

Host Behaviour Based Early Detection of Worm Outbreaks in Internet Backbones

Thomas Dübendorfer[†] Bernhard Plattner

Computer Engineering and Networks Laboratory (TIK)

Swiss Federal Institute of Technology, Zurich

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

Abstract

We propose a novel near real-time method for early detection of worm outbreaks in high-speed Internet backbones. Our method attributes several behavioural properties to individual hosts like ratio of outgoing to incoming traffic, responsiveness and number of connections. These properties are used to group hosts into distinct behaviour classes. We use flow-level (Cisco NetFlow) information exported by the border routers of a Swiss Internet backbone provider (AS559/SWITCH). By tracking the cardinality of each class over time and alarming on fast increases and other significant changes, we can early and reliably detect worm outbreaks. We successfully validated our method with archived flow-level traces of recent major Internet e-mail based worms such as MyDoom.A and Sobig.F, and fast spreading network worms like Witty and Blaster. Our method is generic in the sense that it does not require any previous knowledge about the exploits and scanning method used by the worms. It can give a set of suspicious hosts in near real-time that have recently and drastically changed their network behaviour and hence are highly likely to be infected.

1. Introduction

As we currently approach one billion Internet users [14], more and more cyber criminals join in and misuse this worldwide network by setting free malicious worm code that infects hosts and aggressively spreads over the network. Surprisingly, many large operators of Internet backbones still do not see enough incentives for deploying security elements and for analysing their network traffic proactively in near real-time for new massive security incidents like worm outbreaks. Security in the Internet is mostly regarded as being the duty of the network users that should protect their end systems by a firewall and a virus scanner.

Based on the observation that hosts infected by the same worm execute the same code for scanning and transferring exploit and worm code, we assume that during a worm outbreak the network behaviour of many hosts will suddenly change in a similar way. In this paper, we propose a novel near real-time method for early detection of worm outbreaks in high-speed Internet backbones. By analysing backbone traffic at flow-level, we can attribute various behavioural properties to hosts like ratio of outgoing to incoming traffic, responsiveness and number of connections, which all are strongly influenced by a worm outbreak. These properties are used to group hosts into distinct classes according to their current behaviour. We show that by tracking the cardinality* of these classes for significant changes over time, worm outbreak events can reliably be detected and a set of potentially infected hosts can be identified.

The outline of this paper is as follows: After a survey of related work in Section 2 and NetFlow traces in Section 3, we present in Section 4 our host behaviour based worm detection method. In Sections 5 and 6, we validate our method on NetFlow traces of real e-mail and network worms, and show corresponding cross-checks with “non-worm” Internet days. Finally, we draw our conclusions in Section 7.

2. Related Work

Intrusion detection for local area networks is a well established research area. In 1990, the Network Security Monitor [9] was one of the first intrusion detection tools that implemented the “connection counter” algorithm of the University of California in Davis. This algorithm is used to identify infected hosts and is based on the observation that worm infected hosts normally open connections at higher rates than uninfected ones. Monitoring bandwidth usage on network links and alerting upon reaching a threshold by statistical-based intrusion detection systems is similar to our traffic class cardinality tracking but less accurate.

[†]Partially funded by the Swiss Academic Research Network.

*“Cardinality” denotes the number of hosts in a single class.

Intrusion detection in backbone networks is a rather new research area. For worm detection in the global Internet, methods based on distributed intrusion detection systems like NetBait [6], firewall logs, or honeypots or a detection method using ICMP error messages [2] have been published. Most detection methods proposed for local area networks require packet payloads, which are expensive to collect and process in high-speed networks. Even analyses of real worms in backbones are extremely rare due to the required large efforts of handling such data and due to privacy concerns. In the DDoSVax [7] project, we analysed major Internet worm outbreaks in the AS559 backbone on flow-level. CAIDA analysed backscatter traffic of the Code Red, Slammer and Witty worms collected in a large unused IP address space with their Network Telescope [13]. Our proposed method is based on flow-level information of backbone routers and does not need packet payloads. Worm detection in backbones is a challenge: It has to be efficient for large traffic volumes, and there is no detailed information such as installed software about the active hosts, which e.g. is strongly relied on by many commercial intrusion detection systems for local area networks.

3. Flow-Level Backbone Traffic

We used flow-level data from the AS559 (SWITCH) backbone border routers that is captured in the DDoSVax [7] project since March 2003. In an average hour on a workday, network traffic between approximately 200'000 SWITCH-internal hosts and of approximately 800'000 hosts from outside the SWITCH network can be observed. The majority of internal hosts is operated by SWITCH's customers, who are Swiss universities, colleges of higher education, and international research labs. In Cisco's NetFlow v5 format that we use, consecutive network packets in the same direction between the same two hosts (i.e. IP addresses) using the same protocol (ICMP, UDP, TCP, others) and port numbers are reported as a single flow record together with the number of packets, total number of bytes in the IP layer, start and end time of the flow. In an average hour, approximately 60 mill. flows are reported from all SWITCH border routers. Our NetFlow records contain no TCP flags due to router restrictions.

4. Host Behaviour Based Worm Detection

4.1. Method

From manual analyses of flow-level worm traffic, we gained insights into characteristic network traffic during worm outbreaks. It seems that the classical client/server paradigm does not hold for Internet hosts and grouping

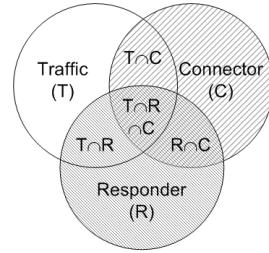


Figure 1. Host Behaviour Classes

Class	Parameter	Threshold
Traffic	(Bytes sent)/(Bytes received)	>3/min
Responder	bidirectional connections	>1/min
Connector	outgoing connections	>10/min

Table 1. Threshold Conditions for Host Behaviour Core Classes

hosts into clients and servers is unwise. Popular P2P services, uploads of large files or the use of active FTP have intermixed client and server roles for Internet hosts. For our detection method, we define three core host behaviour classes "Traffic", "Connector" and "Responder" that overlap each other as shown in Figure 1. The observed flow-level traffic of a host must satisfy a given threshold condition within a fixed time interval (1 minute in this paper) in order to be member of a core class. We list the threshold parameters and the values used for all behaviour class plots in this paper in Table 1. These classes were chosen such that a host's membership in these classes is more likely if it was infected by a worm and less for hosts exhibiting "normal" network usage. The given parameter values proved useful for AS559 but might need adjustments for different backbones. During a worm outbreak, we expect a sudden and significant increase in the cardinality of one or several classes.

Hosts in the "Traffic" core class send several times more traffic than what they receive. This is typical for e.g. worms that send out exploit code or that spread in e-mail attachments. Hosts in the "Responder" core class hold bidirectional connections. We define a bidirectional connection as a pair of flows in opposite directions between a pair of hosts, where the start time of the flows falls within an interval of less than 50 ms. Hosts that respond to TCP scans or handshake initiations typically have such bidirectional connections during a worm outbreak. Hosts in the "Connector" core class initiated many outgoing connections that are not necessarily bidirectional, which is typical for hosts that scan others.

The possible overlap of the core classes results in a total of eight distinct host behaviour classes that a host can be in: T (raffic), R (esponder), C (onnecter), $T \cap C$, $T \cap R$, $R \cap C$, $T \cap C \cap R$, and "no class".

4.2. Implementation

To accommodate large volumes of NetFlow data, a filter stage that selects flows by protocol type (one or several of TCP, UDP or ICMP) and optionally by source and destination port numbers was prepended to our algorithm. We call it a near real-time algorithm as there are several small delays in the traffic processing path: network packets in the routers are aggregated to flow records, which are exported every four seconds, these flows are collected and preprocessed by our algorithm during a one minute interval and at the end of each interval, the class cardinalities are finally calculated.

The algorithm accepts incoming flow records passing the filter for the current and the next one minute interval. For each interval, a hash table stores the hosts seen together with the parameters for the amount of traffic sent and received, and the number of outgoing connections. Bidirectional connections are handled with nested hash tables to achieve fast lookups and to minimise memory requirements. A hash table stores the source IP address for each observed host. A lookup of such a host returns a hash table with all flows originating at this host in the current interval. An efficient lookup by a hash key of destination IP address, source and destination port in this returned hash table is used to match a new flow with an existing one in the opposite direction.

The implementation of the algorithm was done in C as a plug-in for UPFrame [15], our open source UDP packet processing framework that was designed for near real-time processing of bursty NetFlow data. To ensure robust operation during attacks, an upper plug-in memory limit was set. If more memory than the limit were needed, new flow records for the current interval are discarded and an error code for the interval is reported. When flows for the next time interval arrive, the algorithm resumes its operation. This ensures that the plug-in can also handle large-scale attacks without crashing. The memory consumption of the plug-in can be roughly estimated in bytes by $\text{memory} = \text{number of active intervals} \cdot (\text{number of unique IP addresses per interval} + 209 + \text{number of flows not discarded by protocol/port filter} \cdot 44)$. There is some additional management overhead of approx. 6 MB. As an example, if we consider 200'000 active unique hosts and 1 mill. flows per minute, in total 170 MB of main memory are used. The plug-in can also detect missing data (e.g. due to an interruption of incoming NetFlow records) and will automatically start a new time interval if it suddenly receives NetFlow records that carry timestamps outside the currently observed two time intervals. The output of the plug-in is one binary statistic file of 6 KB for each hour of NetFlow data processed that contains all class cardinalities and an operational state (i.e. missing data, memory limit reached etc.) for each of the 60 minutes.

5. Validation on E-Mail Worms

In this section, we analyse the detection effectiveness of our algorithm on two major Internet e-mail worms, namely Sobig.F (2003) and MyDoom.A (2004). We applied our host behaviour algorithm to SMTP traffic, i.e. we considered only flows to and from port 25/TCP.

5.1. Sobig.F

On August 19th, 2003, around 9:00 UTC [8], the Sobig.F [4] e-mail worm that runs on Microsoft Windows 95 or higher first appeared in the Internet. Besides spreading via executable e-mail attachments of varying sizes and providing its own SMTP engine for sending infected e-mails, the worm is programmed to update itself at predefined dates by downloading new code from predefined computers. By timely intervention of network operators and system administrators this update mechanism could be blocked by shutting down all 20 predefined servers. The original worm was programmed to disable itself on September 10th, 2003. Date and time are taken from a small set of hardcoded global NTP time servers. The e-mails sent use an arbitrary sender and recipient address taken from local files of the infected host.

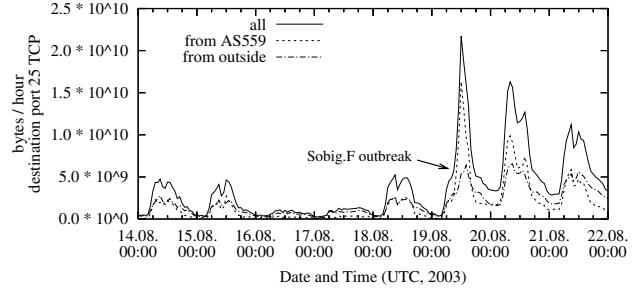


Figure 2. Sobig.F caused an almost fivefold increase of e-mail traffic volume per hour

The graph in Fig. 2 shows the total number of bytes per hour transferred as e-mail (SMTP) traffic over the SWITCH border routers. The daily rhythm can clearly be seen. The workdays have traffic with a maximum around 5 GB/hour, on Saturdays and Sundays the traffic is considerably less. On a workday, the lunch break can easily be identified. The peak of the Sobig.F outbreak is reached on Tuesday, August 19th, 2003 in the hour of 12:00-13:00 UTC with 21.7 GB/hour, which is almost a fivefold increase compared to “normal”.

By tracking the cardinality of the host behaviour classes $T \cap C$ in Figure 3 and $C \cap R$ in Figure 4, the outbreak could have been detected early around 10:00 UTC by comparing with traffic two weeks before (see lower plots in the figures) or by cardinality threshold alerts. A fast reaction (e.g.

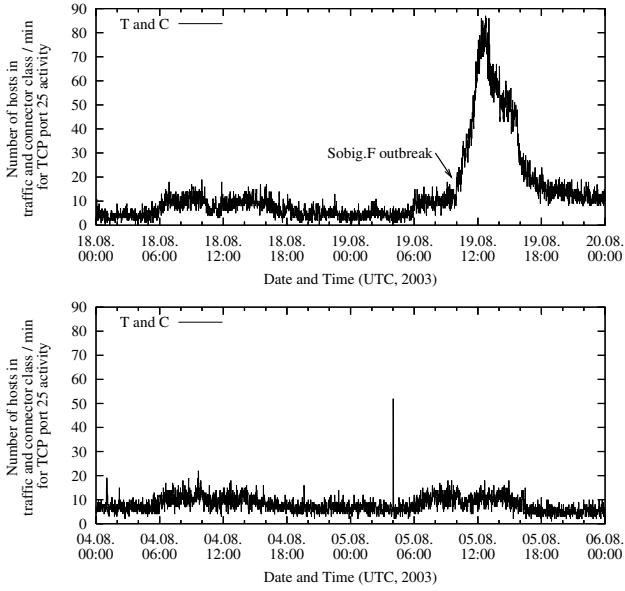


Figure 3. Cardinalities of the “Traffic \cap Connector” class two weeks before and during Sobig.F

timely information to users about the new e-mail worm and e.g. traffic filter rules to block new SMTP senders) could have strongly mitigated the impact.

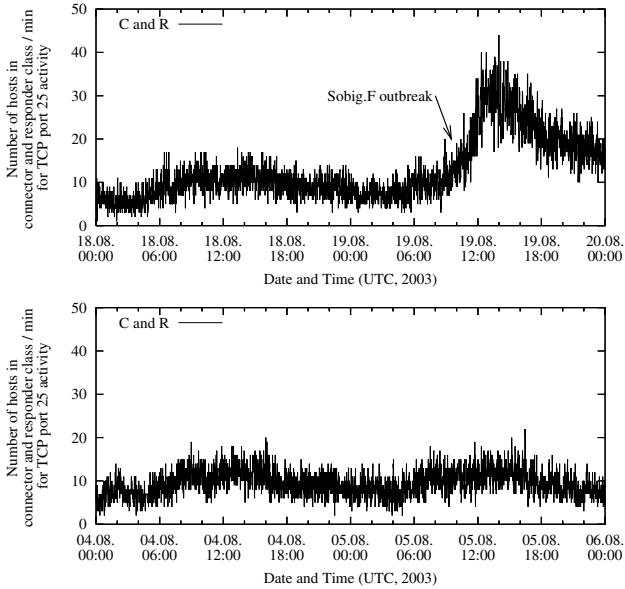


Figure 4. Cardinalities of the “Connector \cap Responder” class two weeks before and during Sobig.F

5.2. MyDoom.A

On January 27th, 2004, the MyDoom.A [5] (a.k.a. W32/Novarg.A) e-mail worm began to massively spread worldwide. It spreads via executable e-mail attachments and provides its own SMTP engine for sending infected e-mails. It uses various different attachment file extensions such as .scr, .exe, .pif, .cmd, .bat, or .zip. An arbitrary recipient and sender are taken from e-mail addresses found on the infected host. The recipient must explicitly open the malicious attachment to infect the system. Additional “features” of this worm are that it copies itself to the local KaZaA P2P file sharing directory if available and that it starts a DDoS attack on www.sco.com after a reboot on Feb 1st 16:09:18 UTC. After Feb 12th, this attack ceases. The worm also installs a backdoor on ports 3127-3198/TCP, which remains active.

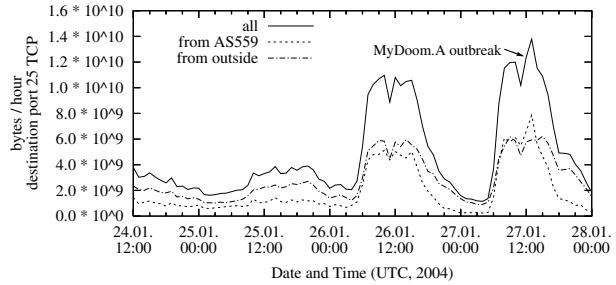


Figure 5. MyDoom.A: E-mail traffic volume

The graph in Fig. 5 shows the total number of bytes per hour transferred as e-mail (SMTP) traffic over all SWITCH border routers. The daily rhythm can clearly be seen. The two workdays in the graph show e-mail traffic with a maximum around 11 GB/hour, whereas on Saturday and Sunday the traffic is considerably less as can be easily explained by the absence of business related e-mails. On a workday, the lunch break can easily be identified. The additional MyDoom.A e-mail traffic elevated the traffic volume level of e-mails since early on Tuesday 27th. Its spreading became evident in the traffic volume statistic just after lunch time as indicated with an arrow in the graph.

During the spreading of MyDoom.A, we observed an increase of 14% (for Tue, 27th Jan 2004) to 30% (for Wed, 28th Jan 2004) in e-mail traffic volume in bytes per hour. Compared to the e-mail worm Sobig.F of August 2003, which caused an almost fivefold increase in e-mail traffic load within the initial hour of its spreading, the MyDoom.A worm seems to be quite harmless.

A prevalent behavioural pattern of MyDoom.A is that new hosts become active by sending e-mails and that existing hosts (like NAT gateways) send massively more e-mails. This fact can clearly be seen in our behaviour class analysis in Figure 6. Comparing the cardinality of host behaviour classes to just before the outbreak or to the same

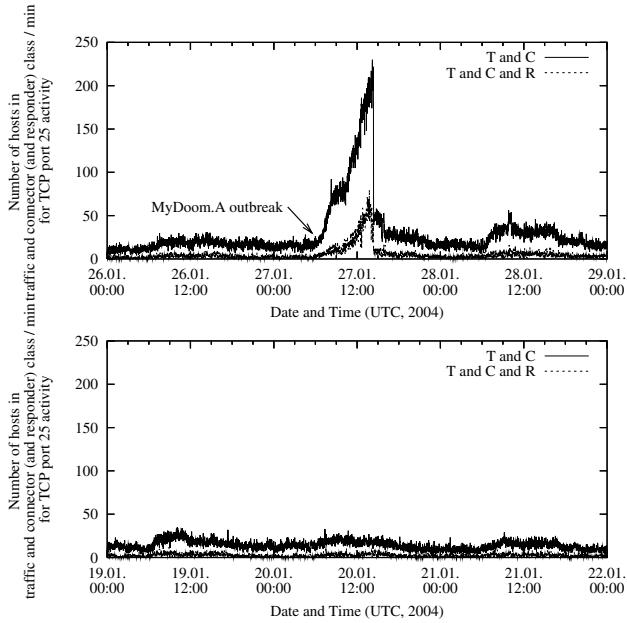


Figure 6. Cardinality of the $T \cap C$ and $T \cap C \cap R$ classes before and during MyDoom.A

days of the previous week (lower graph in Figure 6), the outbreak becomes evident. In contrast to the volume statistic, in which the outbreak becomes evident around 11:00 UTC, the outbreak could have been noticed already around 8:00 UTC if such an early worm detection tool had been in place. The sudden drop in the cardinality of the observed classes in the afternoon was due to manually installed SMTP traffic blocking filters in some networks that were strongly affected by this worm. Possible reasons for the rather moderate e-mail worm traffic volume of MyDoom.A in comparison to Sobig.F's explosive traffic volume are a considerably smaller worm size (22 KB vs. 71 KB), public awareness of e-mail worms, up-to-date anti-virus software, and firewall SMTP filtering.

As seen, our method can effectively and reliably detect fast spreading e-mail worms in the early stage of their outbreak by tracking the $T \cap C$, $C \cap R$ and possibly also the $T \cap C \cap R$ behaviour classes for significant changes. We presented only the classes with the most significant changes in the plots. Watching merely the cardinalities of any of the three core classes T , C or R would not be sufficient.

6. Validation on Network Worms

In this Section, we validate our method on fast spreading network worms, namely Witty and Blaster. Our algorithm does not need previous knowledge about the specific ports used or the type of the exploited vulnerability in order to

detect such worms. We merely need to track the cardinality of the classes for all UDP (Witty) or all UDP and TCP traffic (Blaster) for detecting the outbreaks.

6.1. Witty worm

The Witty worm [17] was first observed in the Internet on Friday, March 20th, 2004, at approximately 4:45 UTC. It exploits a bug in several products by Internet Security Systems (ISS [10]). Witty uses UDP packets with source port 4000 and a randomised target port. The vulnerable host population for Witty was approximately 12'000 hosts. Witty was the first fast worm which demonstrated that even a small population can be infected in a few hours. Unlike most other worms, it carried a destructive payload for deleting arbitrary blocks on an infected computer's harddrive.

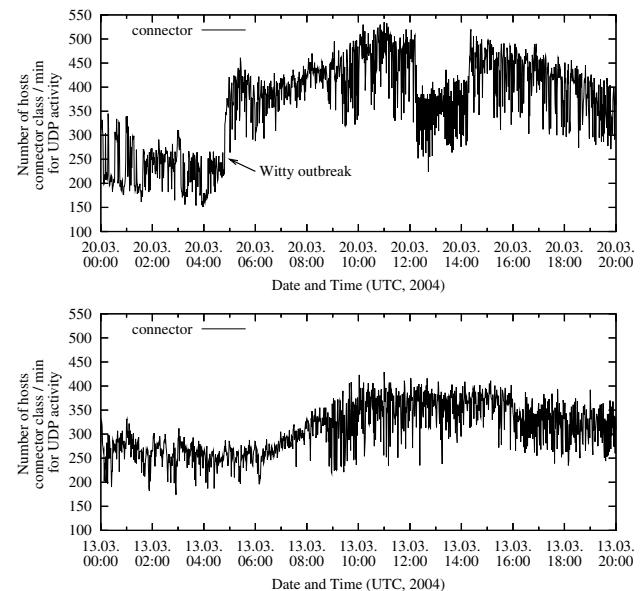


Figure 7. Cardinality of the “Connector” class before and during Witty

Hosts infected by Witty created many outgoing UDP flows (“connections”) and show up clearly in the “connector” class plot in Figure 7. Around noon, it seems that for almost three hours a filter was installed, which was later deactivated again. By comparing the Connector class cardinality of the “Witty day” with the regular “Witty-free” Saturday one week before (lower graph in Figure 7) the worm outbreak can be seen even clearer.

6.2. Blaster worm

The W32.Blaster [3] worm was first observed in the Internet on August 11th, 2003, with a massive outbreak start-

ing 16:35 UTC [8]. The worm exploited a remote procedure call (RPC) vulnerability of Microsoft Windows 2000 and Windows XP operating systems that was made public in July 2003 [16] and is described in the Microsoft Security Bulletin MS03-026 [12] as critical. Estimates of Blaster infected hosts range from 200'000 [1] to 8 millions [11].

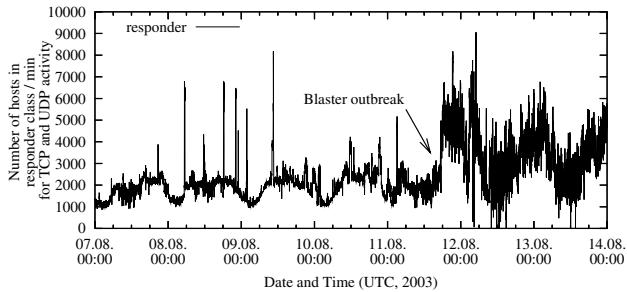


Figure 8. Cardinality of the “Responder” class before and during Blaster.

The host behaviour method notices Blaster activity by a huge increase of approx. 4500 in the number of hosts responding per minute and a higher fluctuation rate (for any TCP and UDP flows). Previously silent and client-like hosts suddenly start to respond to connection requests. This is due to infection attempts by Blaster, mostly on port 135/TCP. We attribute the short peaks in the number of responders before the Blaster outbreak to TCP scanning activities. A “Connector” class analysis of Blaster revealed a limited spreading Pre-Blaster variant in the late afternoon on August 10th of 250 additional hosts, 160 of which also connected to 69/UDP that was used by Blaster for worm code transfer. The cardinality of the Traffic class or of hosts in a combination of classes did not show significant changes during the Blaster outbreak.

7. Conclusions and Outlook

We presented a generic near real-time method for early detection of worm outbreaks in high-speed Internet backbones. By classifying the network behaviour of Internet hosts in a way that is highly sensitive to newly infected hosts, and by tracking the cardinalities of these classes and alarming on sudden and rapid increases, we can warn early of worm outbreaks. We successfully validated our method on real flow-level backbone traffic of four major past Internet worms. We showed that our method works well for e-mail based worms like Sobig.F and MyDoom.A as well as for fast spreading network worms like Witty and Blaster. Future work will be to establish automatic alarming mechanisms based on class cardinality tracking and adjust them for an optimal security incident sensitivity.

8. Acknowledgements

We thank our student Bernhard Tellenbach for the initial plug-in implementation, Placi Flury and Arno Wagner for fruitful discussions and SWITCH for continuously providing us with NetFlow data.

References

- [1] <http://www.sans.org/>.
- [2] V. Berk, G. Bakos, and R. Morris. Designing a Framework for Active Worm Detection on Global Networks. In *Proceedings of the first IEEE International Workshop on Information Assurance (IWIA’03)*, March 2003, Darmstadt, Germany, Mar. 2003.
- [3] CERT. Advisory CA-2003-20 W32/Blaster worm. <http://www.cert.org/advisories/CA-2003-20.html>, 2003.
- [4] CERT. Incident Note IN-2003-03 W32/Sobig.F. http://www.cert.org/incident_notes/IN-2003-03.html, 2003.
- [5] CERT. Incident Note IN-2004-01. http://www.cert.org/incident_notes/IN-2004-01.html, Jan. 2004.
- [6] B. Chun, J. Lee, and H. Weatherspoon. Netbait: a distributed worm detection service. <http://netbait.planet-lab.org/>, 2003.
- [7] DDoSVax - In Search of a Vaccine against DDoS Attacks. <http://www.tik.ee.ethz.ch/~ddosvax/>.
- [8] T. Dübendorfer, A. Wagner, T. Hossmann, and B. Plattner. Flow-level Traffic Analysis of the Blaster and Sobig Worm Outbreaks in an Internet Backbone. In *Proceedings of DIMVA 2005, LNCS 3548*, July 2005.
- [9] L. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber. A Network Security Monitor. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pp. 296-304, 1990.
- [10] ISS. Internet Security Systems. <http://www.iss.net/>, 2004.
- [11] R. Lemos. MSBlast epidemic far larger than believed. http://news.com.com/MSBlast+epidemic+far+larger+than+believed/2100-7349%_3-5184439.html, 2004.
- [12] Microsoft Corporation. Microsoft Security Bulletin MS03-026. <http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx>, 2003.
- [13] D. Moore. CAIDA’s Network Telescope. <http://www.caida.org/analysis/security/telescope/>, 2004.
- [14] How many online worldwide? <http://www.era.ro/howmanyonline.htm>, 2004.
- [15] C. Schlegel and T. Dübendorfer. UPFrame - A generic open source UDP processing framework. <http://www.tik.ee.ethz.ch/~ddosvax/upframe/>, January 2005.
- [16] The Last Stage Of Delirium. Buffer Overrun in Windows RPC Interface. <http://lsd-pl.net/special.html>, 2004.
- [17] US-CERT. Vulnerability Note: Witty (VU#947254). <http://www.kb.cert.org/vuls/id/947254>, 2004.