

MS Blaster

There was a significant amount of Denial of Service activity in the period, widely covered by the media. The first of the set were two worms, dubbed “MS Blaster-A” and “MS Blaster-B”. Both worms take advantage of a buffer overrun exploit present in Microsoft’s RPC services. Those services ship with Windows XP, Windows 2000, and Windows NT.

Evidence of MS Blaster infection attempt activity was first noticed on Abilene’s backbone at approximately 16:00 (UTC, 11AM EDT) on August 11th (see Appendix B) and ramped up very rapidly thereafter, peaking roughly sixteen hours later. Actual system infections appear to have peaked roughly two hours after infection attempts were initiated. In other words, the height of successful infections appears to have occurred roughly two hours after the worm gained its greatest strength. Although infections show a strong peak, it should be noted that they continued at high levels for several days.

Attack and Infection sequence

During the initial phase of the attack, several packets are directed from the attacking system to an Microsoft RPC TCP port (typically 135 although others were used as well) on the system under attack. These packets cause a buffer overflow (on unpatched systems) and subsequent execution of Trojan code. This Trojan bootstrap then opens a TCP socket (4444) on the infected system and connects the Microsoft command-line shell (CMD.EXE) that the socket. At this point, the initial bootstrap Trojan exits on the infected system.

After a short pause, presumably while it waits for the Trojan to start the CMD.EXE executable, the attacking system attempts to reconnect to the system it just attacked. At this point, however, it attempts a connection to port 4444 hoping (if computer programs can hope) to connect to the CMD.EXE executable. Patched systems and systems not vulnerable to the worm (such as UNIX, Linux, Macintosh, etc.) at this point respond either with a port unreachable message or with an otherwise inappropriate response. In those cases, the attacking system ceases trying to further infect the host in question and moves onto another target.

If the initial attack bootstrap was successful, the attacked system will respond on port 4444 with a standard “DOS” command prompt. At this point the attacking system issues commands over port 4444. These commands, which start the Trivial File Transfer Protocol (TFTP) client, are identical in form to what a human might type were they sitting at the keyboard of the infected machine. Using the TFTP client, the infected machine connects back to the attacker (using TCP port 69) and initiates a download of the main worm code (called MSBLAST.EXE in the Blaster-A variant).

At this point the attacked machine is infected and will begin attempting to infect other machines. Furthermore Blaster, unlike worms such as SQL Slammer, not only installs a copy of its code on the infected system’s hard drives but it also alters system information

on the infected computer. In particular, it adds an entry to the system registry which automatically re-starts the MSBLAST.EXE executable on either a reboot or a power cycle. This means, again unlike SQL Slammer, that a restart of the computer system is not sufficient to decontaminate the system. At a minimum both the registry entry as well as the MSBLAST.EXE code must be removed. Of course, the system must also be patched with the relevant patches to prevent re-infection.

Pathology

Blaster's effects were wide-ranging. Symantec estimated that approximately half a million computers were infected. This number matches within an order of magnitude the estimates that were done at the Advanced Network Management Lab using the Abilene network. We counted approximately 30,000 successful infections across Abilene (from 10 billion infection *attempts*, a success rate of just 0.0003%). Abilene is estimated to carry roughly 15% of the globally routable IPv4 addresses thus extrapolation from our viewpoint yields about 200,000 systems infected worldwide.

Blaster's most pernicious effect, like that of other worms (in particular the Morris worm) was the result of a "bug" in its code. Although all suspect platforms (Windows XP, 2000, and NT) contained the same buffer overrun vulnerability, the exploit of that vulnerability required slightly different code on each. Blaster used a simple algorithm to try and determine the type of host, and the operating system in use, that it was attacking. In a significant number of incidents (between 10 and 20% approximately), Blaster made the wrong choice.

The consequence of making the wrong choice was that Blaster attempted to inject an improper version of the worm into the victim host. While this worm was incapable of infecting the host with the Blaster worm it nonetheless left the victim in an unstable state which usually required a system reboot within moments (see "unstable version" path in Appendix C). As attack attempts, especially within the first five days of onset (see Appendix B), were relentless it meant that a very large number of systems, while not infected, were rendered unstable.

This was further complicated by the fact that an unstable system would not remain operational long enough to download the necessary patch files from Microsoft (while, counter-intuitively, an infected system might). Note that ANML analyzed several systems that became unstable but that also harbored older Blaster infections. This leads us to the conclusion that occasionally Blaster made the right decision about host operating systems on systems on which it would subsequently make the wrong decision.

Mitigation

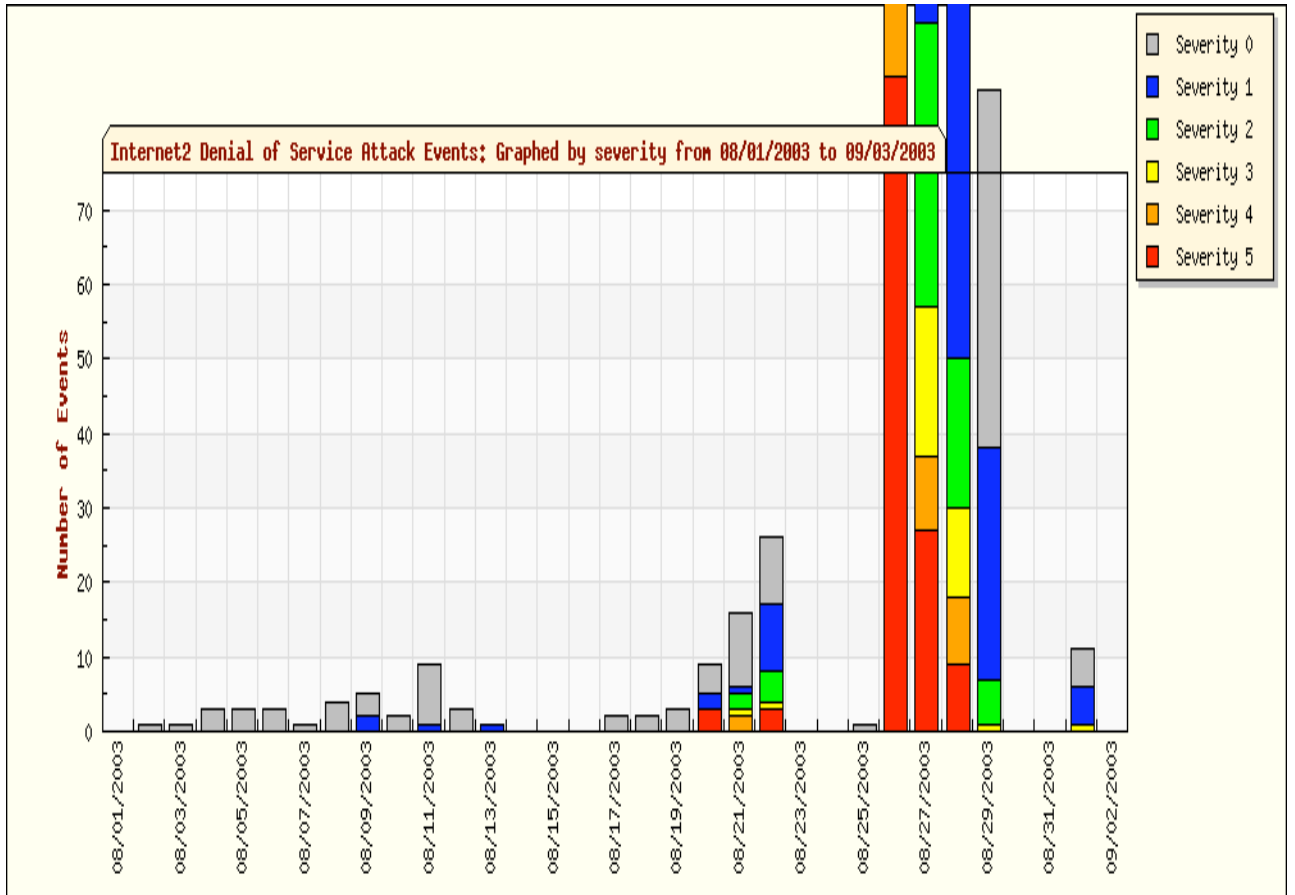
ANML worked with several stakeholders in mitigating the effect of the MS Blaster

worm. Our first action included the development of a single “patch” CD which could be conveyed to vulnerable systems—systems which could not be patched online due to the situation described below. This CD was made available by the morning of August 12th.

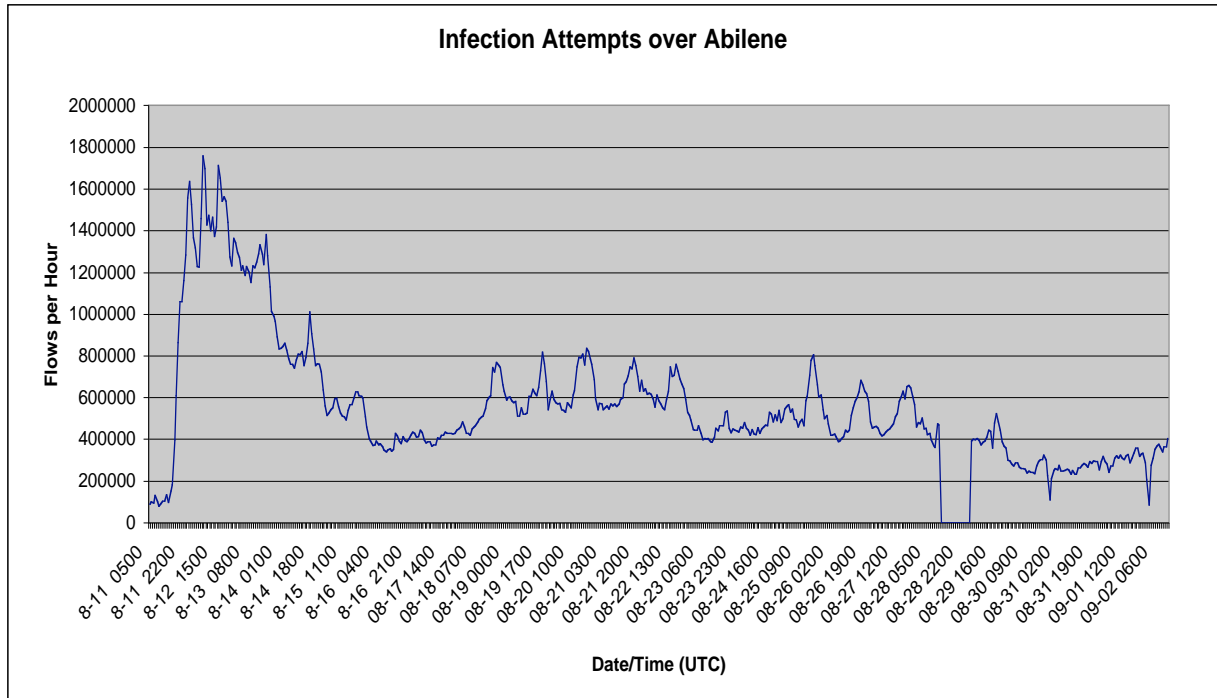
ANML also worked closely with the REN-ISAC in tracking infection attempts, actual infections, sources of infection, and infected systems. In doing so we developed an extension to our incident tracking database involving a spreadsheet frontend (an example of which was sent to AFRES on 8/13).

Finally, ANML, together with the REN-ISAC worked directly with Microsoft in developing scenarios to mitigate the effect of the worm’s ultimate attack, that scheduled for Saturday 8/16. At that time the worm was scheduled to morph into a more traditional distributed denial of service attack, aimed at the host called windowsupdate.com. Although it ultimately proved unnecessary, one potential solution discussed was that of using Abilene’s prodigious capacity to “draw fire” from the Microsoft hosts. Schematically, that would have been accomplished by injecting routes into the global Internet specifying an Abilene “black hole” as the destination windowsupdate.com.

Appendix A, Internet2 Monitoring summary by severity and type of attack



Appendix B, Internet2 MS RPC worm activity



Appendix C, MS BLASTER Worm Flowchart

