITCH backbone with 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 (GEANT2, GBLX, CIXP).
- Router 3: Located in Basel (SwissIx).
- Router 4: Located in Geneva (GEANT2[11], 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 [33] format[12] which is neither anonymized nor sampled. Thus we get a complete view of the traffic flowing in and out of this network[13].

The above network is also known as autonomous system 559 or simply AS559. According to RFC 1812 [10], an AS is a connected segment of a network topology controlled by a single operations and maintenance organization (here: SWITCH) presenting a consistent picture of the destinations reachable through the respective AS. Note that reachable trough an AS can either refer to destinations that are part of the the respective AS or destinations located in another AS for which this AS is a transit AS.

In the case of the SWITCH network, the largest share of the 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., GEANT2). In our thesis, we only look at network traffic ending or

[11] backup only

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

[13] 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, we rarely loose information because of these effects.

rooting in AS559 and ignore other traffic.

		#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)*

.

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 User Datagram Protocol (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 those of 2009. With a median of around 76 million flows, 10^12 bytes[14] 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

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

and showed a growth rate of around 400 GiB per week.

1.5 Thesis Overview

The reminder of this thesis is structured as follows: In Chapter 2 we review related work in the field of anomaly detection and classification. We start with 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 the chapter with a brief review of literature about 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 the traffic mix. In Chapter 4 we continue our assessment by analyzing whether commonly used volume- and entropy metrics provide valuable and largely independent information. We present and discuss the results of our analysis and discuss why our analysis refutes findings of previous work reporting a supposedly strong correlation between different entropy metrics. Furthermore, we introduce new metrics reflecting the geographical structure of the traffic and show that they add another layer of potentially useful information.

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

Finally, in Chapter 6 we introduce our anomaly detection and classification system called the entropy telescope and 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. There exists a considerable amount of related work analyzing and addressing the various aspects of the problem. This chapter reviews a number of publications relevant to the field of anomaly detection with focus on anomaly detection and classification in high-speed network.

This chapter is structured as follows: Starting from common generic 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 identifying or characterizing (prevalent) anomaly types and continue with publications on anomaly detection methods We then concentrate on different works on anomaly classification and conclude this section with a review of publications on the use of generalized forms of entropy in biomedical engineering, physics and complex systems.

2.1 Anomaly: A Definition

Events like e.g., a massive Distributed Denial of Service (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. However, despite different meanings associated with the term anomaly, the following common generic definitions are used [29, 64, 149, 175]: 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 e.g., just be a normal behavior of a system rarely seen and therefore difficult or impossible to include in models for the normal behavior. Nassim N. Taleb coined the terms Black Swan or Grey Swan to refer to exactly this problem. His book [167] raised a lot of interest mainly because of its significance in the context of the crash of the financial markets in 2008. But the ideas presented by Taleb are not entirely new: The roots of the black swan concept is the Knightian uncertainty[2] which refers to an immeasurable risk. In contrast to Talebs concept of black swans, the concept of dragon-kings [153] claims that black swans do almost not exist and that the concept of black swans is largely flawed when studying real systems. In [153], Sornette defines dragon-kings as extreme events that do not belong to the same population as the other events. The hypothesis articulated in [153] and elaborated in [154] is that dragon-kings appear as a result of amplifying mechanisms that are absent or not fully active for the rest of the population. This gives rise to specific properties and signatures that may be unique characteristics of dragon-kings. Sornette and Ouillon [154] explore the generality and applicability of this concept and find significant relevance for the natural, biological and social sciences.

As a consequence, many researchers measure deviations from a *baseline* rather than from ‘normal’ [149]. Or they avoid this problem by designing and evaluating detectors for a set of (prevalent) anomaly types identified and specified beforehand therewith losing the ability to detect new types of anomalies. Another aspect often forgotten is that deciding whether something is an *anomaly* is not an objective zero or one decision [149]: Do we require 50, 500 or 5’000 clients connecting to an otherwise unpopular web server per minute to trigger an alert? From an operational point of view, the policies of the operator specify what is considered to be an anomaly. Maybe they define some sort of minimum size or decide to ignore some types of anomalies entirely because they do not pose a threat to the operators infrastructure. However, since a policy-based view can typically be obtained from any detection and

[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 [91, 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”.

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 discussion 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 relationsship. The reason for using this term this way is that while it is possible to detect 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 classificatoin systems, the term *anomaly type* might also be used to refer to anomalies that share certain properties independent of the classification system while the term *anomaly class* might be used to refer to the label attributed to an anomaly by the anomaly classification system[4].

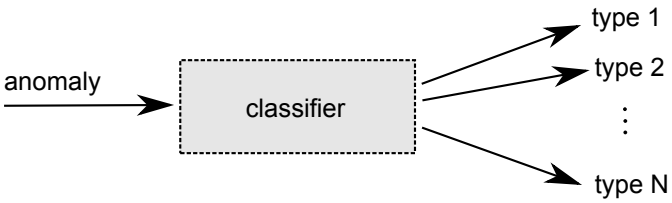


Figure 2.1: *When discussing anomaly classification, the term* anomaly *is mainly used to refer to a specific set of anomaly types or classes.*

2.2 Anomaly Types

As mentioned in the previous section, our thesis focuses on a set of well-known anomaly types since it is generally difficult if not impossible to classify unknown anomalies. The following review of related work on taxonomies

[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 e.g., classes that include several anomaly types, there is no one-to-one relationship.

gives an overview of the well-known anomaly types we can choose from. In our thesis, we make use of a subset of them only: *Probing/Scanning*, *DoS*, *DDoS* and *Worms* whereas 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 prefered to use a smaller set of anomaly types but to 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 occurance[5]. Another reason for not extending our set further was that the two most porminent anomaly types not included in our set are the types *Outage* and *Heavy Hitter (or alpha flows)* which are both realatively easy to spot using volume-based anomaly detectors only[6].

In addition to these considerations, our choice of the types of anomalies 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 the selected anomaly types at the end of this section shows, that they all meet the first criteria. The review of related work on anomaly detection and classification does the same for the second cirteria. Note that the review of related work on the prevalence of the selected anomaly types also includes the Flash Crowd anomaly type to 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 [130] Plonka and Barford propose a taxonomy of network anomalies which is based on what is used operationally at the University of Wisconsin to log anomalies. Their taxonomy includes the following rather generic 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

[5] A volume anomaly might e.g., be easier to spot in times where the expected variance is small

[6] Note that depending on the actual size of the anomaly (also with respect to the dynamics of the normal traffic), the same might but does not have to be true for anomalies of the types included in our set. Only when doing classification, these types might profit from additional information provided by the entropy features.

taxonomy is shown in Figure 2.2 as a tree rooted at the *Anomaly* node. Note that both, *Pouplar Content Exchange* and *Prohibited Content Exchange* are considered to be of the *Flash Crowd* type. If e.g., a hacker manages to break into a protected area of a webserver removing the protection and finally causing a large number of people wanting to access the now publicly available content (which is actually not intended for sharing), this can not be distinguished from *Flash Crowds* caused by *Popular Content Exchange*[7] without knowledge about the content that is in fact exchanged.

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 rather generic anomaly types into a set of more specific types. The taxonomy from Hussain *et al.* [74] splits the denial of service anomalies into: Single-source, Multi-source and Reflector anomalies. Other names for them are DoS, DDoS and Reflector DDoS or RDDoS. There exist even more detailed taxonomies but they are rarely used with systems for high-speed networks, the main reason being that collecting and processing of the therefore required information does not scale. An example of such a taxonomy is presented in [109] where the authors define a large set of parameters such as attack dynamics, impact of the attack, victim type to distinguish various types of DDoS anomalies.

Other important anomaly types that do not have their own node in the taxonomy of Plonka and Barford are *alpha flows*, worms and botnet activity or anomalous activities from Peer-to-Peer (P2P) networks. *alpha flows* are a small number of flows that have a very large quantity of packets or bytes or that represent an unusually high rate point to point byte transfer. Note that *alpha flows* are sometimes also called *Heavy Hitters*. Examples of publications using these types are: [7, 95, 140] for *alpha flows*, [7, 43, 95, 185] for *Worms*, [15, 27] for botnet activity and [121] for anomalous P2P network activity. Figure 2.2 shows a possible integration of these types into Plonka and Barford's taxonomy: Alpha flows could be inserted as an additional node to the Anomalous Measurement type while worms could be seen as new form of abuse being a combination of *Prohibited Content Exchange* and *Probing/Scanning*. Botnet activity is more difficult to integrate since 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

[7] E.g., breaking news published on a news site attracts a massive number of visitors in a short time frame.

node might be replaced with one for each virtual network type.

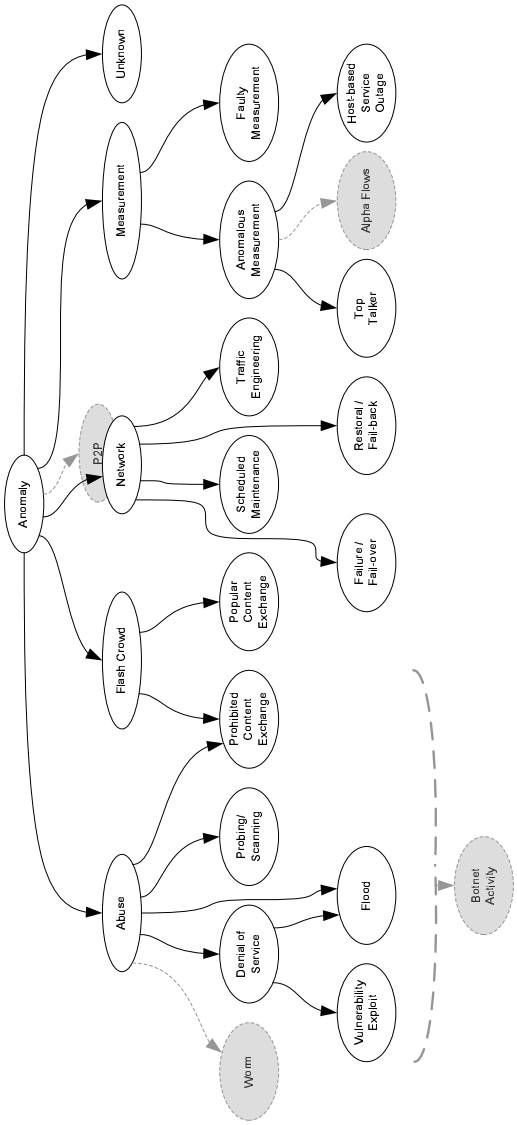


Figure 2.2: *Extended version of the network anomaly taxonomy from Plonka and Barford [130]. Extensions are nodes in dark gray and all dashed lines.*

2.2.2 Prevalence of Anomaly Types

Probing/Scanning and Worms:

The root causes of probing and scan traffic are manifold: it might originate from attackers looking for a target to attack and compromise but also from productive activity such as monitoring services like pingdom[8] or web crawlers from search engines like Google.

Another source of scan traffic is worms. Or more precisely, 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 themselfes without user intervention. Furthermore, whenever a worm tries to propagate and infect another systems, it typically starts to scan for appropriate targets. If it finds a vulnerable target, it attacks the target and copies itself to it. Since the whole scanning and spreading process is fully automated, a worm might quickly infect most, if not all vulnerable and reachable hosts. Consequently, a worm would cause scan traffic which looks quite different in both volume and spatial distribution from the scan traffic from attackers looking for a target to attack and compromise or from monitoring activity or other productive scanning activity.

In [192], Yegneswaran *et al.* study the prevalence and distribution of anomalies of different categories. Based on a large set of firewall logs from more than 1600 networks, they analyzed their quantity and variety and projected the results to the entire Internet. Their analysis with focus on different kinds of worm activity and four categories of port scans[10] determined that with an estimate of 25 billion anomalies per day theses anomalies are indeed prevalent. 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 System 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 [5] where Allman *et al.* study the behavior of scanners from 1995 to 2007.

While the findings of [192] about the persistence of worm traffic are con-

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

[9] See [107] for a more detailed definition of different types of worms and other types of malware.

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

firmed by several sources (e.g. [9, 188]), many thought that after observing a significant decline in the volume of network worms starting with 2006 (see e.g., [164, 165]), the days of the worms[11] would be over. In September 2006, Symantec wrote in its Internet Security Threat Report [164, p.2]:

" 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 appearing in November 2008 [108, 166] was a wake-up call to many who thought these kind of worms a memory from the past. In its Internet Threat Report for 2008 [166, p.7] published in April 2009, Symantec wrote:

> 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 [103][13] worm (2010) or the Morto [16] worm (2011) demonstrated that worms still pose a very realistic threat and (2) because worm traffic from past worms is still around.

[11] Mainly those of the 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 use self propagation techniques mainly in the local area network to find and infect SCADA systems.

DoS and DDoS:

A first quantitative estimate and characterization of denial-of-service anomalies is presented in [115] and refined in [114] where Moore *et al.* analyzed a set of 22 at least week-long traces from the years 2001 to 2004. The authors propose and use an analysis technique called backscatter analysis 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. A consequence of their methodology is that while they account for DoS and DDoS anomalies [15] , they are likely to undersetimate their true number since they do not account for anomalies caused by attackers applying different spoofing schemes or no spoofing at all. With their technique, they observed over 68,000 attacks to over 34,000 distinct victim IP addresses. In all of their traces, they found at least slightly less than 1000 anomalies per week or roughly more than 125 anomalies per day. While their data does not allow to identify whether anomalies of this type are on the rise, they provide evidence that they are indeed prevalent. The yearly Worldwide Infrastructure Security Reports from Arbor Networks [79–84] shows 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 are 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. [84] also contains some quantitative information about the number of attacks: 47% of the survey's participants experienced 1 to 10, 41% 10 to 500 and another 6% 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 refering to background radiation resulting from denial-of-service attacks using multiple spoofed addresses. According to [125], 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 is not clear whether , it is a DoS or DDoS anomaly.

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

Flash-Crowds

An anomaly type which looks very similar to a denial of service attack at the network level is the Flash Crowd. During a Flash Crowd, the number of machines accessing a specific resource in the network increases significantly reaching an abnormal level. Events that trigger it are typically high-profile events such as the Fédération Internationale de Football Association. English: International Federation of Association Football. (FIFA) world-cup final, presidential elections or a new version of a popular software being released suggesting that Flash Crowds are prevalent and distinguishing them from DDoS anomalies is important. [193] summarizes results from various studies on Flash Crowds including [85] where Jung *et al.* presents an analysis of Flash Crowds and (D)DoS attacks on web servers with special attention to characteristics to distinguish the two. They find that in contrast to (D)DoS attacks, during Flash Crowds, the client population has a significant overlap with the normal population. 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 whereas in the case of (D)DoS attacks, they are stable and higher than usual. Based on these findings, [95] 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, this and other promising approaches (e.g., [98, 122]) typically depend on behavioral aspects which are expensive to track in high-speed networks.

2.3 Anomaly Detection Methods in High-Speed Networks

Since the first attempts at detecting outliers or anomalies in the 19th century by Edgeworth [47], a vast amount of anomaly detection methods have been developed and used. An overview of the most important methods in different application domains such as intrusion detection, fraud detection, medical and public health anomaly detection, industrial damage detection, anomaly detection in text data or sensor networks and some other domains are presented in [29]. Surveys with a narrower focus are [52,127] with [127] focusing on the field of (network) intrusion detection in general and [52] also having a separate section on flow-based detection methods. The tutorial by Callegari [25] as well as [175] provide an overview on the most prominent methods used for

detecting anomalies in network traffic:

- 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 reporting anomalies directly or outputting residual signals to be used with a simple threshold detector. The last three items however, are not anomaly detection methods in a strict sense but rather a pre-processing tool to summarize distribution-based features.

Feature distributions provide a more detailed view on whats going on in a network than traditional counter-based features while still being lightweight enough to scale to large-scale and hight-speed networks. This is typically not the case for methods relying on detailed (behavioral) information of single hosts or groups of hosts.

We now first review related work on approaches based on counter- or distribution information before we switch to discussing approaches relying 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 which can be derived from the counters found in routers or other networked devices. In [14], 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 [13], 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 count metrics from before but also add new count metrics such as the average packet size.

They do a true positive analysis using a trace from their campus network and 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 [156], 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 do a thorough evaluation based on a combination of realistic workloads from a backbone network and synthetic anomalies and found that the conventional generalized likelihood ratio (GLR) test [71] method performs best with a true positive rate of 100% and a false positive rate of 7%.

In [97] Lee *et al.* present an 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 some minor differences only.

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, others use summarization techniques such as histograms [159, 160] or Sketch data structures [39, 41, 93, 99]. 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 tuning of the amount of data to be stored and analyzed. Histogram-based methods by choosing an appropriate binning method and bin size and sketch-based approaches by selecting 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 [152, 159] 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 ($\alpha = 1$) [4]

2.3.2 Entropy-Based Anomaly Detection Methods

Approaches that rely on entropy use Shannon-Entropy [17, 27, 94–96, 99], an approximation of (Shannon-)entropy based on compression algorithms [185], the Titchener's entropy (T-Entropy) [49, 157] or some generalized form of entropy [143, 196]. In [196], 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 a q-value around 0.9 is best for detecting DoS attacks, Shafiq *et al.* [143] could optimize the detection of port scans of malware using a q-value equal to 0.5. A different application of entropy is presented in [68], where the authors introduce an approach to detect anomalies 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 does the classification with respect to the mechanisms or processes involved. The reason for this is that symptoms are typically straightforward to see or measure while the underlying mechanisms or processes are not: the same symptoms might e.g., have quite different root causes from a mechanism or process perspective. An example of two network anomalies with similar symptoms but quite different underlying mechanisms are e.g., Flash Crowds and Distributed Denial of Service (DDoS) attacks. While a classification according to the mechanisms and processes would be more accurate, it is - as in medicine - a question of cost and benefit: the extra analysis and/or measurements required to identify the mechanism or process is only done when needed. Unfortunately, the same approach as in medicine does typically not work for network anomalies in high-speed networks. The therefore required analysis processes and measurements (e.g., full

packet traces) are typically too expensive or simply not available[18].

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

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

When it comes to symptoms based anomaly classification, we can basically choose from two fundamentally different approaches: The first one is to extract a *fingerprint* of an anomaly and to compare it to a series of known fingerprints. The other (ideally) does not require a priori knowledge: It applyes data mining and (unsupervised) learning technologies to identify the characteristics of a yet unknown anomaly and 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 with “high-profile TV event (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 specifying a characteristic sequences of bits and bytes in data streams (e.g., described by regular expressions), the term *fingerprint* is used when refering to a description specifying the parameters and their values or the required value ranges of a predefined parameter space (e.g., the parameters measured when doing an operating system fingerprinting of a host) and

[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.

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, there is no such thing as patterns, signatures or fingerprints. However, note that if we do not repeat the unsupervised learning for each new classification task but map the attribute vectors to clusters generated earlier, those 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 not only known problems and threats but also 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 the anomalies and tries to find groups of similar anomalies by comparing anomalies with each other and then outputing descriptions for these groups. In a next step, these descriptions have to be analyzed by an expert to identify whether they reflect a known or unknown anomaly or are not representing anything useful at all. Such systems are discussed in the next subsection. The other way to resolve this contradiction is that anomaly signatures are only used 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 [57]. In their paper *Fault Detection In an Ethernet Network Using Anomaly Signature Matching* they 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. A Fault Feature Vector specifies the performance parameters and their corresponding value or value range required to match a specific fault. Performance parameters are e.g., 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 based on expert knowledge and experience with faults and their manifestations. However, since they also used a refinement strategy after the initial definitions have been tested with real data, they also depend on ob-

[20] For anomaly detection, they use statistical methods (PAMS and AADS) described in [58].

servations made in “training data”. Another example of a system that uses signatures that are at least partially created using expert knowledge and experience on anomalies and their manifestations is [96]. In [96], 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 dispersion of a specific traffic feature distribution and propose to use these characteristics to identify the type of the anomaly. They first turn to an unsupervised clustering approach since it can potentially reveal yet unknown anomalies. In a next step, they analyze the clusters produced by the clustering algorithm and found that the resulting clusters do indeed represent different anomaly types and assign labels to them. Finally, they propose to use the labeled clusters (=signatures) for automated classification.

With the increasing popularity and maturity of data mining methods, 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 handles just the anomaly type worm, is described in [3]. An example of a signature-based anomaly classifier for anomalies found in the Border Gateway Protocol (BGP) routing update messages is presented in [42]. There, 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 recend example of decision tree based anomaly classification is [126], where Paredes-Oliva *et al.* make use of the descriptive power of these data structures to classify different types of network anoamlies. 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 to 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.

Furthermore, if we also include the waste literature from the intrusion detection domain, we can find many approaches relying on learing techiniques

[21] 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 [139] as an improvement to the C4.5 algorithm [133]

to automatically generate signatures from labeled “anomalies”. An early example is [116] where the authors compare and use Support Vector Machine (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 typcially just try to create clusters (groups) of similar items by looking at the data that is fed to the classification system. These clusters could then be analyzed by an expert to find out what kind of anaomaly each of them represents (if any) and whether it was already known or a yet unknown anomaly. This final step is typically not required when one wants to do anomaly detection rather than anomaly classification only. Separation of normal and abnormal data is possible if the following two assumptions about the data hold: First, the amount of normal data is far bigger than the amount of anomalous data and second, the abnormal data is statistically different from normal data [179]. The lack for the need for the cluster analysis by an expert is most likely the reason why unsupervised learning methods are far more frequently used with anomaly detection systems (e.g. [179, 191], than with anomaly classification systems. Actually, publications that make use of unsupervised learning methods in the context of anomaly classification use them mainly to support building the “signatures” for the anomaly classification component. One example of such usage is the presented by Lakhina *et al.* [96] which we discussed already in the section on signature based approaches. Another example of such usage is [60] 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, their classification algorithm works as follows: First, the anomalous flow set is added to the flow sets that have already been labeled. Next, these flow sets are clusteed using the following features as coordinates of the input vector: the average flow size in packets, the average packet size in bytes and the entropy of the distribution of packets as well as 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 leafs in the smallest subtree containing all the siblings of the new anomalous flow set do all have the same label, 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, biomedical engineering or in a broader context, in complex systems.

One example from biomedical engineering is e.g., [176], 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 [123] algorithm to identify abrupt changes in its mean.

However, since applications in fields other than anomaly detection are not in the focus of this thesis, we refer the reader to the book *Introduction to Nonextensive Statistical Mechanics: Approaching a Complex World* [180] from Constantino Tsallis which presents and discusses a selection of paradigmatic applications in various sciences or to [182] for a complete bibliography on Tsallis entropy related publications.

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

[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 e.g., Gates *et al.* [64], Ringberg*et al.* [137] and Sommer and Paxson [149], anomaly detection and more specifically the evaluation of anomaly detection systems is full of pitfalls and 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 since the omnipresence of attack traffic from activity such as network or port scans is a known fact just as the existence of benign anomalous activity such as Flash Crowds. This is also the reason why most approaches do exactly what the authors of [70] recommend: First, determine what malicious activities should be detected. Next, check which (if any) characteristics of these activities appear as anomalous, and finally, design the system to detect them.

Another assumption that is problematic is the assumption 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, it might be easier to get some confidence that this modified assumption is true. After all, an anomaly is something that deviates from normal while an attack does not have to. Moreover, if a system should focus 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.

Anoter issue raised by Gates *et al.* is that authors often forget that low false positive percentages do not necessarely imply that the system is actually useful in practice. The flase 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. But if it is equal to tens or hundreds of anomalies per day, he would not. Hence, a false positive rate should always be accompanied by information on the to number of false positives per time interval.

Finally, they point out that an anomaly detection system should also provide some labeling component since the administrators should not 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 into which the operator is interested in. Thereby adressing 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 not be based on real data with some known anomalies in them, but on simulation. Their main point is, that real data typically contains a 'few istances' of a certain anaomaly and at a specific time of day: With just a few instances, it is difficult to account for the fact that different instances of an anomaly of a certain type might differ quite signifficantly (in volume, timing or other features). Furthermore, there is also the highly dynamic background traffic which typically plays an important role in the detection process.

Simulation is a way to account for this problem: Anomalies can be parameterized and their many variants e.g., injected in background traffic that is either simulated or taken from virually anomaly free sections of real network traces. Unfortunately, comprehensive studies on how to parameterize anomalies are lacking [19, 109]. But if we look for bits and pieces only, there are few studies that provide them [13, 14, 85, 95, 114, 115, 125]. These studies discuss e.g., 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, timing, etc. for the rather broad set of volume anomalies. While they do not provide statistics detailed enough to compile recipies for the parameteriza-

tion of those anomalies, they provide valuable insight when setting up one's own simulation models and parameters. In [19], Brauckhoff *et al.* make a step ahead and present three models for anomalies which they derived from real anomalies found in their network traces and also present the Flow-Level Anomaly Modeling Engine (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 relying on algorithms that are first trained with reference input to learn its specifics (threshold(s), model(s), etc.) prior to being exposed to input for the actual detection process. The reason why they consider it to be harder 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 is finding activity similar to something previously seen without a precise a priori description of that activity. And since novel attacks are only novel if they were not previously seen, they conclude that this premise is not well aligned with the strength of machine learning.

But this premise is not the only characteristic of anomaly detection is not well aligned with 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 for conducting sound evaluation

In summary, Sommer and Paxson say that using machine learning to find something previously seen without a precise a priori description, is a move on less dangerous grounds. But the other requirements such as sufficient training data and a sound evaluation methodology are still a challenge.

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-ciritcal information such as packet payload and/or IP addresses. Packet payload could be used to build full-fledged user profiles. But also IP addresses are critical: if the mapping

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

of an IP addresses to a user is known, they allow to link a certain packet or flow to this specific user. This again could be exploit to build profiles of who communicates with whom.

Anonymization of network traces could resolve these issues. However, the vast amount of methods and tools adressing (network) data anonymization [54, 134, 189] and/or privacy preserving sharing of network data [24, 88], does not seem to foster public access to network traces. The root cause for this situation might be twofold: One the one hand, it has been shown [23, 35, 124, 135] that it might be very difficult to anonymize traces in a way that they do not leak any valuable information. On the other hand, it is difficult to quantify the impact of anonymization on the utility of the data. Brauckhoff *et al.* [21, 22] studied this impact for the anomaly detection use case. They find that the anonymization degrades anomaly detection performance significantly; with ingreasing strength of the anonymization, 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 details. Our findings are similar to those presented by the authors of [101, 155].

Some notable sources of network traces are:

- The packet traces published and maintained by the MAWI [104] Working Group of the WIDE [187] project. Among other traces, they provide daily packet header traces for different, mostly trans-Pacific lines with link speeds of up to 150Mbps and some. Per sampling point and day, a rougly 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 [11]. The repository contains a series of six packet header traces[28] from relatively small sized stub networks and a NetFlow v5 trace from August

[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.

2007 captured in a /16 university network. The IP addresses of these traces have been anonymized using a prefix-preserving anonymization strategy

- The NetFlow data repository of the Internet2 backbone network [76](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 Geant [66] backbone network: Geant 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 largest problem. A notable exception are the daily packet traces published and maintained by the MAWI [104] Working Group. In [40], Dewaele *et al.* made a first attempt at labeling the traces from 2001 to 2006. First, they proposed and used a sketch-based anomaly detection approach to locate anomalous traffic patterns. In a second step, they performed a manual inspection[30] of the anomalous traffic to confirm and label it or to discard it.

In [61], a second attempt is made toward providing labels for this dataset. This time, the anomalies are located and labeled based on the outputs of four different anomaly detectors.[31] The labels are accessible through MAWILab [105] [32], a database that assists researchers to evaluate their traffic anomaly detection methods.

An alternative way to address the problem of unlabeled traces is to take a community-based approach. In [65], Gates *et al.* discuss requirements for evaluation data and propose a community-based approach for labeling released traffic traces. A similar proposal can be found in [138], where Ringberg *et al.* present Webclass, a tool to store and compare labels that have been assigned by different domain experts to a trace. And according to the MAWILab [105] homepage, the MAWILab people do also ask for help from

[29] As of August 2012, IPv6 addresses are anonymized by zeroing the last 80 bits [77].

[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 either their own results or their detectors. However, the collaborative system planed to simplify and automate contributions has not been implemented to date.

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 privacy critical 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 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 [100] - sometimes also called the Lincoln Lab data sets - are probably among the first synthetic datasets released to the research community. At first, it might therefore be surprising that e.g., the DARPA data set from 1998 still plays an role in intrusion detection research (e.g., [92, 146]). After all, it does not only lack recent attacks but has also been criticized for its traffic being unrealistic[33] (e.g., in [106]). Only when considering that better options are rare, it becomes clear why researchers still use it. According to [169] these data have provided significant contributions to the research on anomaly detection. Of 276 research studies published between 2000 and 2008, the LL data or its derivative, the KDD dataset, have been used in over 50% of these studies. Another 15% of the studies used additional attack data.

An alternative to ready-made traces offer tools to build custom traces by generating either synthetic background traffic or attack traffic or both. While there exist quite a number of tools to generate such traces at the packet level (e.g., [12, 110, 117, 151]), Harpoon [150] and FLAME [19] are the only notable tools that do the same at the flow level[34].

In [148, p.1], the authors of Harpoon write that

> ...Harpoon is a flow-level traffic generator. It uses a set of distributional parameters that can be automatically extracted from

[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.

> 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 tool of the two, FLAME, offers additional flexibility in that it can not only be used to generate flow traces from traffic models but also allows 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: Denial of Service (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 do also serve a second purpose: similar to [19, 110, 151], we use them to extract our models. However, our foucs is different from those of [110, 151]: 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 basically four possible outcomes when the detector decides whether or not the data under scrutiny contains an anomaly the detector is expected to 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 mistakingly reported to be anomalous and the FNR denotes the share of anomalyies missed by the detector.

Precision and recall are two related measures. Precision is defined as $\frac{\#TP}{\#TP+\#FP}$ which is the proportion of anomalies reported by the detector turnig 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 compared to the time spent on 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 some (sensitivity) parameter S, these measures are often compiled into Precision-Recall (PR) curves. These curves are optained by ploting 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 ROC curves [48, 162]. Instead of plotting precision versus recall, ROC curves plot the True Positive Rate (TPR) versus the FPR for different values of S. If the axis 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 [37] 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 neigbouring 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 estimate 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 [132], 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 analysis tool which is somewhat related[38] to ROC curves is the error digram. In [113] Molchan introduces the error diagram to measure 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. And 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 rather economical view at the system's performance: both, alarms - be it a true or a false alarm - and missing a true alarm costs money. Hence, minimizing both of the components of the error diagram is directly related to saving money. Furthermore, in [112], 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 costs of an alarm or a 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 [55]:** Detecor performance and therewith the ROC curves might vary significantly accross different datasets. This has two implications: (1) a direct comparison detection performance based on ROC curves should only be done if the ROC curves are derived from the same dataset and (2) to draw a more general conclusion about detector superiority, one shold use a series of representative data sets.
- **McHugh [106]:** Different detectors use different units of measure and/or follow different strategies to match detection results to the ground truth. Examples of different units of measure are, differnt sizes of data aggregation intervals or different bases for input metrics. One detector might e.g., log the number of packets seen per 5 minute interval while another doest the same for a 4 minute interval. Or one detector might calculate the entropy of source IP addresses based on the number of packets per source IP address but another does the same based on the number of flows. To avoid bias, the detectors should use the same units

[38] According to [147], the FPR and the alarm rate τ are of similar size, the ROC curve is almost a mirror image of the error diagram.

of measure. For the strategy to match the detection results to the ground truth[39], McHugh argues that it should be chosen with care.

- **Axelsson [8]:** 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 in case the costs for FNs and FPs are identical and fixed[40], it is far from being straight forward otherwise. Solutions mitigating this problem are e.g., cost-based modeling approaches [161], transformations of ROC curves facilitating cost-related comparison [2] or explicit computation of the expected costs for each detector operating point [63].

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 but 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 to reduce the amount of traffic data measured. Depending on the traffic data measured, e.g., packet traces or flow traces, and the purpose of the measurement, the sampling strategies vary significantly. A common sampling strategy to measure e.g., 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 such as smaller flow tables [2], less load 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 inherently a lossy process resulting in an incomplete and more importantly, biased approximation of the underlying traffic trace. If we apply e.g., one of the popular sampling strategies *random packet sampling* or *every n-th packet*, small flows consisting of a few packets only are likely to

[1] For the Protocol and Application Classification Enging from *ipoque*, this is e.g., 1-3 packets for unencrypted traffic and 1-20 packets for encrypted traffic [111].

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

be missed entirely. The more packets a flow contains, the higher is the chance that at least one of its packets is sampled. As a consequence, the bias of the approximation does largely depend on the underlying traffic mix.

As a consequence, sampling might also have a significant impact on the entropy or volume timeseries serving as input to many anomaly detection system for 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 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 to summarize 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 [33]). 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.

But, while being attractive because of efficiency and availability, sampling is inherently a lossy process, where many packets are discarded without inspection. Thus sampled traffic is an incomplete, and more importantly, a biased approximation of the underlying traffic trace, as small flows are likely to be missed entirely. Previous work has largely focused on analyzing this bias, devising better sampling strategies [31], and recovering statistics (moments and distribution) of the underlying traffic trace using inference [44, 46, 72].

[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.

It might therefore be surprising that sampled traffic views have been used for anomaly detection with considerable success (e.g., [89, 96]). Since this contradicts intuition, it is important to understand how this is possible. Is it e.g., because if an anomaly is large enough, 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 is the impact of the traffic mix that we see when no anomaly is present? This mix is likely to be quite different when we compare e.g., the characteristics reported from sensors placed on a link interconnecting two research networks with those reported from sensors placed on a link connectin a residential network to the Internet. In contrast to the second case, we might see little or no web traffic on the link betwee the research networks: 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 from and to 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, in particular, anomaly detection. Most publications related to the impact of packet sampling focused on different aspects. In [45], Duffield *et al.* study e.g., the problem of estimating flow distributions from sampled flow statistics. And in [50], 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 [18] are [102] and [101]. In [102], Mai *et al.* analyzed how packet sampling impacts three specific portscan detection methods, TRWSYN [86], TAPS [158] and entropy-based profiling method of [96, 190]. This work was extended to analyze the impact of other sampling schemes in [101]. 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 the more basic question: How does packet sampling impact on the *input data* to the detectors? If we look at the impact of sampling on the *input data* instead of the detectino 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 on the impact of traffic mix characteristics on anomaly detection on sampled views is [155]. Based on sampled views from routers of two large scale networks, the authors in-

spect 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 empirically evaluation of the impact of packet sampling and traffic characteristics on various volume and entropy metrics used by many detection methods [13, 20, 94, 96, 185]. 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. While this is basically good news, byte- and packet volume metrics are typically highly variable which makes it very difficult or impossible to use them to identify small- and medium scale events. This is even more true for anomalies impacting mainly flow counts such as distributed scans, or several forms of (distributed) denial of service attacks. To detect those 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 grossly impacted by packet sampling, entropy metrics are disturbed little. In particular, the Blaster worm in our data when measured in flow counts is dwarfed significantly by sampling but remains largely unaffected when looking at entropy metrics. Our findings support 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 collec-

tion 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 or even boosts 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) [163]. Since we have collected the traces 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 that the networks to which the border routers are attached are quite different - private peering with an international research network only, peering with

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

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 thoroughly 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 probability around 60%) or an IP address derived from the local IP (with probability 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 as well as it is a representative of the many anomalies visible in the number of flows per time bin, a metric that is biased significantly by packet sampling, but hardly in the other volume metrics. The Blaster worm is therefore an ideal candidate anomaly 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 and 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 do rarely exist since they are difficult to collect and hard to store consuming hundreds of gigabytes of storage space per hour.

As an alternative, we reconstruct packet traces from flow data. We claim that the reconstruction is fairly accurate if things such as the packet content or inter arrival times of packets are not of interest but rather 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 if most flows have a duration significantly smaller than T. In this case, the problem of distributing the packets of a flow to the correct time windows[5] is less relevant since the chance is high 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 and we can confirm that most flows last less than one minute (more than 99% of the total number of flows). Thus 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 (e.g., [51], [85]) constant throughput property of flows to reduce errors in case of splitting a flow across interval boundaries:

[5] with 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

```
1:  for all f in flowtrace do
2:      packetsize = ⌊f.bytes / f.packets⌋
3:      reminder = f.bytes − packetsize · f.packets
4:      packet = get_packet_from_flow(f);
5:      for i = 0 to f.packets do
6:          packet.time = get_random_time_in_interval(flow.start, flow.end);
7:          if i < reminder then
8:              packet.size++;
9:          end if
10:         write(packet);
11:     end for
12: end for
```

The additional byte for *reminder* packets is due to the fact that the number of bytes of a packet is an integer but the number of bytes in a flow divided by it's number of packets does not have to be.

Note that the constant throughput assumption is supported by [186] 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 [66] network with a sampling rate of 1 out of 1000 or the Abilene network (now part of the Internet2 network [1]) 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 10000

only in the second part of our study. The reason for this is that for the second part, we found that we needed higher sampling 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. Thus, 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 herewith constructed sampled traces, 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 it is done by e.g. NetFlow routers. Unfortunately, the process of how routers construct flows is not entirely deterministic[6], making reconstruction in this way problematic [118]. Thus, instead of trying to emulate a certain router behavior, for each time window, we simply aggregate all packets with the same five-tupple {source IP, destination IP, source port, destination port, protocol} into a single flow. While this might introduce some error, we believe that the chance that two hosts use the exactly same ports for two or more connections within several minutes is small. Operating systems do not reuse ports immediately but wait for some time before doing so. And even if it would be reused within a single time window, it is unsure whether the port is 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 $\rightarrow$ **Fcnt**
  - + number of packets $\rightarrow$ **Pcnt**
  - + number of bytes $\rightarrow$ **Bcnt**

[6] Well, actually it is. But it depends on many factors such as timeouts or the fill level of the flow table, which are impossible to simulate in retrospective wihtout having more information than just the flow level data.

  + 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 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 $P(X)$:

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

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 but 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:

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

In our context, if we calculate e.g., the Shannon entropy of a source IP address distribution, a_i would refer to the number of occurrences of IP address x_i.

Note that 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 referst 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 won't discuss the results for all of these metrics but focus on those revealing the most relevant or most surprising results. More specifically, when discussing results from the dataset around the Blaster worm, we are focusing on the TCP protocol and flows with direction IN. And 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 attack) vulnerable hosts and that there were almost no Blaster or Witty infected hosts inside of 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 can not be modeled accurately[7] rendering an accurate measurement of the disturbance caused by an anomaly impossible. To compensate 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 in our trace very well. Thus, we are able to construct an

[7] for reasons discussed in 2.2

"ideal" baseline by removing the traffic that constitutes to the anomaly.

Blaster baseline

To obtain the baseline for the Blaster worm, we removed the initial connection attempt only and left the remaining worm traffic in the trace. We did this for two reasons: First, removing the initial connection attempt can be done with rather simple rules and second, the share of the remaining worm traffic is less than $1‰$ of the total worm traffic. The rule to remove the initial scan packet is as simple as: remove all TCP packets with destination port 135 and packet sizes of 40, 44, or 48. We are aware of the fact that this heuristic does not only remove packets from connection attempts from the Blaster worm but also a share of packets of non-Blaster port scans to TCP port 135. However, before the blaster outbreak, this number was very low and not making this distinction seems reasonable. A detailed analysis of the Blaster worm with respect to the flow traces used in this study can be found in [38].

Witty baseline

To obtain the baseline for the Witty worm, we removed all UDP packets matching the following heuristic: 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 heuristics 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 some past time period. This is similar to what anomaly detection methods with a baseline model based on past behavior would do. Yet, our approach of removing the anomaly from the trace has two advantages: (1) it produces the "best-case" baseline model that any anomaly detection method could achieve, and (2) it is more general and is independent of the applied detection methods.

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

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 by calculating the absolute and relative distance or anomaly visibility between a sampled view x and the corresponding sampled baseline $\hat{x}$:

- 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 how much sampling boosts or attenuates this distance.

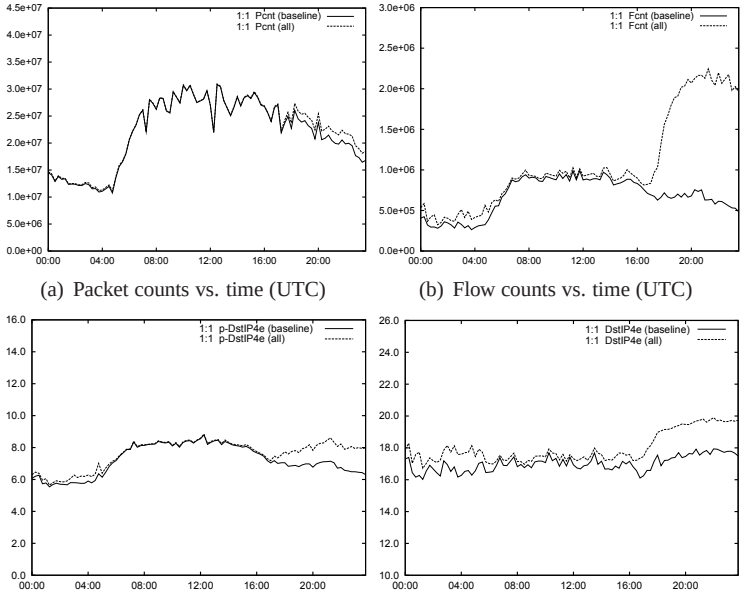
Note that while absolute distance $(x - \hat{x})$ is a result of adding the anomaly to the baseline, the distance is typically not equal to the value of the metric obtained for the anomalous traffic only. Such a direct relationship requires the metric to be an additive metric. In our set of metrics,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

3.3 Impact of Sampling on Entropy and Volume Metrics

As a first step in our analysis of the impact of sampling, we sampled our one-week data set around the Blaster worm outbreak at four different sampling rates of 1 out of 10, 100, 250 and 1000 and computed the timeseries of volume- and entropy metrics as described before.

To illustrate the following discussion on sampling effects, a selection of the most meaningful timeseries: packet counts, flow counts, packet destination IP address entropy, and flow destination IP address entropy is depicted in Figure 3.2 and Figure 3.1. 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 with a factor of $1/p$. Likewise, byte counts (not shown)



(a) Packet counts vs. time (UTC)

(b) Flow counts vs. time (UTC)

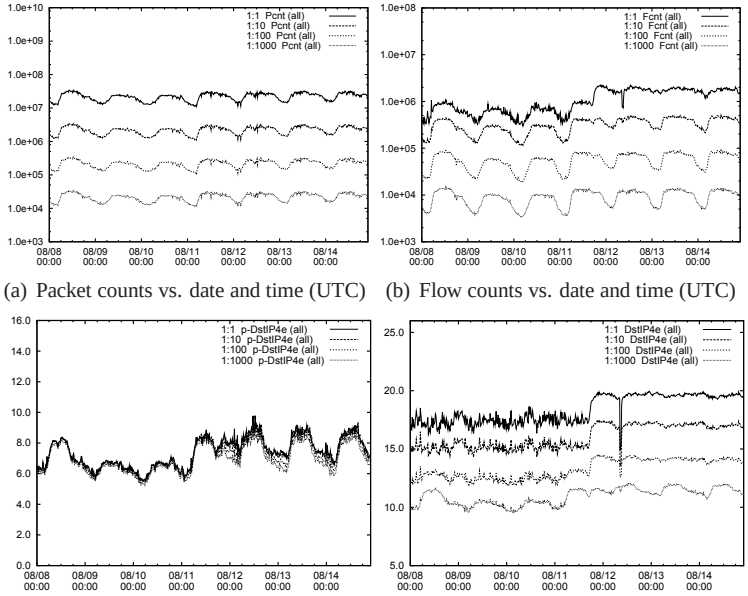
(c) Packet destination IP address entropy vs. time (UTC)

(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 both, the results for the traces 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 is depicted in Figure 3.1(a), packet counts do 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 (e.g., scan-traffic or other background radiation) and few packets imply a smaller probability to get sampled compared to larger flows [46].



(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 timeseries 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.*

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 main motivation for investigating the use of entropy metrics in more detail back then when I started my research.

3.3.1 Anomaly Size

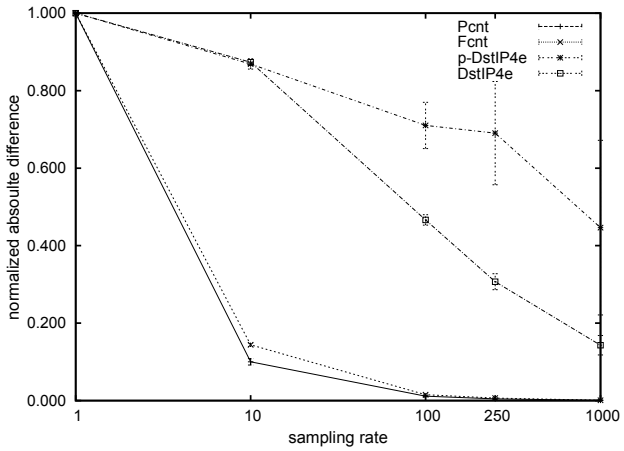
In Figure 3.3 we plot the sampling rate vs. the *absolute distance* (noralized to the respective value of each metric in the unsampled trace) as well as the sampling rate vs. the *relative distance* for packet counts, flow counts, flow

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 as a representative interval, the first interval after the Blaster outbreak around 17:00 UTC.

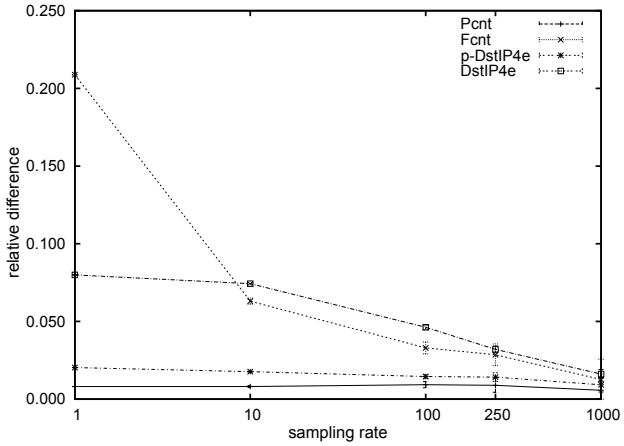
Let us consider the results for volume metrics first. For the flow count metrics the absolute as well as the relative distance decrease drastically when sampling is applied. Thus, we confirm the results of previous work, namely that flow counts, while exposing Blaster very well in the unsampled data, 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 better fit to capture 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 little 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 studied the effect of packet sampling on the Blaster anomaly as originally contained in our data, we evaluate how effective entropy is at capturing Blaster-like anomalies of varying intensities. Therefore, we use the given trace and attenuate or amplify the strength of the Blaster anomaly. Specifically, 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.

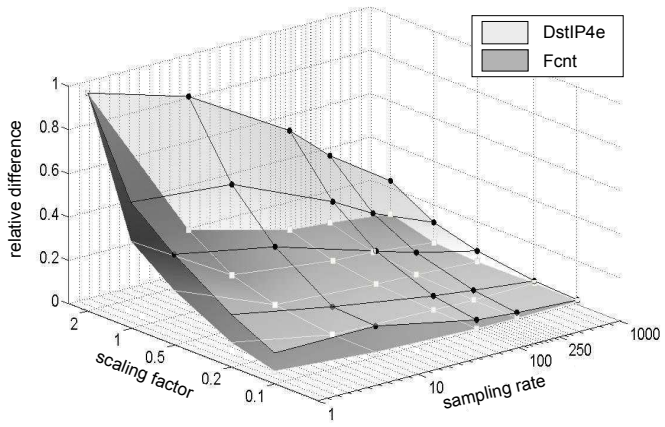


Figure 3.4: *Blaster flow trace: Relative distance from the baseline for flow counts and flow destination IP address entropy for flows with direction IN and protocol TCP across increasing sampling rates and different intensities.*

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 [8]. 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.

[8] 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.

As expected, the stronger the anomaly the larger is the relative difference for both metrics. But, 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.

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

Furthermore, while our study focuses on the Blaster worm anomaly, we argue that this result is not specific to the Blaster worm anomaly but to 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 e.g., 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 it is not met, 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[9]. 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 in the game. In contrast, there is no such indirect impact in the case of the flow count metric: If a flow is not sampled, it does no longer contribute to this metric.

Fortunately, if we look at the definition and the models of e.g., the anomalies discussed and referenced in Section 2.2, property (1) and (2) are expected to be met by e.g., Distributed Denial of Service Attacks (DDoS), Reflector Distributed Denial of Service Attacks (Refl.DDoS) or various types of (large-scale) scanning. Furthermore, successful applications of entropy based anomaly detection in combination with sampled traces (e.g., in [96]) with many different types of anomalies support our claim.

Having said this, it is clear that while these properties should be met, it does not only depend on the anomalous traffic but also on the baseline traffic if and how well an anomaly is visible with respect to our set of metrics. The next section discusses this aspect in more detail.

[9] 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. Thus we need to investigate this in more detail.

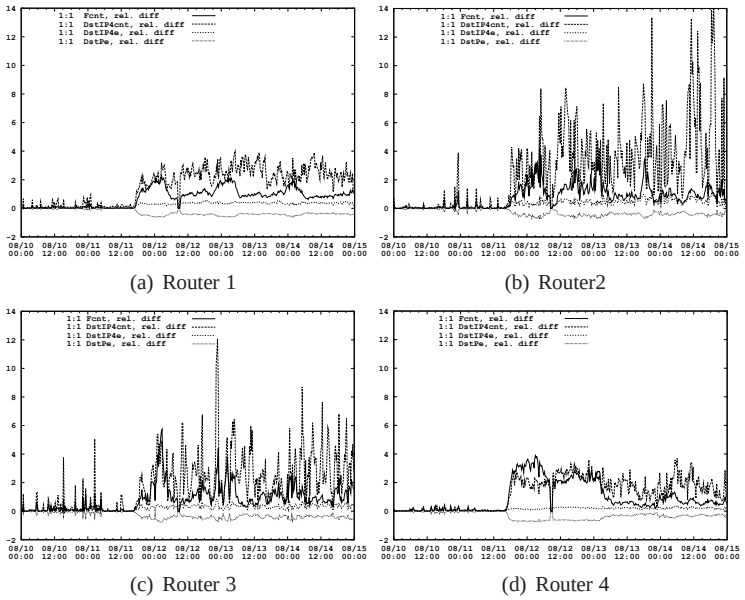
Since the traffic from the four routers from which we get our flow data appears to have quite differing characteristics, especially when comparing traffic from routers 2 and 3 to routers one and four, we looked at the differences in the exposure of the Witty and Blaster worm for each of these routers separately. Details on the characteristics of the traffic of the four routers can be found in our technical report [171].

Our results might be counter-intuitive at first. Intuitively, packet sampling reduces the amount of information contained in our traces. Hence, 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 inverse 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. To illustrate these, i.e., the impact of the traffic mix on the anomaly visibility, 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 1:10000. Furthermore, the basic structure of the relative distance curves for the unsampled traffic is very well preserved by the curves for the sampled traffic. But if we compare the flow destination IP address entropy metric (on the right side), we can see that the impact of



(a) Router 1 (b) Router2

(c) Router 3 (d) Router 4

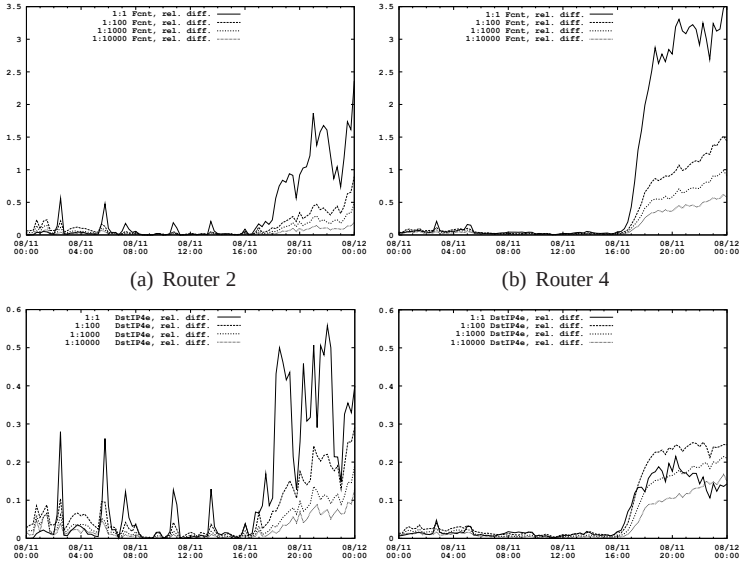
Figure 3.5: *Relative difference of flow count, unique destination IP address count, flow destination IP address entropy and flow destination port entropy metric on our four border routers vs. date and time (UTC). No sampling.*

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.

3.4.2 Discussion

The increase in visibility for sampled traffic is significant and might be counter-intuitive at first. In particular, the fact that the increase is big enough to preserve the visibility of the Blaster (Figure 3.6(d), Router 4) and Witty (Fig-



(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.*

ure 3.7(b) and 3.7(d)) worm up to sampling rate of 1 out of 10000.

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

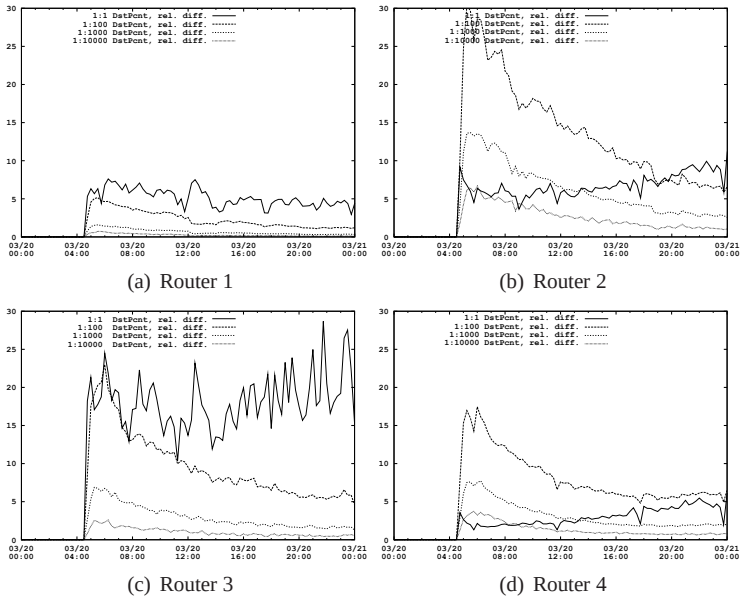
$$\frac{(x_p - \hat{x}_p)/\hat{x}_p}{(x - \hat{x})/\hat{x}} > 1 \tag{3.3}$$

for sample probability p?

To understand this effect, we first need to discuss the characteristics of the baseline- and anomalous traffic that determine the impact of sampling and how they do it.

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[10]. Depending on

[10] such as e.g., a specific source port number



(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 metric under scrutiny, only one of them or both matter:

Flow count: 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, it is sufficient to analyze the distribution of the number of packets per flow in the baseline traffic and in the anomalous traffic. For the 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 traffic, the baseline is expected to be less affected. And in the opposite case, the same is true for the anomalous traffic.

If the lines cross each other at least once, it depends 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 like "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 that involve flows with one packet per flow only. Or we observe anomalies such as Flash Crowd or application-layer DDoS attacks with the number of packets per flow in a specific (and narrow) range. In the first case, we can expect the anomaly visibility to decrease or remain the same[11]. 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 generic problem with sampling since it is theoretically possible that if you are really unlucky, you might not see an anomaly until it consist 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. At least when using one of the popular sampling strategies *uniform random packet sampling* or *every n-th packet*. If it can be avoided, sampling should therefore not be used in security related applications. Unfortunately, this is not always the case as the broad application of sampling in high-speed networks suggest.

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 since 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.

[11] If the baseline too consists mainly of one packet flows.

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 e.g., 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. Hence, if there are e.g., 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 get again smaller than the number of x_i removed from the anomalous traffic. Since todays "normal" traffic contains e.g., 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 do a certain sampling rate.
- The x_i in the anomalous traffic and the baseline traffic are identical. A consequence of this is that 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 n the full traffic traces: the x_i in the full traffic trace have more packets per x_i. Furthermore, anomaly visibility increases as long as the number of
- 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 relationship to the x_i in the baseline traffic. Since 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 depends now on the combined impact on the separate portions of the traffic. Maybe the visibility would 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 it.

Entropy: For entropy metrics, it is not only important how many pack-

ets originate from the baseline and from the anomaly but also how they are distributed into flows. If e.g. 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. And in the opposite case it would be the other way round.

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 with 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 to roughly 3000 ports per 15 minutes bin if we consider that Witty used random destinatino ports for its attack. For router 1 it should be around 11000 ports, for router 2 around 6000 ports and for router 4 around 16000 ports. We could 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 with the set of ports contributed by the baseline. Hence, 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 visibilty for the port count metric. By looking at the distribution of the number of packets per port with and without the anomaly, we found e.g., 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 there 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 visibilty for sampling rates of up to one out of 100 packets shown in Figure 3.6 is harder to explain. This time, it is 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 is 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 contanis many IP addresses which are hit by more than one attack source. Hence, the increase can 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 - to evaluate the fidelity of sampled traffic in exposing anomalies. Our first finding is somewhat 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 show in previous work, small (single packet) flows are entirely missed. Thus, anomalies that only impact 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 then 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 it is 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 find that the visibility of certain metrics, e.g., flow destination IP entropy, is even more pronounced by packet sampling up to sampling rates of one out of 10’000, and elaborate on possible root causes. In retrospective, this was certainly a surprising finding, even though possible explanations are manifold. However, since in order to observe this effect,

both the anomaly and the baseline traffic must be shaped accordingly, we do not expect it to be relevant to a larger number of anomalies and (real-world) baseline traffic mixes. Nevertheless, it shows us that we must be carefull with statements saying that sampling is inherently a 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 one or multiple of these features are largely redundant. In contrast to the analysis in [121], 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 reason for the different findings is the different approaches used to build the traffic feature distributions: Nychis et al. measures the number of packets per feature instance while 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 information

[1] In practice, packet-based approaches are hard to find. Most works use a flow-based approach [17, 95, 96, 174, 185]

content. Any input that does not contribute toward a better detection performance should not be fed to the detector in order to optimize resource usage and and 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 redundant information only. From these approaches, the first one 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 especially 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 series of observations where most observations are not anomalous[2], we find those components relevant to capturing and describing the current state of the communication network. The drawback of this approach is that not all of those components relevant to the current state must also be relevant to detecting an anomalous state. Some might never be affected by anomalies. But since we can not 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.* [121] 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, e.g., traffic volume, would already provide enough information for reliably detecting 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 per packet entropy. To aid detection and especially classification of network

[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.

anomalies, we therefore suggest to use 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 [59, 90, 194, 195].

Concerning the last two features, we do not know of any (entropy based) anomaly detection system making use of them. For now, we just state that these features may expose certain anomalies that might otherwise just vanish in 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: First, 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. Or more specifically, if we take e.g. 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 timeseries. 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 timeseries X and Y consisting of n data points is the Pearson product-moment correlation r, as used by [121]. The Pearson correlation coefficient r_{xy} is defined as

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

where $\bar{x}$ and $\bar{y}$ 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 4.2 from the SWITCH backbone.

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.

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

4.3 Results

The absolute value of the Spearman coefficients in percent are presented in the tables 4.1, 4.3, and 4.4. A value of 100 denotes maximum correlation where on the other hand 0 means no correlation.

Table 4.1 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 some traces, but low in general. This is, for instance, the case for (Sip, Dip) with 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 81 average correlation. All other pairs have an average correlation of less than 80. Suming up, for almost all feature pairs, there is no fixed correlation value that can be attributed to them. If we look at the minimum and maximum of the correlation values for each pair, we see that they span quit a large range of values.

Table 4.3 and 4.4 show correlation statistics for traces 10a-d, studying 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, again correlation is in general very low, the only exception being (Sp, Dp) with correlations between 96 and 98. Surprisingly, the three most correlated pairs from table 4.1 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. Or in other words, in general, we can neither assume that a specific feature pair is correlated nor that it is not correlated. For UDP, there is quite a number of pairs with high maximum correlations. However, it is usually not stable over

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

ID	Description	Start	Days	75p Fcnt	
				TCP	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 4.2: *Overview of traces used. To indicate the size of traces, we list the 75-percentile (75p) of flow counts computed in 5-minute windows.*

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 the nine traces before: For almost all feature pairs, there is no fixed correlation value that can be attributed to them.

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, but 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 in percent. 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 in percent. 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.* [121]. In our results, the pairwise correlations tend to be weaker but also quite variable. We think 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 [17, 95, 96, 174, 185] 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 from Nychis *et al.* 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 be solely 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.* [121] regarding a persisten 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 persistent strong correlation between traffic feature entropies. Our analysis did not expose *strong pariwaise 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: either flow- or packet based. 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 packe-based method. But it also varies quite signifficantly 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 loose 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 (TES)

In the previous two chapters, we presented and discussed empirical evidence for the first two claims made by this thesis: First, entropy is robust to packet sampling techniques often used with measurement infrastructures in high speed networks. And second, 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 parametrized form of entropy; the Tsallis entropy. After introducing the concept and the properties of the TES, we demonstrate its descriptive power using traffic data from different (massive) 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 e.g., traffic volume, connection count or packet count [13]; others look at changes in traffic feature distributions [141] 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 [43]. However, content inspection or storing state information on a per host basis are usually limited to small- and medium-scale networks. If feasible at all, the link speeds and traffic volumes in large-scale networks hinder a reasonable return on investment from such methods.

Most approaches designed for large-scale networks have therefore two things in common: First, they reduce the amount of input data by looking at flow-level information only (e.g., Cisco NetFlow [33] or IPFIX [75]). Second, 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 those 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. The form that is probably the most popular one is the Shannon entropy analysis. 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 [96, 99, 185].

However, the good detection and partially even classification performance of these systems is mainly reached with regard to massive anomalies only. Furthermore, there is some empirical evidence that parametrized forms of entropy such as the Tsallis entropy might be superior to Shannon entropy. In [196], Ziviani *et al.* [196] study for which value of the parameter q of the Tsallis entropy Distributed Denial of Service attacks are detected best. And in [143], 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. Thus, a generalization of these preliminary results to arbitrary types of anomalies as well as appropriate detection and classification systems are yet missing.

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

- We define the *Traffic Entropy Spectrum (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 and classification as well as for visualization of their characteristics.
- We confirm the finding of [143, 196] for a broader set of anomalies.

Furthermore, we propose to add Autonomous System (AS) entropy to the set of commonly used traffic features and 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 Shannon entropy. Next, we introduce the Traffic Entropy Spectrum (TES) and explain how it is used to capture and visualize distributional changes. Section 5.4 we propose 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 evaluatoin before we present our results in Section 5.7. Finally, we conclude this chapter with Section 5.8.

5.2 Shannon and Tsallis Entropy

The Shannon entropy [144]

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

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 on the distribution, it is clear that using other generalized entropies may reveal different aspects of the data. Two of such generalized entropies relying on *moments* different from the *log-moment* are the Renyi and Tsallis entropies, the latter being an expansion of the former. A comprehensive introduction to entropy in general and Tsallis entropy specifically can be found in Constantino Tsallis book *Introduction to Nonextensive Statistical Mechanics: Approaching a Complex World* [180].

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

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

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

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 Autonomous System (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 as a metric measuring whether a distribution is concentrated or dispersed. The main difference to approaches using Shannon entropy in the same manner is that Tsallis entropy allows to concentrate on different regions of the distribution.

When using the Tsallis entropy, there is not one Tsallis entropy but as many as there are possible choices for q. Each q reveals different aspects of distributions used to characterize the system under study.

But what is the role of the parameter q? 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's consider, the following example: In a time window of size T we observed that IP address A was the source of 1000 and IP address B the source of 10 connections. And 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 q's 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 no or low 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 affecting parts of the distribution only 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 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 qs to high activity elements for $q > 1$. Figure5.1 shows a sample of such a Traffic Entropy Spectrum.

5.3.1 Selection of the q-Vector

But 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 years (2003 to 2008) showing largely differing traffic characteristics, we found that the selection $q = -2, -1.75, ..., 1.75, 2$ gives sufficient information to detect network anomalies in all of those traces. Large values $q > 2$ or smaller values $q < -2$ did not provide notable gains. We consider this finding a strong empirical evidence that the parameters of the TES require little or no tuning to different traffic characteristics.

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

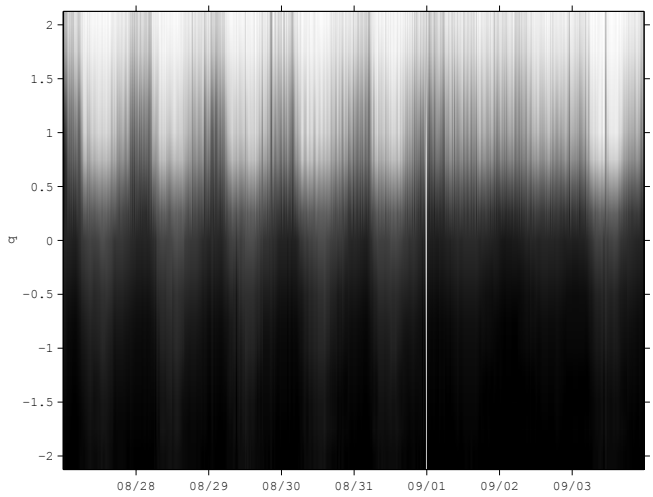


Figure 5.1: *Example of a Traffic Entropy Spectrum (TES): On the y-axis, we see the set of q-values for which the Tsallis entropy values are plotted as colored rectangles versus the time on the x-axis. Hence, a rectangle* (x,y,c) *color-encodes the (normalized) Tsallis entropy for* $q = y$ *and time* $T = x$*. This TES is the TES of the autonomous system activity in the incoming traffic around a DDoS attack in 2007. The color scale used ranges from black for the minimum* S_q *to white for the maximum* S_q*.*

we identify exactly three different regions. Each region contains elements that show either *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 # of elements decreases entropy for $q > 1$
- medium activity: reducing the # of elements decreases entropy for $-1 < q < 1$
- low activity: reducing the activity of some elements increases entropy for $q < -1$

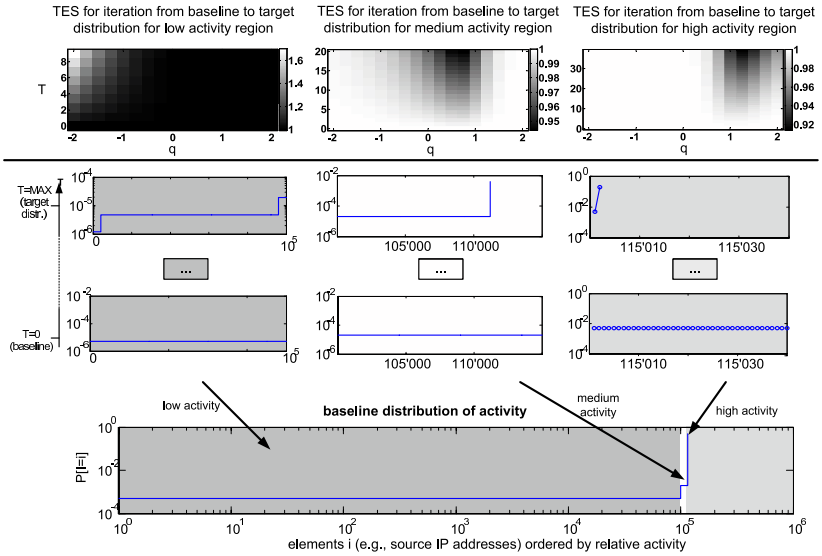


Figure 5.2: *Impact of changes to different regions of the distribution. Bottom: Baseline distribution. Center: Visualization of the baseline distribution (at $T = 0$) which is then iteratively transformed into the distribution shown at $T = MAX$ for low, medium and high activity regions. We call the 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, even sized, steps.*

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 and minimum entropy value for a given q during a training period as follows $S_{normalized,q} = \frac{S_q - minS_q}{maxS_q - minS_q}$. This maps all entropy values to the range [0,1]. Figure 5.1 shows an example of a TES ploted using this method.
- Normalization using the maximum and minimum entropy for a given q during a training period, for instance just from 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 ploted 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 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 the anomalous parts of the TES only. Even though this detection procedure is very straight-forward, 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's from two different time intervals, would imply a change of the activity pattern in the respective region. For instance, a significant change of the S_q's 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 understand 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, e.g., 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[1] in the same region must be decreased by Δa_i such that the total activity $\sum_{j=1}^{n} a_j$ remains constant.

But what happens now, 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's assume that we want to detect changes in the activity of TCP port numbers in the interval T_n with those in the next interval T_{n+1}. Let's 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.

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

Note that the ports are sorted according to their activity. Port 80 is the port with the highest activity in both intervals.

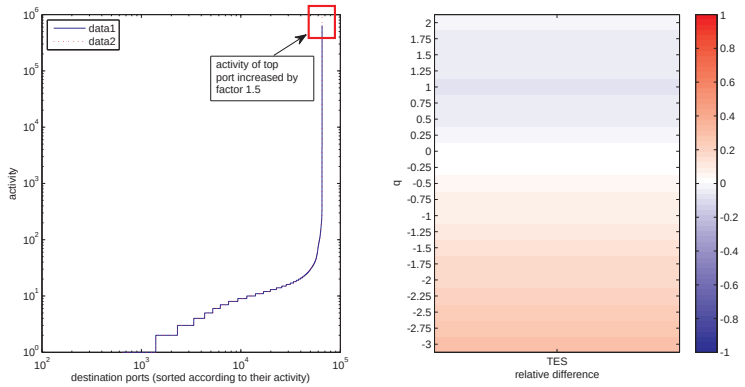


Figure 5.3: *Left: Two activity distributions differing only in the activity of the most active element: in the modified distribution, it's activity is 1.5 times than in the original distribution. The original distribution corresponds to the activity distribution of the destination ports of the TCP traffic flowing into the SWITCH network in the time from 09:45 to 10:00 on the 31st of July 2008. Right: Relative difference of the TES of the original distribution and the modified distribution.*

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:

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

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 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 modified only the acitvity 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 e.g., the entropy for negative q-values decreases compared to the entropy in the first interval.

An important insight of 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 minimum change required to start considering a q-entropy to be abnormal. The problem with this is that depending on the actual activity distributions, this might result in bounds for the minimum change that are e.g., dominated by the overall changes in one activity region only. If most changes to the overall activity are e.g., due to changes in the high activity region, the minimum change required to start considering a Tsallis entropy value to be abnormal might reflect reality quite well if $q \geq 1$ but it would be far to pessimistic for others 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 rather unlikely that e.g., the decrease in activity of some elements is compensated by an increase in activity of other elements in the same region. If we take e.g., an anomaly as an example, a typical impact of it is that the activity of a single element (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 the change in the overall activity since 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 the unwanted normalization effects by computing a *pruned* entropy in a two-step approach. For each time interval, we start with calculating the TES consisting of the entropy values S_q for a set of q-values. We then zoom in on the most contributing elements responsible for p percent 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 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

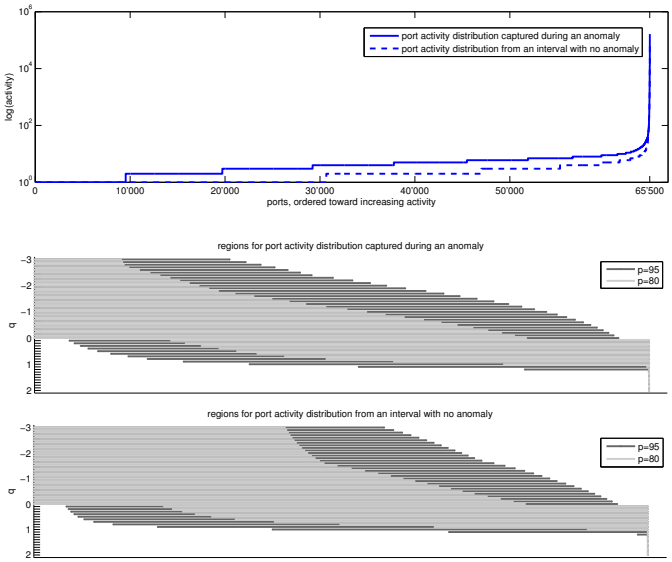


Figure 5.4: *Destination port activity distributions (top) and selected regions for TES_p (bottom). On the x-axis all ports are ordered by rank, i.e., with increasing activity to the right.*

plotted on the y-axis (i) during an anomaly and during normal activity.

For both distributions there is one plot below, showing the selected elements for different values of q and p. At a specific coordinate (x,q) there is a point if element x was selected for the pruned entropy $S_{q,p}$ and no point otherwise. 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 the different activity levels more tightly at the cost of being probably too tight: $q = -3, p = 80$ does e.g., not 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, lets turn back to the example used when we introduced

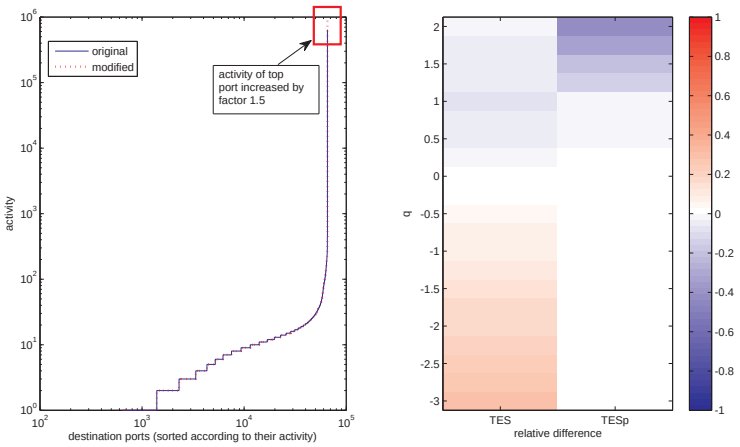


Figure 5.5: *Left: Two activity distributions differing only in the activity of the most active element: in the modified distribution, it's activity is 1.5 times than in the original distribution. The original distribution corresponds to the activity distribution of the destination ports of the TCP traffic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

the inter-region dependency problem. But 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.

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, e.g., 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, 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.

For 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_q’s 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 integrate the TES in 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’s[2] 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 occurance in our traces is known. Since we started out with the original TES and refined it only later, the results published in [174] do still contain significant inter-region dependencies. To

[2] or more precisely, the anomaly scores related to the S_q values

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 of the following set of traffic feature distributions:

- source IP address (SrcIP) and destination IP address (DstIP)
- source port number (SrcPort) and destination port number (DstPort)
- autonomous system number (AS)

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[3] protocols.

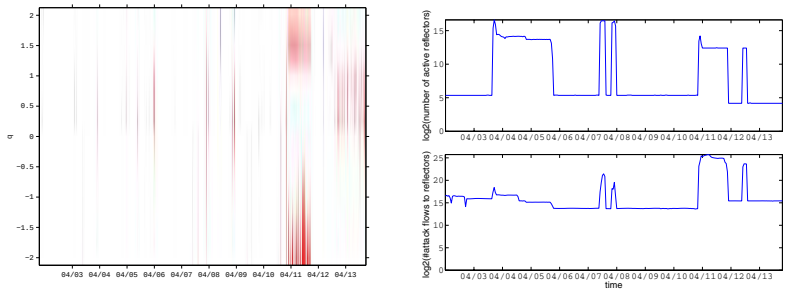
5.6.2 Set of Anomalies

Our evaluation focuses on the following set of well-known anomalies:[4]

- **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 min.) 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 Autonomous System numbers. The attack is nicely 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.

[3] includes traffic for all protocols except TCP, UDP and ICMP.

[4] They are well-known either because we or other researchers 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.*

- **DDoS 2:** A long (13h) 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).

5.6.3 Compiling the TES

Since we know the characteristics and time of occurance of the aforementioned anomalies, compiling the TES for the corresponding portions of our network trace archiv[5] is all we need to do.

For this purpose, we extended our netlow processing framework[6] with an module that compiles the TES (and also the TES_p) in a two step approach: First, the module reads incoming flows, determines the interval to which they belong in order to update the traffic feature distributions of this interval. In a second step, if no more flows are expected to arrive for an interval, it calcu-

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

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

late 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 human readable form. 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.

Another feature that we make use of in our evaluation is that it can not only output the TES but also the actual activity distributions. Either in binary (to save space) or human readable form.

5.6.4 Data Analysis

To get now 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: First, we browsed through the more than 600 TES [7] to check whether we can spot the anomalies simply by looking at the (normalized) TES. In a next step, we 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 characteritics of our anomalies.

To speed up the process of browsing through the output of our netlow 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[8] and controls for the selection and configuration of different normalization methods (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 autonomous system traffic feature. The plot at the bottom displays the corresponding TES. More details on this tool can be found in Section A.2.3.

[7] 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

[8] flow exporter {ALL, router 1 to 4} $\rightarrow$ protocol {TCP, UDP, ICMP, OTHER} $\rightarrow$ direction of the traffic {IN, OUT} $\rightarrow$ traffic feature

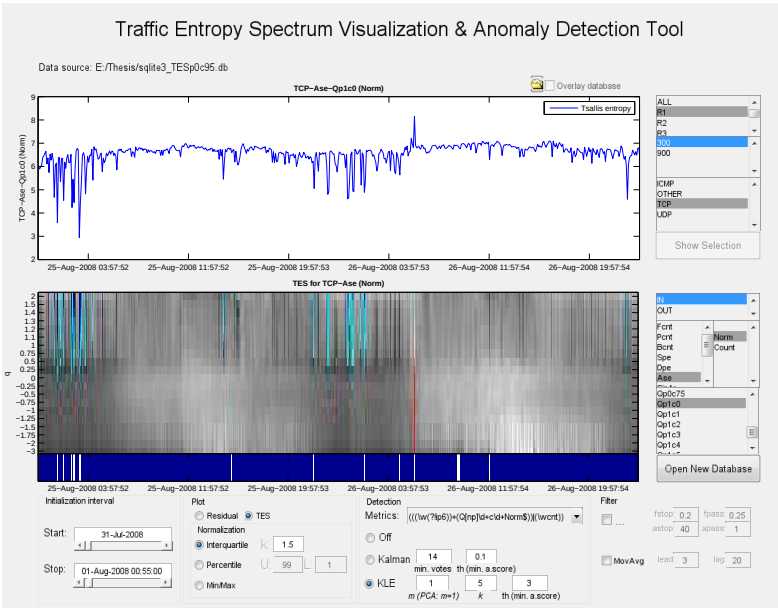


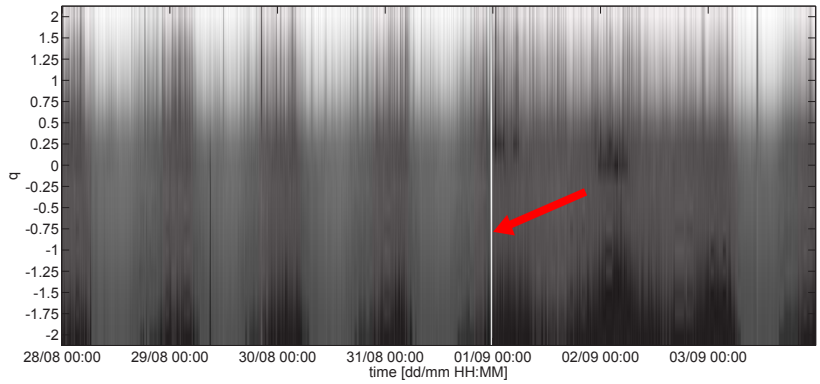
Figure 5.7: *Screenshot of the Traffic Entropy Spectrum Visualization & Anomaly Detection Tool. The plot at the top displays time series data of the selected time series: the Tsallis entropy for $q = 1.0$ for the autonomous system traffic feature. The plot at the bottom displays the corresponding TES.*

5.7 Results

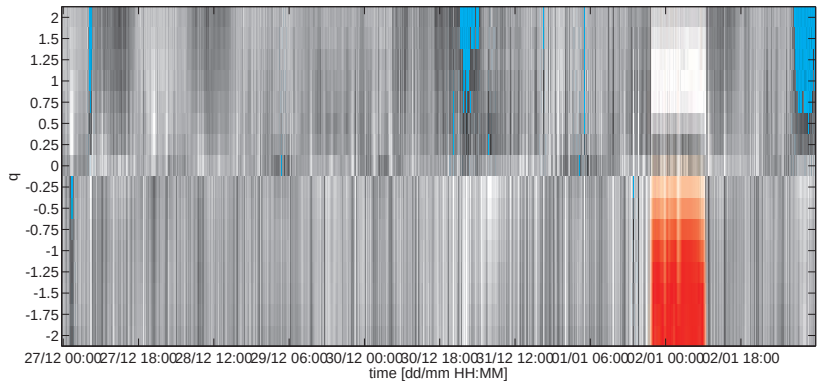
5.7.1 Spoting the anomalies

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 did also check 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 point out anomalies that last only tens of seconds or a few minutes.

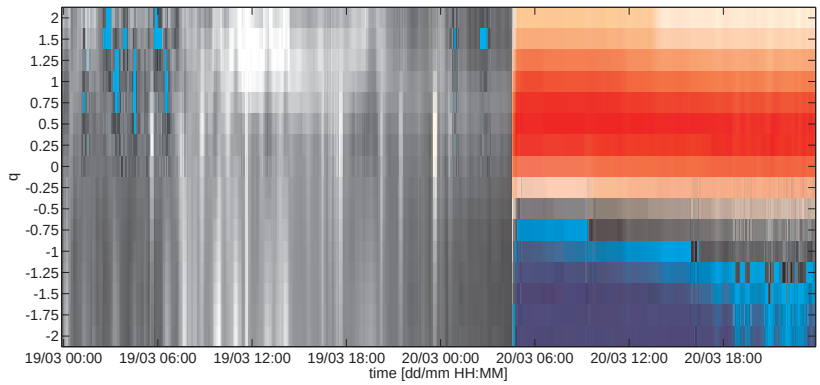


(a) TES of origin Autonomous Systems in the incoming traffic during the DDoS 1 attack represented with global normalization

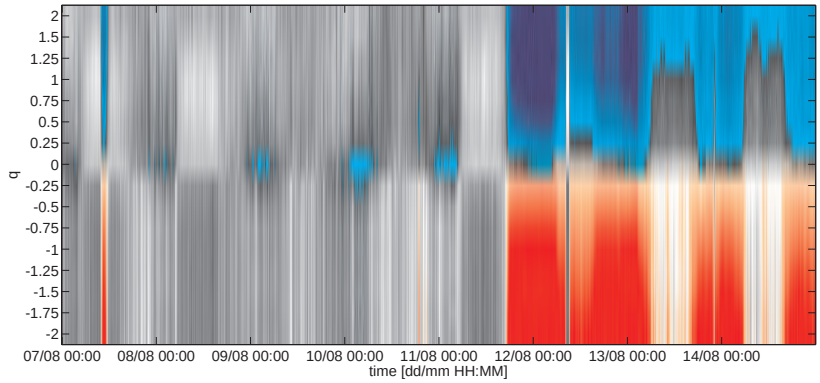


(b) TES of SrcPort numbers for flows into our network during the DDoS 2 attack. The TES is normalized using the inter-quartile range approach. Areas in red (resp. blue) represent locations where the TES is above (resp. below) the threshold.)

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 previously. The following table shows the spectrum patterns for all of the five anomalies, five traffic features[9] and both for 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
	IN	TES	+	+	0	-	-	+	0	0	0	+	-	-	0	+	0	+	+	0	-	-
		TES_p	0	0	0	-	-	0	0	0	+	+	+	-	0	+	0	0	0	0	-	-
Refl. DDoS	OUT	TES	+	0	0	+	0	+	+	0	-	-	+	+	0	-	-	-	-	0	+	0
		TES_p	0	0	0	+	+	0	0	0	-	-	0	0	0	-	-	+	-	0	+	
	IN	TES	+	+	+	+	0	+	+	0	-	-	+	+	0	-	-	0	0	0	-	-
		TES_p	+	+	+	+	0	0	0	0	-	-	0	0	0	-	+	0	0	0	-	-
DDoS 1	OUT	TES	+	+	0	-	-	+	+	+	+		0	0	0	-	-	+	+	0	-	0
		TES_p	0	0	0	-	-	+	+	+	+		0	0	0	0	-	0	0	0	-	+
	IN	TES	+	+	+	+	0	+	+	0	-	-	+	+	0	+	+	+	+	0	-	-
		TES_p	+	+	+	+	0	0	0	0	-	-	-	-	0	+	+	0	0	0	-	-
DDoS 2	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	-	-
	IN	TES	+	+	+	-	0	+	+	+	+	0	+	+	0	-	0	+	+	-	-	-
		TES_p	+	+	+	-	0	+	+	+	+	0	+	+	0	0	0	0	0	0	-	-
Blaster W.	OUT	TES	+	+	0	0	0	+	+	+	+	0	+	+	0	-	0	+	+	0	-	-
		TES_p	+	+	0	0	0	+	+	+	+	0	0	0	0	-	+	0	0	0	-	-
	IN	TES	0	0	0	-	-	+	+	+	+	0	+	+	0	-	-	+	+	+	+	0
		TES_p	0	0	0	-	-	+	+	+	+	0	0	0	0	-	-	+	+	+	+	0
Witty W.	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. A `-` sign and the color *blue* denote a significant decrease, no color and the A `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. We rather 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 (web) servers used as reflectors in the Refl. DDoS attack appear in the incoming destination IP addresses (as requests from the real at-

[9]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.

tackers). 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 then 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 does also seem to report an increase in the destination IP address entropy for negative qs. However, the Refl. DDoS attack did hardly contribute 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 the total activity. Hence, the inter-region dependency [10] then 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 does report 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 refelctors 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 victime into one of the most active IP addresses [11] seen in the outgoing traffic. Consequently, the high-activity region becomes more concentrated which 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 entroy in the low activity region. Note that the TES_p does once more not suffer from this problem.

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

In contrast, the spectrum pattern 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 dominates now the distribution of the rare ports which leads to an increase in the

[10] Remember that the activities of the elements is 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

[11] Most of he 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: There, a few source ports which were significantly more often seen in the attack traffic, caused the upper medium activity region to become more uniform too.

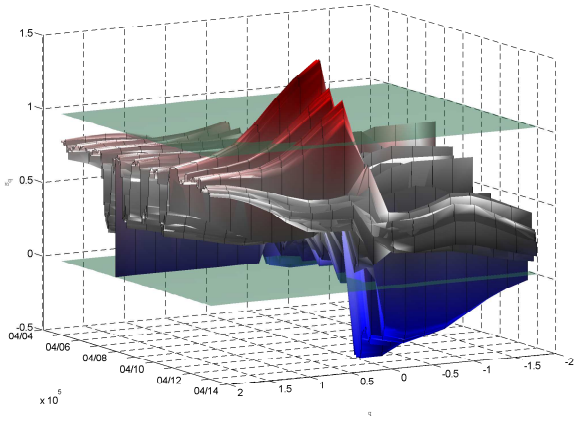


Figure 5.10: *3D TES for incoming SrcPorts before and during refl. DDoS attack for $q = -2...2$. Diagonal axis: date (10 days), vertical axis: normalized entropy. Transparent layers: MIN and MAX at normal week days*

In case of the TES, the modification to the low activity region is again hidden by the inter-region dependency: The increase in the 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 beteen the TES and the TES_p. The reason for this is that in this case, the

increase in the total activity is not e.g., mainly attributed to a the medium or high activity region, but also to the low activity region: a lot of IP addresses with just a few occurances 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. And the low activity region becomes more concentrated since it contains less 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 less autonomous systems than before and after. One possible explanation for this is that during this time, most people spent their time with something different than sitting on a computer and browsing the web.

Blaster and Witty: For both, the Blaster and the Witty worm, destination addresses of spreading attack traffic were generated randomly, much the same way as sources were spoofed during the DDoS attacks. And 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

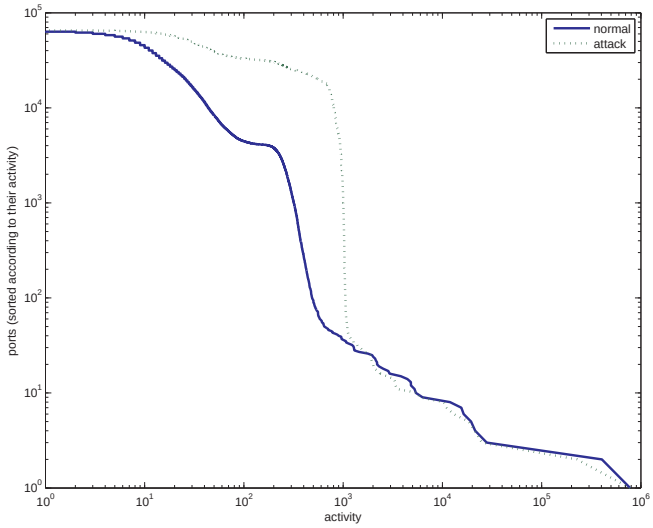


Figure 5.11: *Log-log plot of the source port activity distribution before and during the DDoS 2 attack. The source ports, sorted according to their activity, are on the y-axis.*

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 shows 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 65535 values. Thus, if intensive random port scanning is performed, all ports are often revisited and become frequent, basically replacing the low activity area. This is what happened in the refl. DDoS case, indeed. 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, e.g., the distribution of flow sizes and durations. The final 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 come along 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 from 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.

- 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 can not yet be identified, filter flows that do not match the identified values or patterns to get the candidate flows.

Analyze the candidate flows using frequent itemset mining.

We apply the same procedure for all of the drill-down work done in the context of our thesis.

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 occuring on the 26th of August 2008 around 06:00. Using our *Traffic Entropy Spectrum Visualization & Anomaly Detection Tool*, we browsed thorugh 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 this 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 autonomous system traffic feature shed some light on whether or not the anomaly involves traffic from many or just a few autonomous systems. A very pronounced increase of the entropies in the low- and medium-activity region hinted at an attack involving quite a lot of autonomous systems 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 someone might spoof the source IP addresses used.
- **Source IP address:** Here, we see a similar impact on the TES_p as was the case with the TES_p for the autonomous systems 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 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. There, we saw an increase in the entropies of the medium-activity region. This caused us to believe that the attack might probably not involve random source IP addresses but as we will see later, 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 e.g., the changes in the TES_p for the autonomous systems 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 just of 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(s) of the anomaly is not yet clear. To shed some light on this question, we looked at the flow count metric and saw that the number of incoming flows went from 1.3 in the preceeding 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 1.3 million flows only. 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 makes it still difficult to spot an increase of around 500'000 flows by just 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 not for the strange behavior of the TES_p for the source port traffic feature, our first guess would be a DDoS anomaly whith 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 request. 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 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 just had to check whether our guess is 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 1024 and 3072. 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 autonomous systems responsible for less than 10 flows increased significantly from rougly 4000 to 8000. Furthermore, the number of flows from the top autonomous systems 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 occuring in less than 10 flows. From a total of 100'000 IP addresses matching this 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 [12]. Furthermore, the increase in the number of flows to the top IP address

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

matched the increase expected from the increase in the flow count metric.

- **Country code:** A closer look at the distribution of the flows with respect to their country of origin, did not reveal anything that clearly stands out. The only thing which is a bit different it the total number of countries represented in this distribution: 218. This is typically a bit lower (around 210). But not finding anything that stands out is also a finding: It tells us that the flows contributing to the anomaly might nevertheless come from spoofed IP addresses since if they were e.g., originating from a botnet, the change in the country distribution should reflect the distribution of the bots with respect to countries. And 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.

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

- Flow size: 48 bytes
- Source ports: 1024 and 3072
- 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 [120] describing this pattern and relating it to the DDoS attack tool *juno* whose code can be found here [87]. The reason for the strange behavior with respect to ports is that the code selecting the source port is broken. At least we assume this since the code makes use of the *random()* function but finally results in

just either port 1024 or 3072. Since this tool is indeed generating flows with the characteristics identified and no others, we can be almost sure[13] that we 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 6501 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 1433 is typically used for remote access to Microsoft SQL Servers for which several remot-exploitable vulnerabilities have been documented. The massive scan could e.g. have been triggered by some malware such as the Gaobot family[14] 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 as well as the count metrics 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 the the number of flows involved in this anomaly was simply too small. However, we can not prove this assumption.

[13] There is always the possibility that someone wanted the attack to look like it is generated by the *juno* tool. However, we have no chance to tell this based on flow data only

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

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 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 does also expose a weakness of the TES: the inter-region dependency. We addresse this problem with a modified version of the TES, the TES_p, and show that the TES_p captures the changes in a more convenient way. Furthermore, we demonstrate that we can capture changes introduced by different types of anomalies using just a few Tsallis entropy values and find that 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 speific. 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 our claim that generalized entropy is an accurate tool to characterize anomalous changes in traffic feature distributions of high-speed and large-scale networks extracted at the network flow level holds. 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 integrating the TES in a fully automated way.

In this chapter, we propose a comprehensive anomaly detection and classification system called the entropy telescope and 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.

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

6.1 Introduction

The attractiveness of entropy metrics stems from their capability of condensing an entire feature distribution into a single number and at the same time retaining important information about the overall state of the distribution. Thus, it is possible to detect concentration and dispersion of feature distributions typical for certain types of attacks, e.g., DDoS attacks or worm outbreaks.

Compared to merely detecting an anomalous state, it is significantly harder to *classify* an ongoing anomaly and identify its root cause. Attempts of combining changes in multiple features to establish anomaly patterns are very promising (e.g., [96]), but the accurate automatic classification of anomalies is still a major challenge, especially if anomaly sizes and affected host populations vary. The TES method introduced in the previous chapter allows to focus on specific areas of distributions; for instance on the area of heavy-hitters or rare elements. By doing this, it retains the advantages of entropy metrics in general but provides additional information about the nature of the changes to help with distinguishing anomalies. Specifically, the traffic entropy spectrum (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 the previous chapter 5 showed how the TES could be used to visually match occurring patterns against known patterns to identify different types of anomalies and 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 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 [156], PCA [95], and KLE [17] (see Figure 6.1).

We rigorously evaluated the entropy telescope with a combination of simulation and real background traffic. As we outlined in Section 2.6, we share the concerns regarding AD evaluation practice expressed in [137] 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 [19], it is possible to inject our anomalies to arbitrary trace files. Re-

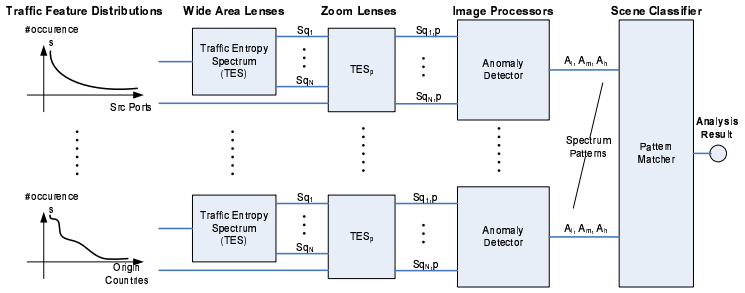


Figure 6.1: *Entropy Telescope building blocks.*

producibility and fair comparison of methods is crucial for scientific progress. For these reasons and to foster further research in this direction, we make the set of anomaly models designed for this study publicly available [170]. Furthermore, we provide access (on request) to the labeled timeseries data along with a MATLAB toolset to process them. Some of the most important findings related to the evaluation of the entropy telescope are 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 report on the prevalence of traffic anomalies. In summary, 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 we evaluate the detection and classification accuracy of several techniques in Section 6.4. Section 6.5 concludes this chapter.

6.2 Entropy Telescope

In this section we describe the entropy telescope consisting 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 tell 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 to be anomalous, the composed image 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 the entropies related to the different regions involves the 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 now to use the TES from the Wide Angle Lenses to determine for each of the Tsallis entropies, which elements (e.g., which IP addresses) contribute most to the respective entropy value. In a next step, the Zoom Lense then uses just those elements, when it recalculates the TES. With this, the Zoom Lense left the global focus and zoomed in on the elements most relevant for the Tsallis entropies for the different values of q. The only parameter of the Zoom Lense is the cut-off condition p. It defines the percentage of the original entropy value that must be reached until the Zoom Lense stops adding elements to the set of elements 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 diffeent 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 does not have to work 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 afore mentioned 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 dominated by generating element distributions, in the first place. 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 [13], Kalman filter [156], Principal Component Analysis (PCA) [96], and Karhunen-Loeve Expansion (KLE) [17], we selected the Kalman filter due to its simplicity as well as the PCA and the KLE method because they reflect the current state of the art:

- **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

[1] source and destination port number, source and destination ip address, autonomous system number, country code, flow size in bytes and bytes per packet

[2] flow, packet, and byte count

normal activity. We discuss the impact of k in our evaluation section.

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

We point out that 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 quantile-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 is 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 and detection is done using a two parameter approach. First, 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 combination 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 reflects indeed normal behavior, we selected the training data based on manual analysis of the time series using box plots and raw time series plots. While there remains an uncertainty whether our selection of training data is really 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 or worse, 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 and without potential bias because of anomalies that are more frequent than others, as typically the case with any real world

traces. And (2), our traffic trace used as 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 [174]. The assumption underlying our anomaly classification is that each anomaly class leaves a characteristic and (to some degree) 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: First, removing trend and daily patterns form the signals is difficult but has to be done for most supervised pattern recognition approaches. And second, we are not interested in the exact amplitude of the signals but rather a conservative estimate whether they are abnormal and if yes, how much.

An obvious choice for the input of the classification component is therefore the output of the Kalman detector: 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 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 for achieving a classification accuracy of around 85 percent.

In a next step, 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 in 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 e.g. 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 overfitting 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, e.g., a signal that has a more or less bi-modal distribution of its values during normal activity.

In a last step, 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 [30], 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 [73]: 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 [163]. 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.

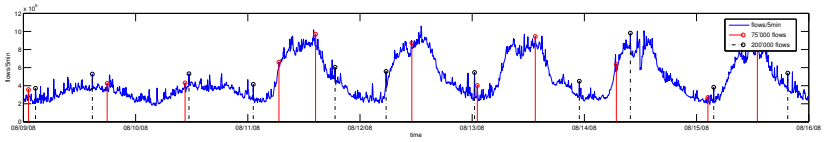


Figure 6.2: *The number of flows per 5min bin of our baseline trace with injected anomalies of intensities 75K and 200K.*

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:
  - Autonomous System (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 did 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 [173] 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 [19]. This approach has two main advantages. First, it provides well-defined ground truth independent of an expert labeling the events. Second, it allows 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 [137]. 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 [62] 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 and dispersed activity (e.g., the spoofed source addresses of the same DDoS attack) leads to an increase in entropy [96, 174, 196]. 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. 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 [170]. 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 but vanish 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, confirm this imbalance for most anomalies. Most

victims of scans do e.g., not reply because the scan is blocked by a firewall, and victims of a DDoS attack do not reply to (all) requests because they e.g., crashed or are simply too busy to serve all requests.

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

To take this aspect 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 of those set 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.

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 Crowd, 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 form studies of the Blaster [43] and Witty [36, 145] 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.

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, Attackers: OUT
7	DDoS II	Flash Crowd/ Botnet DDoS 2 (HTTP GET requests)	Victim: IN, Attacker:s OUT
8	DDoS III	DDoS with spoofed sources (SYN flood)	Victim: IN, Attackers: 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, Attackers: 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 means high/low activity region.*

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.*

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

6.4.1 Detection

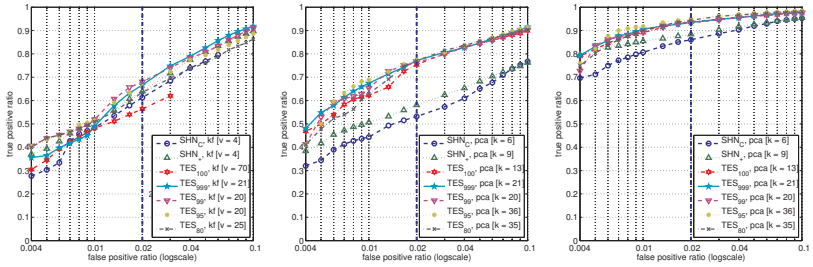
In Section 6.2.3 we defined a metric to be anomalous, denoted by a *vote*, if its anomaly score is bigger than a threshold t, i.e., $|A(x)| > t$. Moreover, for an anomaly alarm to be raised in a time slot, a number of v votes need to be present. For the PCA and KLE method, v is equal to one since they 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 Receiver Operating Characteristics (ROC) curves [56, 129] 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 readability reasons, 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 1 false positive per day for an FP rate of 0.04% to 1 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 [17], 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



(a) (b) (c)

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

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 is changing 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.

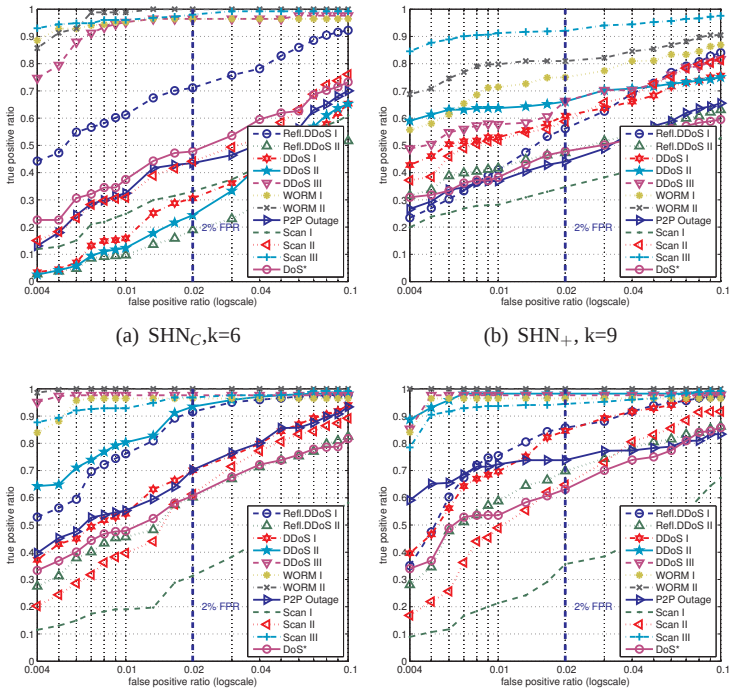
From the plots in Figure 6.3 we can see that a switch to TES, improves the detection accuracy for PCA by up to 20%. However, for the Kalman filter approach, the gain is rather small and lies around 5% for TES feature sets other than TES or TES_{80}. It seems that while the TES adds features carrying valuable information, it also adds noise with which the simple per-feature detection and voting scheme of the Kalman detector does not cope well. Unlike PCA, our Kalman detector does not make use of inter-feature relations. This being the main reason for the worse performance is supported by the Kalman filter's very bad 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 heavily 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

But 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. But since this result 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. Except for TES for an afore mentioned reason: The Kalman method does not make use of inter-feature relationships, such as the correlations between high and low activity regions in 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 only leads to 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, for the detection, we do not care whether an anomaly is seen in two (correlated) metrics or just one (uncorrelated) metric. At least 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 a better signal to noise ratio for anomalies affecting the low activity region only. In TES, such anomalies could be concealed by large (but not yet anomalous) changes in the overall activity.

6.4.2 Classification

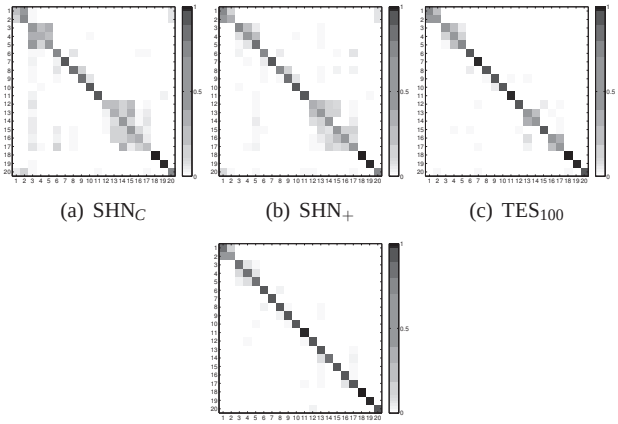
It is important for detection and classification to rely on models that are robust with respect to varying intensities. That is, if we train an SVM with DDoS models of a certain intensity, we do not want to miss the same attacks only because the real attack size differs slightly from the training size. 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 of *SHN* and TES feature sets would be even bigger in practice because a detector based on the *SHN* feature set 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 ($\Rightarrow$) 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 between 7.14% and 14.21% across all intensities. Using TES gives and 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 introduction 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 av-

Train	Evaluation	SHN_C	$\Rightarrow$	SHN_+	$\Rightarrow$	TES_{100}	$\Rightarrow$	$TES_{99.9}$	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 in percent for different sets of features and for different training and validation data set constraints.*

erage 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.



(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.*

In Figure 6.5 we provide a detailed view on which base anomalies were classified correctly and which not. Each point in the plots indicates the probability that the anomaly on the y-axis was classified as the anomaly on the x-axis. 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.

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, read 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

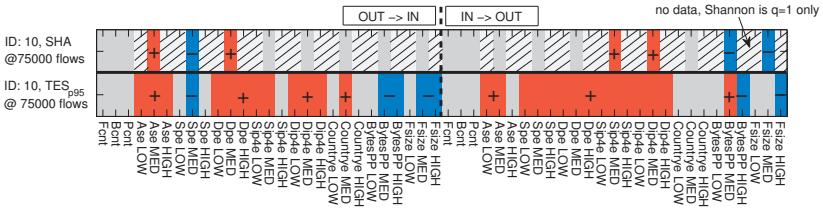
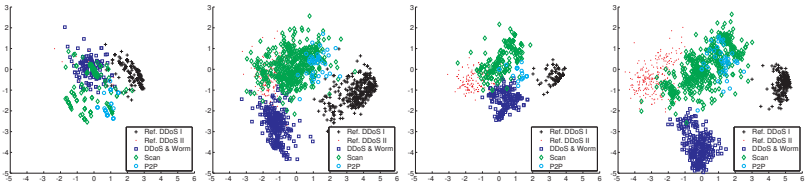


Figure 6.6: *Comparison of anomaly patterns for SHN_+ and TES_{95}. Obviously, SHN_+ misses crucial information captured by TES_{95}.*

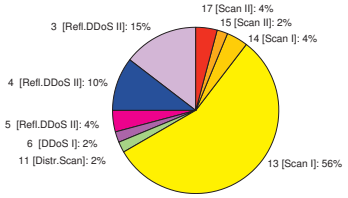


(a) SHN_+ with 50K (b) TES_{95} with 50K (c) SHN_+ with 200K (d) TES_{95} with 200K

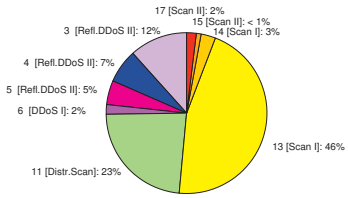
Figure 6.7: *Fisher's LDA plots of SHN_+ versus TES_{95}.*

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, the decrease occurred in the high activity region while in the medium region there was actually an increase of entropy.

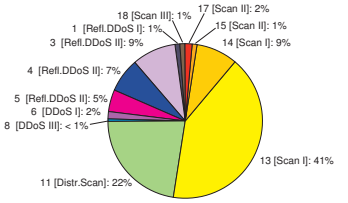
To give a graphical intuition of cluster centers and boundaries for different anomaly types, we show Fisher's LDA (Linear discriminant analysis) in Fig 6.7. LDA is typically used in machine learning to find a linear combination of features which characterize or separate two or more classes of objects. 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}.



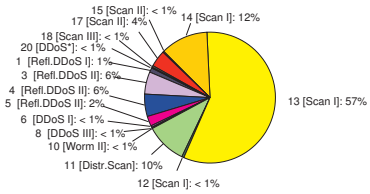
(a) Alert rate: 0.5% (=1 alert every 16.7 hours)
Total alerts: 49 (out of 9792 time bins)



(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-days 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 asigned a color to easily compare its share accross all four pie-charts.*

6.4.3 Prevalence of Anomalies in Real Backbone Traffic

As a last step in our evaluation, we report and discuss the results from applying our entropy telescope to a 34-days 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 with duration 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 expect 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 [5] 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) goes up 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 triggered and also the network operator is 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 [114]. VeriSign, drawing from different sources, estimates between 1000 and 10,000 DDoS attacks per day in 2008 [183]. The CSI computer crime and security survey 2008 [136] states that from the 522 responders, 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.

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 show that using the pruned TES instead of classical Shannon-only approaches improves detection accuracy by up to 20% and classification accuracy by 22.3% on average. 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 most prevalent anomalies are different types of scanning (69%-84%) and reflector DDoS attacks (15%-29%).

Chapter 7

Conclusions

In this chapter, we conclude our work on detection, classification and visualization of anomalies using generalized entropy metrics. We first give 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 which we discuss in the following.

A study on 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 a possible bias introduced

with this method is almost 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. A comparison of measurements obtained from the trace where we removed the respective anomaly, with those from the trace including them, 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 and 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 the anomaly contributes mostly 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 to analyze and visualize 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 makes it difficult to interprete the TES in certain situations. We called this problem the inter-region dependency since a change in one region of the TES, e.g., the high-activity region, can have a strong influence on what the we see in other regions of the TES, e.g., 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 e.g., 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 be a suitable anomaly detection approach for a 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 that 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 or this thesis was to find answers to two quite generic questions: Are entropies a useful tool in the context of anomaly detection? And: Can generalized entropy metrics help to improve on the results obtained wwhen using non-parameterized forms of entropy only? In the following, we assess to which extent we answered these questions.

In the first part, we addressed the first question in that 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 one network and two specific anomalies only. To compensate this to some extent, we did scale 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. Only because another team of researchers published similar results for their setting at the same time as we published our work, did we not follow up on these results.

In the second part, we addressed the question whether or not generalized entropy metrics are a useful tool to use for anomaly detection and classification. While we could 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. While this is now considered mandatory for any sound evaluation of anomaly detection and classification systems, it is unclear whether or not the differ-

ent 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, it is a rather small set of anomalies. Moreover, even if we did some drill-down work to check wheter what the TES detects is truly an anomaly, we can not provide any information about what the TES does not expose. For this, multiple labeled trace with a large and diverse set of real world anomalies would be required. But these limitations are limitations which are probably impossible to overcome. Furthermore, our set of anomalies is far from complete - there are so many different shapes of e.g., something as simple as a DDoS anomaly, that it is hard to model them all. Nevertheless, when we compare our evaluation to those of others, our set of anomalies was quite large.

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: On the one hand, the number of networked devices as well as network bandwiths are still growing with no end in sight. To cope with this growth, we can not rely on tools that require a fine-grained view on the data: such tools are likely to be way too expensive for a widespread use. Flow data however, provides an abstraction which proved to be good enough for things such as accounting or capacity planning. On the other hand, the attention might also be due to the fact that anomaly detection based on flow data is a challenging research field with many research questions that still need to be answered. Two issues that should receive attention from the research community are discussed in the following.

The first issue is the non-availability of standardized and recent data sets for the evaluation of anomaly detection and classification systems. Without such data sets for networks of varous sizes and purposes (e.g., backbone networks and different residential, company or military networks), a comparison of the many different anomaly detection systems is hardly possible. A comparison with respect to the traffic in one network only does not guarantee that the results would be the same for another network. However, this is truly a challenging topic: Creating such traces using e.g., a combination of deep packet inspection and manual labeling is not enought. Real world traces

often contain quite a lot of anomalies of the same type and similar sizes [1] which would make an evaluation quite biased. 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 standardized models, it is difficult to compare e.g., the detection quality related to DDoS attacks: While one used e.g. a model corresponding to the *juno* attack tool, another 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. Probably with the exception of networks not connected to the Internet or traces collected in testbeds, anomaly-free traffic traces are most likely a myth. The problem already starts with the definition of an anomaly: Is backscatter traffic an anomaly? Is it an anomaly if we see more of it than usual? And how much is ”more of it“? The same holds for things like the regular password guessing attacks on any remotely accessible machine. Do we consider them ”normal“ or ”abnormal“? In order to get sound and comparable results, these questions should be addressed in a standardized way. Maybe in the form of policies which define what we consider to be anomalous and what not. Until know, this question has largely been left unanswered.

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

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

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 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 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 to 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 and Dominik Schatzmann 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 (SWITCH) 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, 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 gain insight into the 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 thesis 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 build. 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 accompanying and helping me in my quest in one way or the other and beg them for forgiveness to not having mentioned them explicitly.

Bibliography

[1] Internet2 network. `http://www.internet2.edu/network/`.

[2] Niall M. Adams and David J. Hand. Comparing classifiers when the misallocation costs are uncertain. *Pattern Recognition*, pages 1139–1147, 1999.

[3] 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.

[4] 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.

[5] M. Allman, V. Paxson, and J. Terrell. A brief history of scanning. In *ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2007.

[6] P. Almquist. Type of Service in the Internet Protocol Suite. RFC 1349 (Proposed Standard), July 1992. Obsoleted by RFC 2474.

[7] Georgios Androulidakis, Vassilis Chatzigiannakis, and Symeon Papavassiliou. Network anomaly detection and classification via opportunistic sampling. *Netwrk. Mag. of Global Internetwkg.*, 23(1):6–12, 2009.

[8] Stefan Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *ACM Trans. Inf. Syst. Secur.*, 3(3):186–205, August 2000.

[9] 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.

[10] F. Baker. Requirements for IP Version 4 Routers. RFC 1812 (Proposed Standard), June 1995. Updated by RFC 2644.

[11] Rafael Ramos Regis Barbosa, Ramin Sadre, Aiko Pras, and Remco Meent van de. Simpleweb/university of twente traffic traces data repository, April 2010.

[12] 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.

[13] 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.

[14] 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.

[15] James R. Binkley. An algorithm for anomaly-based botnet detection. In *In Proceedings of USENIX Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI*, pages 43–48, 2006.

[16] Tomer Bitton. Morto post mortem: Dissecting a worm. http://blog.imperva.com/2011/09/morto-post-mortem-a-worm-deep-dive.html, September 2011.

[17] D. Brauckhoff, K. Salamatian, and M. May. Applying pca for traffic anomaly detection: Problems and solutions. In *INFOCOM*, 2009.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] Christian Callegari. Statistical approaches for network anomaly. detection. In *ICIMP*, 2009.

[26] Jeffrey Carr. *Inside Cyber Warfare: Mapping the Cyber Underworld*. O'Reilly Media, Inc., 2009.

[27] 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 2008*, pages 216 –220, 13-16 2008.

[28] V. Cerf, Y. Dalal, and C. Sunshine. Specification of Internet Transmission Control Program. RFC 675, December 1974.

[29] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):1–58, 2009.

[30] 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`.

[31] 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 vol.3, may. 2003.

[32] Cisco Systems Inc. *NetFlow Services and Applications - White paper*.

[33] Cisco Systems Inc. Netflow services solutions guide. `http://www.cisco.com`.

[34] B. Claise. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 5101 (Proposed Standard), January 2008.

[35] 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.

[36] Alberto Dainotti, Antonio PescapÃ©, and Giorgio Ventre. Worm traffic analysis and characterization. *IEEE ICC*, 2007.

[37] 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.

[38] Thomas DÃ¼bendorfer, Arno Wagner, Theus Hossmann, and Bernhard Plattner. Flow-level traffic analysis of the blaster and sobig worm outbreaks in an internet backbone. In Klaus Julisch and Christopher Kruegel, editors, *Intrusion and Malware Detection and Vulnerability Assessment*, volume 3548 of *Lecture Notes in Computer Science*, pages 103–122. Springer Berlin / Heidelberg, 2005.

[39] Guillaume Dewaele, Kensuke Fukuda, Pierre Borgnat, Patrice Abry, and Kenjiro Cho. Extracting hidden anomalies using sketch and non gaussian multiresolution statistical detection procedures. In *LSAD '07: Proceedings of the 2007 workshop on Large scale attack defense*, pages 145–152, New York, NY, USA, 2007. ACM.

[40] 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.

[41] Xenofontas Dimitropoulos, Marc Stoecklin, Paul Hurley, and Andreas Kind. The eternal sunshine of the sketch data structure. *Comput. Netw.*, 52(17):3248–3257, 2008.

[42] Dejing Dou, Jun Li, Han Qin, and Shiwoong Kim. S.: Understanding and utilizing the hierarchy of abnormal bgp events. In *In: SIAM International Conference on Data Mining*, pages 457–462, 2007.

[43] T. Duebendorfer 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.

[44] 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.

[45] 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.

[46] Nick Duffield, Carsten Lund, and Mikkel Thorup. Estimating flow distributions from sampled flow statistics. *IEEE/ACM Trans. Netw.*, 13(5):933–946, 2005.

[47] F. Y. Edgeworth. On discordant observations. *Philosophical Magazine Series 5*, 23(143):364 – 375, April 1887.

[48] J.P. Egan. *Signal detection theory and ROC-analysis*. Academic Press series in cognition and perception. Academic Press, 1975.

[49] Raimund E. A. Eimann. *Network Event Detection with Entropy Measures*. PhD thesis, University of Auckland, 2008.

[50] Cristian Estan and George Varghese. New directions in traffic measurement and accounting. *SIGCOMM Comput. Commun. Rev.*, 32(4):323–336, August 2002.

[51] Cristian Estan and George Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Trans. Comput. Syst.*, 21(3):270–313, 2003.

[52] 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.

[53] 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. Last visited: October 2010.

[54] 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. *Comput. Netw.*, 46(2):253–272, October 2004.

[55] T. Fawcett. ROC graphs: Notes and practical considerations for researchers, 2004.

[56] T. Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

[57] 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.

[58] 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.

[59] Laura Feinstein, Dan Schnackenberg, Ravindra Balupari, and Darrell Kindred. Ddos tolerant networks. In *DISCEX (2)*, pages 73–75, 2003.

[60] Fernando J. S. Filho. *Unsupervised Diagnosis of Network Traffic Anomalies*. PhD thesis, UniversitÃ© Pierre et Marie CURIE, 2010.

[61] 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 6th International COnference*, Co-NEXT '10, pages 8:1–8:12, New York, NY, USA, 2010. ACM.

[62] Harry Frank and Steven G. Althoen. *Statistics: Concepts and Applications*. Cambridge University Press, 1994.

[63] 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–, Washington, DC, USA, 2001. IEEE Computer Society.

[64] 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.

[65] Carrie Gates, Carol Taylor, and Matt Bishop. Dependable security: testing network intrusion detection systems. In *Proceedings of the 3rd workshop on on Hot Topics in System Dependability*, HotDep'07, Berkeley, CA, USA, 2007. USENIX Association.

[66] GEANT Network. `http://www.geant.net`.

[67] MaxMind® GeoIP Geolocation Technology . `http://www.maxmind.com/`.

[68] 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.

[69] Guavus. `http://www.guavus.com`.

[70] 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, oct. 2006.

[71] 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.

[72] Nicolas Hohn and Darryl Veitch. Inverting sampled traffic. *IEEE/ACM Trans. Netw.*, 14(1):68–80, 2006.

[73] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classication. Technical report, Department of Computer Science, National Taiwan University, 2009.

[74] 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.

[75] IETF Working Group. Internet Protocol Flow Information eXport (IPFIX). `http://tools.ietf.org/wg/ipfix/`.

[76] Internet2 observatory data collections. `http://www.internet2.edu/observatory/archive/data-collections.html`.

[77] Interim Internet2 IPv6 Netflow Anonymization Policy, v1.0. http://www.internet2.edu/policies/ipv6-mask.html, 2010.

[78] Internet systems consortium: Internet host count history. `http://www.isc.org/solutions/survey/history`, July 2012 (last visited).

[79] Worldwide Infrastructure Security Repor I. `http://www.arbornetworks.com/report`, 2005.

[80] Worldwide Infrastructure Security Repor II. `http://www.arbornetworks.com/report`, 2006.

[81] Worldwide Infrastructure Security Repor III. `http://www.arbornetworks.com/report`, 2007.

[82] Worldwide Infrastructure Security Repor IV. http://www.arbornetworks.com/report, 2008.

[83] Worldwide Infrastructure Security Repor V. http://www.arbornetworks.com/report, 2009. Last visited: 07.02.2012.

[84] Worldwide Infrastructure Security Repor VI. http://www.arbornetworks.com/report, 2010.

[85] 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.

[86] 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.

[87] juno, a DDoS attack tool. http://packetstormsecurity.org/DoS/juno.c.

[88] 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, Oct 2009.

[89] 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, apr. 2004.

[90] 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.

[91] F.H. Knight. *Risk, Uncertainty and Profit*. Series of reprints of scarce tracts in economic and political science. Houghton Mifflin Company, 1921.

[92] S.P. Kozaitis and W. Petsuwan. Improved anomaly detection using block-matching denoising. *Computer Communications*, 35(7):875 – 884, 2012.

[93] 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.

[94] Anukool Lakhina, Marc Crovella, and Christoph Diot. Diagnosing network-wide traffic anomalies. In *ACM SIGCOMM*, Portland, August 2004.

[95] 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.

[96] Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining anomalies using traffic feature distributions. In *ACM SIGCOMM*, 2005.

[97] 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.

[98] 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.

[99] 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.

[100] 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 vol.2, 2000.

[101] 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.

[102] Jianning Mai, A. Sridharan, Chen-Nee Chuah, Hui Zang, and Tao Ye. Impact of packet sampling on portscan detection. *Selected Areas in Communications, IEEE Journal on*, 24(12):2285 –2298, dec. 2006.

[103] 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. Last visited: 05.02.2012.

[104] MAWI working group traffic archive. `http://mawi.wide.ad.jp/mawi/`.

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

[106] John McHugh. Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Trans. Inf. Syst. Secur.*, 3(4):262–294, Nov. 2000.

[107] P. Mell, K. Karen, and J. Nusbaum. *Guide to Malware Prevention and Handling: Recommendations of the National Institute of Standards and Technology*. National Institute of Standards and Technology (U.S.), 2005.

[108] Microsoft. Microsoft security intelligence report, volume 7. http://www.microsoft.com/security/sir/archive/default.aspx, November 2009. Last visited: 05.02.2012.

[109] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2):39–53, 2004.

[110] 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.

[111] 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.

[112] G. Molchan and V. Keilis-Borok. Earthquake prediction: probabilistic aspect. *Geophysical Journal International*, 173:1012–1017, June 2008.

[113] G. M. Molchan. Strategies in strong earthquake prediction. *Physics of the Earth and Planetary Interiors*, 61:84–98, 1990.

[114] 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.

[115] 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.

[116] 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.

[117] 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.

[118] Cisco Netflow. *White Paper: NetFlow Services and Applications.*

[119] 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.

[120] node9. A lethal Denial of service attack. `http://www.webhostingtalk.com/showthread.php?t=12230`, 2001.

[121] 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.

[122] 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.

[123] E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[124] Ruoming Pang, Mark Allman, Vern Paxson, and Jason Lee. The devil and packet trace anonymization. *SIGCOMM Comput. Commun. Rev.*, 36(1):29–38, January 2006.

[125] 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.

[126] 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.

[127] 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.

[128] 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. Last visited: October 2010.

[129] 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.

[130] 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.

[131] J. Postel. Internet Protocol. RFC 791 (Standard), September 1981. Updated by RFC 1349.

[132] 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.

[133] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[134] 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, feb. 2007.

[135] Bruno Ribeiro, Weifeng Chen, Gerome Miklau, and Don Towsley. Analyzing privacy in enterprise packet trace anonymization. In *In Proceedings of the 15 th Network and Distributed Systems Security Symposium*, 2008.

[136] Robert Richardson. CSI computer crime and security survey 2008. *Computer Security Institute*, 2008.

[137] Haakon Ringberg, Matthew Roughan, and Jennifer Rexford. The need for simulation in evaluating anomaly detectors. *SIGCOMM Comput. Commun. Rev.*, 38(1):55–59, 2008.

[138] Haakon Ringberg, Augustin Soule, and Jennifer Rexford. Webclass: adding rigor to manual labeling of traffic anomalies. *SIGCOMM Comput. Commun. Rev.*, 38(1):35–38, January 2008.

[139] RuleQuest Research . `http://www.rulequest.com/`.

[140] 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.

[141] 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.

[142] 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.

[143] 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.

[144] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[145] Colleen Shannon and David Moore. The spread of the witty worm. *IEEE Security and Privacy*, 2(4):46–50, 2004.

[146] Alok Sharma and Sunil Pranit Lal. Tanimoto based similarity measure for intrusion detection system. *J. Information Security*, 2(4):195–201, 2011.

[147] 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.

[148] Joel Sommer. *Harpoon - A Flow-level Traffic Generator*.

[149] 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.

[150] 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.

[151] 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.

[152] 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.

[153] Didier Sornette. Dragon-Kings, Black Swans and the Prediction of Crises. *International Journal of Terraspace Science and Engineering*, 2(1):1–18, July 2009.

[154] Didier Sornette and Guy Ouillon. Dragon-kings: Mechanisms, statistical methods and empirical evidence. *European Physical Journal Special Topics*, 205, 2012.

[155] 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.

[156] 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.

[157] U. Speidel, R. Eimann, and N. Brownlee. Detecting network events via t-entropy. In *Information, Communications Signal Processing, 2007 6th International Conference on*, pages 1 –5, 10-13 2007.

[158] Avinash Sridharan, Tao Ye, and Supratik Bhattacharyya. Connectionless port scan detection on the backbone. *Performance, Computing, and Communications Conference, 2006. IPCCC 2006. 25th IEEE International*, pages 10 pp. –576, apr. 2006.

[159] 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.

[160] Marc Ph. 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, Berlin, Heidelberg, 2008. Springer-Verlag.

[161] 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.

[162] 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.

[163] SWITCH. The swiss education and research network. `http://www.switch.ch`.

[164] Symantec. Symantec Internet Security Threat Report, Trends for January 06 - June 06, 2006.

[165] Symantec. Symantec Internet Security Threat Report Trends for 2007, 2007.

[166] Symantec. Symantec Internet Security Threat Report Trends for 2008, 2009.

[167] Nassim Nicholas Taleb. *The Black Swan: Second Edition: The Impact of the Highly Improbable*. Random House Trade Paperbacks, 2010.

[168] Andrew S. Tanenbaum. *Computer Networks (3th Edition)*. Prentice Hall PTR, 3 edition, March 1996.

[169] Mahbod Tavallaee, Natalia Stakhanova, and Ali Akbar Ghorbani. Toward credible evaluation of anomaly-based intrusion-detection methods. *Trans. Sys. Man Cyber Part C*, 40(5):516–524, September 2010.

[170] Bernhard Tellenbach. Collection of FLAME anomaly models. http://people.ee.ethz.ch/~betellen/AnomalyModels, 2010.

[171] 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.

[172] Bernhard Tellenbach, Daniela Brauckhoff, and Martin May. Impact of traffic mix and packet sampling on anomaly visibility. In *ICIMP '08: Proceedings of the 2008 The Third International Conference on Internet Monitoring and Protection*, pages 31–36, Washington, DC, USA, 2008. IEEE Computer Society.

[173] Bernhard Tellenbach, Martin Burkhart, Dominik Schatzmann, David Gugelmann, and Didier Sornette. Accurate network anomaly classification with generalized entropy metrics. *Computer Networks*, 2011.

[174] 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.

[175] M. Thottan, G. Liu, and C. Ji. *Algorithms for Next Generation Networks*, chapter Anomaly Detection Approaches for Communication Networks, pages 239–261. Springer London, 2010.

[176] M. E. Torres, M. M. A nino, 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.

[177] B. Trammell and E. Boschi. Bidirectional Flow Export Using IP Flow Information Export (IPFIX). RFC 5103 (Proposed Standard), January 2008.

[178] 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, Berlin, Heidelberg, 2011. Springer-Verlag.

[179] Cheng-Fa Tsai and Chia-Chen Yen. Unsupervised anomaly detection using hdg-clustering algorithm. In Masumi Ishikawa, Kenji Doya, Hiroyuki Miyamoto, and Takeshi Yamakawa, editors, *Neural Information Processing*, pages 356–365. Springer-Verlag, Berlin, Heidelberg, 2008.

[180] C. Tsallis. *Introduction to Nonextensive Statistical Mechanics: Approaching a Complex World*. Springer, 2009.

[181] Constantino Tsallis. Possible generalization of boltzmann-gibbs statistics. *Journal of Statistical Physics*, 52, 1988.

[182] Nonextensive statistical mechanics and thermodynamics: Bibliography. `http://tsallis.cat.cbpf.br/biblio.htm`, regularely updated 2010.

[183] VeriSign. Distributed Denial of Service (DDoS) Attacks: Latest Motivations and Methods. http://www.verisign.com/static/idefense_ddos_verisign_20080908.pdf, 2008.

[184] Arno Wagner. *Entropy-based worm detection for fast IP networks*. PhD thesis, ETH Zurich, 2008.

[185] 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.

[186] JÃ$\frac{1}{4}$rg Wallerich, Holger Dreger, Anja Feldmann, Balachander Krishnamurthy, and Walter Willinger. A methodology for studying persistency aspects of internet flows. *SIGCOMM Comput. Commun. Rev.*, 35(2):23–36, 2005.

[187] WIDE project. `http://www.wide.ad.jp/`.

[188] 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.

[189] 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, nov. 2002.

[190] 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.

[191] Y. Yasami, S. Khorsandi, S.P. Mozaffari, and A. Jalalian. An unsupervised network anomaly detection approach by k-means clustering & id3 algorithms. In *Computers and Communications, 2008. ISCC 2008. IEEE Symposium on*, pages 398 –403, july 2008.

[192] Vinod Yegneswaran, Paul Barford, and Johannes Ullrich. Internet intrusions: global characteristics and prevalence. *SIGMETRICS Perform. Eval. Rev.*, 31(1):138–147, 2003.

[193] Norihiko Yoshida. *Content Delivery Networks*, chapter Dynamic CDN Against Flash Crowds, pages 275–296. Lecture Notes Electrical Engineering. Springer Berlin Heidelberg, 2008.

[194] 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.

[195] 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 Perform. Eval. Rev.*, 33(1):350–361, June 2005.

[196] 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.

Curriculum Vitae

Bernhard Tellenbach was born in Bern, Switzerland on June 30, 1979.

Educational Curriculum Vitae

2005 – pursuing	**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

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 with our thesis, a fully operational NetFlow collection and processing infrastructure built in 2002/3 was already in place. A detailed description of it can be found in [184]. However, during our thesis, we had to update it's hard- and software several times to match the functional as well as performance and space requirements to manage and process the NetFlow data. We now first provide a brief overview of the NetFlow collection and processing infrastructure from in place in June 2012. Next, we discuss the hardware costs and manpower requirements, the past and current Netflow data volume to be handled and some key numbers with regard to processing power and memory. Finally, we conclude with a summary of our main contributions to setting this infrastructure up and keep it running.

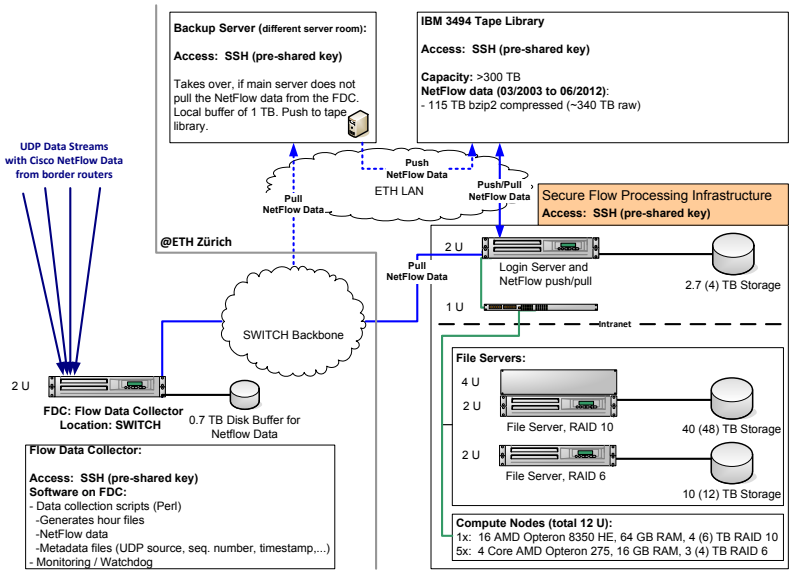


Figure A.1: *The NetFlow collection and processing infrastructure run by the Communication Systems Group (CSG) at ETH Zurich as of June 2012.*

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 consist of the following hardware- and software components:

- **Flow Data Collector (FDC):** This component is responsible for collecting and buffer 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 to 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 e.g., the timestamp, sequence number, and 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 some reasons, compression is done 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 autonomous system of the source and destination of the flow.
  - Performing analysis on the incoming or archived flow data to keep e.g., a database with information on the files on the tape library such as, their size and whether they are missing or corrupt, up to date.
  - Custom flow data processing.

  For research or non out-of-the-box analysis, a flexible software framework is required to parse the data and work with this data. A near real-time analysis (processing one hour of flow data in less than an hour) of flow data might only be achieved through massive parallelization using multiple processing nodes.
- **Backup Server:** If the Secure Flow Processing Infrastructure has downtimes (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. Especially, if as in our case, there is no 24x7 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 critical data. Hence, 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 both at the technical and the management level. Furthermore, the infastructure limits

the attack surface further 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 estimate costs to build and maintain 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 estimate costs to build and maintain our NetFlow collection and processing infrastructure.*

According to this table, two of the bigger cost factors are the initial overhead 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 e.g., 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 comressed 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: *The estimate costs to build and maintain our NetFlow collection and processing infrastructure.*

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 differnt parts of the data set for their research and that (2) they do not just need the raw data but do also produce a large amount of analysis results or pre-processed data optimized for doing e.g., forensics[1], this quickly becomes a challenging problem. As a consequence, most things have to be done on demand: only results and data from events (e.g., from special events such as major attacks or network failures) often used are kept on the storage servers.

A.1.4 Processing Power and Memory Requirements

Decompression of an hour of NetFlow data from 2012 takes typically between 20 and 40 minutes (night/day) using a single core of our AMD Opteron 275 systems. bzip2 decompresses our data approx. at 2 MB/s. If the data is enriched with additional information such as e.g. the country of origin of

[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

the flow or the origin autonomous system, it takes up to 30% longer. As a consequence, if the data comes from our tape library with approx. 20 MB/s expected transfer rate, we can decompress and parse 10 hours of data in approx. 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 require to track e.g., 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

- Concept, 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 this 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 autonomous system 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 (hard- and software) to meet the requirements for data compression and extended flow data buffering in case of 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 spezialiced data structures are only there because

the NetflowVX library is basically a NetFlow v9 extended, refactored and well-documented version of a NetFlow processing library written by Arno Wagner [184]. The library is tailored to parse the NetFlow data as it is stored by our capturing infrastructure: raw NetFlow Protocol Data Units 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 autonomous system number. To do so, the library looks at the source and destination IP address of a flow and resolves the corresponding autonomous system number using the autonomous system 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 autonomous system information since they were not yet 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 after the C++ version was available, no new tools written in C were developed.

Performance

Performance is an important criteria 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 metainfo parsing from *.stat* files.
- **AS+.stat**: PDU parsing from *.dat* files, PDU metainfo parsing from *.stat* files and enrichment of flow data with origin and destination autonomous system numbers.

[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 do only have to be unique per NetFlow export device. Without the information from which device a PDU originates, the library assumes that all PDUs come from the same device.

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 performend using a single core of an AMD Opteron 275 on a system with 16 GBytes of Random Access Memory (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 realtime 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-to-far future, the load can be distributed to multiple cores by e.g., 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 compareing the performances for uncompressed input data, we get the expected result. This suggests that 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 we did not investigate further.

[3] 78000 flows/sec * 3600 sec = 280 million flows

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 filtereing 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 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 envelop used to pass data from module to module. Even though the chain supports to hand over arbitrary data, it is typically a list of flows which is passed form module to module. Figure A.2 shows a sample of such a chain on the module layer.

However, the type of the flows in the list might depend on the acutal processing job: there is an internal flow format (*FlowCompact*) which is optimized for space but does contain 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 alows to extend the framework to handle new input flow formats such as IPFIX without chaning anything expect 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 allow to read NetFlow data from disk using the NetflowVX library but also to read and write files with flows contining 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 the fact that if you do not need them, do not use them: keep the software 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.

[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.

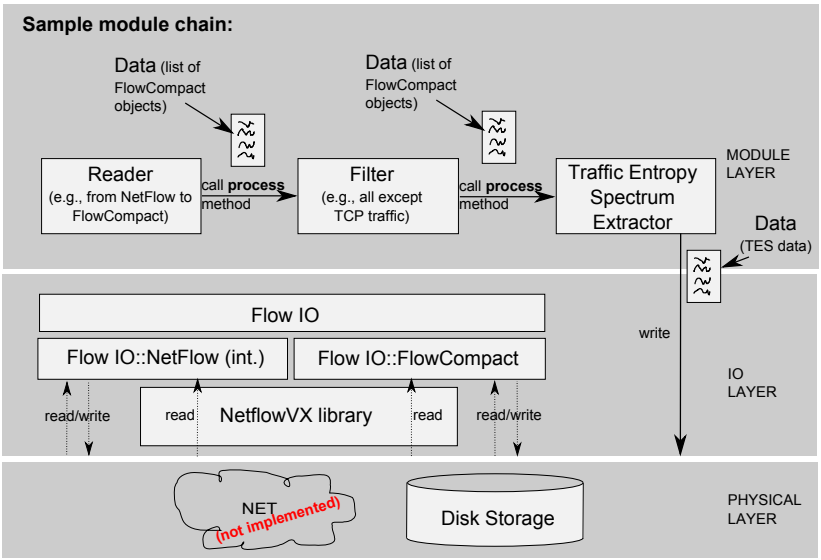


Figure A.2: *Outline of the design of our NetFlow processing framework.*

While we designed and implemented most of the NetFlow Processing Framework, fruitfull discussions and comments from Dominik Schatzmann as well as e.g., 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, Internet Control Message Protocol (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 e.g., 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 schould 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 multipile flow features.
- **Timeing analysis module**: A module to output information on NetFlow time stamps. This module is related to [178] 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 realtime.

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 on 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 manualy 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 class establishes a connection to the data source and allows column-

[5] Note that until release R2008a, only very few Matlab code released on the web was object oriented. Many Matlab users didn't 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.

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 e.g., data points from Mondays between 11:15 and 11:20 to a distribution "Monday, 11:15 to 11:20" etc. These profiles can e.g. 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 [30], 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 e.g., Figure 6.5 in chapter 6.
- **Utilities**: Some Matlab functions to help with ploting, figure export and hierarchical sorting of vectors of structs [9]. Ploting functions include e.g., a function to plot labeled pie-charts of the kind shown in

[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.

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

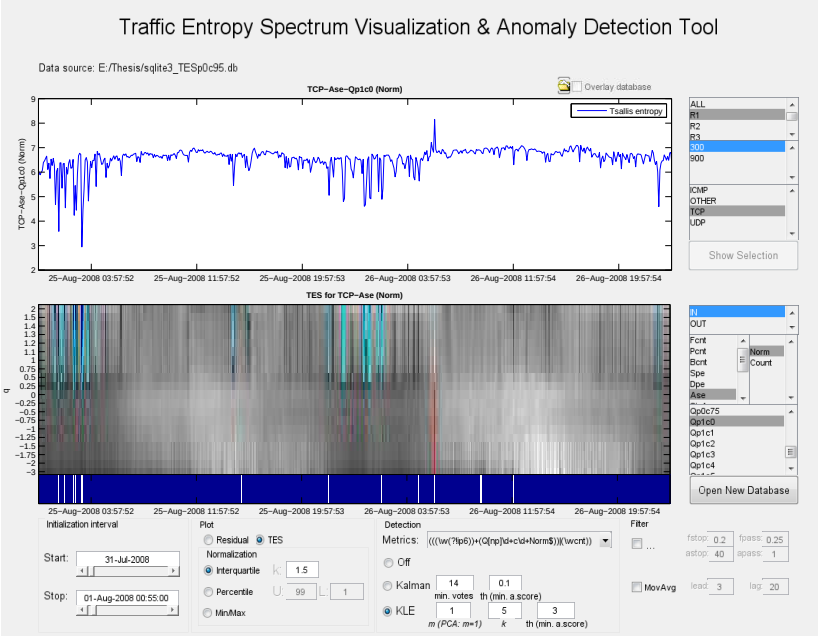


Figure A.3: *Screenshot of the traffic entropy spectrum visualization and anomaly detection tool.*

Week Profile & Statistical Parameter Analysis Tool

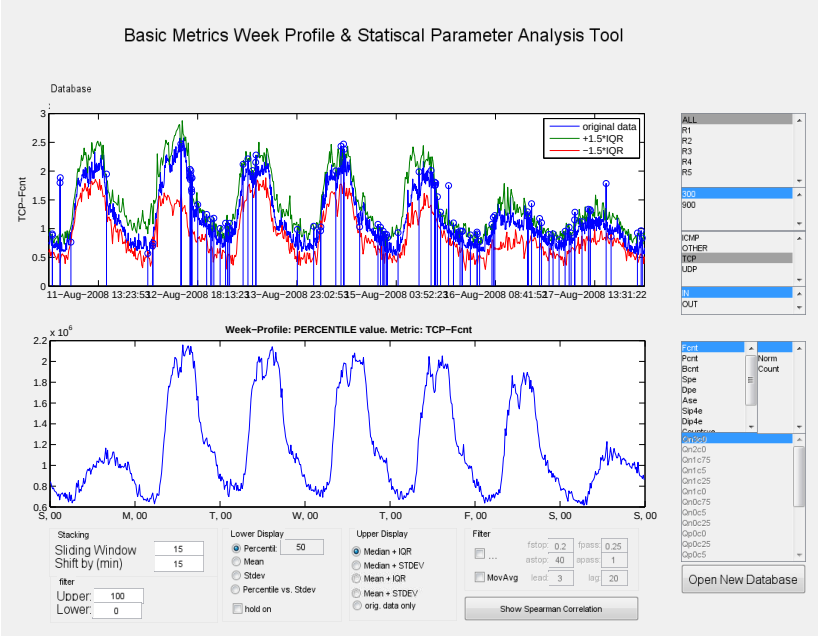


Figure A.4: *Screenshot of the week profile and statistical parameter analysis tool.*

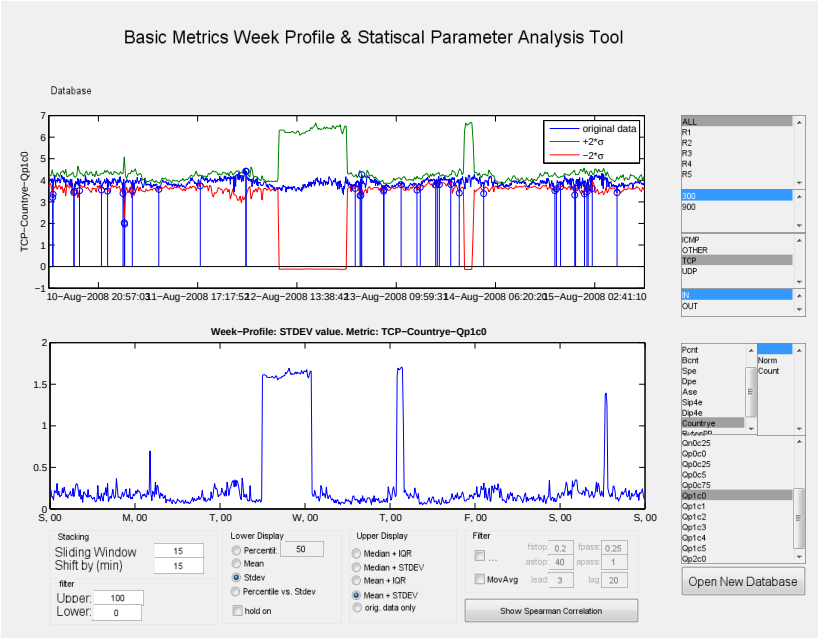


Figure A.5: *Screenshot of the week profile and statistical parameter analysis tool.*

Traffic Feature Distribution Analysis and Visualization Tool

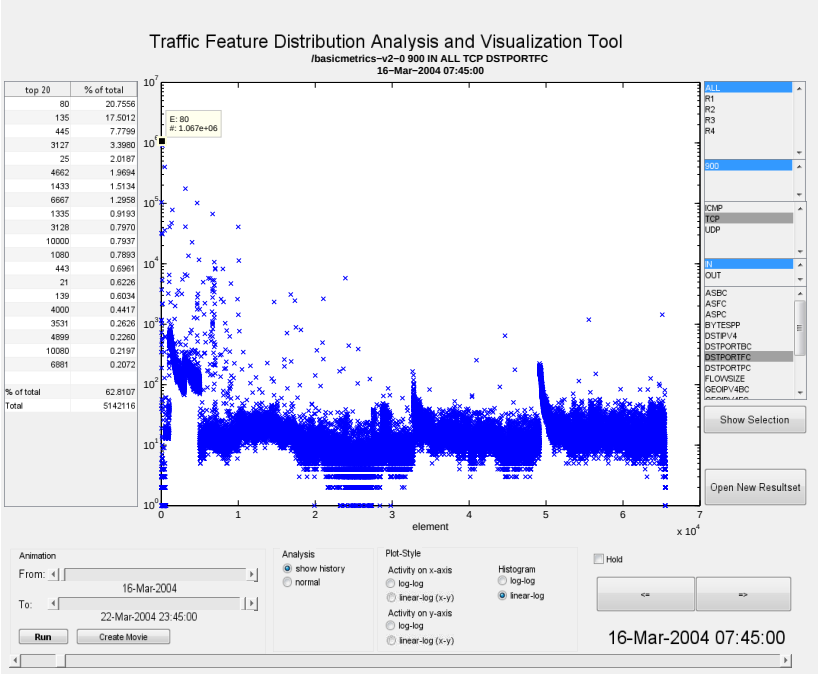


Figure A.6: *Screenshot of the week profile and statistical parameter analysis tool.*

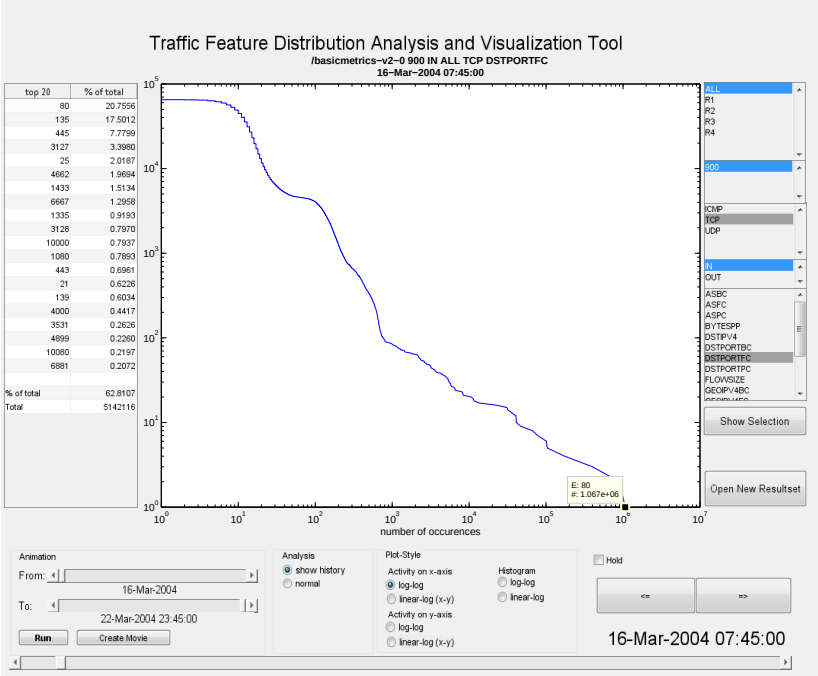


Figure A.7: *Screenshot of the week profile and statistical parameter analysis tool.*