VIRUS ANALYSIS 3

BLAST OFF!

Peter Ferrie, Frédéric Perriot, Péter Ször
Symantec Security Response, USA

On 11 August 2003 – the same day it was completed – a 6176-byte-long UPX-compressed bug started to invade the world using a recent vulnerability described in *Microsoft*'s *MS03-26* security bulletin. Even *Windows Server 2003* was affected by this vulnerability. Patches were made available by *Microsoft*, but on this occasion there was only a short delay between the announcement of the vulnerability and the appearance of the worm that exploited it.

Users of *Windows XP* had a chance to get the patch applied automatically via Windows Automatic Updates. However, the same cannot be said for the *Windows 2000* platforms, where users would need to pay closer attention to the update procedures.

ALL SYSTEMS GO

The first thing Win32/Blaster does when it runs on a system is to create a value 'windows auto update' in the 'HKLM/…/Run' registry key, pointing to the bare file name 'msblast.exe' (for variant .A). This relies on the assumption that the executable has ended up in a directory that *Windows* searches by default, which is usually the case.

Then the worm attempts to create a mutex named 'BILLY', and aborts if the mutex exists already, in order to avoid multiple instances of the worm running at the same time.

Win32/Blaster then waits for an active network connection, and starts searching for machines to infect.

SP4, SP3, SP2, SP1, IGNITION!

The target selection in Blaster is somewhat different from that found in CodeRed and Slammer. Sixty per cent of the time, Blaster will go after entirely random IP addresses, and the other 40 per cent of the time it will attack machines on the same class-B-sized network as the host, hoping to take over pools of vulnerable systems on the local area network.

The scanning for targets is linear (the target address is increased monotonically until it reaches the end of the IP space) and, in the case of a local attack, starts at or slightly below the class-C of the host.

The worm targets machines running *Windows 2000* and *Windows XP*, and intentionally favours the exploitation of *Windows XP* machines (probably because the payload relies on the increased availability of raw sockets there – the requirement to be administrator was removed). Eighty per cent of the time, the exploit is tuned for *Windows XP* systems, the other 20 per cent for *Windows 2000* systems. This selection is made only once, whenever the worm initializes.

All unpatched Service Packs of both *Windows XP* and *Windows* 2000 systems are affected, but because of this random tuning, the worm will sometimes just cause a denial of service (DoS) on the attacked machines, crashing the RPC service.

SECOND STAGE – THE SHELL

The infection of a new machine is a three-phase process, involving quite a lot of network activity in comparison with the single-connection CodeRed and the lightweight Slammer.

First, the worm sends its attack buffer over port 135/tcp, which exploits the RPC DCOM vulnerability and causes the remote machine to bind a shell in the SYSTEM context ('cmd.exe') to port 4444/tcp.

Second, the worm sends a command to the newly created shell to request a download of the worm file from the attacking host to the victim. The transfer is done over port 69/udp using the tftp protocol (the worm implements its own crude tftp server which formats sent data according to RFC 1350, and uses the tftp client that is present by default on most *Windows* systems).

Finally, once msblast.exe has been downloaded successfully, or after 21 seconds, the worm requests the remote system to execute the downloaded file.

HOUSTON, WE HAVE A PROBLEM

Once the shell exits, the hijacked RPC service thread running the shell code calls ExitProcess(), causing the service to terminate. The termination of the RPC service, regardless of how it occurs, triggers a reboot in *Windows XP* systems after one minute. On *Windows 2000* systems, the termination will result in a variety of unusual side effects, among the most critical of which is the inability to use the Windows Update web service.

PAN GALACTIC GARGLE BLASTER

As is common for fast-spreading worms, Win32/Blaster reuses an exploit code that was posted previously to various security mailing lists. The exploit uses two so-called 'universal offsets' as return addresses in a classic stack buffer overflow, each of which is compatible with multiple service packs of one *Windows* version. The vulnerability is located in the code of the rpcss.dll file, in a function related

to the activation of DCOM objects. The buggy function extracts a NetBIOS server name from a UNC path specified by a DCOM client, and attempts to place it into a 32-byte buffer on the stack, without bounds checking.

Once the stack is smashed, the hijacked return address leads to a 'call ebx' instruction (in a 'well-known' constant data table) which then jumps back to a nop ramp in the shell code. This is possible because the ebx register is pointing to a local variable in an earlier stack frame (i.e. at a higher memory address) created by the fourth-level (!) caller of the buggy function (see Figure 1).

The shell code retrieves some useful API addresses, binds to port 4444/tcp, accepts one incoming connection, spawns the shell and ties its input to the port 4444 socket, waits for the shell process to finish, then exits.

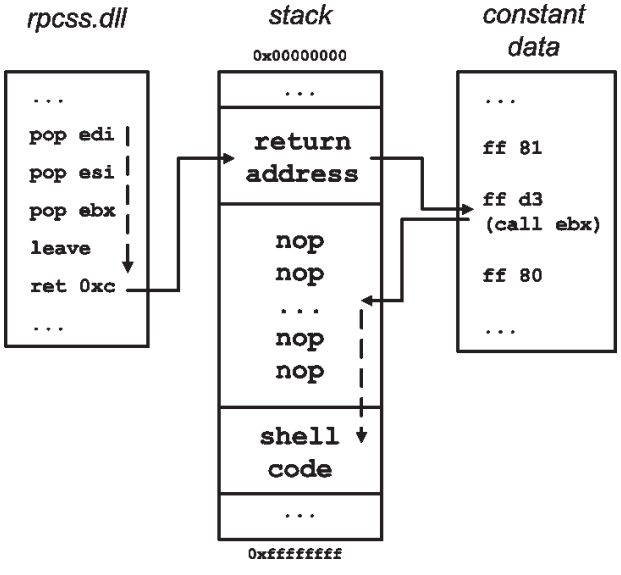

Figure 1: Memory layout and control flow.

MS-DoS

Win32/Blaster implements a SYN-flooding Distributed Denial of Service attack against the website 'windowsupdate.com', an alias of the main *Microsoft* Windows Update site.

The attack is carried out if the day of the month is greater than 15, or the month is greater than 8, i.e. every day from 16 August to the end of the year, and then starting again on 16 January until 31 January, 16 February until 28(/29) February, and so on.

To generate the traffic, the worm uses raw sockets. DoS packets have spoofed source IP addresses, the two low octets are randomized for each sent packet; the high octets are either taken from an IP of the source host, or otherwise initialized once to random values.

The traffic features various characteristics that can help in recognizing it: the two low bytes of the TCP sequence number are always zero, the source port is between 1000 and 1999 (inclusive), and the IP identification field always has a value of 256.

CONCLUSION

The first use of a command shell attack by a Win32 worm has finally arrived. The spawning of a shell had previously been used only by Win32 exploits and by Unix worms executing /bin/sh with system calls such as system() or execve().

Windows worm writers are slowly merging existing exploit code with their creations to make them more harmful. The tendency that started with CodeRed, Slammer and other Unix worms, continues. The delay between the appearances of such creations seems to have decreased from a year to six months.

It is evident that among defensive technologies, proactive behaviour blocking techniques will become essential to fight back against such 'cloned' worms in the future. Peter Ször's paper, 'Attacks of the worm clones – can we prevent them?' (to be presented at this year's RSA Europe conference in November) uncovers the details of how we can get closer to this goal and demonstrates research prototypes that work effectively against this clone.

This time not only corporate servers risked being affected; the threat had the potential to reach the majority of *Windows* desktops. This is 'Buffer Overflow for the Masses'.

Win32/Blaster

Size:	6176 bytes.
Aliases:	W32.Blaster.Worm, W32/Lovsan.worm, Win32.Poza.A.
Type:	Worm, exploits buffer overflow vulnerability in DCOM/RPC (described in Microsoft's MS03-026 Security Bulletin).
Trigger:	DoS attack attempt against windowsupdate.com after 16th of each month or after August each year.