ISSN 0956-9979

APRIL 2003

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

IN THIS ISSUE:

• **Letting the bed bugs bite**: W32/Serot is plagued by programming errors that almost disable it, but the likelihood of some of its capabilities being included in future infectors is high. Peter Ferrie describes the worm that snatches defeat from the jaws of victory. See p.5.

• **Laying down the law:** The National Hi-Tech Crime Unit (NHTCU) is the UK's first national law enforcement organisation dedicated to tackling the new and emerging threat of hi-tech crime. Nina Gaubert explains how the Unit operates. See p.14.

• **Opportunity knocks**: According to the Anti-Spam Research Group, the scale, growth and effect of spam on the Internet have generated considerable interest in addressing the problem. Not only that, but many AV companies have identified the spam problem as a money-making opportunity. Peter Sergeant looks at some of the growing number of anti-spam solutions from anti-virus vendors. See p.19.

CONTENTS

COMMENT

“ It seems customers receive solutions straight from a Frankenstein horror flick, with products forced to fit and work together on a prayer. ”

The New Role of Content Filtering?

2001 was dubbed by some as ‘The Year of the Worm’. From that year to the next we saw developments in worming (*sic*) technology and how it was effective in affecting as many establishments as possible when made into a ‘blended threat’. One could perhaps go so far as to say that the use of worms as a malware propagation technique will be the prevailing trend for many years to come.

Another growing concern is the influx of unsolicited spam. How does one get this? Well, for starters, there are several free email services out there, in many of which the small print states that your signing up constitutes acceptance to receive ‘third-party information’ (read as spam). The ways one can be rid of all this junk range from changing default settings on the next logon, to paying for the commercial version, or worse, responding to each spammed email individually (why not just look for a provider who doesn’t foist all this junk on you?). Examining further we find the unscrupulous collection of email addresses culled from cookies or even web pages using spyware robots which, in turn, are sold to bulk-mailing marketers – it’s that telemarketer from hell once again, but this time online!

Now imagine this scenario: what if one of these bulk-mailing entities gets infected with a mass-mailing worm? How about a compromised news list server spewing backdoor Trojans to unsuspecting subscribers? Unfortunately, one does not have to wait for this scenario to happen. A modification of the same open-relay spamming technique is a common implementation for worms today.

Is there any solution to this nightmare? By degrees, over the years, content filtering has had its purpose and definition changed from merely filtering out insults, threats, racist comments and other text-written undesirables, to becoming a key factor in protecting incoming and outgoing data. Simple content filtering to combat malware can be achieved by the use of rules that block attachments that arrive in messages that contain a certain subject line or even a uniquely identified string in the message itself. That was just right a few years ago, but today’s worms have the capability to use several other extensions and subject lines, as well as message bodies.

What about rogue packets that are used to launch DoS attacks on your servers and the accompanying memory-resident malware that tries to lodge itself in memory? It’s interesting that, as I write this, a sudden burst of CodeRed.F infections started up bells and whistles on one of the distributed monitoring stations. These days the content that needs to be filtered may include stuff that goes through the wire as well as at the destination itself.

Another issue for filtering content is strategic functional positioning. Logically, the best place to integrate is at the gateway. This ensures low resource overheads and one-point control of all content in an enterprise – and this is where everything is usually done today. However, nothing is perfect and thus backup solutions at the desktop level should be at hand. With distributed control given to users, have we considered the ways to manage it? It seems customers receive solutions straight from a Frankenstein horror flick, with products forced to fit and work together on a prayer. This is where collaborative anti-virus solutions come to the fore. There is a need for one-point deployment and raw logging feedback, with filters that would assist pinpointing the problem.

So, before you install any of those oddball products on your gateway, ask yourself if their functionality fits the bill. Can the product safeguard incoming as well as outgoing information without putting too much strain on your current security policies? Is it heuristically smart enough to determine a threshold before your email servers are bogged down? Will it allow you to assert overall control of your other distributed services? More importantly, will it work with the rest of your current installed products to create rules that are robust enough to ensure that your current filtering software can change with the times?

Collaborative anti-virus solutions are what customers need, not tomorrow, but today.

Jaime Lyndon ‘Jamz’ A. Yaneza, Trend Micro, Philippines

NEWS

Security in the Classroom

Microsoft seems to be taking security education seriously these days. The software company has pledged support – both financial and in the provision of resources – for a new course which will teach students to write secure code, at the University of Leeds in the UK. *Microsoft* will provide material based on its 'stride' methodology, which university staff will incorporate into code-writing classes. Unlike previous instances in which *Microsoft* has pledged support to educational institutions, the staff at Leeds University will be given free rein to use the *Microsoft* materials and funding to draw up their own curriculum as they see fit. The university has also been granted all intellectual rights and the freedom to distribute/sell its code-writing course materials to other educational institutions.

While some may snigger at the mild irony of *Microsoft*, a company famed for producing software with security flaws, embarking upon an initiative to promote secure code writing, the move does indicate a much-needed step towards the introduction of security engineering into mainstream computer education. The course will be available from January 2004 ▮

Same Old, Same Old

Just days after his creation made its first appearance in the Wild, the suspected author of the Iraqi war-themed W32/Ganda worm has been tracked down by Swedish authorities. The man is said to have confessed to having written and distributed the worm, which posed as a screensaver offering spy satellite photographs of Iraq. [*A full analysis of W32/Ganda is scheduled for the May 2003 issue of Virus Bulletin.*] Scoring zero out of ten for originality, the suspect made the predictable claim upon his arrest that he had little idea that the virus would cause so much disruption. This has become something of a standard (and frankly boring) response from those charged with virus-writing crimes: take for instance David Smith, who 'had no idea there would be such profound consequences to others'; Jan De Wit, who 'did it without thinking'; Simon Vallor, who 'didn't have a clue it would do that' and so the list goes on. A pretty clueless bunch, or so they would have us (and a sympathetic jury) believe ▮

What, no Comparative?

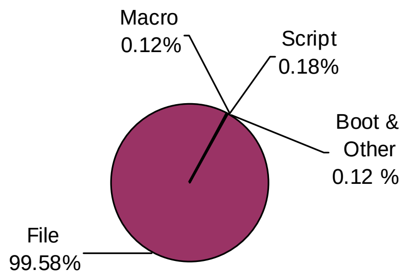
VB apologises to readers expecting to find a comparative review in this month's issue of *Virus Bulletin*. A virus of biological nature can be blamed for holding up proceedings. The health of *VB*'s technical consultant restored, testing for the *RedHat Linux* comparative has commenced and the review will appear in the next issue of *VB* (May 2003) ▮

Prevalence Table – February 2003

Virus	Type	Incidents	Reports
Win32/Opaserv	File	3473	38.42%
Win32/Klez	File	2932	32.43%
Win32/Yaha	File	679	7.51%
Win32/Dupator	File	385	4.26%
Win32/Bugbear	File	260	2.88%
Win32/Lirva	File	229	2.53%
Win32/Funlove	File	191	2.11%
Win32/Sobig	File	156	1.73%
Win32/Magistr	File	146	1.62%
Win32/Gibe	File	136	1.50%
Win95/Spaces	File	120	1.33%
Win32/SirCam	File	44	0.49%
Win32/Hybris	File	41	0.45%
Win32/BadTrans	File	35	0.39%
Win32/Nimda	File	32	0.35%
Win95/Lorez	File	29	0.32%
Win95/CIH	File	25	0.28%
Redlof	Script	11	0.12%
Win32/Lovgate	File	11	0.12%
Win32/Kriz	File	10	0.11%
Win32/Braid	File	8	0.09%
Win32/Parite	File	8	0.09%
Win32/MTX	File	7	0.08%
Others [1]		72	0.80%
Total		9040	100%

[1] The Prevalence Table includes a total of 72 reports across 43 further viruses. Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

Distribution of virus types in reports



LETTERS

All Quiet on the Virus Front?

Last year I predicted that 2003 would be a very quiet year (see *VB*, May 2002, p.16 and *VB* October 2002, p.17). Has it, so far? Well, yes, for viruses, and no, for Trojans.

As far as viruses are concerned, there seems to be little coming, and many in the AV industry have been taking the opportunity to catch up, and tidy up. However, we have been inundated with Trojans – particularly backdoor Trojans – and keeping these up to date is a problem we are just about handling. I am often asked, 'Will the *Microsoft Palladium* project kill the backdoors?' My answer is 'No, but it will dent them. And Bill Gates will ensure that the subsequent *Longhorn* project dents them even more.' In summary, I believe that backdoor Trojans will be nearly-dead by 2009.

However, at the end of the first week in February 2003, I was still blasé about porn dialers (see *VB*, December 2002, p.12). I thought we detected about 350 of them (mostly as apps, not Trojans) – we don't count them. An experiment was in progress in which we added detection for new dialers into our software. When detected, the dialers were classified as 'New PornDial-b', and a request was produced which read 'Please send a copy of the file.'

The following week, we received six CDs, from the University of Innsbruck, Austria. The accompanying letter said the CDs contained about 70,000 dialers, and indicated that our experiment had detected about 80 per cent of them. There was a sting in the tail. The sender expected to have another 400,000 within the next few months, and would send those too! He asked when we would be ready for them.

The first two reactions were immediate, and automatic: i) We must stop requesting copies of the files. You can have too much of a good thing! ii) We must remove the four letters 'Porn' from all the names. (Phredd, who has sweated blood writing his beautiful Internet program, will want our guts for garters, when we false alarm it as a porn dialer!)

The samples will, of course, be available to all AV companies, and each will decide (whether by default, or by conscious decision) whether they process none, some of them, or all of them. They will also have to decide whether they count none, some, or all, irrespective of what they have processed. (Graham Cluley, you can have a ball with this one!) Anyone who processes large quantities of dialers will have no option but to use generic techniques. Since generic detection without generic repair is quite easy, this will not be a problem. But counting may well be.

I have two other comments:

1. My opinion has been reinforced, that over the next few years software reviewers will become more powerful, and their views and activities will push the AV industry in various, perhaps new, directions.

2. Everybody seems to believe that the number of companies in the AV industry will contract over the next 10 years due to take-overs and consolidation. Whilst I agree with this, I suspect that the resulting larger companies will tend to be security companies, with anti-virus software as just one of their products or components.

Peter Morley, NAI, UK

Watching out for Spam

In response to Lavash en-Rangé's questions (see *VB*, January 2003, p.4): '*Should we be looking to anti-virus vendors to go after spamming resellers more vigorously? Is it Symantec's job to track down those pimping cheap copies of Norton Anti-Virus? Is there an address to which I can forward emails trying to flog me McAfee's software at rock-bottom prices?*' Spam from third parties attempting to sell *Symantec* products can be forwarded to spamwatch@symantec.com. See http://www.symantec.com/spamwatch for more details.

Eric Chien, Symantec

Still a Rose?

In Phil Wood's letter in the February issue of *VB* (see *VB*, February 2003, p.4), a reference is made to something called 'VBS/Kakworm' (a description of which can be found at http://www.sophos.com/virusinfo/analyses/vbskakworm.html). All members of this virus family use 'Java Script' language, usually abbreviated to 'JS'. Hence, the correct name for this virus *cannot* contain 'VBS' – its correct name is 'virus://JS/Kak.a@m', or 'JS/Kak.a@m' (for a summary of the CARO naming scheme see *VB*, January 2003, p.7).

It would really be great if AV researchers paid more attention to naming malware correctly (could we always get at least the 'platform' part right?), or followed users' demands for synchronisation of names. As we can see, it is not only the ordinary users that can be confused by the use of the wrong name.

Igor Muttik, NAI, UK

VIRUS ANALYSIS 1

Sleep-Inducing

Peter Ferrie
Symantec Security Response, Australia

W32/Serot@mm is another creation from Benny the virus-writer. Its name is derived from the word 'serotonin', which is a chemical found in the brain that has been linked to the onset of sleep, among other things. If anyone is wondering whether, this time, Benny has released a bug-free virus … the answer is no. Serot is plagued by programming errors that almost disable it, however some of its capabilities are worth describing, in case another virus appears with these bugs fixed.

Serot uses a 'plug-in' architecture – which was very successful in W95/Hybris. However, the plug-ins in Serot are almost completely self-contained, even carrying their own buffers if the buffers are 'small' enough, resulting in an enormous collection, and much redundancy in the code.

Buggy Benny

Serot is designed for *Windows NT* and later operating systems. The virus checks explicitly for *Windows 9x*/*Me*, and will exit if it is run on any of them. In addition, the virus checks whether NTDLL is accessible using the GetModuleHandleA() API (i.e. that it is resident in memory already), which could affect emulators.

The virus assumes that PSAPI.DLL is present on the system – which is not always the case – and the code will crash if the file does not exist. Serot contains some anti-debugging code which is supposed to cause the virus to exit gracefully if a debugger is detected – due to a bug, however, Serot will usually crash instead.

Attack of the Clones

Next the virus checks whether Serot is running on the system already. It attempts to open a mutex called '$serotonin@', but there is no branch to exit if the open is successful. This results in multiple copies of the virus running at the same time.

Serot contains some variables whose contents are increased or decreased by small random values. However, since there are no bounds checks on the alterations, the contents can become negative values – with disastrous consequences. Serot also relies on the result of several APIs being a certain fixed value, even though they are defined as returning either zero or non-zero. If *Microsoft* should ever change this value, Serot will not work at all.

Serot enumerates processes, looking for the Explorer.exe process, which it uses to remain memory-resident. If the process is found, Serot injects some code into the process, and runs that code.

The injection technique has been used several times by viruses since the first time it was demonstrated, in the W32/Dengue virus, in the year 2000. Previous viruses hooked an API in order to gain control and run the code, for compatibility with *Windows 9x*/*Me*. Since Serot is specific to *Windows NT* and later operating systems, it has no need to hook an API, and can use the CreateRemoteThread() API to run the code.

The Cat that ate the Rat that …

The injected code waits for 10 minutes before executing its main routines. This gives the launch process time to terminate, as well as providing an anti-heuristic device. After the time has elapsed, Serot will attempt to delete the file that was used to launch the code, and the registry key that might have been used to launch the file.

At this point, Serot calls a member of the first group of its plug-ins. The plug-ins are in groups based on type. For example, the first group terminates anti-virus and firewall software, based on the window title; the second group gathers email addresses from the 'mailto:' string in files, from the MAPI Address Lists if *Outlook* is running, or from the Windows Address Book; the fourth group sends the virus by email; the seventh group converts *.NET* files into droppers of the virus.

For groups that contain more than one member, the routine that calls the plug-ins will call a single random member from within that group.

After the first plug-in has returned, Serot generates a keypair for use in encrypted communication with other infected machines, then calls the plug-in that gathers email addresses. If that plug-in returns a failure, then a bug in the code results in the loss of the keypair, and a leakage of the cryptographic context.

Let's Swap

Serot carries a list of IP addresses of known infected machines, and looks for other infected machines by attempting to connect on port 194 to IP addresses found in the registry key 'HKCU\Software\Microsoft\Ftp\Accounts'. There is another routine that attempts to connect to random IP addresses – however, as the result of a bug, this routine is never called.

Port 194 is a well-known port, reserved for the Internet Relay Chat (IRC) protocol, so traffic on this port is not unusual. If a machine is found to be listening on this port, Serot will verify that the machine is infected, by attempting to establish a communication with it using a private protocol. This protocol contains data encrypted using public key cryptography, in order to avoid packet sniffing.

The protocol is established by Serot on the local machine by sending its public key to the server on the remote machine, and examining what is returned. If a valid public key is returned from the remote machine, then the copy of Serot on the local machine will send its current configuration to the remote machine, encrypted with the public key that was returned by the copy of the virus on the remote machine. In return, the copy of the virus on the remote machine will send its current configuration to the local machine, encrypted with the public key that was returned by the copy of Serot on the local machine. Thus, the two copies of the virus are able to exchange behavioural characteristics and their plug-ins.

Several bugs exist in this code, resulting in a number of allocated memory buffers that are never freed.

A New Way to Express It

If no other infected machines are found, Serot will attempt to send itself by email. Yet another bug exists in this code, resulting in the occasional failure to send messages.

The email message uses a 'feature' of *Outlook Express* which involves the UTF-7 encoding of plain-text messages. Using UTF-7 encoding, it is possible to encode HTML message bodies that contain scripts, which are run without user interaction. If the *Internet Explorer* security settings allow Active scripting, and the scripting of ActiveX controls that are not marked as safe, then the script will run without prompts, regardless of the zone in which it is executing.

Additionally, the email message contains no attachments, because Serot creates a message body that contains the virus in an encoded text form.

The function of the script is to decode the virus body and drop it as a file called 'c:\setup.exe'. After the file has been dropped, the registry value 'HKCU\Software\Microsoft\Windows\CurrentVersion\Run\Serotonin' is created so that *Windows* will execute the file whenever the current user logs on. This could also be considered a bug, since if there are multiple users, only the current user will spread the virus.

Dot Dot Dot Net

Once all of the other actions have been performed, Serot begins listening on port 194, and runs the final plug-in. The current version of this plug-in will convert *.NET* executable files into droppers of Serot.

The plug-in begins by searching in every subdirectory on drive C: for files whose name ends in 'exe'. Even the name matching is buggy – not checking for the '.' to indicate a suffix results in the matching of names such as 'SerotRexe'. For every file that is found, the virus checks whether it is a *.NET* file that does not contain resources or relocations, and is not protected by a Strong Name hash. If this check passes, then Serot will attempt to infect the file.

The Compiler at Your Service

Serot infects files by using the Compiler Services methods that are exposed by the *.NET* framework. Using these methods, it is possible to decompile an existing file, add new methods and variables, and recompile the file, without requiring any understanding of the underlying structures.

The recompilation is achieved using just a few method calls in *.NET*, requiring far less effort than the manual reconstruction of standard Portable Executable files that was implemented in W95/ZMist (see *VB*, March 2001, p.6).

A virus using the manual reconstruction technique seems unlikely, since the underlying structures in *.NET* are extremely complex and contain many interdependencies. For example, an entry in the method table contains references to the #Strings stream and the #Blob stream. The entry in the #Blob stream contains references into the TypeRefs table. The TypeRefs table contains references into the #Strings stream, and also coded index references into the AssemblyRefs table. The AssemblyRefs table also contains references into the #Strings stream and the #Blob stream. It would take a long time for someone to find out what all of the links are, and work out how to update them manually.

After creating the new code that will drop and run a copy of the virus, Serot attempts to access the file that contains the data that will be dropped. However, errors in the control flow mean that the file the virus attempts to access might not have been created at all.

If the file does exist, Serot will reserve 64Mb of memory for itself, but never free it, resulting in the eventual exhaustion of system resources, since this allocation occurs for every *.NET* file on a machine.

More memory is not freed if a particular *.NET* DLL cannot be loaded, and because of another bug, that dll usually cannot be loaded. Serot also assumes that the infection will always succeed, and uses the MoveFile() API to replace the original file, resulting in deletion of the original file if an error occurred.

Conclusion

Despite its potential, this very buggy creation of Benny's snatches defeat from the jaws of victory. However, it demonstrates the Compiler Services, which seem likely to form the basis for future *.NET* file infectors. A recompiling virus like W95/Anxiety, but without needing the source code, combined with an inserting virus like W95/ZMist, but without rebuilding the file manually … The beast is unleashed, and its full power is unknown.

VIRUS ANALYSIS 2

JunkComp on Air

Jozsef Matrai and Gabor Molnar
VirusBuster Ltd., Hungary

At the end of last year an interesting polymorphic virus appeared. A user sent us a sample, which we examined in detail. The virus runs on all Win32 versions and infects most of the PE files. It contains a lot of junk code, interleaved with the original code.

'Interleaved Virus'

The virus code is interleaved with junk code and the original program code. The virus generates a long polymorphic decoder and chops it up into several parts (see Figure 1). The original program code is XOR-encrypted and saved after the end of the virus code, while the polymorphic decoder takes the place of the original program.

The maximum size of the generated decoder is 32Kb. It contains many blocks of junk code, the end of each of which contains a jump to the next block. The decoder may contain several instructions, including some co-processor instructions. The 'Decoder Loop' and the 'Jump to Decoded Part' instructions are not usually located within the same block, so additional blocks are inserted between them, as shown in Figure 1.

The virus does not change the original Entry Point of the program, but it changes the characteristics of the infected section to writeable.

The 'Virus Body'

When the decoder has finished control passes to the decrypted 'Virus Body'(see Figure 2).

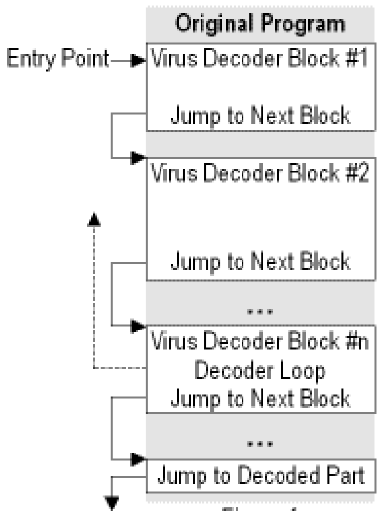


Figure 1

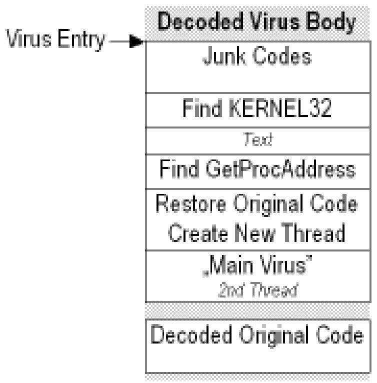


Figure 2

The beginning of the virus body always contains junk code. When the code is executed, the virus searches for KERNEL32.DLL in memory:

- First, the virus determines the address of the default exception handler, which points somewhere into KERNEL.
- Then the virus searches the memory for the DOS-EXE signature ('MZ') and PE-EXE signature ('PE'), searching backwards from the exception handler's address.
- The virus finds the GetProcAddress procedure in the KERNEL32.DLL Export Table, then determines its address.

Next, the virus restores the original code, creates a new thread, which points to the 'Main Virus', and then it executes the original program code.

The newly created thread is the infector part of the virus: it searches for files to infect and checks a number of conditions. If these conditions are met, the virus infects the target file. The decoded virus contains the text: 'JEWS NEVER SURRENDER'.

Digital Infection

Once the virus has located the KERNEL32.DLL and found the GetProcAddress function, the address of every function can be retrieved. The virus simply pushes the function name and DLL module name and gets the procedure address. Before calling any of the located functions JunkComp installs its own exception handling.

The virus uses a known infection method:

- Search the entire hard disk for files with the mask '*.*'.
- Create and opens a selected file.
- Create a file mapping.

- Infect and mark the file.
- Close the file mapping and the file, and sets the file time.
- Find the next file.

When the virus runs under *Windows Me* or *2000*/*XP* (where SFC is available), it checks whether SFC protects the target file. If the file is unprotected, the virus creates a file mapping and infects it.

Infection Criteria

The following conditions must be met for the virus to infect an executable:

- The file must be 'MZ' executable and 'PE' executable.
- The machine code must be i386, i486 or Pentium (0x14C, 0x14D, 0x14E).
- The file characteristics must be executable and not DLL.
- The section number must be greater than or equal to 3, and less than 10.
- The file subsystem must be greater than or equal to 2 (GUI or CUI).
- The file must contain a section which does not contain the Entry Point, the section's SizeOfRawData must be greater than 0x200 and the section's VirtualSize must be greater than 0x200.

If these conditions are met, the virus marks the file as follows:

- It sets the file characteristics by OR 0x01.
- It sets the characteristics of the selected section by OR 0xE0000020.
- It sets Data Directory's BaseReloc Size to 0 and VirtualAddress to 0.
- It increments the selected section's VirtualSize by 0x8000, SizeOfRawData by 0x8000 and SizeOfImage by 0x8000.

The JunkComp Decoder Generator

This generator is a tricky test for emulators, debuggers, and AV software. When building a new decoder, the generator uses almost all Intel 386, 486 and 387 coprocessor instructions, but does not use Pentium MMX, SSE, SSE2 extensions and Win32 function calls. The built-in assembler can randomly generate alias opcode – for example, 'MOV EAX, EBX' can be compiled to machine code in two ways, (0x89, 0xD8) and (0x8B, 0xC3).

VERR and VERW instructions occur in such code, but these never occur in Win32 code. VERR and VERW check the read and write privileges of a segment, respectively. DS, ES and SS are 4 Gb readable and writeable segments, CS is a readable and executable 4 Gb segment, and the size of pointers in C is 32 bits. Only in malicious code do you find instructions like this. ('MOV DS, value' and any other Win32 segment manipulation codes, especially SDLT, SIDT, are used for entering Ring0).

Junk decoders are very slow, and users are easily able to detect that a program is infected, since it starts significantly more slowly than prior to infection.

The redundancy of code in the virus is conspicuous. For example, there is a 'MOV ECX, constant' instruction, followed by some lines that do not contain ECX as a source operand, then a 'MOV ECX, constant' again. There is also a 'XOR EBX,EBX', then a JZ conditional jump. Of course, algorithmically, it is a jump without condition. A close look at the code reveals that it is a very large junk dump.

This virus loves the undocumented features of REPZ and REPNZ. There are two Intel x86 opcodes: 0xF2 (REPNZ, alias REPNE) and 0xF3 (REPE, alias REPZ). The REP assembly mnemonic can be associated with both of them. The Intel manuals claim they can be added to MOVSx, LODSx, STOSx, CMPSx, SCANSx, INSx, OUTSx instructions, and the behaviour is undefined when used with non-string instructions. However, REPZ and REPNZ can be placed before any other instruction without generating an INT 6 exception – in this case, they have no effect. Such opcode can be built twice into an assembly line of the decoder (e.g. 'REPNZ REPNZ JNZ address' equals 'JNZ address'). JunkComp can be decoded only with an efficient emulator or debugger.

There are many backward jumps in the generated code, if the code is disassembled in order of increasing memory addresses instead of execution order, it is not possible to label the assembly lines where the jumps go. The undocumented way of using REP prefixes may be disassembled as DB 0xF2, or DB 0xF3. JunkComp tests the abilities of a disassembler. After the backward jumps you can see a generated instruction list or the original program.

The decoder uses a single-byte XOR and a single-byte SUB encryption key, so the virus can easily be detected with a simple brute force crypto analysis tool. Two 32-bit registers are selected as memory pointer and cycle counter, and the others contain junk data. Most of the instructions operate on junk data but they can use the two selected registers as a source operand. The decoder can be divided into three parts: junk1, decoding loop with junk2, junk3. The decoded area can be divided into two parts: junk and fixed bytes.

Conclusion

The polymorphic decoder in this virus proves a tough test for virus emulators. Detection by emulation is not the best way to catch this virus; it is better to use crypto analysis, because the virus uses only a single-byte key. Fortunately, this virus is not currently in the Wild, and does not cause widespread infection.

VIRUS ANALYSIS 3

The Month of Lov(gate)!

Ronald C. Bautista
Trend Micro Inc., Philippines

February 2003 was the month of the Lovgate family of worms. Three variants of Lovgate, .A, .B and .C, were released consecutively in the same month. The most widespread of the variants was Lovgate.C.

Like the other variants of the worm, Lovgate.C adopts well-known social engineering techniques to aid its propagation via email.

First, it mimics an auto-reply mail by replying to incoming mails and all email messages found in the Inbox folder of the infected system's *Microsoft Outlook* and *Outlook Express*. Secondly, it presents itself as a security patch, as an installer, or as pornographic material, and in these ways it lures users into opening the email attachment – a copy of the worm itself.

In addition to mass-mailing, this worm spreads through the network by copying itself to writable network shared drives and folders.

Lov Hurts

When the worm activates, it drops several copies of itself in the *Windows* system folder using any of the following names: WinRpcsrv.exe, syshelp.exe, winrpc.exe, WinGate.exe, rpcsrv.exe.

Then it adds two entries in the autorun registry key 'HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run'. The entries are:

```
Runsyshelp = %System%\syshelp.exe
WinGate initialize = %System%\WinGate.exe –remoteshell
```

On infected systems running *Windows NT*, *2000*, or *XP*, it adds another entry, 'rpcsrv.exe', to the registry key 'HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows'. The new entries enable copies of the worm to execute at subsequent *Windows* startups.

For the same reason that it made the registry entries, the worm adds a 'run=rpcsrv.exe line' at the [windows] section of the win.ini file of systems that are running *Windows 95*, *98*, or *Me*.

The worm does not end its routine with the autostart entries. It ensures that it is always active on the infected system, even after the system has been restarted in safe mode. To do this, it changes the value of (Default) to 'winrpc.exe %1' in the registry key 'HKEY_CLASSES_ROOT\txtfile\shell\open\command'. The modification activates the worm whenever a .txt file is executed or opened.

The worm marks its existence on an infected system with a unique event named 'My I-WORM-and-IPC-20168 running!'. It checks for the event on target systems and terminates its execution when it finds the event running.

Moreover, when running on *Windows 95*, *98* and *Me*, the worm registers itself as a service process and thus does not display in the Task Manager list or in the window that pops up when the CTRL-ALT-DEL combination of keys is pressed.

A Lov Letter for Everyone

Lovgate.C sends email in two different ways. The first is by sending a reply to incoming email messages and to all other email messages in the Inbox folder of the system's *Microsoft Outlook* and *Outlook Express*. It uses the Messaging Application Program Interface (MAPI) commands to do this. The details of the email it sends are as follows:

```
Subject: RE: <original subject>
Message Body:
'<user name>' wrote:
====
>
>  <original message>
>
====
YAHOO.COM Mail auto-reply:
' I'll try to reply as soon as possible.
Take a look to the attachment and send me your
opinion! '
> Get your FREE YAHOO.COM Mail now! <
```

The email attachment may have any of the following names:

billgt.exe	midsong.exe
card.exe	news_doc.exe
docs.exe	pics.exe
fun.exe	PsPGame.exe
hamster.exe	s3msong.exe
humor.exe	searchURL.exe
images.exe	setup.exe
joke.exe	tamagotxi.exe.

By employing a simple social engineering trick, the worm leads recipients to believe that the email attachment comes from a valid source.

In addition, the worm sends emails to all addresses it obtains from *.ht* files located in the Windows and My Documents folders. The worm uses its own Simple Mail Transfer Protocol (SMTP) engine and connects to smtp.163.com, which is a Chinese domain.

The email content may be presented as a security patch, an installer, a document or as adult material.

The format of the email is selected from any of the following predefined combinations:

```
Subject: Documents
Message Body: Send me your comments
Attachment: Docs.exe

Subject: Roms
Message Body: Test this ROM! IT ROCKS!
Attachment: Roms.exe

Subject: Pr0n!
Message Body: Adult content!!! Use with parental
advisory.
Attachment: Sex.exe

Subject: Evaluation copy
Message Body: Test it 30 days for free.
Attachment: Setup.exe

Subject: Help
Message Body: I'm going crazy... please try to find
the bug!
Attachment: Source.exe

Subject: Beta
Message Body: Send reply if you want to be official
beta tester.
Attachment: _SetupB.exe

Subject: Do not release
Message Body: This is the pack ;)
Attachment: Pack.exe

Subject: Last Update
Message Body: This is the last cumulative update.
Attachment: LUPdate.exe

Subject: The patch
Message Body: I think all will work fine.
Attachment: Patch.exe

Subject: Cracks!
Message Body: Check our list and mail your requests!
Attachment: CrkList.exe
```

Lov is Sharing

In addition to spreading via email, Lovgate.C spreads in a network where there are shared drives. It copies itself to the following file names on all writable network-shared drives:

billgt.exe	midsong.exe
card.exe	news_doc.exe
docs.exe	pics.exe
fun.exe	PsPGame.exe
hamster.exe	s3msong.exe
humor.exe	searchURL.exe
images.exe	setup.exe
joke.exe	tamagotxi.exe

Long-Distance Lov Affair

Lovgate connects to the IPC of remote machines using any of the following passwords to log on as an Administrator:

123	666666
321	888888
123456	abc
654321	abcdef
guest	abcdefg
administrator	12345678
admin	abc123
111111	

In addition to these, the worm also attempts to log on with an empty password.

If the attempt to log on is successful, the worm copies itself to a STG.EXE file in the remote machine's \admin\system32\ folder. It starts the file as a service named 'Microsoft NetWork Services FireWall', then cancels the remote system's network connection.

The worm creates at most 100 threads of its remote infection routine. It creates one thread every 200 seconds and uses semaphores to track the number of threads it has created.

To Lov is to Serve

The worm not only propagates, it also carries a backdoor component, which it extracts and copies to four identical dll files: ily.dll, reg.dll, task.dll and 1.dll. These files open the infected system up as a server backdoor.

The worm invokes the 'Rundll32.exe Task.dll ondll_server' command to create a service named 'Windows Management Extension'. It invokes the 'Rundll32.exe ily.dll ondll_install' command to install itself and invokes the 'Rundll32.exe ily.dll ondll_reg' command to register itself.

To keep the backdoor active in memory, the worm adds the entry 'Module Call initialize = RUNDLL32.EXE reg.dll ondll_reg', in autorun key 'HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run', which allows the component to load on *Windows* startup.

The backdoor notifies the malware author when the server machine is online by sending a message to hacker117@163.com, for example:

```
From:  hacker117@163.com@@<user name>
Subject:  @@@@@@   OR   !@#$%^&*()_+
Message Body:
SYSTEM@<user name>
<ip address>
```

The worm and its backdoor opens TCP port 10168 and 1192, where it listens and waits for connection requests. However, user authentication is required in order to connect to the server. The following is how the authentication process may appear:

```
User Access Verification
Your Password: anypass
Sorry, Your PassWord Not Right.
Your PassWord: xyz123
OK! Please Enter:

Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.
```

Once a connection has been established, the server spawns a command shell (cmd.exe or command.com) for the remote user.

The server backdoor registers itself as a service with a LocalSystem account. The remote user is effectively granted system-level privileges.

With such high-level privileges, the remote user can do almost anything he wishes on the infected machine – for example, delete files, modify the system and steal confidential information.

In addition to the two ports mentioned, the worm also opens TCP port 20168. However, unlike the other two, this does not require user authentication. Once a remote user connects to this port, he will automatically receive a command shell.

Conclusion

W32/Lovgate.C is not significantly different from many other successful worms. Its propagation mechanism, which combines both email and network propagation, is not new, and it uses the same social engineering techniques as those that have been exploited by numerous worms in the past.

Bearing in mind that it uses such old techniques, how did this worm come to be in the wild? Is it because it uses more enticing subjects or attachment names to lure users into opening it? Or could it be because many users rely solely on their anti-virus product for virus protection and will open any email attachment, regardless of where it comes from, as long as their anti-virus software has not indicated that it is infected?

Whatever the reason for its success, the Lovgate worm has proven one thing: there is a pressing need to educate email users – both those within corporations and home users –about the safe handling of email messages and their attachments.

W32/Lovgate.C

Type:	Mass-mailer, network worm.
Removal:	Delete detected files. Registry run entries created by the worm should be deleted. The registry values that have been modified by the worm should be fixed.
Aliases:	WORM_LOVGATE.C, W32/Lovgate.c@M, W32.HLLW.Lovgate.C@mm, Win32/Lovgate.C@mm, I-Worm.Supnot.c.

OVERVIEW

The Next Big Thing: Anti-Virus goes Anti-Spam

Peter Sergeant

Spam is a growing problem – but, chances are, you knew that already, just from checking your own email.

Last month saw the first meeting of the Anti-Spam Research Group (ASRG), a group set up in affiliation with the Internet Engineering Task Force to focus on the problem of unwanted email messages. According to the ASRG the scale, growth and effect of spam on the Internet have generated considerable interest in addressing the problem.

In its December 2002 report, managed email security firm *MessageLabs* predicted that, by July 2003, there will be more spam than legitimate email. Even four months ago, one in three email messages were, according to the same report, unsolicited mail, or spam.

Opportunity to be Seized

Of course, pestering everyday users with herbal remedies to make your mouse size increase by 581% isn't the only way to make money from spam – or so guess various anti-virus companies. *Network Associates*, *MessageLabs*, *Trend Micro* and *Kaspersky Labs* have all launched into the production of some form of spam-repellent.

Anti-virus vendors who already sell email-scanning software are likely to find the switch to anti-spam relatively painless – they already have the customer contacts, they have probably already solved the problems of integrating software with the wide range of mail servers available, and so on. They have experience of parsing email, quirks and all, and of dealing with attempts to break their email parsers.

The Open-Source Complication

Since spam is easy to acquire and safe to work with, a number of very capable free-software anti-spam applications have been created. Anti-virus vendors coming from the luxury of a market into which there are high barriers to entry may find this an unwelcome surprise.

The relationship between free and commercial anti-spam software has been complicated still further by the fact that commercial products have been based upon free software, and shared developers. Confused?

Network Associates Inc. (*NAI*) kicked off the trouble when it acquired *DeerSoft Inc.*, the company that '[created] *SpamAssassin Pro*, the proprietary (*Windows*) version of the

GPL/PAL licensed *SpamAssassin*' in January 2003. The complication is that developers employed both by *DeerSoft Inc.* and by *MessageLabs* contribute to the open-source *SpamAssassin*, from which both products take part of their code.

SpamAssassin is an open-source anti-spam project (the name of which appears to have been trademarked by *DeerSoft Inc.*), which is licensed under the same terms as Perl (the software is written in Perl) – specifically, both the General Public Licence (GPL), and the Perl Artistic Licence (PAL).

DeerSoft Inc. was started by Craig Hughes, a core developer of the open-source *SpamAssassin*, and later the company employed *SpamAssassin* creator, Justin Mason. Under the PAL, *DeerSoft Inc.* may use the open-source *SpamAssassin* code in its products, and developers may modify and enhance the code as they see fit.

MessageLabs does essentially the same thing as part of its *SkyScan Anti-Spam* (*SkyScan AS*) service, the difference being that *MessageLabs* doesn't supply a product *per se*: *SkyScan AS* is hosted on the company's own servers, through which customers' email is routed, taking advantage of a different part of the PAL.

Thus, the two companies start from the same code-base, and developers from both companies had been pooling their code in the open-source *SpamAssassin*. However, since *NAI*'s acquisition of *DeerSoft Inc.*, a developer employed by *MessageLabs* has resigned from working on the open-source product – any other effects have yet to be realized.

Company by Company

So, who's doing what, and with whom?

MessageLabs

MessageLabs offers a managed anti-spam service, in the same way that they offer a managed anti-virus service. This means you don't need to worry about applying updates to their filters, or installing any additional software. However, users can still customise certain aspects of the service.

The service works using 'patented heuristics technology', which is most likely marketing-speak for keyword and phrase detection, as well as 'Bayesian' detection of spam, a concept explained in detail at http://www.paulgraham.com/spam.html.

Despite customers' email being sent through *MessageLabs*' servers, it is claimed that this delays a given email by only one second. In a recent review of anti-spam tools by *PC Magazine* (see http://www.pcmag.com/), *MessageLabs*' anti-spam service was said to have 96.03% accuracy, while the product's nearest competitor was quoted to have 90.66% accuracy.

Network Associates Inc.

NAI offers a product aimed at home users and small enterprises, called *McAfee SpamKiller*. This works with POP3, MAPI, and Hotmail email, and looks for preset phrases within emails. The product arrives with a large number of rules, and will update itself with new rules each day. Customer reviews posted on CNET.com (see http://www.cnet.com/) were very mixed about the stability of this product, and some complained that the product had installed 'nag-ware', in the form of the *McAfee Security Center*, which pops up to tell users they ought to be protected by *McAfee* anti-virus too. This seems a little underhand if true.

Trend Micro

Trend should receive a special award for using the most baffling phrase, 'heuristic technology anti-spam filtering rules', in their press releases. *Trend* has teamed up with a company called *Postini* to add anti-spam protection to its gateway products. This uses 'heuristic-based content analysis', and claims to stop 90% of spam.

Postini is very proud of preventing what it calls 'Directory Harvest Attacks (DHAs)'. These are where a spammer sends many emails to the same domain and notes the addresses which bounce and those which do not. *Trend Micro Spam Prevention Service* for *Windows* and for *Linux* are due for release in the second quarter of this year.

Kaspersky Labs

Kaspersky gives the most away about its anti-spam solution, and it looks impressive. *Kaspersky Anti-Spam* is described as being a 'joint effort' between *Kaspersky* Labs and a company called *Ashmanov and Partners*.

The product blocks mail from open relays, performs pattern matching, and checks whether an email matches their existing spam patterns. Not only this, the product specifically matches Russian spam too. The software is available for *Linux* and *FreeBSD*, and matches text found in RTF files and *MS Word* files.

Conclusion

Anti-spam solutions seem to be the next big area of interest for many in the anti-virus industry, and currently the larger companies are pouncing. [*At the time of going to press Symantec has just announced the availability of its AntiVirus for SMTP Gateways 3.1, which includes anti-spam features - Ed.*]

Unlike the situation in the anti-virus arena, however, AV companies branching into the anti-spam industry will find that open-source software presents some serious competition for their products. The anti-spam industry is young and not yet well established, but it seems clear that there is money to be made from it. Time will tell.

TECHNICAL FEATURE

Stemming the (Over)flow

Yinrong Huang
Independent Researcher, Canada

The buffer overflow bug and its exploitation have been around for several years. The exploitation methods and concept have been well documented by David Litchfield and others (see http://www.nextgenss.com/papers/bufferoverflowpaper.rtf).

In January 2003, the buffer overflow vulnerability in *Microsoft SQL Server 2000* before service pack 3, was exploited to its extremity by the Slammer worm (see *VB*, March 2003, p.6). Slammer caused widespread disruption of Internet traffic across the globe and even prevented some ATM machines from working, causing damage that was reported to have amounted to one billion US dollars (see http://news.com.com/2009-1001-983540.html).

In this article, a new method is proposed, which will make the malicious exploitation of stack buffer overflow by the 'jmp/call esp' technique almost impossible on x86 PCs. If the buffer overflow bug is exploited by a piece of malicious code like Slammer, the overflow will not be prevented, but will be trapped by this method. The method involves inserting a breakpoint opcode, 0CCh, onto the stack before RET in order to prevent the execution of malicious code that has been injected onto the stack.

Insert 0CCh Opcode onto the Stack

The compiler will generate the Call VoidFunction() code as:

```
Push    0       ; just for holding the breakpoint opcode
Call    VoidFunction
Add     esp, 4  ; can be pop ecx etc.
```

The called function, VoidFunction, is coded as:

```
Add     byte ptr [esp+4], 0cch
ret
```

The compiler will generate the Call DllApiFunction(par1, par2 ..) code as:

```
Push    0 ; extra place for 0CCh the breakpoint opcode
Push    ..
Push    par2
Push    par1
Call    DllApiFunction     ; PASCAL-style or WINAPI
Add     esp, 4             ; can be pop ecx, etc
```

The called function, DllApiFunction, is coded as:

```
Mov     byte ptr [esp+4+X << 2], 0cch
; where x is the parameter number
Ret     X << 2
```

The Call ParameterCFunction(par1, par2 ..) code will be generated as follows:

```
Push    ..
Push    par2
Push    par1
Call    ParameterCFunction
Add     esp, X << 2
           ; where X is the parameter number
```

The called function, ParameterCFunction, is coded as:

```
Mov     byte ptr [esp+4], 0CCh
Ret
```

Detailed Explanation

Figure 1 illustrates the execution path during a normal function call.

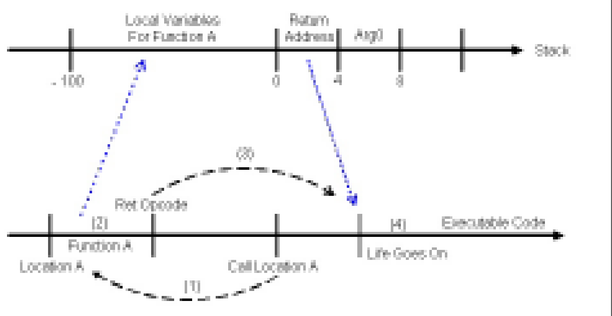


Figure 1. A normal function call and stack usage execution path: (1) → (2) → (3) → (4).

However, if the stack is overflowed due to the exploitation of software bugs, then the mechanism will generate a breakpoint trap and the execution of malicious code is stopped. Figure 2 illustrates the insertion of 0CCh opcode for a function without affecting the normal execution of the program.

Without the protection mechanism proposed in this article, the stack buffer overflow will be exploited (as happened with Slammer) to run a piece of malicious code. Figure 3 illustrates the execution path, where step (3) is hijacked into steps (4), (5), and (6) in order to execute the malicious code.

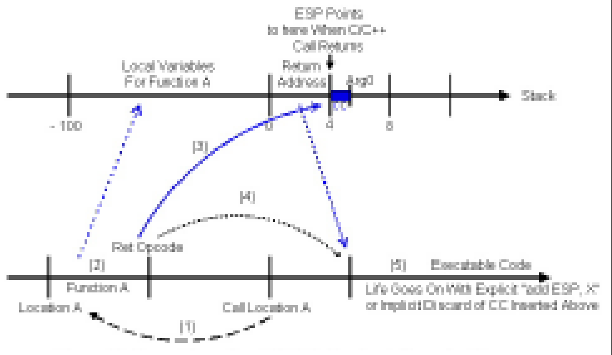

ESP Points to here When C/C++ Call Returns
Local Variables For Function A
Return Address
Arg0
Stack
-100
0
4
8
CC
(3)
(4)
Ret Opcode
(2)
(5)
Executable Code
Function A
Location A
Call Location A
(1)
Life Goes On With Explicit "add ESP, X" or Implicit Discard of CC Inserted Above

Figure 2. A normal function call with breakpoint inserted to safeguard against stack overflow exploitation. Execution path: (1) → (2) → (3) → (4) → (5).

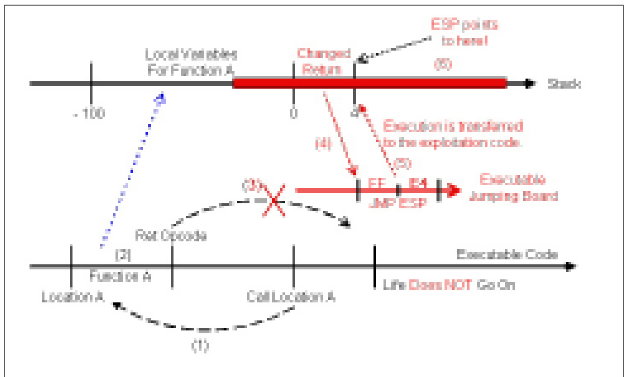


Figure 3. A function call with the stack overflowed and exploited. Execution path: (1) → (2) → (3) → (4) → (5) → (6).

With the insertion of opcode 0CCh onto the stack, the stack overflow still occurs, as illustrated in Figure 4. However, the execution of the malicious code using the ‘call ESP’ or ‘jmp ESP’ method is stopped cold with a trap generated.

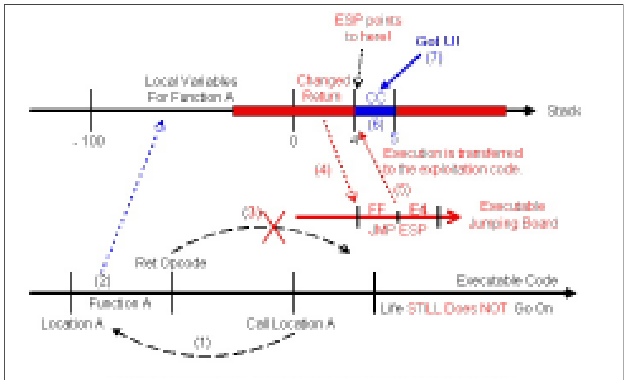


Figure 4. A function call with the stack overflowed and the intended exploitation stopped with 0CCh breakpoint opcode. Execution path: (1) → (2) → (3) → (4) → (5) → (6) → (7).

Performance and Compatibility

This method works in calling old, existing libraries because it wastes a few cycles with ‘push 0’ and ‘add esp, 4’. However, the 0CCh-inserted libraries or object files will not work properly with old call methods (without extra push and pop).

Optimization of the above three functions can be achieved to reduce the extra push and pop needed for holding the breakpoint code. This method will increase the executable image size marginally, and cause minimal runtime degradation of performance.

Some might argue that this method creates a false sense of security. It does. However, it dams the overflow and prevents the malicious code from spreading.

There are three options for the compiler:

1. Use the ‘old’ method, allowing some bugs to be exploited fully.
2. Add breakpoint code 0CCh only to functions with parameters (a risk still exists for the void parameter functions).
3. Add breakpoint code check for void parameter functions, Pascal-style API functions and parameter functions to allow maximum safety check (a compatibility issue arises).

Q&A

Does this mechanism have a prohibitive impact on the execution of a normal program?

The mechanism does not have a prohibitive impact on the execution of a normal program. The breakpoint opcode 0CCh, inserted onto the stack and discarded when a normal function returns, does not usually have any opportunity to be executed until it is triggered by an overflow exploitation using the ‘jmp/call ESP’ scheme.

Does this mechanism prevent an overflow from happening?

If there is an overflow bug which someone intends to flood, this mechanism cannot do anything to prevent it. However, the mechanism puts a dam across the flooded stack so that a ‘jmp/call ESP’ scheme would be unable to execute the malicious code that has been injected.

Does this mechanism prevent an overflow with an exploitation using ‘jmp/call ESP’?

Yes, it does.

Does this mechanism prevent an overflow with an exploitation such as ‘add esp, 4;call/jmp esp’?

No, it does not. However, the chance for the opcode bytes like ‘add esp, 4; call/jmp esp’ is very minimal, if any at all, in an executable section that has been generated normally – unless it is left there intentionally to act as a jumping board. However, the chance for ‘call/jmp esp’ (FF D4/E4 two bytes) in the executable section is much greater.

What is the effect on performance?

There is a very minor performance degradation on a program if this mechanism is used.

What happens when the exploitation returns as shown in Figure 5?

This exploitation first jumps to somewhere with opcodes like ‘add esp 4;ret’. With this new RET, it will transfer to the real ‘jmp/call ESP’. Obviously, all the function calls need to be implemented with a ‘move [esp+4+x], 0CCh’ mechanism. Then, the chance for these opcode bytes

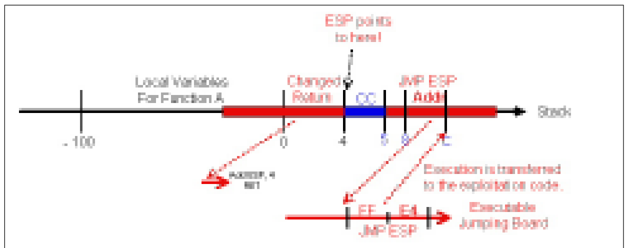


Figure 5. An exploitation with ‘add ESP, 4;RET’ then ‘JMP ESP’.

becomes smaller. The chance for ‘add esp 4;ret’ opcode or byte strings is much smaller than ‘jmp/call ESP’ opcode.2

What happens when the exploitation returns as shown in Figure 6?

Here, the exploitation first jumps to somewhere with opcodes like ‘ret 4’ and utilizing the address ‘XX XX XX CC’ holds a ‘jmp/call ESP’. The ‘mov byte [ESP+X], 0CCh’ can be modified to ‘mov dword [ESP+X], 0CCh’.

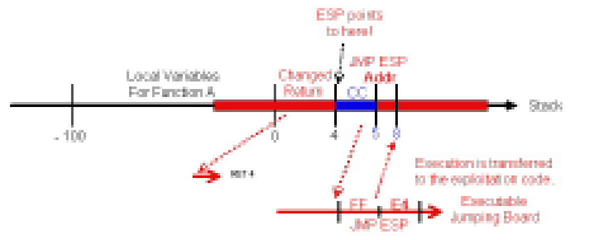


Figure 6. An exploitation with ‘RET 4’.

What happens when the exploitation returns as shown in Figure 7?

The exploitation shown in Figure 7 utilizes the fact that EBX points to an address pointing to somewhere in the flooded stack, and this mechanism will not work. The 0CCh breakpoint is jumped over. It is likely to be difficult to find a case like this, since EBX/ESI/EDI registers might be saved onto the stack during the function call and be populated with some invalid values from the flooded stack itself before RET. But the chance, although small, does exist.

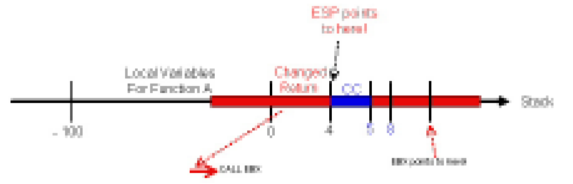


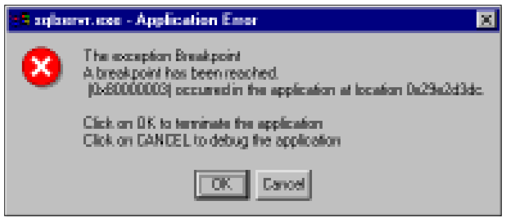
Figure 7. An exploitation with ‘call EBX’.

Is this mechanism ‘bulletproof’?

Unfortunately not.

Can the method be applied to prevent Slammer?

The figure shown below is the result of the runtime test (manual insertion of 0CCh opcode).



```
sqlservr.exe - Application Error

The exception Breakpoint
A breakpoint has been reached.
(0x80000003) occurred in the application at location 0x29a2d3dc.

Click on OK to terminate the application
Click on CANCEL to debug the application
```

Source Code

Note that the only difference between the testOne and testTwo (implemented in assembly) functions is that there is one more line ‘mov [esp+4], 0cch’ for testTwo. These need to be reversed to see the difference.

Source Code For Windows

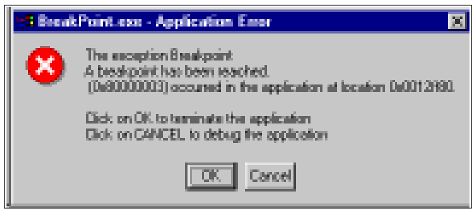
- To generate an exception when an overflowed buffer tries to exploit, run ‘breakpoint x’.
- To allow the exploitation to happen (call the old C function ret method), run ‘breakpoint’.
- If you try to rebuild and run, notice that you cannot run it under debug mode, since exploitation code addresses are hard-coded for release version.

Here is the runtime result I got:

```
H:\>H:\BreakPoint\Release\BreakPoint.exe
Hello World! testOne
Hello World! testTwo
Hello World! done
ABCDEFGHABCDEFGHABCDEFGHABCDEFGHá?@?+ ?@?h¡¦¦+ -
You overflowed to here bffedead

H:\>H:\BreakPoint\Release\BreakPoint.exe 1
Hello World! testOne
Hello World! testTwo
Hello World! done
ABCDEFGHABCDEFGHABCDEFGHABCDEFGHá?@?+ ?@?h¡¦¦+ -
```

The figure below shows the runtime screen.



```
BreakPoint.exe - Application Error

The exception Breakpoint
A breakpoint has been reached.
(0x80000003) occurred in the application at location 0x0012ff80.

Click on OK to terminate the application
Click on CANCEL to debug the application
```

Source Code For Linux

The following is the compiling process as well as the runtime core dump triggered when the exploitation uses the ‘jmp esp’ scheme.

```
[huangy@ph2 pack]$ gcc -c -o breakpoint.o
breakpoint.c
[huangy@ph2 pack]$ nasm -f elf hello.asm
[huangy@ph2 pack]$ gcc -o breakpoint.bin breakpoint.o
hello.o
[huangy@ph2 pack]$ ./breakpoint.bin
Hello World! testOne
Hello World! testTwo
Hello World! done
ABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDABCDABCD~@¸èh-
Þþ¿ÿÐ
You overflowed to here bffedead

[huangy@ph2 pack]$ ./breakpoint.bin 1
Hello World! testOne
Hello World! testTwo
Hello World! done
ABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDABCDABCD~@¸èh-
Þþ¿ÿÐ
Segmentation fault (core dumped)
```

Conclusion

The insertion of breakpoint opcode 0CCh onto the stack before RET on x86 PC is an effective method to protect against the stack-based overflow exploitation as seen in Slammer. This method can be applied to *Windows*, *Linux* and Solaris on x86.

SPOTLIGHT

Policing the Digital Frontier

Nina Gaubert
National Hi-Tech Crime Unit, UK

The National Hi-Tech Crime Unit (NHTCU) is the UK's first national law enforcement organisation dedicated to tackling the new and emerging threat of hi-tech crime, or 'cyber crime'.

The remit of the Unit is combating national and transnational serious and organised hi-tech crime within, or which impacts upon, the United Kingdom.

The Unit was established as part of the UK's National Hi-Tech Crime Strategy announced to Parliament in November 2000, and received £25 million of Government funding, over a three-year period.

Of this funding, £15 million was used to set up the national centre of excellence, the National Hi-Tech Crime Unit, while the remaining £10 million has been allocated to the 43 local police forces of England and Wales, to enable them to create or expand their own computer crime units and bring them up to a recognised benchmark standard.

The NHTCU

The NHTCU was launched in April 2001, and is the UK's first pro-active, multi-agency law enforcement body. The unit consists of police officers from the National Crime Squad (NCS), National Criminal Intelligence Service (NCIS), HM Customs and Excise (HMC&E) and the Ministry of Defence (MoD).

The role of the NHTCU is defined as:

- To support or lead activity against serious and organised hi-tech crime of a national and transnational nature.
- To respond, with an investigative capability, to all threats to and attacks upon the UK Critical National Infrastructure.
- To undertake strategic threat assessments.
- To develop intelligence.
- To support and co-ordinate law enforcement operations.
- To offer 'best advice' to other law enforcement agencies, and to businesses, industry and the IT world.

Within the unit there are four teams: the intelligence section, operations section, forensics section, and the technical and tactical support section.

Intelligence Section

The Intelligence Section of the NHTCU is responsible for providing:

- Strategic intelligence on hi-tech criminality.
- Tactical intelligence development, identifying new hi-tech criminality and targets for investigation.
- Intelligence support to the Unit's operations section.

The Intelligence section provides leadership in the gathering of intelligence and the use of data-mining tools. The NHTCU has developed and implemented a tactical hi-tech crime intelligence database and a confidential source register that allows the identity of sources of information to be protected.

A number of tactical intelligence development operations have been undertaken and are on-going, including work to combat on-line child abuse, hackers and virus writers, hi-tech criminality supporting drugs trafficking and on-line fraud.

Operations Section

The operations section of the NHTCU either leads or supports others in investigations, depending on the severity of the crime, the level of organisation displayed by the targets, the geographical impact of their activity, and – most importantly – the technical complexity involved in investigating the offences.

Since October 2001, the NHTCU has been involved in over 30 pro-active operations, resulting in over 70 arrests. Investigations have been over a broad spectrum of computer related crime including:

Intellectual property theft/software piracy	10%
Hacking, virus writing and DoS attacks	25%
Organised on-line paedophilia	25%
Denial of Service attacks	5%
Extortion	15%
Fraud	15%
Human trafficking	5%

Digital Evidence Recovery

The Unit has a dedicated forensic section, which works in parallel with the other teams in the Unit by examining and producing evidence from computers that have been seized. The NHTCU's digital forensic section also supports the forensic officers attached to the regional police forces across the UK.

Tactical and Technical Support Section (TTS)

The TTS section is the 'shop window' of the Unit and is the interface with law enforcement locally. It provides the focal point for all international requests for assistance within this field. In addition, the TTS section has set up a telephone help line to enable members of the public and industry to report computer crime.

The NHTCU has appointed a dedicated industry liaison officer with sole responsibility for acting as an interface between the NHTCU, business and industry. The success of the Unit depends to a greater or lesser extent on the relationship built between both.

Another recent appointment is that of a crime prevention officer, who currently is trying to identify how best to get the relevant information across to the general public – particularly information regarding the use of firewalls and anti-virus software.

Assistance

So what was the motivation for writing an article in *Virus Bulletin*? In order to contain the level of malicious code authoring and distributing activity, so that the investigation of such crimes remains at 25% of the Unit's overall workload, the Unit needs good intelligence.

This is where the anti-virus industry can assist, by identifying the source of viruses, and/or discussions within relevant newsgroups can assist the Unit in detecting these offenders. On a regular basis the Unit receives intelligence which is not within the UK jurisdiction; this has allowed the Unit to operate closely with 25 international partners.

As governments around the world begin to target hi-tech crime, many of their own crime units are using the NHTCU as a template for setting up dedicated hi-tech law enforcement agencies.

In addition, the European Union has been drafting a protocol whereby all member states will have similar legislation regarding computer crime. Hopefully, this will encourage other non-European countries to follow their lead. [*In March 2003, news agency Reuters reported that new laws approved by European Union justice ministers will mean that, in Europe, virus writers could be imprisoned for up to five years. According to Reuters: 'Hacking and spreading viruses, when committed by organized criminals, will be punished with jail terms of no less than two years – and up to five years – under the new law.' - Ed*]

The Future

The evolution of e-commerce, the growth of multinational companies and the ease and speed with which information can be passed around the world, will only ensure that those behind organised crime will turn their hand to this new, lucrative market.

Charter

A legacy of law enforcement has identified that commercial organisations tend to be reluctant to report crime – particularly that in the hi-tech arena – for fear of adverse publicity. In an attempt to counter this reluctance, the NHTCU has launched a 'Confidential Charter'.

The Charter, which was drawn up with input from the Confederation of British Industry and the Crown Prosecution Service, is designed to help the business community understand how they can interact with the NHTCU in a secure, efficient and confidential manner when there is a need to exchange information.

The accurate and timely reporting of hi-tech crimes and intelligence is of paramount importance. Success against serious and organised hi-tech criminals can only be achieved through a strategy of collaboration and co-operation between the business community and the NHTCU.

The Unit provides reassurance that businesses can report suspicious hi-tech activities and attacks without fear of causing unwelcome interruptions to their business or unfavourable publicity.

Successful Operations

Last year saw the widely reported prosecution, by the Metropolitan Police (one of the UK's regional police forces), of virus writer Simon Vallor, a 22-year-old Welshman who wrote and distributed three viruses – Gokar, Redesi and Admirer – between November 2001 and February 2002.

It was estimated that, between them, the three viruses infected approximately 27,000 machines worldwide and were geographically spread across 42 different countries. It was as a result of the co-operation between the law enforcement agency and the anti-virus industry that a successful prosecution was able to be made. Simon Vallor was sentenced, under the UK's Computer Misuse Act, to two years imprisonment.

Continued Support

The number of viruses released this year indicates that the penalty applied in the case of Simon Vallor has had no deterrent effect upon virus writers.

The NHTCU and worldwide law enforcement requires the continued co-operation both of the anti-virus industry and businesses, in order to identify these offenders.

The National Hi-Tech Crime Unit
Tel: +44 (0)870 2410549
Fax: +44 (0)870 241 5729
Email: admin@nhtcu.org
Web: http://www.nhtcu.org/

OPINION 1

Waiting in the Wings: Linux Viruses

Phil d’Espace
Independent Researcher, UK

In a previous article (see *VB*, September 2002, p.16) I made the point that *Linux* is not impenetrable to malware. In this follow-up, I aim to make that case a little more strongly, and point out that *Linux* is, in fact, *more* susceptible to viruses/worms than *Windows*. I imagine that this will be a highly unpopular opinion, especially among the ‘four legs good, two legs bad’ crowd (see previous article), but I believe it to be the case, and shall attempt to convince you of the same.

User Traits

Believing your system to be impenetrable to any kind of black-hat activity is a good way to get hacked. Likewise, the blind belief that *Linux* viruses represent a toothless threat is a good way to get one. The majority of virus infections happen as a result of the same things, regardless of operating system: user stupidity, user ignorance and user laziness.

Failure to apply patches is, and probably always will be, one of the main contributing factors to the successful spread of viruses. And an unpatched *Linux* machine is far more of a security concern than an unpatched *Windows* machine: unlike machines running *Linux*, *Windows* machines don’t tend to come with remote administration tools (such as ssh) installed and running by default.

In the same way, the type of user who sets their password to ‘password’ is far more of a liability at the helm of a *Linux* machine than an equivalent *Windows* machine. Several existing *Windows* viruses guess passwords to gain access to various services – a trick that could, fairly easily, be ‘ported’ to *Linux* viruses.

The users who do not understand why they should never click on unsolicited email attachments in a *Windows* environment – and who, therefore, rarely take heed of safe-computing guidelines – are unlikely to become security-aware über-users simply because you’ve given them a *Mandrake* box to work on.

Conclusion: a *Linux* machine is much less forgiving of user stupidity when it comes to security.

The Upgrade Path is Not Always Easy

The ease of upgrading on a *Linux* machine is distribution-dependent, but is rarely straightforward. Patching a *Windows* machine is often as simple as browsing to http://windowsupdate.microsoft.com/, allowing the machine to do some black magic, and then rebooting.

If you are very lucky, patching *Linux* software can resemble such an experience (*Debian* makes this task the easiest). If you’re unlucky, however, you will find yourself stuck in a mire of dependencies which need to be fulfilled, battling source code that needs to be told the locations of obscure libraries before it will compile, and so on. This sucks.

And that’s just the standard software. Installing a new kernel is no walk in the park. It has, I believe, become easier since I switched to BSD variants a couple of years ago, but upgrading the *Linux* kernel – for example to patch local root exploits – is not a job for the faint-hearted. For one, it can take a long time, especially on slow machines. The kernel may simply refuse to build, leaving you with little useful information (if you’re not a programmer) as to why. And so on, and on, and on.

Patching is hard work. Some people do it, some people don’t, but as more users move to *Linux*, the more unpatched *Linux* boxes will be run, and an absence of patches leads, inevitably, to virus infections.

Exploits are Easier to Engineer

Being able to see the source code of applications can make it a lot easier to engineer exploits for them. I shall use the example of a particular buffer overflow for which people didn’t really stop to consider the ‘virus-potential’, and describe what I believe to be a fairly viable virus that *could* have been written to take advantage of it.

In January 2003, *Gobbles Security* issued a security advisory involving the RIAA, P2P, and a large dose of fiction. But the advisory also referred to an mpg123 buffer overflow that allowed an attacker to run arbitrary code on the victim’s computer by crafting a special MP3, and persuading the victim to play it.

If that arbitrary code were to grep the user’s home directory for email addresses, and then send the MP3 to every address it found, along with some banal message such as ‘Listen to this funny mp3!?!?!’, how many machines do you think it would infect? Few people imagine they will be hit by a *Linux* virus, few people exercise caution before listening to MP3s, and on many *Linux* systems, a user’s mailbox will be stored in their home directory, making it a trivial exercise to harvest email addresses.

It’s true that there are buffer overflows in *Windows* applications, but *Windows* applications tend to be closed-source, and overflows are a lot easier to find and exploit when you can see the source code.

Linux 'Security Model' Myth

There are a few advantages to running *Linux* when it comes to keeping viruses at bay. But first, there is a myth to be debunked.

A seemingly favourite mantra of *Linux* zealots is that, because of the *Linux* security model, a virus can't actually infect the system – by which they actually mean a virus can't infect the *system executables*. This is a straw man – gone (mostly) are the days when viruses were spread by people sharing executable files on floppy disks; today's vectors are email, *Word* documents, network shares, and server applications (such as *SQL Server* or *IIS*). None of these require the virus to infect system executables in order to be effective. Email clients with vulnerabilities exist on *Linux* already, as do buggy server applications and services – the CrossOver Plugin allows *MS Office* to be run on *Linux* too.

More Open Service by Default

A standard *Windows 98* fresh install isn't very exciting if you port-scan it. However, a standard *Mandrake* or *RedHat* of the same vintage could have all kinds of interesting ports open. Take, for example, the notoriously buggy sendmail, sshd (with the remote root exploit discovered not all that long ago), or *Apache* (of Slapper fame, see *VB*, November 2002, p.7) – admins who are too lazy to patch are almost certainly too lazy to turn off services that overzealous *Linux* distributions happily install and switch on by default.

Conclusion

There are two main obstacles keeping viruses away from *Linux* at the moment. One will change, one will probably stay the same.

The first is a small user base: for virus writers, there's very little 'bang to the buck' to be gained from writing a *Linux* virus – exploitable hosts are, at the moment, few and far between compared to the almost ubiquitous *Windows* boxes. This will probably change, especially if *Microsoft* continues to pursue funky licensing schemes.

Secondly, there is a lot of software diversity on *Linux* machines. A virus that exploits a *Linux* MUA is unlikely to spread very well – since, not only does it need to find other *Linux* users, but it needs to find users with the same MUA. Where *Windows* is concerned, everyone (well, almost everyone) uses *Outlook* for mail, *IE* for web-browsing, *IIS* for a web-server, and *SQLServer* as a database. No surprise, then, that most recent *Windows* viruses have had one or more of those four programs at their core.

Can the dearth of viruses on *Linux* last? No. There is no magic keeping viruses away from *Linux* machines, no 'inherent security' that actually counts for anything. I believe that *Linux* is, in fact, more susceptible to viruses than *Windows*, and as we see a growing *Linux* user base, this will become increasingly apparent.

OPINION 2

Anti-Virus Left Out in the Open

Peter Sergeant

In general computing, open-source software has made huge inroads in displacing proprietary software: the *Linux/Apache* combination is arguably a superior web server to any proprietary system, and is free; BIND is the most popular DNS server implementation.

Despite the success of open-source software in other areas, however, only a limited range of open-source anti-virus software exists, and it seems unlikely that open-source anti-virus software will ever 'make it big'.

Advantages

First, let's take a look at all the potential advantages of open-source anti-virus.

No politics

At VB2002 Graham Cluley highlighted the numerous sticky questions, so-called 'e-Bugs' raised for anti-virus vendors, and we know about the problems encountered by anti-virus companies that have attempted to add detection of porn dialers and the like into their products (see *VB*, December 2002, p.12).

Rainer Link, of the Open Anti-Virus project, says the fact that he and his colleagues are neither tied to any business plans, nor bound by any political restrictions frees them up to do pretty much what they want as far as these two thorny issues go.

How true this is, is not really clear, but one presumes it would prove a lot more difficult for a company that produces porn dialers to bring legal action against an open-source project, than it is for them to sue an anti-virus company with a subsidiary in their own country.

Bugs are shallow

The nature of open-source software is such that many eyes can see the source-code of the applications – and with many eyes, the reasoning goes, all bugs are shallow. That is, the greater the number of people giving your code peer-review, and submitting patches, the better it will become, the more secure it will be.

Furthermore, if the public has access to your CVS tree, then they don't need to wait six months before you release a new version with a feature they want in it – they can grab it from the CVS tree, compile it, test it, and submit a bug report if it's not working properly.

Customers (rightly) expect a lot more from commercial anti-virus vendors. The 'release early, release often' philosophy is difficult to implement when sending out physical CDs, updating printed manuals, retraining technical support staff, and going through several layers of quality assurance.

Existing codebase

If code is released under the GPL (General Public Licence), programmers can make use of existing GPL libraries, and can 'borrow' any GPL-licensed code they want, as and when it is needed.

However, producers of commercial anti-virus software will not be keen to use GPL-licensed code – were they to use GPL-licensed code in their applications they would be obliged to provide the source to customers who requested it.

It's free

The fact that open-source software can be obtained free of charge will encourage many people to try it out. Furthermore, if it reaches a useful state, OS vendors could start to bundle it, thus increasing its user-base enormously, and seeing an increase in the number of bug reports and patches as a consequence.

Commercial anti-virus manufacturers have OEM sales, but this comes at a cost to OS and hardware vendors – if these vendors were able to offer their customers virus protection at an almost-zero cost to themselves, it seems likely that they would take that opportunity.

Disadvantages

Sadly, despite the apparent advantages to open-source anti-virus software, it's not all plain sailing. There are a number of considerable obstacles faced by open-source anti-virus developers.

Non-disclosure agreements

Commercial anti-virus companies often need to sign non-disclosure agreements (NDA) with companies such as *Microsoft* in order to gain access to information that will make their products more effective.

Since by its very nature, an open-source product would most likely give away this information through the code itself, an open-source anti-virus developer is likely to encounter a great deal of trouble getting their hands on the relevant information.

Microsoft's new *Palladium* system (see *VB*, September 2002, p.15) could make this situation worse, certainly where *Microsoft Windows* platforms are concerned. If, as proposed, all software will need to be *Microsoft*-approved, and *Microsoft* approval costs big money, open-source projects are unlikely to be able to run on the *Windows* systems of the future.

Virus samples

Traditionally, anti-virus companies exchange virus samples with each other, but they are wary of sending samples to those outside the 'inner-circle', as it were. Therefore, it is likely that those developing open-source anti-virus products will have trouble procuring virus samples to develop detection against.

Interestingly, Rainer Link, of the Open Anti-Virus project, says this has not proved a problem so far (see *VB*, September 2002, p.6), but notes that at the present time he is devoting more energies towards stabilising the scanning engine than to acquiring a large collection of samples. He may find he encounters a lot more trouble as his search intensifies.

Commercial vendors got there first

A lot of open-source software has been born from a need to solve a particular problem, where problems are related to performing functions on an open-source operating system. Numerous anti-virus vendors offer *nix products already, and some offer well-documented libraries. Therefore, people can solve the problems using proprietary software, and there is no pressing need to develop open-source solutions.

Someone wanting a custom *Linux* mailscanner does not need to re-implement a scanning engine – they can simply use a commercial scanning engine, and wrap their code around that. There are already several fairly popular open-source packages that, essentially, wrap proprietary anti-virus libraries.

Keeping up to date

Most large anti-virus vendors now have more than one virus research lab situated worldwide, so that if a virus hits in a big way when their main development team is asleep, they can still respond quickly to the threat and keep their customers protected. The same luxury is not available for open-source developers, who as well as being hobbyists, will find it hard to get samples and add detection quickly enough.

Conclusion

The anti-virus industry has made it difficult for non-commercial competitors to enter the industry, with a number of considerable barriers to entry: customer trust, collection of virus samples, the ability of existing products to solve most problems.

Of these, sample collection and pre-existing software that does the job well seem to be the biggest barriers for open-source anti-virus software, as well as the lack of access to information available only under NDA.

In time, we will see if and how open-source anti-virus copes with these problems. However, I'm unconvinced it will take off.

PRODUCT REVIEW

eTrust Antivirus

Matt Ham

Computer Associates (*CA*) has a history of using a wide range of engines in its products. In its current form, *eTrust Antivirus* is based primarily upon the *iRis* engine, *iRis Software* having been purchased a number of years ago by *CA* specifically for such purpose. Anti-virus historians will also recall *CA*'s purchase, in 1999, of *Cybec*, the company from which *Vet Anti-Virus* originated.

Although *Vet* exists as a product in its own right, the engine can be used within *eTrust* as an alternative to the default engine. This is useful in those situations where two engines are desirable within an organisation, without the hassle of having to deal with more than one vendor – although the same feature is available in products that use multiple engines sourced from different companies, *eTrust* is, to my knowledge, the only example in which the engines are products of the same company.

Previous reviews of *eTrust* (formerly under the name *InoculateIT*, see *VB*, June 2001, p.17) have, by and large, focused on its scanning capabilities and the options in the desktop portion of the product. In this review the net will be cast a little wider.

Making up part of the *eTrust* suite are a number of tools intended to make large-scale use of the product as efficient and simple as possible. Also contained in the package are scanning solutions for platforms and *Groupware* products other than the standard desktop. It is some of these which will be inspected in more detail here. Even within these solutions a variety of options are available, to such an extent that those reviewed do not cover the entire range.

The Package

The package supplied consists of a box containing two printed manuals and two CDs. Each of the manuals relates specifically to one of the CDs – presumably a more efficient system for distribution than having a monolithic manual or different manuals for each standard combination of product options. The CDs covered 'Desktops, Servers, PDAs and Groupware' and 'Gateway and Perimeter Devices'.

Falling into the category 'Desktops, Servers, PDAs and Groupware' are those familiar scanners which have appeared in *VB*'s comparative and standalone reviews in the past. The CD autoruns to provide access to its contents. The interface is by necessity multi-layered, though it can prove difficult to determine exactly what is on the CD. For example, selecting 'Install Products' brings up a list of products which can either be installed, or for which installation instructions can be read (when running on a *Windows* machine; the installation option is not available for the UNIX/*Linux* and Macintosh products). Each of the installation options leads to a separate application, though the descriptions of these are nebulous in some cases.

The products on this disk are the *Windows Client* and *Server* versions of *eTrust*, together with *Microsoft Exchange* and *Lotus Notes Domino* scanning options. In addition, remote administration tools are included for instances of the scanner installed in a *NetWare* domain and for the *Windows* server product.

An alert manager is provided, which includes the option to interface with *CA*'s *Unicenter* product line. Further support for a networked environment is provided by the Remote Install Wizard, which offers remote installation to a variety of *Microsoft* platforms. A separate application offers the same functionality for *NetWare* versions 3 and upwards. PDA support is currently limited to *Pocket PC* and *Palm OS* (the *Palm* product is not provided on the CD, but can be accessed via a link to the *CA* website).

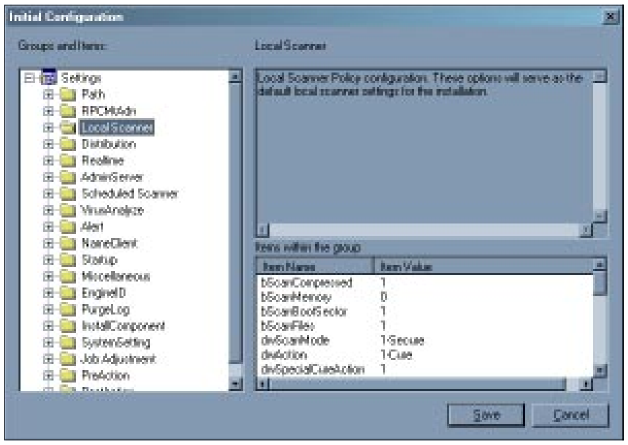
Documentation on the CD is supplied in PDF format, along with the *Acrobat Reader* software. There is also a selection of links leading to documentation on the *CA* website, although these pages are more reminiscent of sales and marketing literature than hands-on guides.

The second disk is devoted to products which are becoming of increasing importance in larger organisations. 'Management Components' is a general term for a collection of three applications: Control Center, Policy Manager and Audit Viewer.

In terms of scanning products rather than tools, there are plug-in detection capabilities for *Check Point FireWall-1*, *Microsoft ISA Server* and *Microsoft Proxy Server*. For installations where gateway protection is required without one of these programs, there is a standalone Gateway Inspection Engine (GIE). Each of the products requires the Control Center to be installed before it can be used.

There are a number of other requirements for the installation of these applications, with DAO (Data Access Objects) a necessary install for the Management Components. This, along with numerous obligatory patches, service packs and the like, is included on the second CD. Ample instruction is provided as to the order in which these should be applied – useful information given the counter-intuitive ways in which service packs and drivers can interact under *Windows.*

The installation procedure gives some information as to the hardware and software requirements, and further details can be found in the documentation. Hard drive requirements varied between 3 and 500 MB, for the Firewall-1 and GIE components, respectively.



Settings
- Path
- RPCMtAdn
- Local Scanner
- Distribution
- Realtime
- AdminServer
- Scheduled Scanner
- VirusAnalyze
- Alert
- NameClient
- Startup
- Miscellaneous
- EngineID
- PurgeLog
- InstallComponent
- SystemSetting
- Job Adjustment
- PreAction

Local Scanner Policy configuration. These options will serve as the default local scanner settings for the installation.

Item Name	Item Value
bScanCompressed	1
bScanMemory	0
bScanBootSector	1
bScanFiles	1
dwScanMode	1-Secure
dwAction	1-Cure
dwSpecialCureAction	1

Web Support and Documentation

The *CA* website can be found at http://www.ca.com/. Specific sections of the site are devoted to virus information and to product information. The content is both useful and well laid out. Of particular note is the extensive listing of aliases for viruses – helpful when searching for information about a threat which may have been named differently by other companies.

As mentioned, documentation is provided in three formats: hard copy manuals, PDF manuals and files on the *CA* website. In addition there are numerous small readme files, mostly in html format, located both on the CDs and associated with installed files.

The documentation proved clear and relevant, with screen captures conveying information well. Help files within the programs were particularly useful, despite context-sensitive help not being available in all circumstances.

Installation and Update

As far as the installation of individual components was concerned, there were no peculiarities in the procedure. Rather than concentrate on these, remote installation was examined in more detail.

This is a separate package, which specifies that it can perform remote installation to *Windows NT*, *2K*, *XP* and .*Net* machines of both the server and client variety. For testing purposes a rather odd network was used, containing both a domain and a workgroup and including a *Linux* machine with a *Samba*-accessible hard drive. (Reasons for such a setup had as much to do with the machines being prepared for the forthcoming *Linux* comparative review as any decision to challenge the capabilities of the installer.)

As expected, a view of the network dominates the GUI and all machines on the test network were identified correctly. Machines may be selected from this view for installation, using a standard account. This may be altered on an individual basis if necessary. Once selections have been made they may be saved for later use so, although the initial setup may be laborious if there is no administration account that is universally applicable for target machines, it should prove trivial to use the same information on subsequent occasions. No problem was caused by the presence of a *Linux* machine on the test network – its presence was detected, but the option to select the machine for installation was not presented.

The files involved in installation are the program files, licence files and a pre-installation executable. By default these are stored locally, though the location can be changed as required.

The specifics of the installation procedure are controlled by an ICF file, which at the most basic level oversees the decisions that need to be made during installation, allowing the remote installation to be performed silently. However, more control is extended than is available from the standard installation options. This file may be hand-edited, as it is a text file, but thankfully there is a custom editing interface within the remote installation utility.

The installation process was not quite invisible – a DOS-style box appeared on the target machine screen, although no information appeared in it. Progress was reported on the source machine's remote installation machine. The on-access scanner was not activated immediately by default – this occurred only on a reboot.

As a simple test an update was initiated on a machine and interrupted by means of a power down on the target machine when the install was only 40% complete. Rather ominously, the installer proclaimed after a short delay that it was deleting temporary files, but did not mention any problems with the installation. This announcement was replaced after a short period by a notice that the helper application was being removed and only after that was it revealed that a problem had occurred and a retry was advised. With power restored, a retry resulted in no untoward occurrences.

A new install was attempted – this time while the machine was waiting for a user logon. Although there were fewer status reports than on a machine that was already logged on, the process continued smoothly to completion. Installing at this point also resulted in immediate installation of the on-access scanner when a user logged on.

Updates are performed from within the Server Administration view of the on-demand scanning interface. It should be noted that there are a large number of network administration and policy control features within this package, although these are not explored in this review. The Administrator offers control of scanning, updates and policies over networks – everything but the initial distribution to machines seems to be covered here.

By default, updates are downloaded from the *CA* servers and distributed from the administrative machine. This can be configured so that local file servers replace the *CA* servers – useful in networks where web access is restricted for enhanced security.

Features: Exchange Scanner

The *Exchange* scanning support offered by *eTrust* differs from others for that platform in one major respect. It is not controlled as a *Microsoft Management Console* snap-in, but rather as a separate application available through the *eTrust* tray icon. This is used mainly to control the on-access file scanner, but has extra functionality too.

When the Mail Options dialog is invoked another tabbed dialog box appears, through which the *Exchange* options can be configured. If activated, the *Domino* scanning options are also regulated here, with slightly different options available. The tabs presented in this dialog are Scan, Selection, Options and Misc.

Rather confusingly, the Scan tab opens with a check box which determines whether scanning overall is on or off. However, the check box is labelled 'Incoming and Outgoing Messages', and a small note below this gives the scanner status – activated, deactivated or in the process of running. At first the note led me to believe that overall control of scanning was selected elsewhere, since the information seemed a little redundant if referring to the check-box directly adjacent to it. The on/off switch for scanning is followed by a choice of scanning engine, and the option to apply heuristics to any scanning. As expected, the engine choices are the *InoculateIT* and *Vet* engines.

Further along the Scan tab is the choice of Reviewer or Secure scanning mode. This amused me as it seemed to be a reflection of both customer psychology and the number of reviews performed simply to provide high detection results and a healthy advertising revenue.

Reviewer mode is described in the documentation as having, in so many words, problems with false positives, but, presumably, it detects in the sort of paranoid fashion that is suitable for bumping up scanning detection rates. Labelling the paranoid setting as 'Reviewer' should scare off real-world users, while labelling the standard configuration 'Secure' does, admittedly, sound better than the strictly more accurate 'sensible compromise'.

The final selection on the Scan tab is as to what action to take when an infection is detected. This is selected from Report only (the default), Delete File, Rename File, Move File and Cure File. Only the Cure File option has a further refinement in its activity, with the default actions in the case of non-disinfectable files being one option configurable. Whether files should be backed up before disinfection, worms simply be cured by deletion and macros disinfected or removed are also choices here.

The second tab is Selection, which is concerned primarily with which files should be scanned. By default all extensions are scanned, whether compressed or not.

Scanning is set to stop with the detection of one infection within an archive and, in order to improve scanning speed, compression type is determined by the extension of the file under consideration. This last feature could be considered something of a weakness should users be sufficiently aware that they change extensions on files before transferring them into or out of an organisation. It is also possible to set a list of those extensions which will simply be blocked as a matter of course, with no scan applied.

It will have been noted that the options mentioned already refer to files and extensions and are thus applicable to attachments more than to the contents of the message body. The Options tab moves away from this attachment-centric view.

By default, message body scanning is activated, though it may be deactivated here. Proactive scanning is another feature which may be toggled here, this being the system by which scanning occurs before direct calls for access to a mail object are issued by a mail client or server. This is activated by default.

The final tab is designated 'Misc'. Here options are concerned with log file sizes, types and the level of information contained within them. Although the documentation recommends strongly that it not be activated, background scanning of the *Exchange* message store can be activated here, if so desired.

Conclusion

Although the cosmetic appearance of *eTrust Antivirus*' main scanning module has not seen a significant change during its passage from version 6 to version 7, the change in the supporting documentation and the information available on the web has been very positive.

The facets of the product covered within this review demonstrate that the support for remote administration, installation and update is also copious. What is more pleasing, at least from a reviewer's point of view, is that the documentation renders relatively painless the task of using these features. As for improvements in scanning capabilities, future comparative reviews will indicate whether there have been any.

Technical details:

Test environment: Identical 1.6 GHz Intel Pentium machines with 512 MB RAM, 20 GB dual hard disks, DVD/CD-Rom and 3.5-inch floppy drive.

Server software: *Windows 2000 Server Service Pack 2* with *Exchange 2000 Server Service pack 2* and *Outlook 98*

Client software: *Windows XP Professional* with *Outlook Express*, *Windows XP Professional* with *Outlook 98*

Other software: a selection of other machines, running a variety of operating systems, were connected to the test Network, but not used as targets for software installation or scanning.

Developer: *Computer Associates International Inc*, One Computer Associates Plaza, Islandia, NY 11749, USA; tel +1 631 342 6000; fax +1 631 342 6800; email cainfo@ca.com; website http://www.ca.com/.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues) including first-class/airmail delivery:

UK £195, Europe £225, International £245 (US$395)

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire, OX14 3YP, England

Tel 01235 555139, International Tel +44 1235 555139
Fax 01235 531889, International Fax +44 1235 531889
Email: editorial@virusbtn.com
World Wide Web: http://www.virusbtn.com/

END NOTES AND NEWS

RSA Conference 2003 takes place 13–17 April 2003 at the Moscone Center, San Francisco, CA, USA. General sessions feature special keynote addresses, expert panels and discussions of general interest. For more information and booking details see http://www.rsaconference.net/.

Information Security World Asia takes place 23–25 April 2003, at Suntec Singapore. For details of what is claimed to be Asia's largest and most dedicated security technology and solutions exhibition see http://www.informationsecurityworld.com/2003/iswa_SG/.

Infosecurity Europe 2002 takes place 29 April to 1 May 2003, at Olympia, London. A free keynote and seminar programme alongside almost 200 exhibitors is expected to attract more than 7,000 dedicated security visitors. See http://www.infosec.co.uk/.

EICAR 2003 will take place 10–13 May 2003 in Copenhagen, Denmark. The 12th Annual EICAR Conference combines academia, industry and media, as well as technical, security and legal experts from civil and military government, law enforcement and privacy protection organisations. Call the conference hotline +45 4055 6966 or +44 709 211 1950, or check http://conference.eicar.org/ for details.

Black Hat Europe 2003 will be held 12–15 May 2003 at the Grand Krasnapolsky, Amsterdam, the Netherlands. Trainings take place 12–13 May and Briefings 14–15 May. For more information see http://www.blackhat.com/.

The DallasCon Wireless Security Conference takes place 24–25 May 2003 in Plano, Texas. A two-day wireless security course precedes the conference, including hands-on lab experience and lectures. For full details see http://www.DallasCon.com/.

Infosecurity Canada Conference and Exhibition takes place 4–5 June 2003 in Toronto, Canada. For registration and exhibitor details see http://www.infosecuritycanada.ca/.

NetSec 2003 Conference and Exhibition takes place at the Hyatt Regency, New Orleans 23–25 June 2003. The CSI NetSec conference is devoted exclusively to network security. For more details, including conference programme, exhibitor list and registration information, see http://www.gocsi.com/.

The Black Hat Training and Briefings USA 2003 take place 28–31 July 2003 at the Caesar's Palace hotel, Las Vegas. A call for papers for the Briefings remains open until 15 May 2003. For more details and registration for the event, see http://www.blackhat.com/.

The Thirteenth *Virus Bulletin* International Conference and Exhibition (VB2003) takes place 25–26 September 2003 at the Fairmont Royal York hotel in Toronto, Canada. For sponsorship details or an exhibitor's pack, contact Bernadette Disborough on +44 1235 555139 or email vb2003@virusbtn.com. For more information see http://www.virusbtn.com/conference/.

Black Hat Federal 2003 takes place 29 September to 2 October 2003 in Washington D.C. See http://www.blackhat.com/.

The Workshop on Rapid Malcode (WORM) will be held in association with the 10th ACM Conference on Computer and Communications Security, 27 October 2003 Washington D.C. The workshop aims to bring together ideas, understanding and experience relating to the worm problem from a wide range of communities including academia, industry and government. The organisers are currently seeking papers for the workshop – these must reach the selection committee by 1 July 2003. Further submission instructions can be found at http://pisa.ucsd.edu/worm03/.

Erratum: *VB* regrets that a typographical error crept into the 'Slamdunk' article in the March 2003 issue of *Virus Bulletin* (see *VB*, March 2003, p.6). The article read 'The worm starts with a header posing as local variables of the buggy function. A new return address (0x42BCC9DC) follows these filler bytes.' In fact, this should have read 'The worm starts with a header posing as local variables of the buggy function. A new return address (0x42B0C9DC) follows these filler bytes.' *VB* apologises for any confusion.

Kasperksy Labs* has released a new version of its home user product, *Kaspersky Anti-Virus Lite 4.5. The new release includes both internal architecture and external changes designed to enhance the software's functionality and make it easier to use. According to *Kapsersky*, a re-worked program interface makes *KAV Lite* more ergonomic, with the buttons placed more logically, and with a new and more attractive design and colour set. See http://www.kaspersky.com/.