

Flow-Level Traffic Analysis of the Blaster and Sobig Worm Outbreaks in an Internet Backbone

Thomas Dübendorfer*, Arno Wagner**, Theus Hossmann, and Bernhard Plattner

Computer Engineering and Networks Laboratory (TIK),
Swiss Federal Institute of Technology, ETH Zurich
{duebendorfer, wagner, plattner}@tik.ee.ethz.ch,
hossmath@ee.ethz.ch
<http://www.tik.ee.ethz.ch/~ddosvax/>

Abstract. We present an extensive flow-level traffic analysis of the network worm Blaster.A and of the e-mail worm Sobig.F. Based on packet-level measurements with these worms in a testbed we defined flow-level filters. We then extracted the flows that carried malicious worm traffic from AS559 (SWITCH) border router backbone traffic that we had captured in the DDoSVax project. We discuss characteristics and anomalies detected during the outbreak phases, and present an in-depth analysis of partially and completely successful Blaster infections. Detailed flow-level traffic plots of the outbreaks are given. We found a short network test of a Blaster pre-release, significant changes of various traffic parameters, backscatter effects due to non-existent hosts, ineffectiveness of certain temporary port blocking countermeasures, and a surprisingly low frequency of successful worm code transmissions due to Blaster's multi-stage nature. Finally, we detected many TCP packet retransmissions due to Sobig.F's far too greedy spreading algorithm.

1 Introduction

In this paper, we examine worm behaviour from a network centric view based on one of the very rare real backbone traffic measurements of the actual worm spreading events. We analyse two major recent Internet worms: Blaster.A [1], that exploits the Microsoft Windows Remote Procedure Call DCOM vulnerability and which spreads without any user interaction and Sobig.F [2], a worm that installs its own SMTP-engine and propagates as e-mail attachment, which has to be executed by the user for an infection.

The remainder of this paper is organised as follows: We describe our measurement setup and survey related work in the rest of Section 1. In Section 2, the infection steps of Blaster and associated network traffic on packet and flow-level is analysed. In Section 3, we discuss our measurements of Sobig.F related e-mail traffic. Finally, we give our conclusions in Section 4.

* Partially funded by the Swiss Academic Research Network (SWITCH).

** Partially funded by the Swiss National Science Foundation under grant 200021-102026/1 and SWITCH.

1.1 Backbone Measurement Setup

In the DDoSVax [3] project at ETH Zurich [4], we are capturing the complete flow-level (Cisco NetFlow v5) traffic of all border routers of AS559 (SWITCH) since March 2003. The SWITCH network connects all Swiss universities (ETH Zurich, EPFL, University of Zurich, University of Geneva, University of St. Gallen HSG etc.), various research labs (CERN, PSI, IBM research etc.), federal technical colleges and colleges of higher education (Fachhochschule ZHW, FHA etc.) to the Internet. The IPv4 address ranges of AS559 and its customers comprise roughly 2.2 million IP addresses. The AS559 backbone carries about 5% of all Swiss Internet traffic [5] or roughly 300 Gigabytes in about 60 million flows per hour. The size of SWITCH is large enough to get a relevant view of Internet traffic activity and small enough such that captured unsampled traces can still be handled rather efficiently.

Cisco's NetFlow [6] format version 5 that we use defines a "flow" as a *unidirectional* stream of packets from one host to another. A flow is reported as a tuple of source/destination IP address, source/destination port, IP protocol type (i.e. TCP, UDP, other), packets/flow, number of network layer bytes/flow, time of first/last packet in this flow, routing-specific and other parameters without any TCP/UDP packet payload.

AS559 transit traffic was excluded from our Blaster.A analysis and ignored in the Sobig.F analysis. Traffic routed through several border routers is reported more than once. We eliminated such flow duplicates by counting flows with the same source and destination IP addresses and ports only once within 50 ms. A different method would be to use Bloom filters [7] for this elimination. It is possible that partial loss of NetFlow data during aggregation in the routers and other worm-unrelated larger network events introduced distortions into the plots presented. As we captured all NetFlow traffic exported by the routers and as no other major network events during the analysed time periods were reported publicly, we believe these effects to be small. Another limitation is that no TCP flags are reported in our traces due to constraints in the routers' hardware-based NetFlow engines.

1.2 Related Work

All major anti-virus software vendors published analyses of the Blaster worm code on their web sites (e.g. Symantec [8], Trend Micro [9]) based on a host centric view of the worm behaviour. We made use of this information to crosscheck our own measurements with the real worm executables in our testbed.

José Nazaris from Arbor Networks describes in [10] some plots of Blaster traffic and explicits the effects of Blaster on routing. Symantec has analysed in [8] the infection rate of Blaster in the days after its initial outbreak.

Long-term archives of network backbone measurement data as we used it for our analyses are rare and difficult to get access to due to privacy laws, data security concerns, the challenge and costs of handling large amounts of real-time statistics data and the possibility of interference with current network operations and accounting.

There are many special-purpose and mostly commercial tools [11] available for processing NetFlow data. Some open source NetFlow tools such as SiLK [12] also exist. Many network operators use such tools to collect NetFlow data for accounting and network planning purposes. They often use only a small sample of all flow records (e.g.

1/400 of all records) and rarely store them for a longer time. We know from several Internet Service Providers and from the network services at ETH that their commercial software used for real-time network monitoring crashed during the Blaster outbreak (mainly due to out of memory errors as a consequence of a huge network activity increase). For long-term capturing and analysis of large amounts of NetFlow data, software is rare. We used the tools developed in our DDoSVax project for capturing and data processing.

The University of California, San Diego (UCSD) operates a “Network Telescope”, which is a /8 subnet that a team of the Cooperative Association for Internet Data Analysis (CAIDA) [13] uses to analyse backscatter traffic of worms and attacks. With this measurement setup one can mostly see traffic due to spoofed source IP addresses and scanning activities. However, traffic of successful infections of productive hosts (especially if a worm uses multiple steps for an infection like Blaster) are not visible in such a passive network setup. They published analyses of the worms Code-Red, Slammer and Witty [14] but nothing on Blaster or Sobig.F.

Research on intrusion detection systems (IDS) was done for more than twenty years. However, in an IDS usually a lot about users, resources, running services, and other installed software of the hosts under attack is known unlike to our backbone measurements. Most IDS research focuses on access networks and does not deal with the specifics of flow-level cross-border traffic in backbones.

Several mathematical models [15, 16, 17, 18] were proposed that simulate and predict worm propagation behaviour. However, they do not model effects due to network operators intervening during the outbreak, their parameters must be carefully adjusted to each new worm and they are valid mostly only for the very early spreading stage. Due to the scarcity of in-depth analyses of real worm measurements in the backbone, very little about real worm behaviour in the Internet is known.

2 Blaster

Blaster is a multi-stage worm: for a successful infection, six sequential steps, which involve traffic on three specific ports, must be completed. We analysed Blaster for the interplay of infection steps and associated network traffic. We gradually added new restrictions to our traffic filters. This allowed us to differentiate how many infection attempts there were, how many were partially successful up to a specific stage and finally how many were successful. In addition, we analysed our traffic traces for further anomalous behaviour in relation to the Blaster worm.

2.1 Outbreak

On August 11th 2003, the W32.Blaster [1] worm was first observed in the Internet. In April 2004, Microsoft estimated the number of all Blaster infected systems since the outbreak to be at least 8 million [19], whereas the Internet Storm Center stated that based on their evaluations of firewall logs provided by thousands of volunteers between 200'000 and 500'000 computers had been infected.

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 by

the “Last Stage of Delirium Research Group” in [20] and that is described as critical in the Microsoft Security Bulletin MS03-026 [21]. The same vulnerability (which requires a slightly different exploit code) is present in Windows NT 4.0 and 2003. However, these systems were not targeted by the main Blaster variant Blaster.A. An infection of a Windows host by Blaster can be prevented by using a firewall that blocks traffic incoming to port 135/TCP and by applying the operating system patch against this RPC vulnerability.

2.2 Worm Variants

As no commonly agreed rule exists for worm and virus naming, W32.Blaster.A (Symantec) is also known as W32/Lovesan.worm.a (McAfee), Win32.Poza.A (CA), Lovesan (F-Secure), WORM_MSBLAST.A (Trend), W32/Blaster-A (Sophos), W32/Blaster (Panda) or Worm.Win32.Lovesan (KAV). Besides the A version of Blaster, many more variants were developed based on the same exploit code. They differ in the name of the executable or have changed or added mostly malicious functionalities.

2.3 Blaster’s Infection Steps

Measurements of Blaster.A infected computer activity in our testbed network supported the machine code analysis described in [22]. The following description holds for Blaster.A, all other variants work very similar. The illustration in Figure 1 shows Blaster’s infection steps with a focus on network flows that can be observed. The following subsections use the same numbering as Figure 1 and explain each infection step in detail.

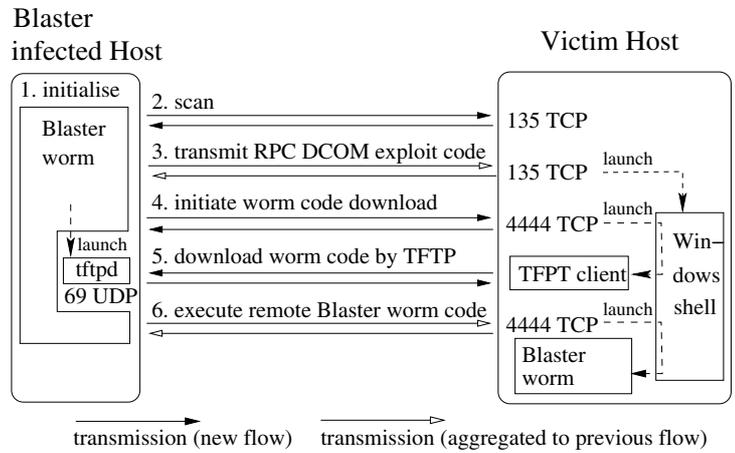


Fig. 1. Blaster’s infection steps

Step 1: Worm Initialisation. When Blaster is launched, it opens a mutex called “BILLY” that is used to prevent multiple infections of the same machine and sets a registry key to assure it is restarted upon each reboot. Then it checks the date. If the

current day is the 16th or later or if the current month is from September to December it starts a TCP SYN flooding attack against windowsupdate.com with a spoofed source address, which consists of the two first bytes of the local address and the two last bytes generated at random. This attack was not successful because Microsoft could simply stop the DNS forwarding from windowsupdate.com to windowsupdate.microsoft.com. We did not further analyse this attack.

Step 2: Victim Scanning on Port 135/TCP. In Blaster’s initialisation phase, the worm decides whether it will use the exploit code for Windows XP (80% probability) or the one for Windows 2000 (20% probability). According to Symantec [8] the worm then generates an IP address to start scanning as follows: With probability 60%, an IPv4 address of the form $X.Y.Z.0$ with X , Y and Z chosen at random is used. With probability 40%, an address of the form $X.Y.\tilde{Z}.0$ derived from the infected computer’s local address $X.Y.Z.U$ is chosen. \tilde{Z} is set to Z unless Z is greater than 20, in which case a random value less than 20 is subtracted from Z to get \tilde{Z} . Blaster always scans blocks of 20 sequential IP addresses simultaneously. The destination IP address value is incremented by one after each scan.

Step 3: Transmission of RPC Exploit Code. If a TCP connection to destination port 135 can be opened, the exploit code is sent to the victim. If it was vulnerable and the correct exploit code was sent, a Windows command shell process is started that listens on port 4444/TCP and allows remote command execution. Unpatched Windows XP computers automatically reboot within one minute after the RPC exploit code is executed.

According to our measurements with a Blaster.A infected computer in our testbed, the exploit code is sent as a remote procedure call (RPC) “bind” (72 bytes), an RPC “request” (1460 bytes) and a TCP packet (244 bytes). Summing these values up and adding the size of the headers (40-48 bytes for TCP/IP without respectively with TCP options) and also counting the two packets for the TCP handshake, we get 1976 to 2016 bytes for the RPC exploit code.

Step 4: Initiation of Worm Code Download. Blaster then initiates a TCP connection to port 4444/TCP. If successful, the command “`tftp -i attacker-IP GET msblast.exe`” is executed to start a Trivial File Transfer Protocol (TFTP) download of `msblast.exe` from the Blaster-infected host. Windows has the TFTP client `tftp` installed by default.

Step 5: Download of Worm Code by TFTP. If the remote download initiation was successful and the victim’s TFTP requests are not blocked (e.g. by a firewall), the Blaster-infected host is contacted on port 69/UDP for a download of the worm code. The size at the TCP layer of the Blaster.A worm code is 6176 bytes. In our own measurements with a Blaster.A infected computer, this code was transmitted in 12 TFTP packets of 512 bytes each and a 13th one of 32 bytes. Accounting for each TFTP packet 32 bytes for IP/UDP/TFTP headers, we get 6592 Bytes on the IP layer.

Step 6: Blaster Worm Code Execution. Finally, the Blaster-infected machine stops its TFTP daemon after a transmission or after 20 seconds of TFTP inactivity. In case of

success, it sends a command to start `msblast.exe` on the already open TCP connection to port 4444 of the victim. Now, the victim is running Blaster and starts to infect other machines.

2.4 Identification of Blaster Infections at Flow-level

Infection Stages. We define five different stages A, B, C, D and E that classify to which extent a Blaster infection attempt on a victim host was successful.

- A. The victim host does not exist or does not respond to a connection attempt on 135/TCP.
- B. The victim host responds but port 135/TCP is closed.
- C. The victim host receives the exploit code but either the exploit code for the wrong operating system was transmitted or the RPC DCOM security patch was already applied.
- D. The victim host is vulnerable and the correct exploit code is successfully transmitted to port 135/TCP and the TFTP commands are sent to the remote shell on 4444/TCP but the TFTP server does not respond.
- E. The infection is completely successful.

Table 1. Flows required for infection stages A - E. ‘A→V’: Flow from attacker to victim, ‘A←V’: Flow from victim to attacker

Stage	135/TCP		4444/TCP		69/UDP	
	A→V	A←V	A→V	A←V	A←V	A→V
A	■	-	-	-	-	-
B	■	■	-	-	-	-
C	■	■	■	■	-	-
D	■	■	■	■	■	-
E	■	■	■	■	■	■

Filtering for Blaster Flows. The infection attempt stages defined in 2.4 can be distinguished by filtering our flow-level backbone traffic for the sequential occurrence of specific flows between any two active hosts that contain certain protocols and ports and that have a size and a number of packets in restricted ranges. We derived this information for each of the five infection stages from the Blaster analysis given in 2.3, from packet-level measurements in our Blaster testbed, and from tracking flow-level traffic of a host that was infected by Blaster during the actual outbreak and that was highly active on August 12th, 2003.

Obviously, the number of infection attempts per time unit is highest in stage A and lower in stages B to E as the filter criteria get more and more restrictive. This filtering for infection stages shows a reduction in the number of infection attempts of several orders of magnitude as can be seen in the Blaster plots in Figures 2 to 6.

Challenges of Malicious Flow Extraction. We faced the following challenges when defining our malicious flow extraction filters:

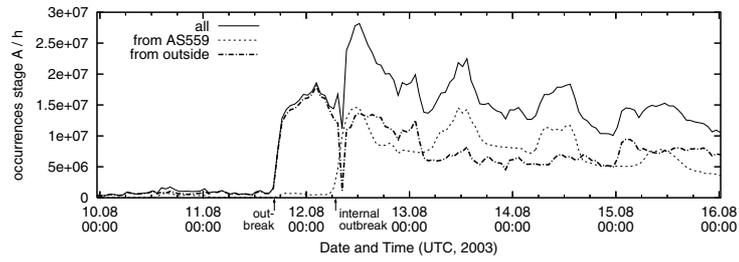


Fig. 2. Number of 'stage A' Blaster infection attempts from Aug 10th to Aug 15th

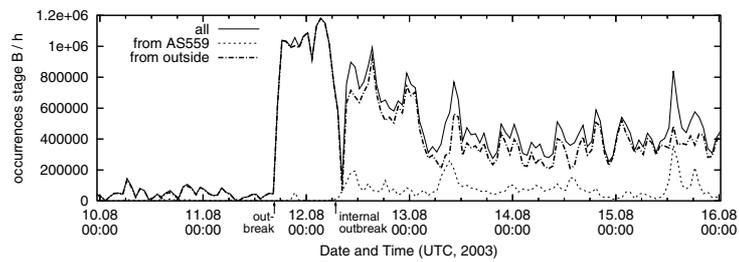


Fig. 3. Number of 'stage B' Blaster infection attempts from Aug 10th to Aug 15th

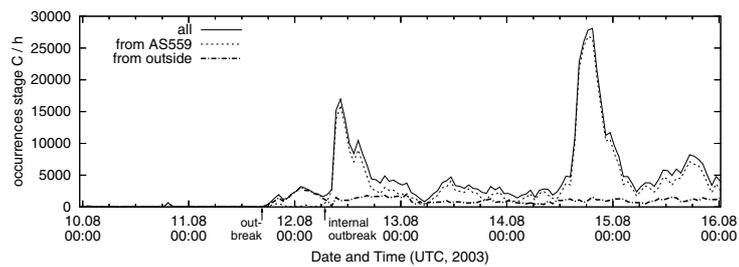


Fig. 4. Number of 'stage C' Blaster infection attempts from Aug 10th to Aug 15th

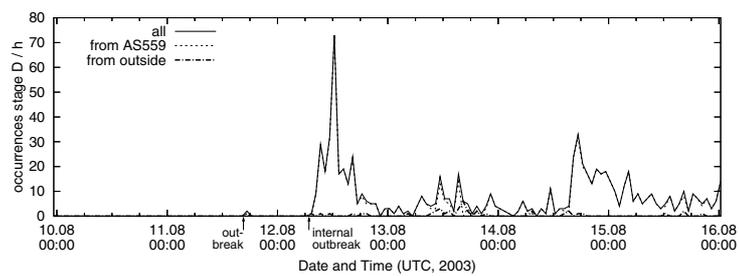


Fig. 5. Number of 'stage D' Blaster infection attempts from Aug 10th to Aug 15th

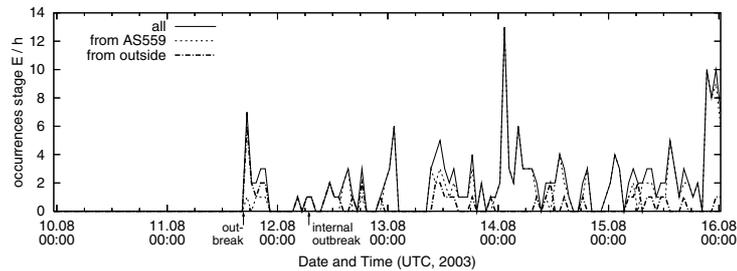


Fig. 6. Number of ‘stage E’ Blaster infection attempts from Aug 10th to Aug 15th

- Retransmissions by TCP due to packet loss (mostly at the receiver) and too short timeouts caused additional packets in the flows; we observed that the initial TCP SYN packets were most likely to be retransmitted (possibly due to an overload of the receiver).
- Different sizes of TCP SYN packets (40 and 48 bytes) due to the presence or absence of the TCP option field of 8 bytes that indicates the maximum segment size accepted.
- Indistinguishability of a TCP SYN packet of a malicious flow and of regular use in case of an unsuccessful connection attempt (e.g. on 135/TCP).
- Inactivity timeouts (30 s) of the NetFlow aggregation engine that cause a split of one flow into multiple flows for slowly responding hosts (e.g. shells on 4444/TCP) requires “glueing” of such flows.
- Preventing to count hosts, which had similar traffic like Blaster but out of order or with non-Blaster payload. Therefore, we also applied a heuristic timing condition, which required that the start time of each flow belonging to an infection attempt must lie within 10 seconds of a first 135/TCP flow seen. In our LAN testbed all such flow start times were below 4 seconds. We chose the larger threshold of 10 seconds due to expected higher delays in a WAN and due to possibly slower processing powers of involved hosts.
- Trivial FTP (69/UDP) used by Blaster is a well-known service. Therefore, we had to further limit our filters to only consider host pairs that had flows on port 135/TCP and 4444/TCP previous to a TFTP transfer attempt.

2.5 Blaster Outbreak Traffic Analysis

Our traffic analyses focus on a time interval starting shortly before the Blaster outbreak on 10th of August 2003 and ending on 16th of August 2003. In the following plots, we have split the total traffic by origin (inside or outside of AS559). With “inside” we mean all IP addresses of hosts belonging to SWITCH (AS559) and its customers.

For each 5 minute interval, all pairs of hosts that had flows matching the criteria of the infection attempt stages A to E defined in 2.4 are identified and accounted to the five stages. Table 1 lists for each infection attempt stage the required flows (marked by symbol ■) and their directions. Congestion in storage tables of the router’s NetFlow engine can lead to a loss of flows. Therefore, we alleviated the requirements such that

only at least one matching flow for each port/protocol type (135/TCP, 4444/TCP, and 69/UDP) needed to be present if the stage required that type at all. However, the effect of this alleviation in the filtering conditions was only minimal.

Infection Attempts. Figures 2 to 6 show the number of infection attempts for each of the five stages A to E defined in Section 2.4. Monday, August 11th, 2003 at around 16:35 UTC can be regarded as the *outbreak* of Blaster. We can see in Figure 2 that at the outbreak the number of unsuccessful connection attempts to port 135/TCP (stage A) drastically increases from around 0.7 mill. to 1.5 mill. and in the next two hours to 13 mill. flows per hour. In the three hours after the outbreak, the number of stage B infection attempts (victim responding but port 135/TCP is closed) grows from about 50'000 to 1 mill. connection attempts per hour. The number of stage C (Figure 4) occurrences jumps from 0 to around 650, while stage D (Figure 5) occurrences show only a single host in total during the first three hours of Blaster. The very first successful infection from the outside to a SWITCH-internal host happened at 17:42 UTC. Quite late, at 18:20 UTC, the first external host is successfully infected from a host in the SWITCH network. In the hour from 17:20 to 18:20, a total of seven infections can be seen in Figure 6. More than a full hour passed after the significant increase of port 135/TCP activity and before the first successful infection in the AS559 network happened.

Before August 12th, the vast majority of Blaster traffic originated from outside the SWITCH network. This changed around 6:50 UTC and can be considered as the *internal outbreak*. Before that, only few hosts within AS559 had been infected. The reason for the delay of the internal outbreak is that the external outbreak happened not during Swiss work time and most internal Windows hosts were switched off during the night. In the time 23:55 on Aug 12th to 2:01 UTC on Aug 13th, only a single internal host was successful and infected 11 outside hosts that were all in the same /16 network range.

In the plots for stages A and B, we can observe a *drop in the number of connections* from external hosts from 08:30 to 09:10 on August 12th. This was caused by an inbound port 135/TCP filter installed at a border router of AS559. We can observe another but smaller drop of infection attempts coming from external hosts, decreasing since 2:40 on August 13th with its lowest point around 5:00. This is most probably also an effect of manual port filtering.

The first peak of stage C is between 9:20 and 10:20 on August 12th, with around 15'000 infection attempts. Our analysis showed that around 70% of the stage C infection attempts in that interval came from one single /16 network. The vast majority of the victims of these infection attempts were in the next higher /16 net lying outside of AS559. These connections were probably generated by Blaster scanning the local subnet, but the scanned addresses were constantly increased by one and suddenly pointed out of the local subnet and therefore the infection attempts were routed over the SWITCH border gateways. At the same time interval the infected hosts of that subnet generated only 29 stage D and not a single stage E infection attempt. The reason for this lack of successful infections may be that in the destination subnet the hosts were already patched.

Many similar IP-neighbourhood attacks happen during the second significant increase of stage C occurrences starting around 15:20 on August 14th. The majority of attacks originate in one single /16 network and most destinations are in the next

higher /16 network lying outside of AS559. In that network, most hosts were apparently also already patched as almost no successful infections were observed. We can deduce, that choosing as backscatter or honeypot network one with IP addresses adjacent to small internal subnetworks can help reducing the time for the detection of worms that scan IP addresses linearly increasing.

The reason why these scans show up as peaks in the plot is probably that most of the hosts were infected in a small time range internally and therefore started their scanning around the same time. Consequently, they also reach the next network at the same time and when they have passed the address space of that subnet, they came probably to a network less populated or with some filtering, which caused a drop of stage C infection attempts. Their scanning then appears as stage A or stage B.

The plot of stage C shows a small peak of 631 infection attempts on August 10th in the hour of 19:20 - 20:20 UTC before the massive outbreak of Blaster. A single host becoming active around 19:40 is responsible for 80% (522) of these attempts. At that time, the exploit code used by Blaster was already published. From that specific IP address we observed a scanning of port 135/TCP and for the addresses that the scanning was successful the exploit code was sent. It is possible that this was some testing in the development phase of Blaster, but more likely someone just tried out the exploit code for fun or for some abuse.

Successful Infections. The stage E plot of successful Blaster infections shows a peak at the right end with 35 infections within 3 hours, from 21:20 to 0:20 on August 15th. 29 of these infections originate from one host and have their victims in the same /17 network range. This host obviously scanned by chance a network with many vulnerable hosts. A surprise in Figure 6 is, that despite the high number of Blaster infected hosts worldwide, we can only observe very few successful infections going over SWITCH's border routers. Over the analysed time period, from the outbreak, on August 11th, to August 16th, 0:20, we observed only 215 successful infections in total. 76% of the observed infections originate from within AS599 and 24% are from external hosts. 73 different hosts have successfully infected others. The reason for this low number is that the vast majority of successful infections happened within the local networks and did not cross the backbone border routers. The ten most successful hosts have performed 138 (64%) of all observed infections. The hosts in the top ten list scanned networks with high numbers of vulnerable computers. The 47 infections of the "winner" all go to addresses in a range of 13 adjacent /16 networks. The fact that 11 out of the top 21 infecting hosts were found to be in the same /16 network is an evidence that this network suffers from *slow patching procedures*.

2.6 Worm Code of Multi-stage Worms: Low Frequency vs. High Threat Potential

From our backbone measurements we conclude that for multi-stage worms, which use several different steps before actual worm code is transmitted, the number of observable hosts that successfully infect others is extremely low (4 hosts during the initial outbreak per hour in Fig. 6). This is in heavy contrast to the high number of hosts scanning for vulnerable hosts in Fig. 2.

The design of Blaster relies on three different connections, one of which was on a port rarely used (4444/TCP) and the other involved a modestly popular service (TFTP). As such connections are filtered in many networks, this is a source of errors.

As a consequence, actual worm code (but not exploit code) transmissions are quite rare in the backbone. This has consequences for e.g. sampling for malicious code in a backbone, as sampled packet sniffing will almost never capture real worm code. Automatic detection of worm code and blocking infecting source hosts for multi-stage worms becomes almost infeasible. Missing even a single successfully infecting host will destroy the effectiveness of backbone border router worm filtering efforts. Even a very low frequency of malicious worm code occurrence in the backbone has apparently still a high threat potential.

2.7 Coarse Grained Analysis

Due to the huge number of possible combinations of protocols, ports, sizes and number of packets involved in a new worm outbreak, it would be very resource consuming to constantly watch host activity for new worms on such a fine grained level as we used it in our Blaster infection attempt analyses. Therefore, we also present the Blaster outbreak on a more coarse grained level disregarding size and numbers of packets per flow constraints in the remainder of this section.

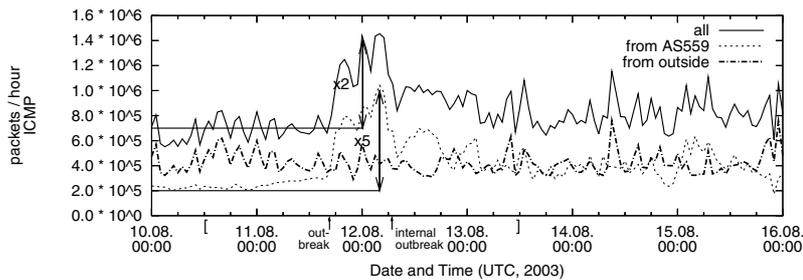


Fig. 7. Blaster worm: ICMP packets per hour

ICMP Measurements. The graph in Fig. 7 shows the total number of ICMP packets per hour. The ICMP traffic sent from hosts within AS559 and from outside of AS559 are shown separately. We noticed approximately a fivefold increase in the rate of ICMP packets per hour sent from AS559 and a twofold increase for the ICMP packet rate per hour sent in total during peak time compared to the base level before the outbreak. This large backscatter traffic of ICMP messages can be explained by many error notifications caused by unsuccessful connection attempts to blocked ports and non-existent hosts.

Activity on Port 135/TCP. The graphs in Fig. 8 and 9 show the number of unique IPv4 source addresses from which connections to port 135/TCP were initiated. Source hosts are separated into 1) all hosts, 2) hosts within AS559 and 3) others. The plots use aggregation over one hour respectively 5 minutes observation intervals to build the set of active hosts. The brackets [and] in the hour plots indicate the smaller time window of the 5 min. plots.

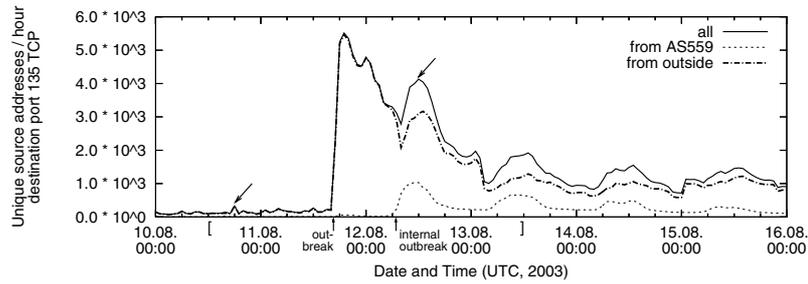


Fig. 8. Blaster worm: Unique source addresses per hour

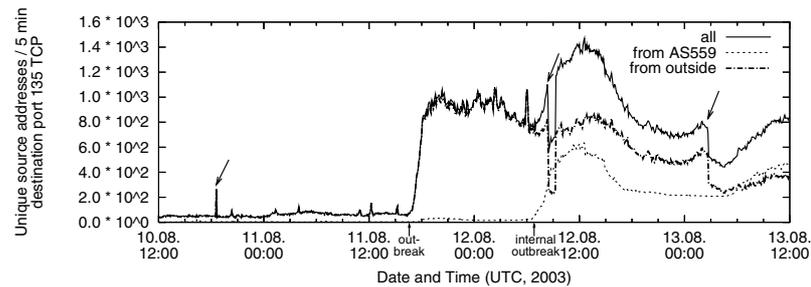


Fig. 9. Blaster worm: Unique source addresses per 5 minutes

We observed around 140 hosts/hour connecting to port 135/TCP in the hours before the outbreak. There is an interesting small peak of 327 hosts/hour on Sunday, August 10th, 2003, 18:00-19:00 UTC indicated with an arrow. Figure 9 shows that this peak stems from a single five minute interval starting at 18:35. In this interval, 263 hosts connect to port 135/TCP. We assume that the peak was either a preliminary version of Blaster that was used to test-run a generation limited infection or that it was a scan to identify an initial host population to be infected or someone just playing with the RPC DCOM exploit code. There might have been more such tests, but they are less visible. From the stage C analysis in Section 2.5, we remember the increased infection attempt activity also involving 4444/TCP connections around 19:40 - 20:10 UTC.

The primary Blaster outbreak, which is indicated by a small vertical arrow on the time axis in all Blaster plots, starts on Monday, August 11th, 2003, around 16:35 UTC with 64 hosts from outside AS559 connecting per 5 minutes (Fig. 9), but increases to 96 hosts at 16:55 and then sharply to 832 hosts active per 5 min at 18:15. A rather chaotic phase without major increase follows. The hour plot shows a peak of about 5'500 hosts scanning on 135/TCP on 11th during the hour 19:00-20:00. The number of active source hosts increases again when hosts within AS559 begin to contribute in the interval August 12th, 2003, 6:00-7:00 (Fig. 8), reaching 1030 active internal hosts per hour in the interval 11:00-12:00. Figure 9 shows that around 6:50 (8:50 CEST) many hosts within AS559 became infected by Blaster. This can be explained by the start of the working day in Switzerland. We assume that most of the vulnerable hosts in AS559 were not running during the night.

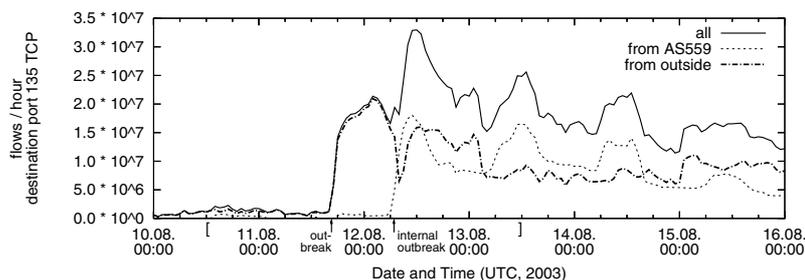


Fig. 10. Blaster worm: Flows to 135/TCP per hour

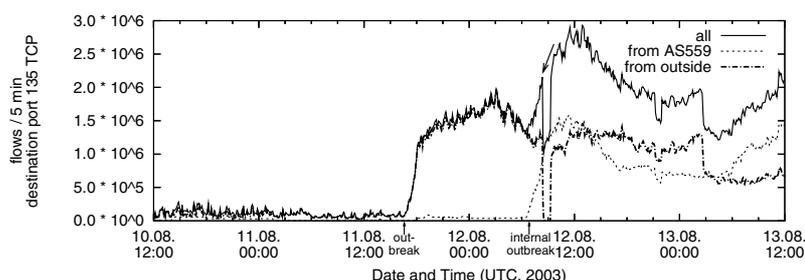


Fig. 11. Blaster worm: Flows to 135/TCP per 5 minutes

Another remarkable event is the sudden drop of outside connection attempts on August 12th, 2003, 8:30-9:10. This drop is due to temporary blocking of port 135/TCP on some of the AS559 border routers for incoming connections by SWITCH. This ingress filter proves mostly ineffective as a countermeasure to stop the fast increase in the number of new internal host infections. However, if a complementary egress filter were installed at the same time as the ingress filter was activated, this would have prevented up to a thousand AS559 internal hosts per hour from trying to infect AS559 external hosts. A similar filtering effect can be seen around 2:40 on the 13th. This port filter is also only partially effective.

Activity on Port 4444/TCP. As explained in Section 2.3, a successful transmission of the exploit code to 135/TCP makes Blaster connect to 4444/TCP, where it tries to initiate a remote download. Figure 12 shows the number of flows per hour to destination port 4444/TCP. Several significant short peaks of scan traffic to this port from hosts outside of AS559 can be seen. An analysis with 5 minute buckets revealed that these traffic peaks were constrained to 15-20 minutes of high activity and that the number of unique source IP addresses connecting to port 4444/TCP did not show significant changes during these flow peaks. We conclude that the first flow peak might result from a pre-test of a propagation-limited Blaster-like worm, and the other peaks might result from a few network operators scanning the network for open 4444/TCP ports.

Activity on Port 69/UDP. A Blaster victim that has received the correct RPC exploit code and the TFTP download command initiates a connection to the Blaster-infected

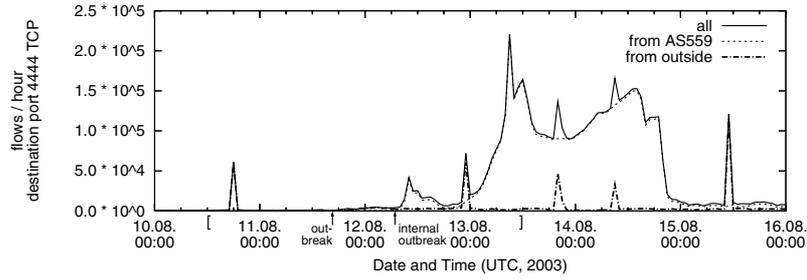


Fig. 12. Blaster worm: Flows to 4444/TCP per hour

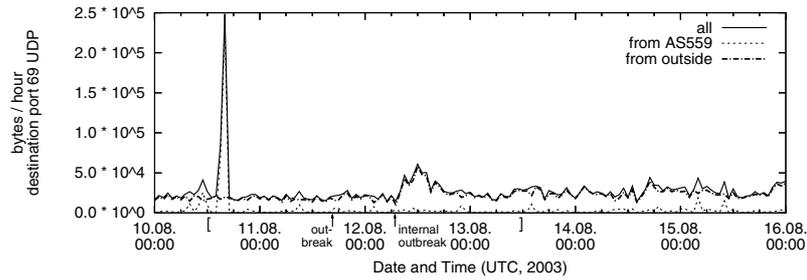


Fig. 13. Blaster worm: Bytes to 69/UDP per hour

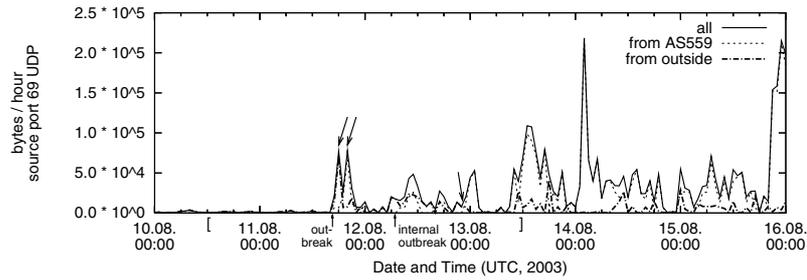


Fig. 14. Blaster worm: Bytes from 69/UDP per hour

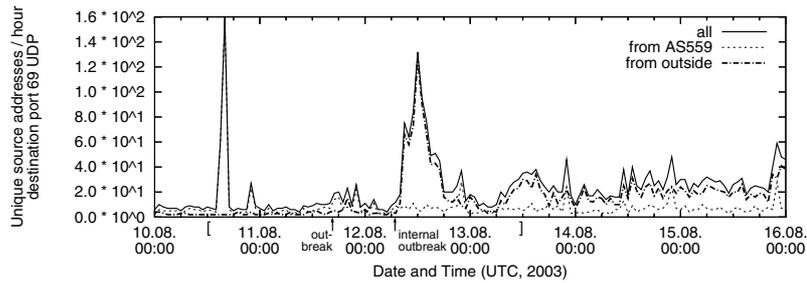


Fig. 15. Blaster worm: Unique source addresses (dest. 69/UDP) per hour

host on port 69/UDP and tries to download the worm code with the trivial file transfer protocol (TFTP). Hence, we expect to see many connections with little payload to this port containing mainly the TFTP commands to fetch the worm code. If this is successful, we should see larger amounts of data being sent from source port 69/UDP of the Blaster-infected host back to the victim.

The plots of bytes per hour to destination port 69/UDP in Figure 13 shows a base level of about $15 \cdot 10^3$ to $20 \cdot 10^3$ bytes per hour. There is a huge peak of $2.5 \cdot 10^5$ bytes in the hour from 16:00-17:00 on August 10th. For 92% of this peak traffic, hosts from AS559 are responsible. The plot for the traffic originating from port 69/UDP in Figure 14 reveals that these connections were apparently unsuccessful as almost no data was downloaded. It also shows (indicated by the first arrow from left) that between 18:00-19:00 on 11th worm code was almost exclusively downloaded from hosts outside AS559. With a two hour delay (indicated by the second arrow from left), worm code is almost exclusively uploaded. However, these peaks of roughly 70'000 bytes each only account for about 10 worm code copies of 6'592 bytes transmitted during each peak. The third arrow from left in the plot of Figure 14 indicates, that after 12th 23:00 the vast majority of total bytes transmitted from source port 69/UDP was sent from infected hosts within AS559 to outside hosts.

The analysis of the activity of unique source addresses sending traffic to destination port 69/UDP as shown in Figure 15 reveals a peak of about 160 unique IP addresses that were involved in the probable pre-test phase of the worm. About 250 flows with an average size of 1.4 kB go to port 69/UDP from AS559. Figure 13 shows the increased bytes per hour activity. The small number of 1.5 flows per involved host on average indicates that this was not UDP scan traffic to port 69/UDP as one might have expected but rather small file transfers.

3 E-Mail Worm Sobig.F

3.1 Outbreak of Sobig.F

On August 19th, 2003, the W32/Sobig.F [2] 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 MTA for sending e-mail, 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 the 10th of September 2003. Date and time are taken from a small set of hardcoded global time servers (NTP). The e-mails sent use an arbitrary sender and recipient address taken from local files of the infected host. This type of social engineering is obviously intended to fool users to open attachments seemingly from people they know.

The graph in Fig. 17 shows the total number of bytes per hour transmitted as e-mail (SMTP) traffic over the SWITCH border routers. A daily rhythm can clearly be seen. The five working-days have rather heavy traffic with a maximum around 5 Gigabytes per hour, whereas on Saturdays and Sundays the traffic is considerably less. The lunch break can be identified easily during weekdays.

On Tuesday, August 19, 2003 there is a huge increase in bytes transmitted over SMTP that rises up to around 21.7 Gigabytes/hour at 12:00-13:00 UTC, which is four to five times more than ordinary. This can be regarded as the outbreak of the Sobig.F worm. The plot clearly shows that the vast majority of the border e-mail traffic during the massive outbreak is originating from within AS559.

The graph in Fig. 18 shows the number of flows per hour split by origin of the e-mail sender. Interestingly, late on Monday, 18th of August 2003 there is a short peak of flows coming from outside AS559. An analysis showed that the number of unique hosts did not rise significantly during this peak. Therefore we assume this to be scanning traffic for SMTP hosts originating from a few hosts only. During the actual outbreak, the number of unique hosts sending e-mail from AS559 shows significant peaks.

3.2 Identification of Sobig.F E-Mails

In our NetFlow flow-level data, normally one flow corresponds to one e-mail delivered by SMTP. We used the size of Sobig.F infected e-mails to filter out Sobig.F e-mails from the total SMTP traffic observed.

The Testbed. In order to observe Sobig.F traffic at packet-level we used a testbed with an attacking host (Sobig.F on Windows XP) and a server (see Fig. 16). On the server (Linux Fedora Core1) we installed the services NTP, MTA and DNS. Sobig.F uses a hardcoded list of NTP servers to check that the current date is earlier than September 10th, 2003 before activation. We chose 129.132.2.21 for our server from this list. The DNS service (bind) was configured to resolve all name queries (for A and MX DNS records) from the attacker to the server IP address (129.132.2.21) so that the e-mails from Sobig were all sent to the MTA (sendmail) running on our server. The packet capturing was done on the server machine.

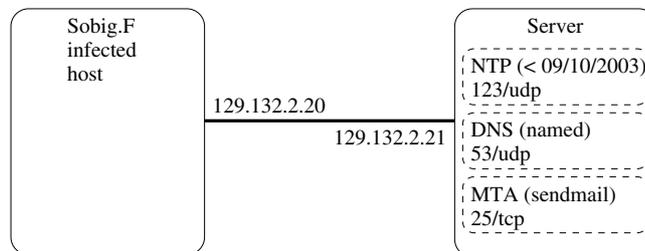


Fig. 16. The testbed for Sobig.F

Observed Worm Transmissions. In the testbed we captured the packets of several successful Sobig.F transmissions and observed an average of about 100 port 25/TCP packets sent from attacker to MTA with a total size (including IP and TCP headers) of about 104'000 bytes. The flows in the other direction consisted of about 40 packets with a total size of about 2'000 bytes.

For e-mails rejected by black- or whitelist spam filters, the flow from attacker to MTA consists of 8 packets with a total size of about 400 bytes, while the flow in the opposite direction shows 11 packets with a total size of about 850 bytes.

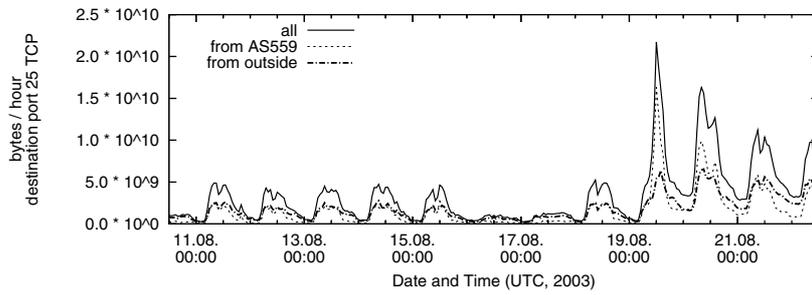


Fig. 17. Sobig.F worm: SMTP traffic volume per hour

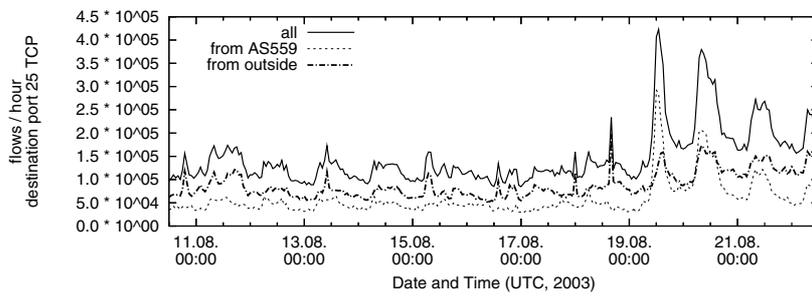


Fig. 18. Sobig.F worm: SMTP flows per hour

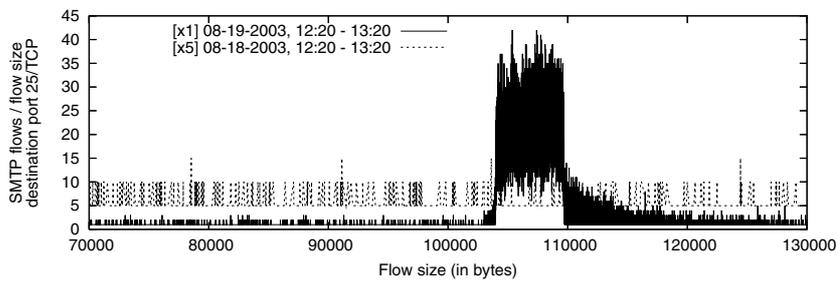


Fig. 19. SMTP flow size distribution before and during Sobig.F

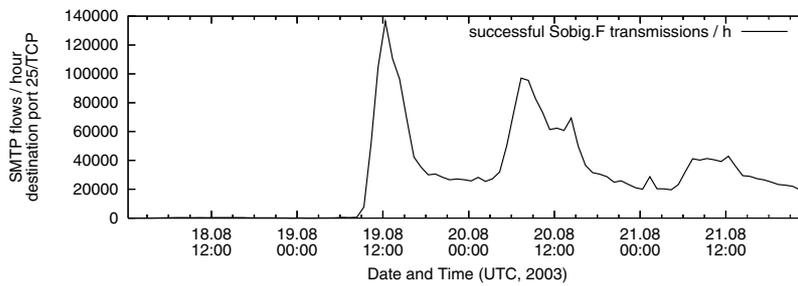


Fig. 20. Number of Sobig.F transmissions over time

Flow Size Distribution. The large worm size of about 100 Kbytes and the aggressive spreading algorithm caused many retransmissions by TCP during Sobig.F propagation as can be clearly seen in Figure 19 that shows the two histograms of e-mail sizes for a one hour interval during Sobig.F (on August 19th, from 12:20 to 13:20) and for the same hour the day before. The wide peak of successfully transmitted Sobig.F e-mails, starting at about 103'000 bytes then decreasing at about 109'000 bytes but still being significant up to about 125'000 bytes can easily be seen.

Further analyses showed that there are about twice as many flows of size 0 - 1'000 bytes (probably rejected e-mails) during the initial outbreak hour as compared to the day before. There were also some noticeable sharp peaks between 4'800 bytes and 5'100 bytes. Further analyses showed that all these peaks originate from flows with only two source addresses in the same subnet. As Sobig.F infected hosts could be used as open mail relay, these servers might have been abused for sending spam.

Number of Worm Transmissions. Figure 20 shows the plot of the number of flows with a size between 103'000 and 125'000 bytes, from which we can assume that they originate from successfully transmitted Sobig.F e-mails. As the number of e-mails in that size range was 350 on August 18th, 12:20 to 13:20 and starts to rapidly increase on August 19 at about 09:00, this can be regarded as the Sobig.F outbreak. The number of successful transmissions raises drastically until about 12:00 and then starts to decrease until the end of the working day at about 18:00. The peak of 137'000 transmissions on August 19 is by far the highest, on August 20 the peak reaches about 100'000 and on August 21 50'000 transmissions were counted. The decreasing heights of the peaks can be explained by people updating their anti-virus software and cleaning up their machines.

4 Conclusions

Our observations have shown that spreading events of massive worms can clearly be seen in overall traffic statistics of Internet backbones. Worms are a major threat to the reliability of Internet services, especially as those worms seen so far did not aim at disrupting Internet services by launching attack code but merely focused on fast and worldwide propagation.

We have seen some indication for test runs or preliminary scanning several hours before the actual Blaster outbreak. One consistent effect in all our observations is the time-skew between incoming infection traffic and infection traffic originating from AS559. This is due to the fact that most vulnerable computers were switched off during the night and that e-mail worms like Sobig.F require the attention by a user (e.g. executing an attachment) for an infection. This time window could be used for taking preventative countermeasures if early detection of new worms were available in backbone networks.

Blaster is a multi-stage worm, which uses several protocols and connections for data exchanges before actual worm code is transmitted. Our analyses have shown that this multi-stage nature together with Blaster's preference for local scanning over global scanning for vulnerable hosts has surprising consequences: only very few successfully infecting hosts and consequently almost no worm code can be detected (and possibly

filtered) in the backbone traffic. Nevertheless, these few successful infections over the international backbone had devastating consequences for the local networks. Consequently, automated effective blocking of actual worm code on backbone border routers is almost infeasible as only a few missed worm code transmissions will completely destroy the success of such security efforts. Furthermore, automated efficient capturing of new worm code in the backbone becomes a challenge due to the scarcity of such transmissions (this holds at least in the case of Blaster). The few worm code instances observed are not to be confused with the heavy scanning traffic and RPC exploit code that is sent in the first steps of a Blaster infection and which were transmitted quite frequently.

In addition, the ineffectiveness of simple ingress port blocking filters on routers in the hope to stop a further increase of internal infections was illustrated for Blaster. It was also shown, that AS559-external networks with IP addresses adjacent to AS559-internal networks were more heavily attacked than others due to Blaster's incremental scanning algorithm. Choosing as backscatter or honeypot network one with IP addresses adjacent to small internal subnetworks can help reducing the time for detection of worms that scan IP addresses linearly increasing. Several challenges to extracting actual malicious traffic at flow-level were stated such as the sporadic use of a TCP option field for the maximum segment size in SYN packets that enlarges the packet header and frequent packet retransmissions by TCP that both let the measured flow vary in size and consequently lower the accuracy of simple flow size filters. Finally, we discovered that 11 hosts of the top 21 successfully infecting hosts were in the same /16 network, which is an evidence that this specific network suffers from slow patching procedures.

As a consequence of the Blaster and Sobig.F analyses, the authors developed algorithms for early detection of worm outbreaks in the backbone that were successfully validated on our archived DDoSVax NetFlow data of past worms. One is based on a classification of the hosts' traffic behaviour [23], another one tracks the entropy of the IP addresses and the TCP and UDP ports [24] seen.

Further research on and measurement analyses of worms and countermeasures are vital for a better understanding of worm propagation and the development of effective countermeasures, especially as worm authors get more and more sophisticated.

References

1. CERT: Security Advisory: MS.Blaster (CA-2003-20). <http://www.cert.org/advisories/CA-2003-20.html> (2003)
2. CERT: Incident Report: Sobig.F (IN-2003-03). http://www.cert.org/incident_notes/IN-2003-03.html (2003)
3. Wagner, A., Dübendorfer, T., Plattner, B.: The DDoSVax project at ETH Zürich. <http://www.tik.ee.ethz.ch/~ddosvax/> (2004)
4. ETH: Swiss Federal Institute of Technology. <http://www.ethz.ch/> (2004)
5. Müller, O., Graf, D., Oppermann, A., Weibel, H.: Swiss Internet Analysis. <http://www.swiss-internet-analysis.org/> (2003)
6. Cisco: White Paper: NetFlow Services and Applications. http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm (2002)
7. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* **13** (1970) 422–426

8. Symantec Corporation: Symantec Security Response - W32.Blaster.Worm. <http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html> (2003)
9. TREND micro: Technical details of WORM MSBLAST.A . http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_MSBLAST.A&Vsect=T (2003)
10. J. Nazario: The Blaster Worm: The View From 10'000 Feet. <http://www.nanog.org/mtg-0310/pdf/nazario.pdf> (2003)
11. SWITCH: FloMA: Pointers and Software (Netflow tools). <http://www.switch.ch/tf-tant/floma/software.html> (2004)
12. CERT/CC at SEI/CMU: SiLK GPL Netflow tools. <http://silktools.sourceforge.net/> (2002)
13. CAIDA: Cooperative Association for Internet Data Analysis. <http://www.caida.org/> (2004)
14. Shannon, C., Moore, D.: The Spread of the Witty Worm. <http://www.caida.org/analysis/security/witty/> (2004)
15. Kim, J., Radhakrishnan, S., Dhall, S.K.: Measurement and Analysis of Worm Propagation on Internet Network Topology. In: Proceedings of ICCN. (2004)
16. Staniford, S., Paxson, V., Weaver, N.: How to Own the Internet in Your Spare Time. In: Proc. USENIX Security Symposium. (2002)
17. Wagner, A., Dübendorfer, T., Plattner, B.: Experiences with worm propagation simulations. In: ACM Workshop on Rapid Malcode (WORM). (2003)
18. Zou, C.C., Gong, W., Towsley, D.: Code Red Worm Propagation Modeling and Analysis. In: Proceedings of the 9th ACM conference on Computer and communications security, Washington, DC, USA. (2002)
19. Lemos, R.: MSBlast epidemic far larger than believed. http://news.com.com/MSBlast+epidemic+far+larger+than+believed/2100-7349_3-5184439.html (2004)
20. The Last Stage Of Delirium: Buffer Overrun in Windows RPC Interface. <http://lsd-pl.net/special.html> (2004)
21. Microsoft Corporation: Microsoft Security Bulletin MS03-026. <http://www.microsoft.com/technet/security/bulletin/MS03-026.msp> (2003)
22. eEye Digital Security: Blaster Worm Analysis. <http://www.eeye.com/html/Research/Advisories/AL20030811.html> (2003)
23. Dübendorfer, T., Plattner, B.: Host Behaviour Based Early Detection of Worm Outbreaks in Internet Backbones. In: Proceedings of 14th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WET ICE); STCA security workshop, IEEE (2005)
24. Wagner, A., Plattner, B.: Entropy Based Worm and Anomaly Detection in Fast IP Networks. In: Proceedings of 14th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WET ICE); STCA security workshop, IEEE (2005)