# VIRUS ANALYSIS 1

## THE WORMPIRE STRIKES BACK

*Peter Ferrie, Frédéric Perriot*
Symantec Security Response, USA

It took less than six months before W32/Welchia (see *VB*, October 2003, p.10) returned to plague us. The new version has been upgraded to attack different worms and exploit more vulnerabilities. Once again, the author of the worm intended to make a 'good' worm, disregarding the master's warning: 'A Jedi uses the Force for knowledge and defence, never for attack.'

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

After creating its mutex, the worm attempts to open a service called 'WksPatch' and query its status. If this service is set to start automatically, then the worm attempts to delete a file called 'svchost.exe' and start the service. Otherwise, the worm copies itself to the '%system%\drivers' directory as 'svchost.exe', and creates a service called 'WksPatch', using a random display name. The display name is composed of one entry from the list:

System Remote Performance License
Security Routing Network Internet

followed by one entry from the list:

Logging Procedure Event Manager Accounts

followed by one entry from the list:

Provider Messaging Sharing Client

The worm copies the existing service description from the 'MSDTC' or 'Browser' service, if available, otherwise it uses 'Network configuration manages by updating DNS names' as the description.

### YOU SEEK YODA

The replication code begins by querying the W*indows* version number and the locale identification information. The worm pays particular attention to Japanese locales, which are used to activate the payload, and increase the attack range and strength.

On Japanese systems, the worm creates a file called 'temp.htm' in the current directory. This file contains dates that are politically-sensitive to the Japanese. The worm queries the '/' value in the 'HKLM\SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\Virtual Roots' registry key, which returns the list of directories provided by *IIS*. In the first of these directories, the worm copies the 'temp.htm' file over any file whose extension is one of: asp, htm, html, php, cgi, stm, shtm, or shtml. The worm also does this in the '%windir%\Help\iisHelp\common' directory, then deletes 'temp.htm'.

### WE'RE DOOMED!

On non-Japanese systems, the worm attempts to remove W32/Mydoom.A and W32/Mydoom.B, if either of them is present. Mydoom.A is removed by deleting the 'RpcPatch' service, deleting the 'TaskMon' value from the 'Software\Microsoft\Windows\CurrentVersion\Run' registry key in both HKLM and HKCU, and deleting the 'TaskMon.exe' and 'shimgapi.dll' files from the '%system%' directory. Mydoom.B is removed by deleting the 'Explorer' value from the 'Software\Microsoft\Windows\CurrentVersion\Run' registry key in both HKLM and HKCU, and deleting the 'Explorer.exe' and 'ctfmon.dll' files from the '%system%' directory.

During Mydoom.B removal, the worm overwrites the 'hosts' file in the '%system%\drivers\etc' directory with a single default entry, but without checking the contents first. This can cause problems if the proper contents (which were replaced by Mydoom.B) have been restored already. Welchia.B also sets the 'InProcServer32' value in the 'HKCU\CLSID\{E6FB5E20-DE35-11CF-9C87-00AA005127ED}' registry key to '%SystemRoot%\System32\webcheck.dll'. This will cause a problem if the worm is run on *Windows 9x/Me*, where the correct value is '%windir%\system\webcheck.dll'.

### NOT MUCH TIME

The worm checks the current date of the local machine and will remove itself if the date is after 31 May 2004, or if it is more than 120 days after the worm file was created on the local machine. Removal is performed by deleting the 'WksPatch' service, and deleting the running file using a fairly well-known routine (which is a good trick, since it is a common assumption that a running file cannot be deleted).

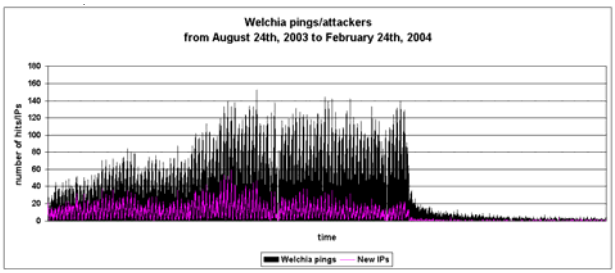



*Figure 1: Daily frequencies of attacks recorded on a typical DSL machine from August 2003 to February 2004. The dramatic drop in the rate of Welchia.A ping sweeps occurred around 1 January 2004.*

Welchia.A (see *VB*, October 2003 p.10) also had a cutoff date: 1 January 2004. We wondered how effective such a population control method would be. Looking back at some firewall logs, it appears to have worked very well. Figure 1 depicts the daily frequencies of attacks recorded on a typical DSL machine from August 2003 to February 2004. A dramatic drop in the rate of Welchia.A ping sweeps is clearly visible – this occurred around 1 January 2004.

## IT'S NOW OR NEVER

If the worm has not expired yet, it will begin its replication phase. Generally, the worm checks for an active Internet connection, except on Japanese systems where it assumes that one exists 5 per cent of the time. The check for Internet connectivity is performed by attempting to resolve one of the names to an IP address: 'microsoft.com', 'intel.com', or 'google.com'. If a connection is not available immediately, then the worm checks again every 20 minutes.

If the locale identification information matches US-English, Korean, or Chinese PRC, then the worm checks the operating system type. If the operating system type is *Windows XP*, then the worm checks for the presence of the Messenger patch, by querying the presence of the 'SP1\KB828035' or 'SP2\KB828035' registry key in the 'HKLM\SOFTWARE\Microsoft\Updates\Windows XP' registry key. If the operating system type is not *Windows XP*, then the worm queries for the presence of the 'HKLM\SOFTWARE\Microsoft\Updates\Windows 2000\SP5\KB828749' registry key.

If the registry key(s) does not exist, the worm will wait for a random period of time, up to an hour, then silently download and install the corresponding patch. Note that since these patches require the reboot of the machine to become active, the machine remains unprotected, though it might appear to be patched correctly. (Another problem, though independent of the worm, is that the installation of [at least the initial version of] the Messenger service patch requires that the Messenger service is running. Thus, in order to patch your potentially vulnerable system, you must first make it vulnerable by starting the service!)

## QUICKER, EASIER, MORE SEDUCTIVE

Welchia.B uses four vectors to propagate: exploits for RPC DCOM, Locator and Workstation vulnerabilities, which share a common transmission mechanism, and an exploit for WebDAV using a different transmission method.

The first three exploits inject a 'connect-back' shellcode inside the respective exploited services. Unlike Welchia.A, the shellcode does not use the socket API to connect back to the attacking machine. Instead it uses the API URLDownloadToFile() to download a copy of the worm from the attacker, and WinExec() to launch it. An advantage of using URLDownloadToFile() rather than connect() is that the IP address is embedded in the shellcode as a text string rather than as a binary. As a result, Welchia.B is no longer unable to attack IP addresses containing certain bytes.

The download location used by the shellcode is 'drivers\svchost.exe' (the worm assumes that the current directory is '%system%') but since URLDownloadToFile() first downloads files to the Temporary Internet Files directory, copies of the worm can appear in somewhat surprising locations such as: '\Document and Settings\ <current user>\Local Settings\Temporary Internet Files\ Content.IE5\<random directory name>\WksPatch[1].exe'.

The WebDAV exploit does not use a connect-back shellcode. It does not need to download the worm to the local machine at all, because the worm binary is encoded in the attack URL! Thus the shellcode just decodes and writes the worm binary to a file, then executes it.

From the point of view of network traffic, Welchia.B is a little more difficult to pinpoint than Welchia.A, since the pings with a peculiar payload, the TFTP transfers, and connections to port 707 are all absent, and have no equivalent in the new variant. Besides the attack ports themselves (tcp/445 for Locator and Workstation, tcp/135 for RPC DCOM, tcp/80 for WebDAV), all other ports are variable. However, the content of the packets used for fingerprinting is a giveaway.

## GOT TO FIND A SAFE PORT

For URLDownloadToFile() to work, the attacking copy of the worm must open a pseudo-HTTP server on the attacking host, to which the victim host will connect. Welchia.B picks a random port for this server. It picks a first random port candidate, attempts to bind to it, and if this fails it cycles through up to 30,000 ports looking for one that can be bound successfully.

The pseudo-HTTP server of the worm accepts incoming connections and searches within the requests for the following strings in the following order: 'GET ', '/ WksPatch.exe ', 'HTTP/1.', and one blank line.

If all four strings are found, the worm sends a copy of itself to the requesting machine. However the worm does not send an exact copy. It patches three spots of its UPX-compressed binary image, corresponding to the PE header time-date stamp field, the PE header linker version field, and a region of the UPX header containing information used by UPX to decompress the file. Thus the weekend reverse-engineers will not be able to decompress the image with UPX.

However, it is possible to decompress the file by other means, and to recover the original values of the PE header fields.

In the case of WebDAV, where the pseudo-HTTP server is not used to transfer the worm, the worm image is patched by altering nybbles of the attack URL corresponding to the same spots of the PE file described above.

## I AM YOUR FATHER

Following in the footsteps of its predecessor, Welchia.B uses a binary strategy, attacking nearby networks in one way, and distant ones in another. Against the local class B of the attacking host, and the class Bs above and below, Welchia.B uses the RPC DCOM and Workstation exploits. Against randomly picked class Bs, Welchia.B uses the Locator and WebDAV exploits. This is probably because the WebDAV exploit, executed against web servers, is more likely to succeed against random hosts than the other exploits using ports that are often closed at the firewall. The RPC DCOM and Workstation exploits, on the other hand, are more likely to succeed against nearby hosts that belong to the same organisation. (We wonder why the Locator exploit is used remotely, and invite anyone with a good explanation to contact us.)

The attack cycle of the worm is influenced by the locale of the attacking system. If the worm is running on a Japanese system, it will never stop attacking new machines, using 600 attack threads. Otherwise, the worm will attack machines at an average rate of one class B per three-hour period, using only 100 attack threads, and will also check its cutoff date and possibly remove itself.

Within each cycle, a single exploit is picked, and attempted against all hosts of the target network, or a pool of 64kb random hosts. The approximate probabilities for the exploit choice are as follows: 33 per cent chance for WebDAV, 33 per cent chance for Locator, 22 per cent chance for RPC DCOM, 11 per cent chance for Workstation.

When the worm uses IP address randomization, it selects a class-A network ID between 2 and 239, and random network and host identifiers within this class A. However, it avoids all non-routable IP addresses used in local networks: 192.168.x.x, 10.x.x.x, 172.16.x.x-172.32.x.x.

## WEBDAV EXPLOIT

Like Welchia.A, Welchia.B attempts to exploit the targeted web server only after probing it and determining that it is running *IIS 5* and has WebDAV enabled. The exploit itself is based on the hijacking of an exception handler on the stack, leading to the execution of two stages of shellcode, and eventually the worm binary.

The same 'well-known' area of memory is referenced in the hijacked exception record that was used by W32/Blaster.A (see *VB*, September 2003 p.10) and W32/Welchia.A. Unlike Welchia.A, Welchia.B chose to use a 'jmp ebx' instruction to transfer control to the exception record. Right after the record comes a first stage shellcode, whose purpose is to decode the second stage shellcode and jump to it. The first stage shellcode is encoded in the attack URL with a series of '%u' characters (this means that Welchia.B suffers from locale dependency, just as Welchia.A did). The second stage shellcode is split into nybbles, and encoded in the URL as lower case characters, in a very similar way to the second stage shellcode of Welchia.A. In fact, the entire worm body is encoded in a similar manner, and follows the second stage shellcode in the URL. This was not the case in Welchia.A, which embedded the binary worm in the body of the SEARCH request. It is the role of the first stage shellcode to put all of these nybbles back together into a sequence of instructions. Once executed, the second stage shellcode creates the file 'svchost.exe' in the root directory of the current drive, and runs it.

The WebDAV exploit now targets random IP addresses. Welchia.B no longer uses a set of fixed class A-sized networks from which to pick its victims.

## RPC DCOM EXPLOIT

The RPC DCOM exploit code is almost identical to that in W32/Blaster.A and Welchia.A. Like Welchia.A, Welchia.B targets *Windows XP* using a stack-smashing attack.

## WORKSTATION SERVICE EXPLOIT

The *Windows* Workstation Service exploit in Welchia.B is twofold: one variant of it is designed to work against *Windows 2000* systems, and the other against *Windows XP*. The overall SMB transaction is very similar for both *Windows 2000* and *Windows XP*: the worm begins by fingerprinting the operating system of the target machine remotely, by sending a protocol negotiation request (on behalf of '*Windows Server 2008*') and comparing the *Windows* version to 5.0 (*Windows 2000*) or 5.1 (*Windows XP*) in the negotiation answer.

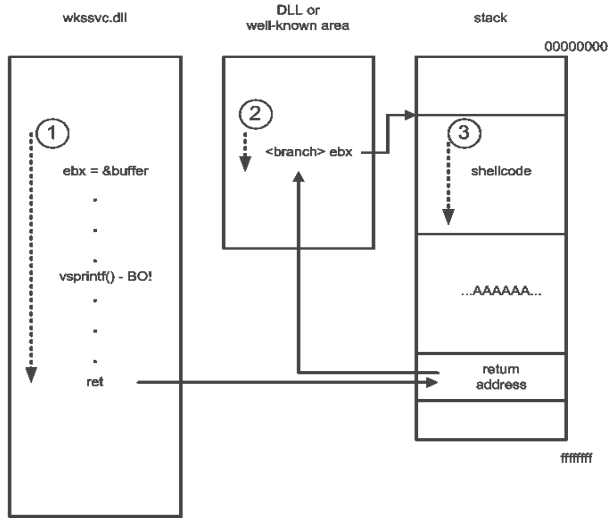
Then the worm connects to the 'wkssvc' pipe, binds to it through the service's RPC interface, and sends it a malformed request. The request is crafted differently for *Windows 2000* and *Windows XP*. In the former case, a 'NetrValidateName2' request is issued, with a shellcode preceding an overlong ASCII name. In the latter case, a 'NetrAddAlternateComputerName' request is issued, composed of an overlong Unicode name with a shellcode embedded in it. The name is composed of all 'A' characters.

The control flow of the Workstation Service exploit also differs depending on the target operating system. On *Windows 2000*, the transfer of control is performed by a 'call ebx' instruction reached by overwriting a return address on the stack. At this point, the 'ebx' register points to the buffer containing the shellcode. On *Windows XP*, a classic flavor of stack-smashing is used with a hijacked return address pointing to a 'jmp esp' instruction leading to a small jumpcode directly following it in memory. The jumpcode jumps backwards to the shellcode.

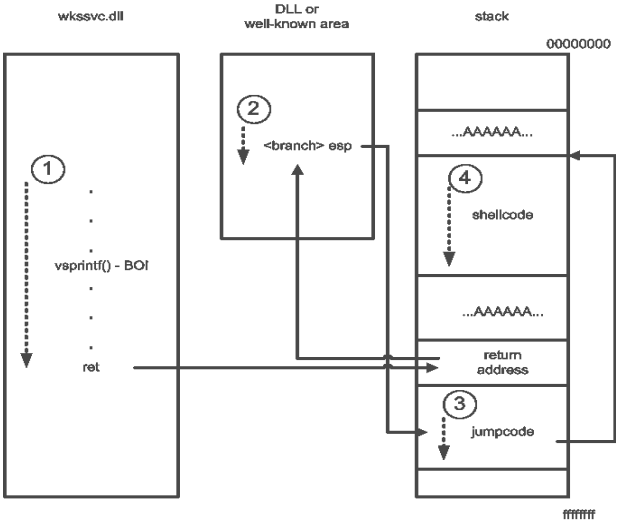
Interestingly, the addresses for the aforementioned trampoline instructions, 'call ebx' and 'jmp esp', are not always picked from a hard-coded list in the worm. Instead they are computed dynamically on systems whose locale is unidentified by the worm, by parsing a set of DLLs for instruction patterns. If the local operating system is *Windows 2000*, the worm searches the libraries 'mprapi.dll' and 'rtutils.dll' for a 'call ebx' or a 'call esp'. If it is *Windows XP*, it searches the libraries 'ws2_32.dll' and 'wshtcpip.dll' for a 'jmp ebx' or 'push esp/ret'. Astute readers will notice that this will lead to picking the wrong trampolines 50 per cent of the time, since the *Windows 2000* exploit requires the use of 'ebx' and the *Windows XP* exploit requires the use of 'esp'. The bug is the result of a miscalculation of the index in the pattern array lookup.

The idea of looking for trampolines on the attacking host and using the obtained addresses against the attacked remote host may seem a bit disconcerting at first. The worm operates on the assumption that the target machine uses the same operating system as the attacker. This is coherent with the scanning strategy used for the Workstation Service exploit, though, since the worm attacks only nearby class Bs.

Workstation Service exploit on Windows 2000




Workstation Service exploit on Windows XP




## LOCATOR SERVICE EXPLOIT

The Locator Service exploit works almost exactly like the *Windows 2000* Workstation Service exploit described above. It smashes a stack buffer and transfers control to the shellcode through a 'jmp ebx' at a known address in memory.

## NO, THERE IS ANOTHER

As we write this article, the latest variant of Welchia is .D, which uses an additional propagation vector in the form of the backdoor installed by W32/Mydoom. Welchia.D also attempts to clean systems from more competing worms. The worm war is raging!

### W32/Welchia.B

| | |
|---|---|
| Size: | 12,800 bytes, UPX packed. |
| Aliases: | W32/Nachi.worm.b, W32/Nachi-B, Win32.Nachi.B, WORM_NACHI.B, W32/Welchi.B, Worm.Win32.Welchia.b |
| Type: | Exploits RPC DCOM vulnerability (MS03-026), WebDAV vulnerability (MS03-007), Workstation vulnerability (MS03-049), Locator Service vulnerability (MS03-001). |
| Payload: | Removes W32/Mydoom.A and W32/Mydoom.B. Patches some systems against MS03-043 and MS03-049. Defaces websites. |