Entropy Based Worm and Anomaly Detection in Fast IP Networks

Arno Wagner* Bernhard Plattner

Communication Systems Laboratory, Swiss Federal Institute of Technology Zurich

Gloriastr. 35, CH-8092 Zurich

{wagner, plattner}@tik.ee.ethz.ch

Abstract

Detecting massive network events like worm outbreaks in fast IP networks, such as Internet backbones, is hard. One problem is that the amount of traffic data does not allow real-time analysis of details. Another problem is that the specific characteristics of these events are not known in advance. There is a need for analysis methods that are real-time capable and can handle large amounts of traffic data. We have developed an entropy-based approach, that determines and reports entropy contents of traffic parameters such as IP addresses. Changes in the entropy content indicate a massive network event. We give analyses on two Internet worms as proof-of-concept. While our primary focus is detection of fast worms, our approach should also be able to detect other network events. We discuss implementation alternatives and give benchmark results. We also show that our approach scales very well.

1. Introduction

Large-scale network events, such as outbreaks of a fast Internet worm are difficult to detect in real-time from observed traffic, even more so if the observed network carries a large amount of traffic. Even during worm outbreaks, actual attack traffic is only a small fraction of the overall traffic. Its precise characteristics are usually unknown beforehand, making direct matching hard. It is desirable to derive parameters that describe large amounts of traffic data in such a way that details are hidden, but the type of changes associated with worm outbreaks and other network events are visible.

We have developed an entropy based analysis method that highlights outbreaks of fast worms in near real-time. The analysis method works on flow-level

data exported from the observed network. Entropy is estimated using data compression algorithms.

The paper is structured as follows: Section 2 explains the fundamental idea and its characteristics. Section 3 gives proof-of-concept results obtained on traffic recorded on a medium-sized Internet backbone network. A discussion of implementation details and characteristics, including speed and scalability is given in section 4. Sections 5 and 6 give concluding remarks and discuss related work.

2. Approach - Why Entropy?

Generally speaking entropy is a measure of how random a data-set is. The more random it is, the more entropy it contains. Entropy contents of a (finite) sequence of values can be measured by representing the sequence in binary form and then using data compression on that sequence. The size of the compressed object corresponds to the entropy contents of the sequence. If the compression algorithm is perfect (in the mathematical sense), the measurement is exact.

So what is the relation to worm detection? When a fast scanning worm propagates through the Internet, the propagation activity looks like this: a smaller number of infected hosts tries to find other hosts to be infected by attempting to connect to them in a random fashion. This scanning activity is different from normal Internet traffic.

The connection between entropy and worm propagation is that worm traffic is more uniform or structured than normal traffic in some respects and a more random in others. The change in IP address characteristics seen on a *flow* level (i.e. when packets belonging to a TCP connection or UDP data stream with same source/destination IP address and port numbers are reported aggregated into one "flow") is relatively intuitive: infected, scanning hosts try to connect to a lot of other hosts. If these flows grow to be a significant part of the set of flows seen in total, the source IP addresses of the scanning hosts will be seen in many flows and since they are relatively few hosts, the source IP address fields will contain less entropy per address seen

than normal traffic. On the other hand the target IP addresses seen in flows will be much more random than in normal traffic.

A similar thing happens on the port level. If an attacker scans for a specific vulnerability, these scans often have to go to a specific target port. The source ports of these connection are usually selected in some weakly random fashion form a range of possible source ports. If scanning traffic with this characteristic becomes a significant component of the overall network traffic, the entropy contents of the destination port fields in flows seen in the network will decrease significantly.

The second important insight is that during a random scan, most targeted addresses will not respond because the addresses are unused or protected by firewalls. As a consequence scanning traffic is far more asymmetrical than normal traffic.

Target ports in connection initiating flows are well known ports and contain very little entropy. However the *answering* flows will use the port numbers in reverse. In normal traffic the entropy of source and destination port fields is almost equal. During an attack, many of the answering flow are missing and source port entropy increases while destination port entropy decreases.

One important difference between IP addresses and ports is that while IP addresses have to follow this pattern, i.e. few trying to connect many, port numbers do not need to. It is possible to write worms that use fixed *source* port numbers and variable *target* port numbers in their scans. Completely fixed port numbers and completely random port numbers are also possible. While IP address entropy changes can be used to detect worms, port address entropy changes are more useful to get first insights into the the way port numbers are used in a scanning strategy. Port address changes may still be used to support detection, but should not be relied on since they are not guaranteed to happen in a worm outbreak.

On the theoretical side it is important to understand that not entropy is the relevant traffic characteristic, but Kolmogorov Complexity [16] of an interval of data. While entropy describes the average expected information content of a symbol that is chosen in a specific randomised way from a specific symbol set, Kolmogorov Complexity describes the specific information content of a specific object given, e.g. as a binary string of finite length. Please refer to the literature on Information Theory for the details.

Kolmogorov Complexity and entropy are closely related. We will use the term "entropy" in this paper loosely and in places were we really should have said "Kolmogorov Complexity".

We estimate entropy by using standard, well-known data compression algorithms. The data field to be ex-

amined (e.g. source IP address) of the flow records belonging to the relevant data interval are fed into the compression algorithm in the order the router exported them. The resulting compressed size serves as the entropy estimation. The entropy estimation obtained in this way can (and often will) be very inaccurate, but it turns out that it is well suited for relative comparison between different data intervals.

3. Experimental Findings

This section describes the archived network data our measurements were done on. We then give two real-world proof-of-concept examples based on the outbreak events of two fast worms.

In most figures we plot the entropy estimations over time for four values, namely source and destination IP addresses and ports. The plotted time intervals start before the outbreaks to illustrate normal traffic entropy characteristics. The outbreak times of both worms are marked with arrows.

3.1. Network Data Used

We are collecting NetFlow v5 [10] data from the SWITCH (Swiss Academic and Research Network [4], AS559) network, a medium-sized Swiss backbone operator, which connects all Swiss universities and various research labs (e.g. CERN) to the Internet. Unsampled NetFlow data from all four SWITCH border routers is captured and stored for research purposes in the context of the DDoSVax project [11] since early 2003. The SWITCH IP address range contains about 2.2 million IP addresses. In 2003 SWITCH carried around 5% of all Swiss Internet traffic [17]. In 2004, we captured on average 60 million NetFlow records per hour, which is the full, non-sampled number of flows seen by the SWITCH border routers.

3.2. Example Entropy Analyses

In order to demonstrate the effect of a worm outbreak on entropy statistics we will now give plots on two examples, namely the Blaster [9, 6, 15] and Witty [18, 20] worm.

Both worms are relatively well understood and well documented. First observed on August 11th, 2003, Blaster uses a TCP random scanning strategy with fixed destination and variable source port to identify potential infection targets. Blaster is estimated to have infected 200'000...500'000 hosts worldwide in the initial outbreak.

The Witty worm, first observed on March 20th, 2004, is a good second example, because it has some unexpected characteristics. Witty attacks a specific firewall product. It uses UDP random scans with *fixed*

source port and *variable* destination port. Witty infected only about 15'000 hosts, but was clearly visible in the UDP entropy statistics. (There is far less UDP traffic than TCP traffic in the Internet, but because UDP traffic is aggregated far less in the routers there are almost as many UDP flows recorded as TCP flows.)

The given figures all contain normalised plots. The y-axis is the number of compressed bits per uncompressed bit input data. This is also the estimation in bits of entropy per raw input bit. The possible range for this value is $0.0 \ldots 1.0$ if we assume that compression can not make the object to be compressed larger.

Figure 1 shows the compressibility changes before and during the Blaster worm outbreak. The y-axis gives the size of the compression result in relation to the raw data size for the indicated address fields. Before the outbreak time it can be seen that source IP addresses are a bit more uniform than destination IP addresses. At the same time destination port numbers are more uniform than source port numbers. Port numbers are less uniform than IP addresses, but should not be compared directly, since ports are 16 bit while IP address are 32 bit. This leaves the relation sensitive to characteristics of the compressor used. (In Subsection 4.4 we will also demonstrate that IP addresses and ports are affected differently by sampling.) Analysis of samples taken from the stored 2003 and 2004 data indicate that this configuration is typical.

During the outbreak of the Blaster worm the compressibility characteristics change massively. The most obvious is that they diverge, i.e. source and destination plots change their values in different directions, both for the IP and the port plots. This effects can be used to automate detection of large-scale anomalies. Regarding the individual plots, it can be seen clearly that source IP addresses become (since a smaller number of hosts starts to generate more flows) more uniform, and hence contain less entropy and the compressibility plot graph drops, while destination IP addresses become more random (since packets to many more IP addresses than normally seen are sent by the random scanning algorithm used in Blaster), leading to the opposite effect.

The equivalent plot for the Witty worm in Figure 2 a bit different. For the time before the attack we see a similar configuration as for TCP, with IP address and port compressibility shifted against each other. Samples taken from the 2003 and 2004 data also indicate that the observed pre-attack configuration is typical for UDP traffic in the absence of large-scale attacks.

During the Witty outbreak the general observations are the same as for Blaster: The source and destination graphs diverge significantly. In addition, because of the fixed source and variable destination port the source and destination graphs for ports actually cross over and change their vertical ordering. Source and destination

IP plots still show the same trends as for Blaster, but the source IP address is less affected and destination IP address more.

4. Design and Performance Aspects

We compared three different lossless compression methods, the well-known bzip2 [2] and gzip [3] compressors as well as the lzo (Lempel-Ziv-Oberhumer) [1] real-time compressor. We did not consider lossy compressors. Bzip2 is slow and compresses very well, gzip is average in all regards and lzo is fast but does not compress well.

Direct comparison of the three compressors on network data shows that while the compression ratios are different, the changes in compressibility are very similar. Figure 5 gives an example plot that compares the compression statistics for destination IP addresses before and during the Witty worm outbreak (see Section 3). Because of its speed advantage lzo was selected as preferred algorithm for our work.

4.1. Reporting Latency

We aim at near real-time event reporting. So what reporting latency is to be expected? Data is processed in time intervals and every interval will result in entropy estimations for several parameters, such as the source IP address. The maximum reporting latency for an interval is therefore the interval length (since the event of interest can be at the start of the interval) plus the processing time for that interval.

Compression of a data stream can be done in two fashions. In *block mode* a larger amount of data, e.g. the sequence of all destination port numbers for a full measurement interval, is stored in memory and compressed in one step. In *streaming mode* data is processed as it is being read. (Streaming mode is equivalent to block mode with small block size in practice.) Streaming mode needs little memory besides the internal state of the compressor. If, as in our examples, several entropy statistics are calculated at the same time, then several compressor instance have to be used in parallel. (We determine 16 compression statistics in parallel: Source and destination IP and port statistics for four protocol classes.)

4.2. Sliding Window and Smoothing

Measurement interval length choice involves a trade-off. Short intervals give fast observation, but are sensitivity to short-term effects, e.g. memory issues on a router. Longer intervals smoothen out the resulting graphs, but cause a longer reporting latency. In our experiments we found that 1 minute intervals are too short, since the routers may export flows with up to 30
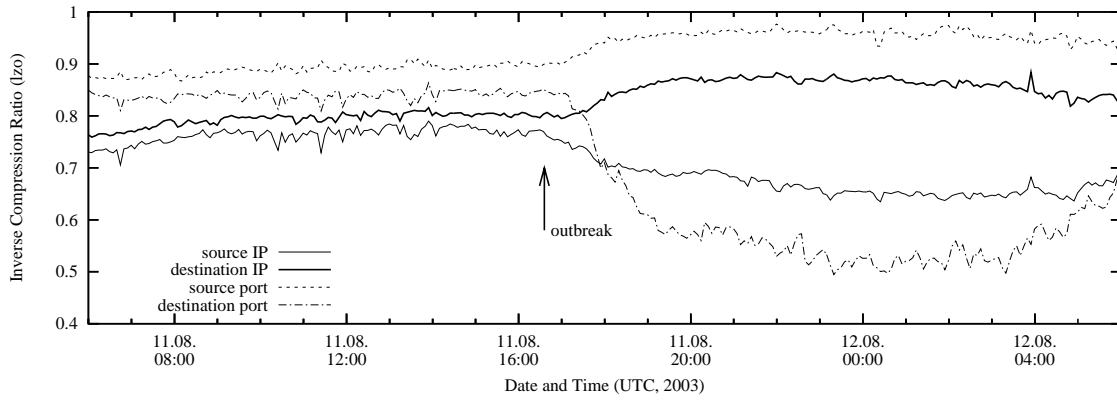
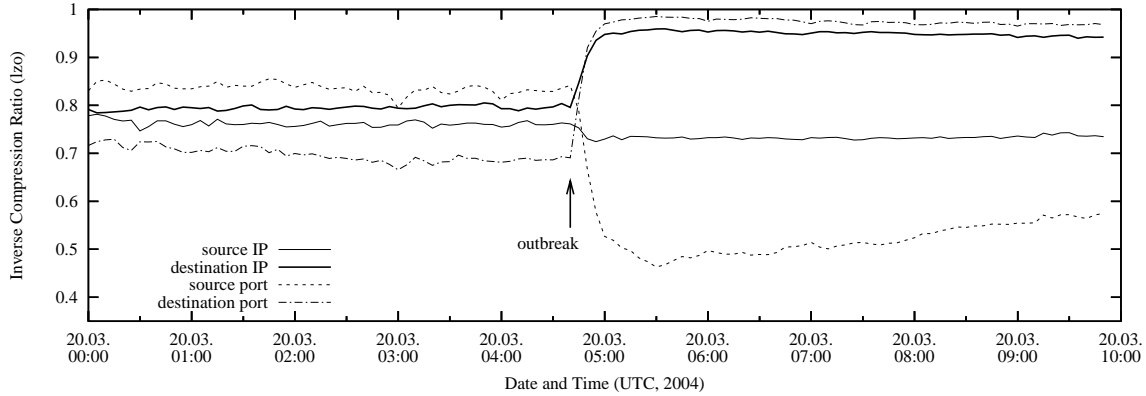**Figure 1. Blaster - TCP address parameter compressibility**

Figure 2. Witty - UDP address parameter compressibility

second delay. A middle ground can be found by using a *sliding window* approach, where, e.g. 5 minute intervals are used that are shifted by one minute against the neighbouring intervals.

4.3. Benchmark Results

All our measurements were done with Linux and a prototype implemented in C. The CPU figures given in Table 6 are for processing of source and destination IP addresses and ports for all flows, in non-overlapping measurement intervals of 5 minute length. All overhead, like parsing the raw flow records, is included.

Method (Library)	CPU time / hour (60'000'000 flows/hour)
bzip2 (libbz2-1.0)	169 s
gzip (zlib1g 1.2.1.1-3)	52 s
lzo1x-1 (liblzo1 1.08-1)	7 s

Figure 6. CPU time (Linux, Athlon XP 2800+)

The memory figures in Table 7 are for one compressor instance. In streaming mode they have to be multiplied with the number of concurrent analyses done. In block mode storage for at least one measurement interval of data has to be added.

| Method	Memory per instance
bzip2	7600 kB
gzip	256 kB
lzo1x-1	64 kB

Figure 7. Compression memory needs

4.4. Sampling

Sampling is a technique often found in NetFlow based network monitoring set-ups. Sampled data is easier to handle, since smaller, and accurate enough for accounting and network monitoring purposes. For relatively frequent sampling entropy based anomaly detection works well on SWITCH network data. Figures 3 and 4 show plots with input data randomly sampled at 1 in 20 input records. The figures were generated in the same fashion as Figures 1, 2.

It can be seen that the main effects remain intact, while the absolute values and amount of change during the outbreaks have changed. IP addresses and ports are noticeably shifted to a different degree. The Blaster plots show more "noise" in the pre-outbreak phase, especially in the port plots.
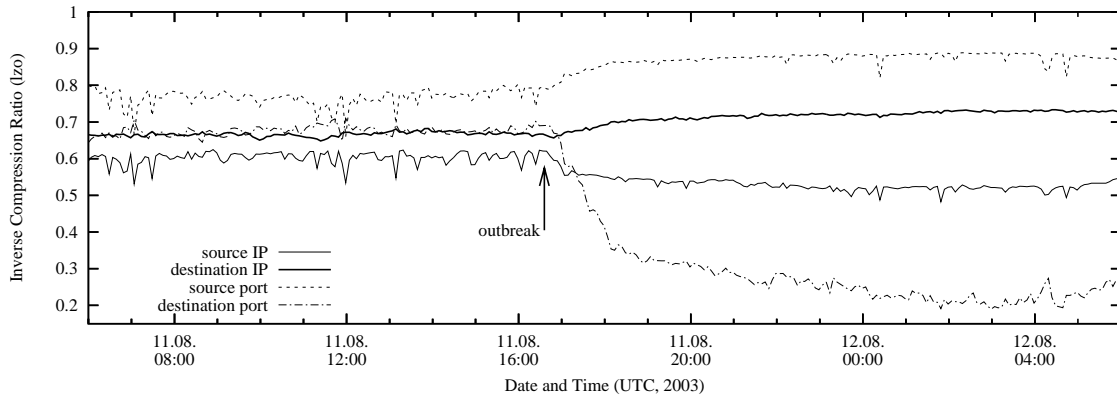
Figure 3. Blaster - TCP, randomly sampled at 1 in 20 flows
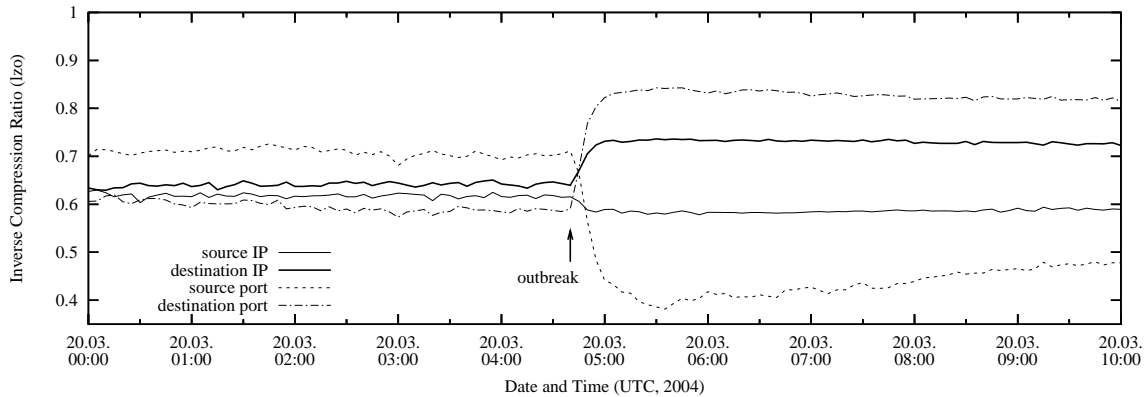
Figure 4. Witty - UDP, randomly sampled at 1 in 20 flows

4.5. Scalability

The compression algorithms all have time complexity linear in the input size. Memory usage remains constant if streaming mode is used. Lzo is efficient enough to analyse much larger numbers of flows per hour that the SWITCH network generates on a current PC. At around $3 \times 10^9 \ldots 10 \times 10^9$ Flows/hour (i.e. $50 \ldots 160$ times the SWITCH data rate) a typical 32bit/33MHz PCI bus becomes a bottleneck. At the upper end of this range Gigabit Ethernet gets too slow. Sampling in the routers should allow at least one order of magnitude more traffic to be processed on a current PC. All in all we feel confident that our analysis method scales very well.

5. Conclusion

We have presented and given a proof-of-concept for an entropy based worm detection method that can also detect other types of network events that involve large amounts of scanning traffic. The approach is completely generic and does not need parameterisation in order to be effective.

The presented method is most useful in combination with other, more targeted analysis techniques. It can generate first insights and is suitable for initial alarm-ing, but has limited analytic capability. We are currently investigating how the entropy-based approach can be used to generate a more detailed analysis of a massive network event, that goes beyond the work presented in this paper.

In our experiments we found that the proposed entropy based detection method does not work well for slow worms or small-scale attacks. While entropy changes can also be significant for slow worms, other detection methods will likely be faster and more conclusive for them. Entropy based detection of massive network events is a powerful, efficient and fast method to detect massive network events like worm outbreaks and to provide a first analysis of the characteristics of scanning-like traffic. As such it is very suitable for use in an early warning system. Its primary strength, is its generality. It will work for any type of fast scanning worm or other massive event with scanning-like characteristics, regardless of scanning method used or vulnerability exploited.

6. Related Work

The idea to use some entropy measurements to detect worms has been floating around the worm research community for some time. Yet we are not aware of any publication(s) describing concrete approaches, systems or measurements. The authors of this paper were
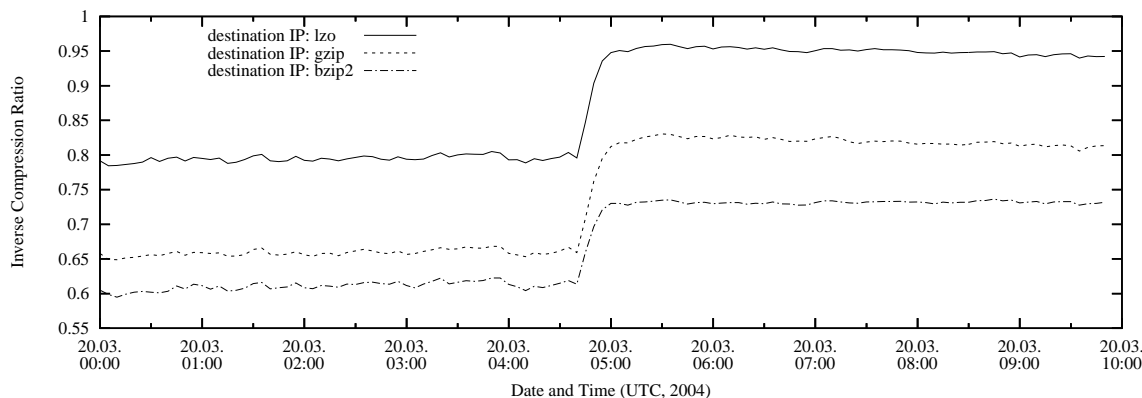
Figure 5. Witty - compressor comparison

prompted to investigate this idea by an observation on the Nachi [12, 7, 5] worm: Nachi generated about as many additional ICMP flow records as there were total flow records exported before the outbreak, yet the compressed size of the storage files increased only marginally.

In [19] the authors describe *behaviour-based clustering*, an approach that groups alerts from intrusion detection systems by looking at similarities in the observed packet header fields. The clusters are then prioritised for operator review. Principal Component Analysis is used in [14] to separate normal and attack traffic on a network-wide scale in a post-mortem fashion. Detection of exponential behaviour in a worm outbreak is studied in [8]. In [13] the authors study how worms propagate through the Internet.

7. Acknowledgements

The authors thank Thomas Dübendorfer and Placi Flury for valuable feedback and helpful discussion on the subject matter.

References

[1] http://www.oberhumer.com/opensource/lzo/. LZO compression library.

[2] The bzip2 and libbzip2 official home page. http://sources.redhat.com/bzip2/.

[3] The gzip home page. http://www.gzip.org/.

[4] The swiss education & research network. http://www.switch.ch.

[5] Mcafee: W32/nachi.worm. http://vil.nai.com/vil/content/v_100559.htm, August 2003.

[6] Symantec Security Response - W32.Blaster.Worm. http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html, 2003.

[7] W32.welchia.worm. http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.html, August 2003.

[8] C. C. Z. an L. Gao, W. Gong, and D. Towsley. Monitoring and Early Warning for Internet Worms. In *Proceedings of the 10th ACM Conference on Computer and Comminication Security*, 2003.

[9] CERT. Security Advisory: MS.Blaster (CA-2003-20). http://www.cert.org/advisories/CA-2003-20.html, 2004.

[10] Cisco. White Paper: NetFlow Services and Applications. http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm, 2002.

[11] DDoSVax. http://www.tik.ee.ethz.ch/~ddosvax/.

[12] H. Gabor Szappanos VirusBuster. Virus Bulletin: Virus information and overview - W32/Welchia. http://www.virusbtn.com/resources/viruses/welchia.xml, Apr. 2004.

[13] J. Kim, S. Radhakrishnan, and S. K. Dhall. Measurement and Analysis of Worm Propagation on Internet Network Topology. In *Proceedings of the International Conference on Computer Communications and Networks*, 2004.

[14] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *SIGCOMM*, pages 219–230, 2004.

[15] R. Lemos. MSBlast epidemic far larger than believed. http://news.com/MSBlast+epidemic+far+larger+than+believed/2100-7349_3-5184439.html, 2004.

[16] M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Verlag, second edition edition, 1997.

[17] O. Müller, D. Graf, A. Oppermann, and H. Weibel. Swiss Internet Analysis. http://www.swiss-internet-analysis.org/, 2004.

[18] C. Shannon and D. Moore. CAIDA: The Spread of the Witty Worm. http://www.caida.org/analysis/security/witty/, 2004.

[19] K. Theriault, D. Vukelich, W. Farell, D. Kong, and J. Lowry. Network traffic analysis unsing behaviour-based clustering. Whitepaper, BBN Technologies, http://www.bbn.com/docs/whitepapers/NetTrafficAn-Clustering-Theriault10-02.pdf.

[20] US-CERT. Vulnerability Note: Witty (VU#947254). http://www.kb.cert.org/vuls/id/947254, 2004.