VIRUS ANALYSIS 2

WORM WARS

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

Around 1966 Robert Morris Sr., the future NSA chief scientist, decided to create a new game environment with two of his friends, Victor Vyssotsky and Dennis Ritchie. They coded it for the PDP-1 at *Bell Labs*, and named their game 'Darwin'. Later 'Darwin' became 'Core War', a computer game played to this day by many programmers and mathematicians as well as hackers.

The object of the game is to kill your opponents' programs by overwriting them. The original game is played between two programs written in the Redcode language, a form of assembly language. The warrior programs run in the core of a virtual machine called MARS (Memory Array Redcode Simulator). The fight between the warrior programs was referred to as Core Wars.

Well, the world used to be a better place with the fights between genies in a bottle. Who let the worms out?

INSTALLATION

When Win32/Welchia first runs on a machine, it checks for the presence of a mutex called 'RpcPatch_Mutex', and aborts if the mutex already exists, in order to avoid running multiple instances of itself.

After creating its mutex, Welchia creates two services, one for the worm itself, configured to start automatically, and one for a TFTP server used during replication, configured to start manually. The service display names are 'WINS Client' and 'Network Connections Sharing', and the worm attempts to assign their descriptions from the legitimate 'Computer Browser' and 'Distributed Transaction Coordinator' services, respectively. The service executables are located in the %system%\wins directory and named DLLHOST.EXE and SVCHOST.EXE.

When started as a service, Welchia registers a basic service handler procedure with the Service Control Manager. (The handler procedure is kind enough to honour the STOP control requests, which makes it easy to stop the worm process on infected machines.)

BLASTER-BOMBER

Welchia attempts to remove Win32/Blaster.A from the machines it infects. More precisely, it kills any 'msblast' process by name (regardless of the extension, and the process name comparison is case-insensitive), and it deletes from the system directory any file named 'msblast.exe' (after stripping from it a possible read-only attribute). This 'cleaning process' takes place immediately before the worm starts attacking new hosts, and periodically at the end of every infection cycle.

Welchia also checks for and attempts to install the MS03-026 patch for the RPC DCOM vulnerability on systems whose code page and locale match one of the following: China, Korea, Japan, Taiwan or US English. It duly reboots the machine after installing the patch.

By virtue of this curative effect, Welchia has been deemed by some to be a 'good worm'. This term is arguable for many reasons (incomplete coverage of the patching routine, unwanted side-effects, lack of control, etc.). The concept of a 'good worm' has been researched repeatedly, with little success. We would rather call it a 'jealous worm'.

COMPETITIVE ADVANTAGE

Welchia exploits two common vulnerabilities to infect new systems: the recent RPC DCOM vulnerability (MS03-026) already used by Win32/Blaster; and the NTDLL overflow from March 2003 (MS03-007) that the world came to know as the 'WebDAV' vulnerability, after one of the multiple paths that lead to its exploitation. (Indeed, Welchia uses the *IIS 5.0* WebDAV functionality to exploit the latter vulnerability.)

The use of multiple buffer overflow exploits to spread is new to Win32 worms (although it has been used in the *Linux* world by worms like Millen, Ramen or Adore). For Welchia, the use of multiple vectors is a competitive advantage in its fight against Blaster, because the pool of potential targets is increased compared to Blaster.

As a result of Welchia patching some machines against the RPC DCOM vulnerability, the digital environment becomes tougher for both Blaster and Welchia, but Welchia still enjoys a large base of unpatched IIS systems, giving it the edge in a 'survival of the fittest' race between the two worms.

ECOLOGICAL NICHE

After starting the TFTP server service that was created upon installation, and preparing the attack buffers for both of its exploits, Welchia starts its infection cycle, which has four phases targeting various ranges of IP addresses with different methods.

Before each attack phase, Welchia checks if its host is connected to the Internet by attempting to resolve the DNS name 'microsoft.com' to an IP address. If it can do so, it carries on with the attack, otherwise it waits 10 minutes and checks again.

The first attack phase targets the class-B network of the host with the RPC DCOM exploit. The class-B sized network is scanned linearly from top to bottom. Each IP address in this range is pinged. If the machine replies to the ping, the worm attempts to exploit it.

The second phase targets three class-B-sized networks (i.e. about 200,000 IP addresses) located nearby the host class-B with the RPC DCOM exploit.

The third phase targets one class-B-sized network, picked randomly from a hard-coded list of 76 target networks, with the WebDAV exploit. The target networks, most of which belong to Chinese organizations, were probably pre-scanned to find vulnerable machines. The WebDAV exploit carried by Welchia works only on some double-byte character platforms, which correlates with the geographic distribution of these targets.

Finally, in the fourth phase, the worm randomly selects either the RPC DCOM exploit or the WebDAV exploit and uses it against 65,536 random IP addresses. The addresses are selected from the following class-A network ids:
60 to 66, 128 to 172, 192 to 200, 202, 203, 210, 211, 218, 219, 220.

During all of these attacks, the worm avoids IP addresses containing the byte 0xc5, because it needs to patch the shell code with the bytes of the IP address xored with 0x99. Since 0xc5 xored with 0x99 would be 0x5c, which turns out to be the backslash character ('\'), the worm needs to avoid using it in both exploits, because they both involve over-long paths. As a consequence, the lucky few with IP addresses containing 197 (0xc5) will never be attacked!

SPL

When a machine is successfully exploited, be it with the RPC DCOM exploit or the WebDAV exploit, a connect-back shell code is executed on the victim. Unlike Blaster, which uses a binding shell code, Welchia expects the victim to open a connection to the attacking instance of the worm. The attacking instance binds a server that provides commands to the remote shell.

The server port was meant to be random in the range 666 to 765, but on most *Windows* systems the port ends up being 707 all the time. This is because the worm is calling srand() on the current tick count when the main thread starts, and then calls rand() from the server thread.

The random seed being a Thread Local Storage value on most versions of MSVCRT.DLL, the random number generator is practically uninitialized and always returns 41 on the first call (666 + 41 = 707). (For a description of Thread Local Storage, see *VB*, June 2002 p.4.)

Once a connection is established between the attacking instance and the remote shell, the shell command server on the attacking side issues some commands whose purpose is to make the victim download the worm through a TFTP transfer.

Unlike Blaster, Welchia does not implement its own TFTP server. Instead it carries around a TFTP server when it spreads. However, if Welchia comes to a new machine that has a file named tftpd.exe in the %system%\dllcache directory (this file is usually present on server flavours of *Windows 2000*), it will abandon the old TFTP server and snatch this new executable to carry around.

RPC VAMPIRE

The RPC DCOM exploit in Welchia is a stack buffer overflow very similar to that used in Blaster. The hard-coded return address to a 'call ebx' instruction used to hijack control is specific to *Windows XP* systems.

The exploit uses a connect-back shell code that opens a connection to the attacking machine, spawns a 'cmd' process whose input and output get associated with the socket connected to the attacker, and finally calls ExitThread().

The use of the ExitThread() API instead of the ExitProcess() API (used by the Blaster shell code) ensures that the RPC service survives the attack. Thus, machines infected by Welchia will not present the same symptoms as those machines infected by Blaster (unexpected rebooting, unavailability of some services, and so on …)

IMP SPIRAL

Welchia's WebDAV exploit hijacks an exception handler on the stack by overflowing a buffer in RtlDosPathNameToNtPathName_U (as do most published WebDAV exploits). From KiUserExceptionDispatcher, control goes to a 'call ebx' instruction in a 'well-known' data area of the inetinfo process (the same 'well-known' area to which we referred in our previous *VB* article on Blaster, see *VB*, September 2003, p.10). Details on the KiUserExceptionDispatcher function and how it relates to exception handler hijacking, can be found in Péter Ször and Bruce McCorkendale's article on CodeRed (see *VB*, September 2001, p.4).

Register ebx is then pointing to the hijacked exception record on the stack, one entry in a series of eight 8-byte exception records. Interpreted as CPU instructions, this series of exception records forms a ramp of pushes and pops designed to avoid execution of the exception handler addresses. Eventually, control flows to the beginning of a

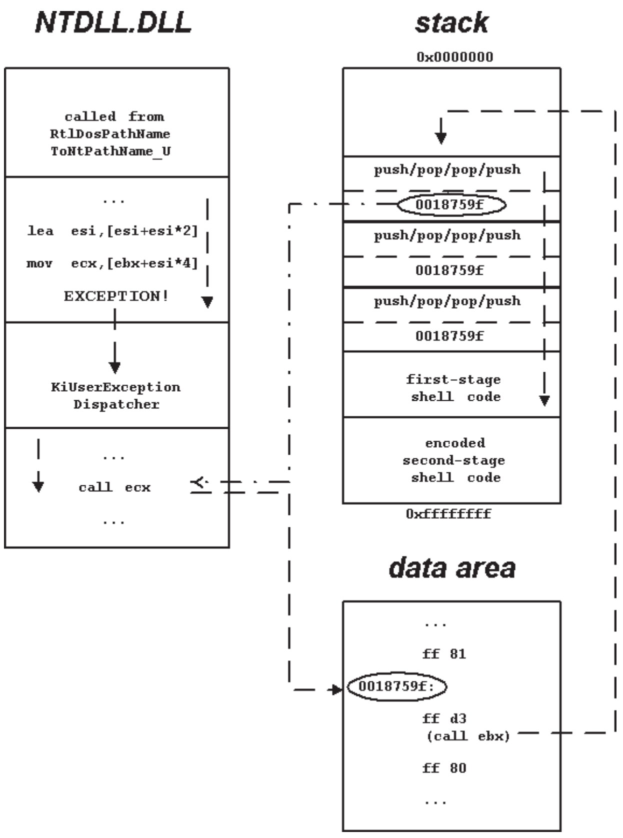

Figure 1: WebDAV exploit diagram.

first-stage shellcode. (See figure 1 for a diagram of the exploit phases.)

The exception record data are supplied as %u Unicode-encoded characters in the request URL. The Unicode-encoded characters are properly decoded only on double-byte character systems. In a typical US-English *Windows 2000* setup, question marks are substituted for the Unicode-encoded characters, and the attack results in a Denial of Service against IIS.

Interestingly, this WebDAV exploit uses a chain of three consecutive shell codes. It may seem suboptimal at first glance, because one shell code would be enough to achieve the same goal, but it results from a desire to separate the shell code functionality into reusable components. (Indeed, variants of the first-stage decoder, described shortly, were published previously.)

Once the control flow of IIS is hijacked, the first-stage shell code (supplied as Unicode-encoded characters) gets control, locates and decodes the second-stage shell code and jumps to it.

The second-stage shell code is encoded as a stream of lowercase letters that gets expanded by IIS, from ASCII to

Unicode, by inserting a zero byte in between consecutive letters. Each letter encodes four bits of information (a nibble). The first-stage shell code thus decodes one expanded dword into one byte of second-stage shell code.

The purpose of the second-stage shell code is to locate the third-stage and final shell code (shared between the RPC and WebDAV exploits). To achieve this, the second-stage shell code sets up an exception handler and strides through the process memory space, starting at 1 megabyte, in 16 kb increments (with 4 kb hops in case of page faults), looking for the ramp of Ns that constitutes most of the attack URL.

Once the ramp is found, the second-stage shell code scans forward byte-by-byte for the 'YXYX' marker that indicates the entry-point of the third-stage shell code, then jumps to this entry-point.

The third-stage shell code is the same as the one in the RPC DCOM exploit, described above.

POPULATION

As we write this article, Welchia attacks show no sign of slowing down. One of the authors of this article collected data on the ping probes coming from systems infected with Welchia (they can be distinguished from regular pings by their peculiar payload).

The data in figure 2 were collected from 24 August 2003, 17h00 (PST) to 3 September 2003, 23h59 (PST) and represent the number of hits from Welchia per one-hour time slice coming to one DSL IP address.

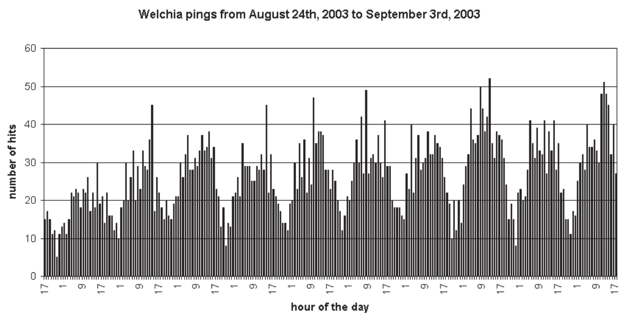

Figure 2: Welchia pings statistics.

LEMMING OF THE YEAR

Come 2004 and Welchia will jump off the cliff. Each time the worm starts it checks the current date, and if the year is 2004 it deletes its services and its files, and exits. It should be emphasized that the worm only checks the date when it starts, so that infected machines running unattended will not automatically stop attacking new systems on new year's eve. The service needs to be restarted, or the machine rebooted, for the wormicide to take place.

(As a side note, Peter Ferrie insists that lemmings do not really jump off cliffs to commit suicide. He claims they are in fact migrating to another island, and trying to get a head start because they don't swim very well …)

CONCLUSION

One genie jumped out of the bottle in about 1988, providing an early warning to all of us. Did we all learn the lessons of Morris worm? Sadly, it appears 15 years was enough to forget the warning.

Evidently worms will use multiple exploits in the future and will do so in more and more obfuscated ways. Under the attack of more than one major worm outbreak it is going to be increasingly difficult to provide accurate information about worm attacks.

Learn about the vulnerabilities in time, foresee all possible exploitation vectors ('WebDAV' was challenging in this respect), and recognize the exploit code in the blink of an eye. Without that, the information about protection might be only half-right or worse.

We have already highlighted the importance of worm blocking techniques. Not surprisingly, the basic principles to stop these attacks are the same for Morris, Blaster or Welchia (for both exploits) – however their description is beyond the scope of this article.

Will the entire Core Wars move to real networks in the form of new worm attacks? You could protect yourself, or play the ignorant for yet another 15 years. You decide!

Win32/Welchia

Size:	10,240 bytes.
Aliases:	W32.Welchia.Worm, W32/Nachi.worm, WORM_MSBLAST.D, Lovsan.D, W32/Nachi-A, Win32.Nachi.A, Worm.Win32.Welchia.
Type:	Exploits RPC DCOM vulnerability (MS03-026) and WebDAV vulnerability (MS03-007).
Payload:	Removes Win32/Blaster.A, patches some systems against MS03-026.