ISSN 0956-9979 MARCH 2003

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

Editor: **Helen Martin**

Technical Consultant: **Matt Ham**

Technical Editor: **Jakub Kaminski**

Consulting Editors:

Nick FitzGerald, Independent consultant, NZ
Ian Whalley, IBM Research, USA
Richard Ford, Independent consultant, USA
Edward Wilding, Data Genetics, UK

IN THIS ISSUE:

• **Making the race more even:** The idea of concentrating on the possible harm that a virus-infected machine can cause, rather than on harm that could be done to it, is one that was taken up by researchers at *Hewlett Packard Labs*. As a result, the 'virus throttling' technique was developed. See p.8.

• **And now for something completely different:** What is the psychological toll inflicted by computer viruses? Psychiatrist Dr H.W. LeBourgeois, of Tulane University of New Orleans, ran a study to attempt to answer this question, revealing some interesting findings. See p.12.

• **More thorny issues:** In response to *VB*'s previous article on the move to formalise and update the CARO virus-naming conventions, Sarah Gordon sets out her own thoughts and suggests that the road historically travelled is not necessarily the one we should continue to take. See p.14.

CONTENTS

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England.** **http://www.virusbtn.com/ /2003/$0.00+2.50** No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

COMMENT

Viruses: Is the Battle Really Changing?

There is a lot of gossip these days about computer viruses. If you go to a seminar about *Microsoft*'s *.NET* framework, or to a presentation of *Canon*'s brand new digital camera, it seems that everybody is talking about computer security and viruses. On the train, at the airport, everywhere I go I hear people talking about it. Everyone seems to be an anti-virus expert. However, if you listen carefully, you can overhear the most ridiculous statements, ranging from: 'If you are not connected to the Internet you can't get viruses!' to 'My firewall should block all the viruses I have, but still I'm spreading Klez and Bugbear … strange!'. Here are some of the statements I have overheard:

"Viruses and other malware could break the Internet even with all the filtering software in place."

'*Gateway protection was not needed and didn't exist 15 years ago!*' Let's go back to the year 1988. If you had a good anti-virus policy at that time you probably worked with what we called 'sheep-dip' PCs. These workstations, positioned in strategic places, were there to scan every incoming document, spreadsheet and program on diskette. This was the 'Gateway' protection *avant-la-lettre.* Anybody who says that this kind of protection didn't exist years ago is wrong. Email was rarely used in those days and certainly not in the format we know it in today.

'*The EICAR test string is going to change, and that will be a problem for detection of the old string.*' The string itself will not change. The string remains the same as it was in the early 1990s. It is the definition that will be changed slightly from 1 May 2003. The change is in order to make it impossible to include the EICAR test file within any virus and to make it easier for any anti-virus vendor to detect it as the unique EICAR test file. More information will be provided at the EICAR conference this year and on the EICAR website (http://www.eicar.org/).

'*The virus battle is changing!*' Is this really the case? From some of the poorest people to presidents and royalty, almost everybody is using the largest network in the world: the Internet. And, as a result, everyone gets viruses, spam, chain letters, and so on. We already have three times more spam than last year and it's still on the increase. So, as the Internet 'matures', governments, corporations, universities and service providers are erecting fences.

The Internet worked well when computers did no more than their assigned roles: pass along data packets to the next computer. Now those computers, in control of several parties, are increasingly being called upon to make social judgements: is that packet advertising, pornography, a virus or terrorist communication?

Of course, without content filtering in the workplace, employers lose productivity and risk lawsuits if workers access illegal material. But the fight against junk email sometimes backfires – legitimate mail such as newsletters for support groups is sometimes blocked mistakenly, often without senders or recipients knowing. Other barriers are also on the way as wireless access becomes more common worldwide.

So, if you look very closely, you can see that there is indeed a change in the battle. Spam is just one problem. Unfortunately, it's not the only one. That's one of the reasons why the anti-virus industry is making moves to include firewalls, anti-spam and other security breach detecting techniques in their products. And it's not only the anti-virus industry who seem to be starting to bundle everything. Within the new era of *.NET*, *Microsoft* is gathering everything together to make it more user-friendly. Look at the W32/SQLSlammer worm or W32/Nimda or W32/CodeRed or even W32/Klez. What if virus writers start to combine their techniques too? What could happen if the virus writer combined some DDoS attacks with a worm which sent out spam and made the payloads more dormant and then the AV industry overlooked it for, let's say, a few days …

Is this fiction? I don't think so. Viruses and other malware could break the Internet even with all the filtering software in place. It nearly happened last year (attack on the 13 DNS root servers) … will it happen this year?

Eddy Willems, Data Alert International, EICAR Director Information & Press, Belgium

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. Tel +44 1235 555139. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

NEWS

Lighter Sentences for Virus Writers?

The USA's largest group of defence lawyers has backed a report claiming that sentences for computer-related crimes are too harsh. In a set of comments submitted to the US Sentencing Commission and signed by the National Association of Criminal Defense Lawyers, the Electronic Frontier Foundation and the Sentencing Project (a group that focuses on perceived injustices in penalties), sentences that have been awarded for computer-related crimes were criticised for being tougher than those for comparable, non-computer-related crimes.

According to the report the 'typical' computer crime involves the misuse of a company's computers by a disgruntled current or former employee and the severity of sentencing often exceeds that of the crime.

The author of the report believes that the serious nature of computer-related offences is often overplayed, with the calculation of loss being both unreliable and open to exaggeration. The report argues that the loss estimation for identical offences can vary widely depending on factors such as the actions taken by the victim (e.g. one victim may simply restore the hard drive from backup, while another spends large amounts of money hiring consultants to assess the damage) and the nature of the victim (i.e. the losses resulting from a compromised system within a small business with a low turnover will be lower than those resulting from a similar attack on a thriving business).

Furthermore, the report argues that 'greater penalties are dangerous' and they 'may chill legitimate computer research, business development and reporting on security vulnerabilities.' The author imagines that, were greater penalties to be instituted, security researchers who uncover and disseminate information on vulnerabilities might refrain from doing so for fear of being charged for their actions.

Last year, the sentencing by US courts of Melissa author David Smith provoked considerable discussion within the anti-virus community. Some considered Smith's 20-month prison sentence a fitting penalty for what they, like the authors of the paper, felt amounted to little more than a 'white collar crime', while others were disappointed by the lenience of the sentence. There was little talk, however, of the sentence being too harsh.

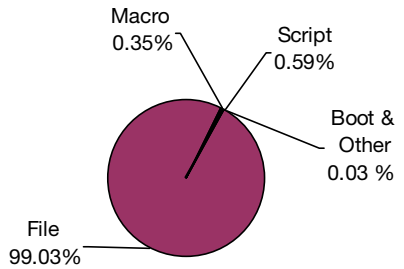
In his comment on David Smith's sentencing (see *VB*, June 2002, p.2) AV researcher James Wolfe said, 'it is nice to see that [US] courts have set a precedent for convicting virus writers.' Let's hope the United States Sentencing Commission doesn't undermine that good work when it reviews and amends the sentencing guidelines for computer-related crimes ▮

Prevalence Table – January 2003

Virus	Type	Incidents	Reports
Win32/Opaserv	File	9228	52.20%
Win32/Klez	File	2961	16.75%
Win32/Lirva	File	1367	7.73%
Win32/Dupator	File	1192	6.74%
Win32/Yaha	File	648	3.67%
Win32/Funlove	File	460	2.60%
Win32/SirCam	File	308	1.74%
Win95/Spaces	File	306	1.73%
Win32/Bugbear	File	293	1.66%
Win32/Magistr	File	207	1.17%
Win32/Sobig	File	112	0.63%
Redlof	Script	77	0.44%
Win95/Lorez	File	62	0.35%
Win32/Nimda	File	53	0.30%
Win32/BadTrans	File	43	0.24%
Win32/Hybris	File	42	0.24%
Win32/Oror	File	36	0.20%
Win32/Gibe	File	35	0.20%
Laroux	Macro	32	0.18%
Win32/Braid	File	25	0.14%
Win95/CIH	File	25	0.14%
Win32/Elkern	File	14	0.08%
Win32/MTX	File	13	0.07%
LoveLetter	Script	8	0.05%
Win32/Aliz	File	7	0.04%
Win95/Whog	File	7	0.04%
Others [1]		118	0.67%
Total		17679	100%

[1] The Prevalence Table includes a total of 118 reports across 63 further viruses. Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

Distribution of virus types in reports



VIRUS BULLETIN ©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. http://www.virusbtn.com/. /2003/$0.00+2.50
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

VIRUS ANALYSIS 1

Looking a Bagift-Horse in the Mouth

Peter Ferrie and Frédéric Perriot
Symantec Security Response, USA

W32/Bagif is a polymorphically encrypted, entry point-obscuring, anti-heuristic, memory resident, parasitic infector of *Windows* Portable Executable files that are not DLLs. It replicates across mapped drives and shared directories on local area networks, and it appears to be based on the code of several existing viruses. In the same way that the author of W95/Bistro had his signature changed in the copy of the virus that was released, it is very likely that the author of W32/Bagif is not the one named in the code.

What Virus is That?

As an anti-heuristic device, files infected with Bagif do not have their entry point altered. Instead, the virus will search for the first call or jump to the ExitProcess() API, and replace the instruction with a transfer of control to near the end of the code section, where the virus will place itself. The technique is very similar to that used by W32/Simile (see *VB*, May 2002, p.4). Additionally, no section attributes are altered, so after infection files look very much as they did beforehand.

When a file infected with Bagif is executed for the first time, and if the virus gains control via execution of the replaced instruction, the virus executes the polymorphic decryptor. The decryptor has characteristics that allow it to be identified immediately as produced by the KME-32 (Kewl Mutation Engine). KME-32 is the engine used by several other viruses, including W95/MTXII, W32/Toal and W95/Zexam. Analysis of Bagif's code allows the engine to be identified as version 5.52, which was released on the first day of 2002.

How Can I Run Thee?

The decryptor uses the floating point unit to perform the decryption, which is an effective attack against the CPU emulators in some anti-virus products. The decryptor places a small (216 bytes) routine on the stack, and then runs this routine.

The routine searches in memory for KERNEL32.DLL and retrieves the API addresses for two functions: GlobalAlloc() and GetModuleHandleA(). The function names are stored as checksums, however the checksum algorithm is the simple checksum used by Delphi applications, among others, rather than the more common CRC32. It is probable that the algorithm was chosen for its smaller size, and considered acceptable despite the increased risk of non-unique checksums. Once the API addresses have been retrieved, the routine allocates memory in which to place the virus body, then decrypts the virus body directly into this memory.

The use of dynamically allocated memory is the method by which an encrypted virus can run from files without altering the section attributes, and the small 'first stage' routine reduces the chance of stack overflow.

Let Me Count the Ways

Once the virus body gains control, it checks whether another copy of it is already running. If no other copies are running, the virus decompresses and creates a file called 'backup.gif', in the directory used for storing temporary files. The compressor that is used is aPLib, a favourite among virus writers.

After the file is written, the virus will infect the file, execute it, then exit, leaving the dropped file as the one that remains running in memory. Whenever the dropped file is executed, it will copy itself to the Windows\System directory as 'ntloader.exe', and to the current user's Startup directory (if it can be found by querying HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Startup in the registry) as 'win32s.exe'.

After copying itself, the virus will install itself as the application that handles requests to run .exe files, by changing the exefile Open key in the Registry (HKCR\exefile\shell\open\command). If the dropped file was executed as a result of the change to the exefile Open key, there is a 50 per cent chance that the virus will infect the file being executed.

We Interrupt this Program

When the dropped file is executed for the first time it will register itself as a service process, if the undocumented RegisterServiceProcess() API is available (it exists in *Windows 9x/Me*), then create a thread that spreads the virus. The first part of the thread enumerates network disk resources; the second part of the thread enumerates drive letters. Intriguingly, the virus body contains the names of some APIs that can be used for email and/or backdoor effects, but there is no reference to these APIs in the virus code. Perhaps the virus author had not completed the routines before the virus was released.

Share and Enjoy

The spreading of this virus across the network is achieved using a method that is very similar to that used by W32/Magistr. Bagif begins by changing to the directory

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. Tel +44 1235 555139. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

that it has found on the network, then attempting to create a file, using a random name and extension, to determine whether the directory is writeable. If the file can be created, then the virus will guess at the name of the *Windows* directory and try to change to that directory. If the change is successful, the virus will copy itself as 'tsoc32.exe' and alter the WIN.INI in that directory to run the copied file whenever *Windows* is started. The virus uses the WritePrivateProfileString() API to do this, because that API allows the path of the .INI file to be specified.

Randumb

The random number generator that is used is very similar to the one in *Microsoft Visual C++*. It has a short period, but it is also small. Random number generators are very common in viruses, and range from the very simple (calling the GetTickCount() API repeatedly) to the very complex (the Mersenne Twister in W32/Chiton). The algorithm that is chosen is often a compromise between randomness, period length, and code size.

The spreading across drive letters is done backwards, from Z until the drive letter that contains the Windows\System directory. For each local and mapped network drive, the virus will check whether the directory is writeable, as described above. If the directory is writeable, the virus will scan recursively into directories, but only to a depth of four subdirectories. For each file that is found, there is a 50 per cent chance that it will be skipped explicitly. This behaviour is identical to that of W32/Simile.

The virus looks for files with extensions EXE or SCR, but whose name does not begin with 'EXPL' or 'UNRE', or whose name is 'HL'. These names would match files such as Explorer, and the game files for *Unreal* and *Half-Life*, all of which are self-checking.

Are You My Type?

Additionally, the file size must be between 4kb and 2Mb, and the file must be executable and not a DLL. The check for the CPU type has its origins in misinformation about the allowed values. The standard allows only for a value of 0x14C in the field, corresponding to Intel 386+ CPUs, but documents exist suggesting that the values 0x14D, 0x14E, and 0x14F, exist, corresponding to Intel 486, 586, and 686, CPUs. In fact, no value for Intel x86 CPUs, other than 0x14C, is supported by *Windows*.

The virus attempts to match the first few characters of every section name against a list of 15 names that the virus carries. Files are avoided if they contain unrecognised section names. After these checks have been made, the virus looks for its infection marker.

Files are considered infected if the difference between the third byte of the COFF Date/Time stamp field and the XOR of the low two bytes is less than eight. Files must also import the ExitProcess() API from KERNEL32.DLL, since the virus requires this function for its EPO implementation, however only the first eight characters are matched in the import name and the dll name, so an API called 'ExitProcrastinator()', for example, in a file called 'KERNEL32R0X', would be accepted too.

If the file imports ExitProcess(), the attributes of the first section are checked, unless the file to be infected is the dropped file. The checking of the first section appears to be a bug, since the number of sections is retrieved, but never used. It is likely that the last section was the one that was intended to be checked, which also matches the behaviour of W32/Simile. The contents of the first section are checked for 'virus-like' strings (a check for 'MZ' followed within 128 bytes by 'PE'). Files are avoided if they contain these strings.

I'm Attached to You

If a file passes all of these checks, the virus will allocate a buffer in which to place itself and the decryptor. The virus begins by filling the buffer with between 255 and 512 bytes of random values, then the virus body is encrypted and placed immediately after these values. The encryption is weak and the original key can be restored very easily. At this point, the virus will decompress and run the KME-32 code, to create a new decryptor for the 'first-stage' routine.

Once the decryptor has been created, the virus will append the buffer to the first section in the file, then update the physical and virtual locations of all of the following sections, as well as fixing the imports, resources, exports, and relocations.

While parsing the relocation table, the virus removes the relocation pointing to the instruction that was altered. This is necessary because the new instruction does not require relocation. If the image base is the *Windows* default value (0x400000), and the relocation section is the last one in the file, then there is a 50 per cent chance that the virus will remove the relocation section completely. If a checksum existed before, then the virus will calculate a new one.

Conclusion

W32/Bagif is an interesting sum of other viruses' parts, with some neat optimisations, but we have seen it all before.

W32/Bagif	
Type:	Polymorphic, EPO, memory-resident, parasitic.
Infects:	PE .EXE and .SCR files.
Self-recognition:	Magic value in PE header of files.
Removal:	Delete infected files and restore from backups.

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. http://www.virusbtn.com/. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

VIRUS ANALYSIS 2

Slamdunk

Péter Ször and Frédéric Perriot
Symantec Security Response, USA

The Slammer worm targets versions of *Microsoft SQL Server 2000* products, as well as *MSDE 2000* and related packages. The outbreak began on 25 January 2003 (GMT). According to early reports, the worm had a very significant presence around the world in less than one hour, and the peak time of the worm lasted for about three hours. During the worm's initial outbreak, Internet users experienced large percentage packet drops that developed into a large-scale DoS attack.

The worm exploits a stack-based overflow that occurs in a DLL implementing the SQL Server Resolution Service. This DLL (ssnetlib.dll) is used by the *SQL Server* service process called SQLSERVR.EXE. The vulnerability had been reported to *Microsoft* by David Litchfield (*NGSSoftware*), along with a few others. Furthermore, exploit code was made available at a *BlackHat* conference in 2002 and it is clear that this code was used as a base from which to develop the worm.

Exploit Setup

The *SQL Server* process listens on TCP as well as UDP ports. The worm targets UDP port 1434, sending a special request (0x04) specified as the first character of the payload. In the datagram this is followed by a specially crafted 'string' that contains the worm code. The worm code is extremely small – 376 bytes, which is the shortest binary worm known today. (376 bytes is the length of the UDP datagram without the protocol headers.)

Since the worm can use a UDP packet for the attack, it is probable that the source IP address of the original attacker was spoofed. The worm spreads to randomly generated IP addresses and, as a result, it is very difficult to determine from which country the attack originated.

The vulnerable function in ssnetlib.dll (as implemented in *SQL Server 2000*) is nested two levels deep inside a thread associated with the incoming request. The function is supposed to build a string for a Registry access by concatenating three strings into a 128-byte buffer. This string will be built on the stack and there are no input validations for the size of the middle string parameter. Strings 1 and 3 are constant and located in the ssnetlib.dll.

(String 1) 'SOFTWARE\Microsoft\Microsoft SQL Server\'

(String 2) String passed in the datagram (starts after the 0x04 type field)

(String 3) '\MSSQLServer\CurrentVersion'

As a result, whenever a string that is too long is passed to the function, the stack is corrupted (smashed). String 2 is an *SQL Server* instance name. According to the *Microsoft Knowledge Base* this string should be 16 characters long at most. However, this is neither enforced in the server, nor even in some of the common clients.

The worm has been crafted carefully. Its code is not only compact but it contains no zeros. This is because the buffer is used as a string parameter to an sprintf() library function call. As a result of the overflow a concatenated string will build on the stack where string 2 is the worm body itself.

Getting Control

Since the worm cannot contain zeros the author uses a lot of 01 filler bytes. Furthermore, attempts are made to use addresses that do not contain any zeros and, in some cases, the code uses XOR to mask zero bytes, which is a known shell code technique.

The worm starts with a header posing as local variables of the buggy function. A new return address (0x42BCC9DC) follows these filler bytes. This address is a pointer to a JMP ESP instruction inside SQLSORT.DLL, another module of the *SQL Server* process.

To make sure the vulnerable function will give control to the worm body, the header section of the worm also uses dummy ('crash test dummies') values (0x42AE7001) to replace function arguments on the stack. It is necessary to do this because these arguments are used after the call to sprintf() triggering the overflow. Failure to replace these arguments would cause an exception and thus the function would not return normally. When the function returns, control flows to the JMP ESP instruction which jumps on the stack to the location immediately after the hijacked return address. The first instruction will be a short jump around fake function arguments to the main worm code.

Initialization

The local variables within the worm header section could change during the time between the actual faulty sprintf() and the function return to the worm body, which means that the worm's header could become corrupted. Thus the worm will rebuild this area first to make sure that its header section remains constant for the next attack. Since the query type field (0x04) is missing from the top of the worm on the stack it is also rebuilt by pushing a 0x04000000 DWORD whose high byte is referenced by the replication code later.

Now the worm needs only a few functions to call. Following the original exploit code the worm's author uses the import address directory of SQLSORT.DLL to make calls to LoadLibraryA() and GetProcAddress() function calls.

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. Tel +44 1235 555139. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

This routine is compatible with different Service Pack releases and patches of *SQL Server*. Therefore GetProcAddress()'s code is checked first to be sure that it is the proper function entry point.

Then the worm gets access to the handles (base addresses) of WS2_32.DLL and KERNEL32.DLL. Next it gets the addresses of socket(), sendto() and GetTickCount() APIs, which is all it needs to replicate.

Replication

The replication method is extremely simple. The worm sends 376 bytes to UDP port 1434 to randomly generated IP addresses in an endless loop. This will cause the server CPU usage to increase and thousands of packets will be sent, effectively causing a DoS attack and at the same time compromising a large number of new systems around the world. The random number used to generate IP addresses is a variant of the *Microsoft Basic* random number generator. It uses the same multiplier. This results in sufficient randomness in the distribution of targeted systems.

Conclusion

A patch had been available for six months to cover both this vulnerability and others related to it (see *Microsoft Security Bulletins MS02-039* and *MS02-061*). Patches would block the worm's attack effectively if applied properly, but they are often too costly to deploy in large corporations. It should also be noted that the patching process was not an easy one, due to the large number of *Microsoft* and third-party products that include *SQL Server* as a component.

Although *SQL Server* would offer various user rights for the installation of the server process, such a server process often enjoys system context or admin privileges. This will provide attackers with access to any resources on the system, since the hijacked thread will run with significant privileges to do further damage on the system.

It appears that modern behaviour-blocking countermeasures will need to take place at host-level to provide a last line of defence to mitigate attacks like these. Such host-based products might be the key to slowing down similar attacks in the future.

Slammer

Size:	376 bytes (worm does not exist as a file on the system).
Aliases:	W32.SQLExp.Worm, SQL Slammer Worm, DDOS.SQLP1434.A, W32/SQLSlammer, Sapphire, W32/SQLSlam-A.
Payload:	None. Large-scale DoS attacks occur as a side-effect of replication.

CALL FOR PAPERS

Final Call for Papers: VB2003

Virus Bulletin is seeking submissions from those wishing to present at VB2003, the Thirteenth Virus Bulletin International Conference and Exhibition, which will take place 25–26 September 2003 at the Fairmont Royal York hotel in Toronto, Canada.

How to Submit a Paper

The conference will consist of 40-minute presentations in two concurrent streams, Corporate and Technical. All anti-virus-related subjects will be considered.

Abstracts of approximately 200 words must reach the Editor of *Virus Bulletin* no later than **Monday 31 March 2003**. Abstracts should be sent as RTF or plain text files to editor@virusbtn.com.

More details, including a list of suggested topics for papers, can be found at http://www.virusbtn.com/conference/.

Authors are advised in advance that the deadline for completed papers selected for the conference programme will be **Friday 6 June 2003**. Full papers should not exceed 6,000 words.

VB Conference

Many of its regular attendees cite the Virus Bulletin Conference as *the* anti-virus event of the year. The VB conference provides a focus for the AV industry, representing an opportunity for experts in the anti-virus arena to share their research interests, discuss methods and technologies and set new standards, as well as meet with – and learn from – those who put their technologies into practice in the real world.

While the conference remains concentrated entirely on computer viruses and malware threats, delegates range from dedicated anti-virus researchers to security experts from government and military organizations, legal, financial and educational institutions and large corporations worldwide.

Sponsorship and Exhibition

For details of the sponsorship opportunities and exhibition packages available at VB2003 please contact Bernadette Disborough at vb2003@virusbtn.com or call +44 1235 555139. Further information about the conference, including online registration, will be available from the *VB* website shortly, see http://www.virusbtn.com/conference/.

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. http://www.virusbtn.com/. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

RESEARCH FEATURE 1

Virus Throttling

Matthew M. Williamson, Jamie Twycross, Jonathan Griffin and Andy Norman
Hewlett Packard Labs, UK

Virus throttling is a new technique to contain the damage caused by fast-spreading worms and viruses. Rather than attempting to prevent a machine from becoming infected, throttling prevents the virus spreading from an infected machine. This reduces damage because the virus is able to spread less quickly, and produces less network traffic.

Throttling is particularly effective against fast-spreading viruses, where signature-based approaches are weak. A signature-based anti-virus approach is really a race between the virus and the signature: a vulnerable machine will be infected if a virus reaches it before the signature does, but not if the signature gets there first.

Unfortunately, not only do modern viruses spread quickly, they also have a head start in the race as the result of any delay in generating the virus signature. In the case of these viruses, it is not just the infected machines that are a problem, the network loading caused by the additional traffic generated by the virus can cause problems for all users, not just those unfortunate enough to be infected.

Virus throttling is based on controlling an infected machine's network behaviour, and so does not rely on details of the specific virus – it does not need a signature. It restricts the virus spread, which is not as effective as preventing infection in the first place.

However, if it is impossible to prevent infection (i.e. the virus has reached the machine first), then restricting the spread of the virus will help contain the damage. The outbreak will grow less rapidly, because there will be fewer machines actively spreading the virus, and the network loading will be reduced.

By damping down the spread of the virus, throttling buys time for signature-based approaches, which are slower but more effective. Throttling makes the race more even.

How Does it Work?

Throttling relies on the difference between the network behaviour of a normal (uninfected) machine, and one that is infected by a virus. The fundamental behaviour of a virus is to replicate and spread itself to as many different machines as possible. For example, Nimda makes about 300–400 connections per second (cps) and SQLSlammer (see *VB*, this issue p.6) sends 850 packets per second, both probing for vulnerable machines. Many email viruses send mail to all the addresses they can find.

Our machines do not normally do this! They tend to contact machines at a much lower rate, and also contact the same machines repeatedly. The rate of connections to *new* machines is more in the order of one connection per second for TCP/UDP and once every ten minutes for email.

A virus throttle is a rate-limiter on interactions with new machines, where 'interactions' could be the initiation of a TCP connection, or the sending of a UDP packet or email, and 'new' is defined as the interaction having a different destination address from anywhere the machine has contacted recently. The throttle delays (as opposed to drops) connections that occur at a higher rate than that allowed.

If a virus attempts to scan for vulnerable machines at a high rate (e.g. 400 cps), the throttle will limit this to something much smaller (e.g. 1 cps). This will slow down the rate at which the virus can spread. If the virus is attempting 400 connections every second, and only one is being allowed, the backlog of delayed connections will grow rapidly. The length of this backlog is a reliable indicator that a virus has infected the system, and more drastic action can be taken. This involves preventing any further propagation e.g. stopping networking, and alerting IT staff.

The throttle thus slows down viruses until they are detected, at which point any further propagation can be stopped. For fast-spreading viruses, this whole process can take less than a second.

The rest of this article describes the throttling algorithm in more detail, discusses which protocols are suitable for throttling and presents some results from a user trial. The following sections then show how quickly the throttle can prevent onward propagation of the virus, as well as some experiments on how using throttles can affect the extent of a virus outbreak.

Throttle Algorithm

The throttle is a rate limiter on connections[1] to new machines, and is shown schematically in Figure 1. Whenever a request is made, the throttle checks to see whether the request is to a new host, by comparing the destination of the request with a short list of recent connections. The length of this list, or 'working set', can be varied – thus altering the sensitivity of the system. For example, if the length of the working set is 1, all requests other than consecutive connections to the same host will be 'new'.

If the host is not new, the request is processed as normal, but if it is new it is added to a 'delay queue' to await processing. Every time a timeout expires (indicated by 'clock' in Fig 1), the rate limiter process pops one request off the delay queue and processes it, thus ensuring that only one connection is made to a new host per timeout period.

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. Tel +44 1235 555139. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

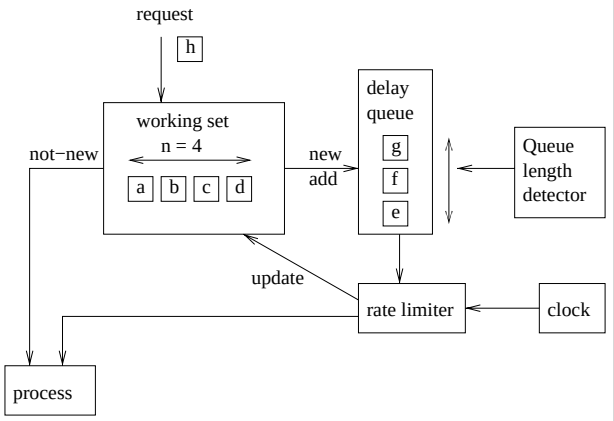


Figure 1: Virus throttling. When a request is made the destination is compared against a 'working set' of recently contacted hosts, to determine whether it is new or not. If new (e.g. the request is to host 'h', which is not in the working set), then the connection is queued in the delay queue. If not new, the request is processed as usual. The requests in the delay queue are popped off and processed at regular intervals, guaranteeing that no more than one connection per time interval is made to a 'new' host. Potential viral activity can be detected by monitoring the length of the delay queue.

The rate limiter processing involves releasing the request at the head of the queue and releasing any others to the same destination, as well as updating the working set with the new destination (removing a host from the working set and replacing it with the new destination).

Since the throttle implements a rate limit, and delays connections made at a higher rate rather than dropping them, if a process makes many connections they will mount up quickly in the delay queue. Therefore, monitoring the length of that queue gives a good indication of whether a process is acting like a virus. If the length of the queue reaches a threshold, the offending process can be halted either by stopping networking or by suspending the process itself. A user or administrator can then be contacted.

The allowed rate of connections to new machines is set to enable normal traffic to pass with minimal delay. If there are occasional periods when connections are made at a higher rate, these will be put on the queue, but the queue will not grow large, and so the delays will be small.

Since different protocols have different characteristics, it is possible to have a throttle per application, per protocol etc. The throttle itself could be implemented in a variety of places. On the host it could be inserted into the network stack e.g. as part of a software firewall, ethernet driver, etc., although being on the host makes it vulnerable to being switched off by a virus. It could also be implemented at various locations in the network. For protocols that use proxies e.g. email, web, etc. the throttling could be carried out at the server.

What Protocols can be Throttled?

In order for a machine to be 'throttled', its normal traffic must not look like a virus spreading i.e. connections made at a low rate, contacting the same machine repeatedly. The traffic for some protocols is nearly always to the same machine. For example SMTP (port 25), IMAP (143), web proxy (8088). Throttling on these protocols could be very tight since the destinations change so rarely. The behaviour for other protocols where the destination address changes is shown in Figure 2.

Normal network traffic was collected and the effect of the throttle was simulated for a range of working set sizes and allowed rates of connection. The average delay for each connection/interaction was then calculated. Figure 2 shows the different values of working set size and rate that give a constant average delay, for different protocols. The average delay for the TCP/UDP data is 0.3 ms, corresponding to three connections in every 1000 being delayed by one second.

The graph (Figure 2) shows that *Microsoft* file sharing (139), http (80), ssl (443) and dns (53) all look conducive to throttling, with a reasonable selection of working set sizes and allowed rates. The best settings are with a small working set and a low allowed rate, on the 'knee' of the curve.

Good values for http are thus a working set of 5, allowed rate 1 etc. Data for email (not shown) suggests that email can also be throttled, but the working set should be larger (around 20) and the allowed rate much lower (say one new email address every ten minutes).

Protocols that were found to be difficult to throttle were *Microsoft* naming (137) and another port used for file sharing (138 udp). The UDP traffic on both of these ports is to many different hosts at high rates, albeit in a bursty manner.

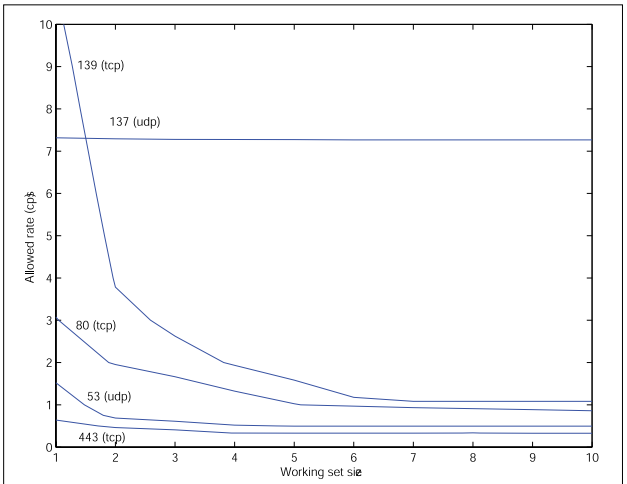


Figure 2: Throttle parameter settings for different protocols. The plot shows the settings for allowed rate and working set size that gives a constant average delay per connection. The different lines correspond to traffic with different destination ports. With the exception of UDP traffic on port 137, the other traffic is suitable for throttling, with different protocols requiring different parameter settings. The best parameter settings are with the smallest working set, and lowest allowed rate.

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. http://www.virusbtn.com/. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

This data was collected for desktop machines, but one would expect similar patterns for servers (which primarily handle incoming connections and make outgoing connections to a limited number of machines).

There are notable exceptions: machines running scanners, web crawlers or notification services or, for UDP, a dns server. These would all look like machines infected with viruses (they have many interactions with different machines at a high rate).

It would thus be difficult to throttle such machines. However this is not the end of the world, it just means that, should a machine running one of these applications become infected with a virus spreading over the same protocol, then the spread from that machine could not be limited.

While the average delay gives some idea of what effect throttling would have on normal usage, the best test is whether those delays cause difficulties for users. One trial we have run involved throttling all TCP connections for three users over a two-month period. The working set size was five, and the allowed rate 1 cps.

Table 1 shows the results of the trial. 98% of connections occurred without delay, and the maximum delay was five seconds, occurring only once in over 80,000 connections. Anecdotally, the users did not notice any of these delays – the most likely explanation being that networks are full of delays of this sort of size.

Delay (s)	No. of requests	Percentage
0	80641	97.8%
1	1428	1.73 %
2	300	0.36%
3	29	0.03%
4	2	0.02%
5	1	0.01%

Table 1: Details of the user trial. For each delay (in seconds), the table shows the number and percentage of connections that were delayed.

To summarise, the analysis suggests that the majority of common protocols have normal traffic patterns that make them suitable for throttling. In addition, the sorts of delays that throttling creates in normal usage are small and not noticeable.

How Quickly does it Stop Onward Propagation?

As mentioned previously, the length of the delay queue can be monitored to detect if a machine is making many connections to many different machines and is thus likely to be infected by a virus. If the queue goes over a defined threshold, further propagation can be stopped.

Setting the delay queue threshold to 100 (fairly high, given that the queue never went above 5 in our user trial), Table 2 shows the time taken to stop virus propagation and the number of connections made for Nimda, and a number of different conditions for a hand-crafted 'test worm'.

Virus	Connections per second	Stopping time (s)	No. of connections allowed
Nimda	120	0.2	1
Test worm	2	106.1	104
Test worm	5	26.5	25
Test worm	10	11.2	11
Test worm	20	5.4	5
Test worm	40	2.3	2
Test worm	60	1.4	1
Test worm	80	1.0	1
Test worm	100	0.9	1
Test worm	150	0.2	0
Test worm	200	0.0	0

Table 2 – Showing time to stop and number of connections before stopping for a throttle with an allowed rate of 1 cps. Fast viruses are stopped very quickly (Nimda in 0.25 seconds), while slow ones are stopped fairly promptly (1.5 minutes for 2 cps). The virus is not able to make many connections before it is stopped.

The table shows some encouraging results, indicating that the faster the virus, the more quickly it is stopped, the time being less than a second for rates higher than 100 cps. Even viruses with a relatively slow connection rate are stopped quickly (just over 100 seconds for 2 cps). The number of connections that the virus is able to make is fairly small in all cases. The throttle is effectively preventing the onward propagation of the virus from the infected machine.

The fact that the throttle can detect the presence of a virus very quickly might make it useful as a monitoring device to provide early warning of viruses and collect data on their behaviour to enable quicker virus signatures.

Effect on Global Spread

A virus throttle is an altruistic idea – a throttled machine may still become infected, but it will not pass the infection on to others. Like most altruistic ideas, throttling will be most effective when it is widely used! Obviously throttling every machine is impossible in practice, so the question is: what is the effect on virus outbreaks if a smaller proportion of machines have the throttle?

That question is difficult to answer. There are many factors that make it hard to predict virus spread, let alone with throttling in the picture too. Virus spread depends on the propagation strategy of the virus, the density of vulnerable machines, the topology of the network, user behaviour, and so on. It also depends to a large extent on the whole process of fighting the virus: how quickly signatures are created and distributed, the vigilance of IT staff and so on.

To assess the impact of throttling, we tested the throttle against real viruses and measured spreading rates. Figure 3 shows the propagation of Nimda against time for an isolated subnet of 16 machines. When all the machines are throttled, Nimda does not spread at all. When no machines are throttled, Nimda takes about 20 minutes to infect all the machines. The other lines show that it is only when more than half of the machines are throttled that the infection is slowed, to about an hour when 75% of the machines have throttles.

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. Tel +44 1235 555139. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

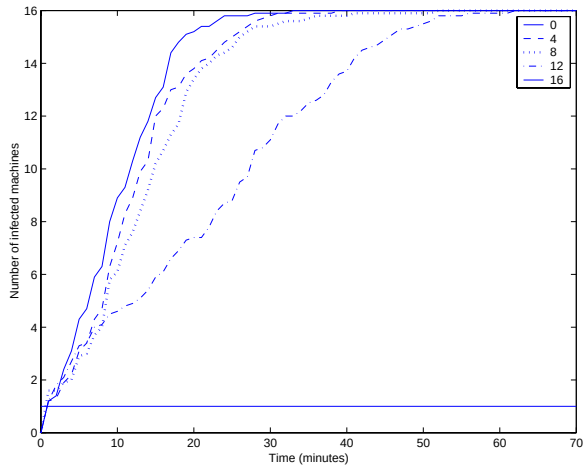


Figure 3: Number of infected machines plotted against time for the Nimda virus, varying the number of machines with throttles. Each line is the average of 10 runs. With no throttles, the virus infects all machines in about 20 minutes, but if all the machines use throttles, the virus does not spread at all. More than half of the machines need to have throttles in order to slow the virus significantly.

While the virus was slowed by a factor of three, it still spread through all the machines in one hour. This is quite a harsh test however, as Nimda's spreading strategy targets the local subnet, and so would be expected to spread most quickly there. However, the number of infected machines is not the only problem, the traffic they produce is important too. For the run with 12 of the 16 machines throttled, the traffic loading would be around one quarter of that in the unthrottled case, ignoring any saturation effects.

Part of the problem is that, because Nimda is a scanning virus, a single unthrottled machine can continue infecting others for as long as it is allowed to run. Other viruses do not scan (e.g. email viruses), and the throttle is likely to be more effective against those, mainly because the damage from an unthrottled machine will be much smaller.

We have also used modelling to look at throttling in the context of signature-based approaches, with a model of virus spread and clean up. The results from that model suggest that if 50–60% of machines (or more) have throttles, the impact of virus outbreaks can be much reduced[2].

Conclusion

Virus throttling is a new approach to containing the damage caused by fast-spreading worms and viruses. It targets the propagation of viruses from infected machines, slowing and stopping it. This reduces the number of machines actively spreading the virus, which in turn means that the outbreak grows more slowly, and the amount of traffic (and therefore disruption) is reduced.

Analysis of normal network traffic shows that the majority of protocols are suitable for throttling without interfering with normal usage, and that virus outbreaks can be reduced if a reasonable proportion (more than 60%) of machines are throttled.

Computer security is an arms race, with the attacking and defending technologies changing to exploit each other's weaknesses. Throttling is, in principle, quite hard to beat, because it targets a fundamental characteristic of a virus: its replication. It is not possible to replicate as a computer virus without contacting different machines!

On the other hand, one way to defeat the throttling technique would be to design slow viruses that pass through the limiter without delay. However, while the throttle would not stop such viruses, there are many other mechanisms already in place to deal with such slow threats.

A simpler way to defeat the throttle would be if malicious code were able to switch it off, as recent viruses (e.g. Bugbear) have begun to do with software firewalls and anti-virus software.

A virus switching off the throttle is worrying, because the throttle is designed to work after infection. There are two solutions for this, one is to hide and obfuscate the software on machines to make it more difficult to disable, and the alternative is to move the throttle into the network. Currently we are researching a variety of ways to do this. The throttle might already be implemented in the network;, for example the logical place for an email throttle is at the outgoing mail server.

In conclusion, throttling looks like a promising technique. The idea of concentrating on the possible harm that a machine can do, rather than on harm that could be done to it is a general and powerful one. If our machines were made so that they could cause less harm to one another, using throttling and other similar technologies[3], then our computer systems would be much more resilient when under attack.

Footnotes

1 The term 'connections' is meant to apply not just to TCP connections, but also to UDP packets and emails. The word should really be 'interactions', but that is clumsy and does not fit any of the protocols.

2 See 'An epidemiological model of virus spread and cleanup', M. M. Williamson and J. Leveille (2003), Submitted to USENIX Security Symposium 2003. This paper is available from http://www.hpl.hp.com/research/bicas.

3 See 'AngeL: a tool to disarm computer systems', D. Bruschi and E. Rosti, *Proceedings of the New Security Paradigms Workshop 2002*, pp.63–69.

Further Reading

M. M. Williamson (2002), 'Throttling Viruses: Restricting propagation to defeat malicious mobile code', *Proceedings of ACSAC Security Conference 2002*, pp.61–68, available from http://www.hpl.hp.com/techreports/2002/HPL-2002-172.html.

J. Twycross and M. M. Williamson (2003), 'Implementing and testing a virus throttle', Submitted to USENIX Security Symposium 2003, available from http://www.hpl.hp.com/research/bicas.

M. M. Williamson (2002), 'Resilient Infrastructure for Network Security', Submitted to "Complexity", available from http://www.hpl.hp.com/techreports/2002/HPL-2002-273.html.

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. http://www.virusbtn.com/. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

RESEARCH FEATURE 2

Computer Viruses: Electronically Transmitted Disease?

H.W. LeBourgeois III, M.D.
Tulane University School of Medicine, USA and
Monique LeBourgeois, M.A.
The University of Southern Mississippi, USA

In the summer of 2001, W32/Sircam (see *VB*, September 2001, p.8) infiltrated the server at Tulane University, New Orleans, infecting both personal and institutional computers. Tulane University Health Sciences Center was the seat from which we began to observe the psychological effects of computer virus infection (CVI).

The self-propagating worm grabbed documents from infected computers and sent them to email address book contacts. As one might imagine, the medical doctors at the Center were especially concerned about the possibility of sensitive documents being sent to other individuals. This stressor was compounded by other potential outcomes, for example: had the worm damaged their personal computers?, could the damage be repaired?, would they lose any important personal documents or information?, how much time and money would they have to expend to repair this damage?

We observed anxiety, frustration and anger among those affected by this CVI. What became increasingly interesting was that the action of a single person (the virus creator) had caused such stress in the lives of people at Tulane University, and Tulane University was a mere 'drop in the bucket' when it came to the number of people affected by CVI worldwide.

What was the actual psychological toll inflicted by computer viruses?

Study

In response to this question, an online survey was developed to explore the emotional, behavioural, and cognitive reactions of computer users who had been affected by CVI. We also wanted to assess risk/protective factors for these psychological reactions (e.g., level of computer experience, number of prior infections) and potential social/interpersonal consequences of transmitting a CVI.

We hypothesized that if CVI was associated with the development of significant psychological symptoms in computer users, the term 'electronically transmitted disease' might be appropriate in describing the syndrome they experience.

Study Design

An online version of the Computer Virus Infection Survey (CVIS) was completed by 308 college students from The University of Southern Mississippi, all of whom had previous experience with CVI. The CVIS assesses personal reactions to CVI across three separate domains: affective, behavioural, and cognitive. For each of these domains, students were asked questions about the presence and severity of their reactions in terms of anger, anxiety and depression.

Findings

Psychological symptoms were a common reaction to CVI across all domains (see Figure 1). The most severe reactions to CVI were feelings and behaviours associated with anger.

The authors identified both risk factors and protective factors in development of significant elevations of anger, anxiety, and depression (as compared to other study subjects). Female gender and African-American race were factors associated with elevated responses on several measures. Level of computer experience was not a significant risk factor. Interestingly, a self-reported history of a psychological problem was a protective factor on several measures, suggesting that absence of a prior psychological problem does not obviate computer users from experiencing psychological symptoms.

Identification of the immediate source of virus transmission was found to protect from development of feelings of anger; however, we did not investigate differences based upon intentional vs. inadvertent virus transmission. This may be an important factor when assessing victims of cybercrime, in which virus transmission is intentional. Another important factor to examine may be identification of the virus writer. For instance, if a religious extremist group, a terrorist group, or a hate organization were to be the perpetrator of a virus, knowledge of this fact may serve to heighten psychological symptoms.

Computer users who had experienced property damage as a result of CVI were found to have significant elevations on affective measures of anger, anxiety, and depression. In contrast, privacy violations and downtime were not found to result in significantly elevated measures of psychological reactions among the survey participants.

Anti-virus software ownership prior to CVI was found to reduce angry and depressive thoughts associated with CVI, suggesting a prophylactic effect against psychological morbidity. We also identified treatment interventions associated with less severe psychological symptoms, such as purchase of anti-virus software following CVI, and rapid removal (< 24 hours) of the computer virus. Frequency

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. Tel +44 1235 555139. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

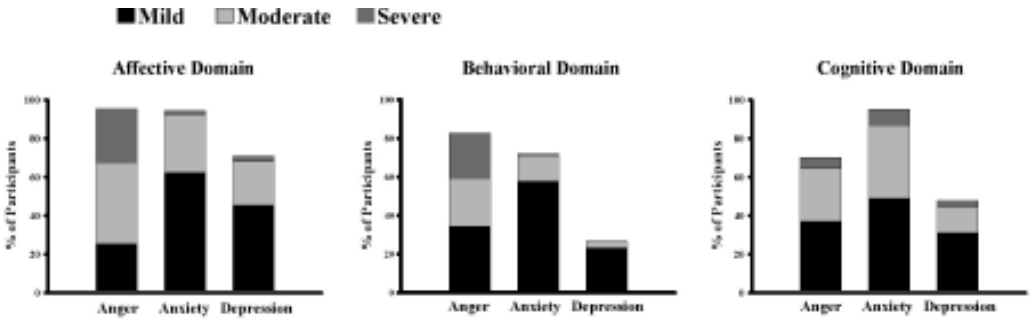


Figure 1: Psychological reactions to CVI across affective, behavioural and cognitive domains.

of anti-virus updates and data backup were not significant factors, despite these practices being important in preventing subsequent CVI or data loss. The brand of anti-virus software employed did not have an impact on psychological reactions.

We also studied the social/interpersonal consequences of CVI. Participants were asked whether they had identified the immediate source of CVI (i.e. the person or institution that had passed the computer virus to them). 35 per cent of participants said they identified the immediate source of the CVI. Of these, 80 per cent said they would change the way they communicated with the person/institution identified as the immediate source: 40 per cent would limit email contact with the source, 36 per cent would not open email attachments sent by the source, and 4 per cent would limit *personal* contact with the source.

Discussion

It is known that computer viruses inflict an enormous financial toll[1] on society. The psychological toll of CVI has previously been alluded to[2], but not formally evaluated. This study demonstrates that CVI is associated with important psychological reactions, and that risk factors for more severe symptoms may be identified. Additionally, treatment interventions may reduce these symptoms.

Professionals who develop anti-virus software and other techniques to prevent CVI may take credit not only for preventing financial loss, but also for protecting computer users from the development of psychological symptoms.

As additional techniques that lessen psychological symptoms are found, these techniques could be incorporated into anti-virus software. For instance, if literature that educates computer users on methods to prevent privacy violations (following a successful virus attack) is found to lessen psychological symptoms, then such literature could be included with anti-virus packages.

For those using email as an integral form of communication, the social/interpersonal implications of being identified as the source of CVI should not be overlooked. A majority of participants who identified the immediate source reported that they would somehow limit/change their future electronic relationship with the person or institution. A CVI-related communication breakdown between businesses and their employees/customers could result in a huge loss in productivity and/or sales. A minority of participants who identified the immediate source affirmed that they would limit *personal* contact with the immediate source, suggesting that CVI may affect social relationships outside of electronic communication.

The punishment for criminals that develop and propagate computer viruses has been relatively lenient in Western societies, considering the damage caused by their criminal acts[3]. For instance, David Smith, creator of the *Melissa* virus, was sentenced to 20 months in federal prison and fined $5,000 for causing more than $80 million in financial damage[1] [see also *VB*, this issue p.3]. Individuals have called for harsher sentences in order to deter further such criminal acts[3]. Perhaps the judges who determine sentencing in such cases should take into consideration the psychological toll inflicted upon society by those who create and propagate computer viruses.

Presently, it is not known whether CVI leads to psychological reactions sufficient to warrant psychiatric diagnoses, such as adjustment disorders with anxious or depressed mood or phobias involving the use of technological equipment. The authors plan to interview those subjects reporting more severe psychological reactions or to incorporate standard measures of psychopathology into subsequent surveys in order to investigate this possibility. Until such time, the authors assert that the term 'electronically transmitted disease' should be considered when referring to the psychological symptoms associated with contraction of a computer virus.

Dr LeBourgeois plans a further study of the psychological effects of CVI and is developing an online survey for this project. He would like to post links on computer virus-related websites directing potential participants to his online survey. For more information, contact Dr LeBourgeois by email: hwliii@yahoo.com.

References

1 'Virus verdict', *Hollywood Reporter* 2 May 2002, v373 i14 p3.

2 Khan L.A., Khan S.A., 'Doctors response to computer virus', *Saudi Medical Journal* 23(2), p.242.

3 Wolfe J., 'Melissa creator vacationing at Club Fed', *Virus Bulletin*, June 2002, p.2.

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. http://www.virusbtn.com/. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

OPINION 1

That Which We Call Rose.A

Sarah Gordon
Symantec Security Response, USA

This article began its life as a series of comments on Nick FitzGerald's excellent article about the recent efforts to extend and formalise the CARO virus naming convention (see *VB*, January 2003, p.7). The comments were far from individual in nature, however, and rather better suited to discussion with a wider audience.

First, I should point out my basic agreement with the rationale behind changes to the naming scheme – virus naming is an important issue, especially when dealing with fast-moving threats (for earlier work on these and related issues see 'Virus and Vulnerability Classification Schemes: Standards and Integration' [October 2002], available from http://www.symantec.com/.)

Furthermore, increased information sharing and threat tracking calls for a simple, scientific and, most importantly, *accepted* naming scheme. This, of course, is the key point: the widespread acceptance of the scheme is its most important attribute, as without ubiquity the utility of any such scheme is negligible. Bravo, Nick.

That said, I also believe that there are several highly critical areas that either were not addressed in the article, or were addressed incompletely. These are pointed out in order that they might become grist to the mill, rather than simply critiques; it is hoped that, by expanding the scope of dialogue, a more complete and useful scheme can be devised.

Standards

Perhaps it is appropriate, before embarking upon a more detailed analysis of naming issues, to explore the underlying reason for a standardized naming scheme. After all, by understanding the ultimate purpose of the system we are creating, we are more likely to develop a system that suits its users' needs.

Standards tend to evolve when communities need to share resources – information or, more frequently, physical things.

The use of standards means that a nut built to a particular specification in England will fit a bolt made to similar specifications in America: that is, standards provide interoperability, allowing information to be exchanged easily and accurately.

One only needs to consider the $250 million Mars Climate Observer crash – reportedly attributed to confusion over imperial versus metric measurements – to see that the lack of standards (or, in this case, confusion over standards) can lead to disaster.

On a (thankfully) smaller scale, the issue of standards is omnipresent in the computer world. The Internet itself exists almost exclusively as a result of standards – the World Wide Web's phenomenal success is possible because content that is rendered on one machine is rendered in a (hopefully) similar fashion on a completely different machine, often running a different operating system and web browser.

In the virus world, there are three different groups of people who would 'use' the naming scheme – and for each of these groups the name serves a different purpose.

- Anti-virus researchers, who are interested in taxonomy and detail. Anti-virus researchers require names to be unique, and to carry with them meaningful information regarding the virus in question, such as infective length, platform and family.
- The educated corporate users, who know enough about computer viruses to understand parts of the virus naming scheme (for example, they could be expected to know that @MM means that a particular virus is a mass mailer). For such a user, a name that is relatively memorable and that conveys some information about the virus is advantageous.
- The average users, who are, by and large, unaware of the complexities of virus naming. For these users, the name represents just that: an identifier one can use to identify the virus, albeit not necessarily rigorously.

Ignoring, momentarily, the needs within the anti-virus industry, let us consider the primary needs of the end users of anti-virus software. For these users, the names should be:

1. As simple as possible.
2. As informative as possible.

Clearly, there is a tension between these two goals: typically, simplicity does not lend itself to rich information content.

However, not all information is equally important to the user: for example, from a purely pragmatic perspective taxonomy is less important than platform. Similarly, knowing that the virus is a mass mailer is of tremendous import to the user community, whereas the exact infective length is of use only as a method for identifying the virus precisely.

From this simple analysis, it can clearly be seen that the purpose of the naming scheme varies depending upon

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. Tel +44 1235 555139. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

one's perspective. However, these different motivations and needs should be in the forefront of one's thoughts during any discussion of virus naming. Note that there is one common thread between each of these groups: the unique identification of the threat one is dealing with, if not the exact virus variant.

Reference Set?

Given that one of the goals of virus naming is identification (labelling) of different viruses, it is regrettable that virus naming conventions are not universally accepted – the sample that one scanner identifies as infected with NastyThing.A is identified by another as W97M.Toast.BO. This confusion occurs for a variety of reasons, but perhaps the most deeply-rooted is the lack of an authoritative reference collection.

Consider, for example, the way plants are named. In each case, a name can be traced back to a particular specimen of a plant – that is, there is a reference collection that can be accessed in order to determine whether a particular plant is of a particular variety. This reference collection provides a certain authority to the scheme – in the event of a dispute there is an authoritative source of knowledge that can be called upon.

Unfortunately, no such reference collection is available in the anti-virus world. While this makes a certain amount of sense given the glut of known viruses, it makes less sense when one considers just those viruses known to be actively spreading in the wild, as this represents a much smaller number. Furthermore, it is these viruses that, by and large, are most important from a naming perspective, as these are the samples that are causing 'real-world' problems for users.

The WildList Organization has made considerable inroads in this area, by the creation of the WildCore sample set – a set of samples comprising all viruses known to be spreading in the wild. This sample set is made available to *bona fide* vendors and testers, allowing them access to a sample-based system for virus identification.

However, one can argue cogently that this set is not large enough, as it is unclear as to which of the Zoo viruses will be next to appear in the wild, requiring either a rapid reconciliation between disparate names or confusion regarding the correct naming of a sample. Despite this, the WildCore collection represents an important step forward in the search for a reliable naming standard.

Wider Issues: Blended Threats

Blended threats are defined as malware that combines the characteristics of viruses, worms, Trojan horses and malicious code with server and Internet vulnerabilities to initiate, transmit, and spread an attack. By utilizing multiple methods and techniques, blended threats can spread rapidly and cause widespread damage.

Characteristics of blended threats include the following:

- Causes harm: launches a denial of service attack at a target IP address, defaces web servers, or plants Trojan horse programs for later execution.
- Propagates by multiple methods: scans for vulnerabilities to compromise a system such as embedding code in html files on a server, infecting visitors to a compromised website, or sending unauthorized email from compromised servers with a malicious attachment.
- Attacks from multiple points: injects malicious code into .exe files on a system, raises the privilege level of the guest account, creates world read and writable network shares, makes numerous Registry changes, and adds script code into html files.
- Spreads without human intervention: scans the Internet continuously for vulnerable machines to attack.
- Exploits vulnerabilities: takes advantage of known vulnerabilities such as buffer overflows, http input validation vulnerabilities and known default passwords to gain unauthorized administrative access.

Blended threats have a considerable impact on naming: due to their nature, they almost *require* information sharing with the more general security world.

Thus, we should examine carefully the information the name of a blended threat should convey to the user: should blended threats (in this case, viruses that use exploits in order to spread) be clearly identified as such by the naming scheme? This question probes more fundamental issues about the overall *goal* of the naming scheme, referred to above.

Information Sharing with the Security World

One of the reasons for re-examining the attributes we attach to a virus name is the increased overlap between the virus world and the security world due to the rising prevalence of blended threats. After all, computer viruses that utilize exploits in order to spread are relevant to both the traditional computer security world and the anti-virus world.

If, for example, a new virus emerged that used a previously unknown exploit, this sample would be of significant interest to virus and security researchers, and should be shared between both. As blended threats become more prevalent, such situations are likely to occur with increasing regularity.

Just like computer viruses, vulnerabilities are classified (named) according to certain rules. However, unlike computer viruses, the security world has chosen a far simpler naming scheme for vulnerability identification.

This sounds great, until one realizes that there are at least three competing schemes in use: the Microsoft

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. http://www.virusbtn.com/. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

Vulnerability ID, the Bugtraq ID, and the CVE vulnerability and candidate IDs. The Microsoft Vulnerability ID provides details of vulnerabilities specific to *Microsoft* applications. The Bugtraq and CVE IDs are numerical identifiers for newly announced vulnerabilities provided respectively by *SecurityFocus* and the CVE Board.

The lack of a single universal naming standard for vulnerabilities makes the problem of dealing with blended threats slightly more difficult, as listing the vulnerabilities associated with a particular blended threat may be somewhat verbose.

However, the users do *need* to know if a particular virus exploits vulnerabilities in order to spread, as the presence of the virus on a machine could signify the presence of an underlying security flaw – such a flaw can have far more serious consequences for the user than viral replication.

Conversely, the absence of system vulnerability could mean the virus presents less of a risk from a security standpoint. Thus, there is a good argument that, just as virus names indicate action and platform, another moniker should be added, indicating the virus is a 'blended threat' and alerting the user to check for exploits. After all, this information seems at least equally important to some of the other pieces of information that the proposed standard should have embedded in the name.

URI Confusion?

One of the aspects of the naming scheme that is, on the surface, attractive, is the way in which the formal name of the virus is structured. Quoting Nick's earlier article:

> 'The general form of malware names under the new guidelines is:
>
> `<malware_type>://<platform>/<family_name>.<group_name>.<infective_length>.<sub_variant><devolution><modifiers>`'

This system looks suspiciously like a URL (Uniform Resource Locator) used for locating resources. However, the way it is structured and described is too loose to be compatible with the correct URL naming scheme (see RFC 2396 for a complete description of URIs and how they relate to URLs).

Furthermore, while the name looks 'scientific' it seems to be a misuse of the purpose of the scheme. Quoting from RFC 2396:

> '**Resource**
> A resource can be anything that has identity. Familiar examples include an electronic document, an image, a service (e.g., 'today's weather report for Los Angeles'), and a collection of other resources. Not all resources are network 'retrievable'; e.g., human beings, corporations, and bound books in a library can also be considered resources. The resource is the conceptual mapping to an entity or set of entities, not necessarily the entity which corresponds to that mapping at any particular instance in time. Thus, a resource can remain constant even when its content – the entities to which it currently corresponds – changes over time, provided that the conceptual mapping is not changed in the process.'

While information about a virus is certainly a resource, the <malware_type> attribute is arguably misplaced: it does not really present a scheme in the strict sense of the word, and creates a different scheme (namespace) for each type of malware.

Possibly, making the scheme a general 'malware' namespace and building out a hierarchy below it would make more sense – this is in keeping with the general approach for URIs, which are generally of the format:

`<scheme>:<scheme-specific-part>`

An additional issue is that one might expect to use a URN format (RFC 2141) for names – as such, the names would fit more neatly into the URI/URN system. However, the names given resemble URLs more closely, despite the fact that there is, in fact, no resource to locate. As Nick's paper gives no discussion of the reasoning behind the URL-like format, it is hard to speculate as to the advantages of this scheme as seen by the CARO members proposing it; on the surface it is confusing, closely resembling, but separate from, a more familiar format.

In summary, I would argue that the URL-like format is confusing and should be re-thought for two reasons:

First, if it is the intent to use a URI format, the name scheme should be *completely* compliant to the various RFCs that describe the format – and this should be stated explicitly. If the intent is not to be a strict URI and not related to a URL, the name should be modified so that it does not look like one. The current format seems to borrow heavily from the URL layout, but with no obvious benefit.

Secondly, the use of a URI/URN for a name seems somewhat unusual – from a user perspective names should be as simple and memorable as possible. I am aware of no other naming scheme of this type that follows such a format, as the classification and naming of things is usually not a good candidate for a URL-like scheme, which is better suited for identifying and locating resources.

A Complementary Solution?

While it is tempting to cram various attributes about the virus into its full name, I have come to believe that this is a pointless task. Ultimately, one is attempting to force *attributes* into what is actually an *identifier*. This will lead to names that are both large and cumbersome (such as the ones proposed) or that are lacking important attributes (i.e. names that do not indicate that an exploit is closely associated with the virus).

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. Tel +44 1235 555139. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

It is unlikely that sufficient information can ever be embedded in a virus name such that no other information about it is needed – indeed, a user should always find out more details about any virus that their anti-virus software has detected on their machine. Thus, the attributes attached to the name are somewhat redundant compared to the information that will be contained in a more detailed analysis of the virus.

A better approach for the industry might be to develop an XML schema that defines a standard format for describing computer viruses. Then, the 'name' of the virus is simply the unique identifier that is used to access the required information that can be rendered to the appropriate format at display time. Thus, many of the naming issues become irrelevant as the name simply becomes a unique identifier used to locate further information.

While such a scheme does not solve the problem of two scanners identifying the same sample differently, it does allow for a simpler naming scheme.

Anti-virus researchers require much richer content than the corporate user who, in turn, requires much richer content than the home user. The name does not, and cannot, provide the type of information required in any of these cases.

Using an XML-based description, researchers can use the richer fields given to them by the schema for taxonomical purposes far more completely than would be possible with the proposed naming scheme, whereas users (or user-facing products) can filter out information that is not directly relevant to the current application.

Furthermore, the use of XML is attractive, as it is highly definable, easily machine-parsable, and extendable. Thus, it is the ideal format for the sharing of information between systems, platforms and communities.

Conclusion

We have gone down the same road in naming, adding bits and pieces of information about the functionality of the virus, for years. However, the fact that this is the road historically travelled does not mean it is the direction we should continue to take.

Consider a *Linux* worm that replicates on both *Windows* and *Linux* platforms, *and* exploits several vulnerabilities, *and* has a data-diddling payload *and* is a mass mailer. Under the proposed scheme, where would that information be shown in the name? This example illustrates perfectly the problems of including such attributes in the virus name – the set is never complete.

The current proposals for virus naming put forward by some in the industry mix two important concepts: attributes and identifiers. Given that a user *always* has to consult the background information regarding a virus, the only real purpose of the name is to provide a unique way of referencing the attributes associated with it.

OPINION 2

What's Pepe Been Doing Lately?

Jong Purisima
TrendLabs, Trend Micro Inc., Philippines

A few weeks ago a friend of mine, Pepe, called me and asked if I was free to join him in a 'Spolarium' – an exclusive, slightly misleading, codename for a drinking session with buddies for no particular reason. After obtaining the necessary permit from my wife and daughter, I dressed up and drove to the agreed venue. On my way, I tried to imagine a possible agenda for the session.

Usually on these occasions we play an extended version of poker called 'pusoy', while drinking some good old San Miguel beer. Sometimes we hold musical jamming sessions which usually consist of three guitars and an old biscuit tin to serve as a drum and convert a friend's lawn into a pseudo-concert arena. Other times, we opt for a karaoke session. There are times, though, when we simply talk. Our discussions tend to cover the staple topics of basketball, music, politics, showbiz rumours and neighbourhood issues. Occasionally, we talk about computers.

When I arrived at the venue I was met by a slightly distraught friend who seemed, unwittingly, to have acquired a computer virus which had been detected by his anti-virus software. He told me that an email had arrived stating that he had just sent an email infected with the notorious Klez virus. I told him not to worry because it might just be the case that his email address had been used by another Klez-infected machine. After I had explained what Klez does, my friend simply sighed and drank some more.

Another drinking buddy then piped up, 'Isn't Klez an old one?' I told him that the first Klez variant appeared in October 2001 and the more prevalent variant, Klez.H, surfaced around April 2002. 'So, what else is new?' came the follow-up question. I mentioned all the other 'prominent' malware of 2002: Bugbear, Opasoft, Yaha, Frethem, and so on …

As expected, the response was, 'I've never heard of them.' I countered, 'I think you are more interested in the next '*big*' thing after Nimda?' – which was met simply by a nod. Unfortunately, there was no answer to that question, yet.

How Big is '*Big*'?

After Nimda, it had been observed that there had been no other malware outbreaks whose magnitude was comparable to that of CIH, Ska, Melissa, LoveLetter and CodeRed. To be considered a 'big' thing, there are a few attributes that a piece of malware should possess:

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. http://www.virusbtn.com/. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

- It should be *cunningly sneaky*. It should catch everyone off-guard. CIH, though accidentally spread via media, was one of the first *Windows* infectors that employed the cavity type of infection, thus avoiding the usual file size increase – the most obvious giveaway of virus infection. Ska, on the other hand, was one of the first pieces of malware to send itself successfully via email. It demonstrated to a largely unprepared world a new truth about virus entry points.
- It should be *exceedingly swift*. It should be able to spread around the world within a timeframe that would seem an astronomical feat at the time of its release. Melissa spread around the world in a matter of days. It posed such a tremendous threat that some people predicted that the Internet was at its end. When LoveLetter came out, it spread around the world in a matter of hours – probably even minutes. During that period, a lot of viruses were utilizing the same techniques as those employed by Melissa. However, LoveLetter's features were relatively new. Of course, there were similar predictions to those made during the Melissa outbreak.
- It should be *adroitly stealthy*. The malware should be so unnoticeable that its date of discovery is far later than the date of first infection. When CodeRed was discovered it was calculated to have started infection days before its first execution took place. Its utilization of an *IIS* vulnerability also introduced a new malware entry point and its memory-only approach was considered the first of its kind.
- It should be *hard to crack*. Suppression of the malware should be a relatively difficult feat and pose a challenging task. Nimda used several vectors of attack: from mass-mailing, to *IIS* vulnerabilities, to network share infection, to file infection. Upon its discovery, cleaning and/or defending one vector was not enough. This is the main reason why it keeps coming back. It was a classic all-or-nothing case.

In general, a 'big' thing usually comes as something totally new, something that is utterly difficult to solve, or a combination of both.

Another question that crops up in relation to this issue of 'bigness' is: 'Why have there been no recent big malware catastrophes?' A possible answer to this question is that 'Pepe' is persistently becoming more aware and is actively doing his part in protecting cyberspace.

Pepe, the Anti-Virus Vendor

Most anti-virus vendors have adjusted their approach in providing solutions to malware problems. The following are some of their more significant contributions to prevent the next 'big' thing from surfacing:

Streamlining solution cycle time. The majority of (if not all) anti-virus vendors have abandoned their old ways of providing solutions. Virus patterns or definitions are now readily available in just a matter of hours, whereas in the old days it used to take days or even weeks to provide solutions to urgent customer problems. Providing speedy solutions to minor problems contributes significantly to the prevention of big ones.

Automated product updates. Nowadays, an anti-virus product comes with modules that check a central server for updates constantly, and that download them automatically. Alternatively, the module will inform the user that an update is available and prompt him/her to download it.

This approach protects relatively vulnerable systems before they can be compromised. The use of an old component for the scanner is also a good reminder in the sense that it warns the user that they may be using an outdated program and that they are, theoretically, not protected in the best possible way.

Eliminating the possibility of being surprised. Most of the pieces of malware that have appeared recently have not been as successful as the 'big ones' in carrying out their malicious ambitions. This is mainly because anti-virus vendors have been able to arm themselves with newer and more reliable methods of anticipating potential threats. A very apt example is the technology of heuristic and generic detection.

This technology raises anti-virus detection capabilities to such an extent that it is capable of stopping a potential 'biggie' from spreading and causing an outbreak. Also, more and more AV researchers are spending their time looking for new malware entry points, thus making it extremely difficult for potential 'biggies' to succeed.

Timely dissemination of information. AV vendors have become more efficient in providing information to users on how to protect their systems. Some vendors even publish the information or analysis of malware before they actually deliver a sound solution for it. This approach makes the user aware of potential threats even before they can be protected by their AV products.

Providing other value-added modules or services. Personal firewalls and content filtering are the more recent additions to most AV product ranges and are proving to be very helpful in protecting customers. There are even 'Outbreak Prevention Policies' or rules which a user can utilize to make sure that certain malware cannot invade the system.

Anti-virus vendors are looking continuously for ways to improve their products and services. At the same time, they are making sure that nothing becomes big by eliminating the threats while they are still small. A stitch in time saves nine.

Pepe, the Operating System Developer

At the end of every outbreak, after much of the 'blame game' has been played, operating system developers always seem to be at the losing end. They often hear such

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. Tel +44 1235 555139. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

not-so-subtle comments as: 'If only Pepe would have done this,' or 'If only Pepe would have done that,' or 'If Pepe had only listened to me.' As a result, operating system developers are becoming increasingly aware of their critical role in maintaining a safe computing environment.

Operating system developers try to solve reported flaws at the soonest possible opportunity, which can be considered a far cry from their previous approach. Moreover, they are also looking into other areas that may be compromised. Their investment in security will also reap rewards in their aim to create a more secure environment for their future products.

The same goes for the applications developers who, these days, make sure that every hole in their architecture gets patched immediately.

It is no longer surprising to see a patch for vulnerabilities only the day after the release of a new product. It goes to show that the commitment that the product developers should have shown in the first place is slowly being demonstrated. It may be a sign of poor testing and/or poor product development, but the key here is taking the responsibility for an unexpected dilemma and providing the solution for it, fast.

Pepe, Who Works for ISPs

Most ISPs (Internet Service Providers) are also doing their share of protecting cyberspace. One of the most notable contributions is their quick response to requests which are made regarding malicious websites that need to be taken down. Another is the fact that they are now responsibly and meticulously scanning email messages and the web pages of the accounts they host.

Pepe, the Law-maker

Although there is a wide array of implementation, laws concerning hooliganism in the digital world are slowly being established. Furthermore, they are the kind that are not very easy to ignore and they make one think twice before carrying out something that is digitally undesirable.

Although recent sentencing of virus writers may have been considered too lenient in some sectors, the impact on society is to instill in the minds of users that cyber-vandalism is punishable by law and, like every crime, you may get away with it – but if you don't, you are in deep trouble.

Pepe, the Corporate Network Administrator

Traditionally perceived as doing nothing and just playing god, these days network administrators spend most of their time setting up policies, reviewing the network and system architecture, checking for exploits and vulnerabilities, deploying anti-virus solutions at different levels, and making sure that each and every system connected to their networks is secure.

Life has not really been easy for network administrators since the occurrence of multiple outbreaks in the year of the worm. Not only did they do the cleaning and restoration after the havoc, but they also painstakingly made sure that each and every system could not become the entry point or origin of such digital mayhem.

When losses are calculated in the aftermath of a major outbreak, it is usually related to the loss in production of a certain company or entity. This means that the assurance of continuous production is in the network administrator's hands and sometimes it could even mean losing one's source of income. It is not an easy job, but just like any other, somebody has to do (and enforce) it.

Pepe, the End User (aka the Usual Suspect)

I tend to agree with Dr Vesselin Bontchev that educating users might be a waste of time and that forcing the users to 'behave properly' might be the key. Furthermore, as increasing numbers of users are joining the Internet bandwagon, the task of educating users is becoming more difficult each day.

Recently, however, it has been noticeable that these users might be behaving 'properly'. Perhaps they are learning slowly, are slowly being educated, or perhaps all the other 'Pepes' are giving them no chance to mess things up.

But this is quite unlikely. Patches to systems are only downloaded and installed if a user allows it. Anti-virus products are only updated based on the configuration set by its user. The choice to click on those potentially malicious attachments lies solely in the hands of the one holding the pointing device.

If none of these things are happening then there might be something going on with Pepe, the end user. Has a learning process taken place?

I can only guess that the most effective method of learning is through experience and, if all the other Pepes are not the reason why Pepe the end user is beginning to invalidate Dr Bontchev's 97% calculation (see *VB* November 2001, p.9), then Pepe the end user might just be becoming a little more knowledgeable about safe computing practices.

Pepe-logue

As my friend Pepe (who is unaware of the fact that I have used him as an example), walked into the room with two more cases of beer, I was busy talking on the phone with a member of *Trend Micro*'s support personnel about a certain worm that seemed to be spreading and was supposedly already all over the world.

I stood up, picked up my stuff and said, 'Sorry guys, gotta go. I think this new worm that attacks SQL on UDP port 1434 may be my answer to your question.' On my way back to the office, I wondered to myself: which Pepe is it this time that did not do his part?

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. http://www.virusbtn.com/. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

PRODUCT REVIEW

Virus Chaser

Matt Ham

New Technology Wave's *Virus Chaser* made its *Virus Bulletin* debut in last month's comparative review (see *VB*, February 2003, p.21), and promptly gained its first VB 100% award.

Products entering *VB*'s comparatives for the first time tend to fall into two categories, those that achieve a VB 100% at first attempt and those that face a long struggle towards eventual success. The products in the former category are, more often than not, rebadged versions of software that has already found success. *Virus Chaser* is exactly this.

The product behind the scenes in this case is *DialogueScience*'s *Dr.Web*, which has gained VB 100% awards consistently in its own right. *Dr.Web* has a number of small idiosyncrasies which might well be transferred to a product that is derived from it, not the least of which is a tendency towards a functional appearance rather than being very aesthetically pleasing. Whether its more practical features are transferred to *Virus Chaser* will become apparent as the review unfolds.

Virus Chaser is another among the recent surge of products sent for review from Korea. There is a booming home market for the Korean-based *New Technology Wave* (*NTW*) to tap into, with some stiff local competition, while the ever-present large international companies are also contenders for market share.

The Package and Documentation

The package supplied for review was the boxed version of the product for Korean consumption, to which were added a recent version of the English language manual and an electronic version of the English language program.

The boxed version contains a jewel-cased CD, a licence agreement with serial number, and a slim manual. However, the manual was out of date, and a new version was provided separately. All of this material was in Korean, although the CD installed an English version of the software.

The newer version of the printed manual showed signs of a less-than-perfect translation process. Although both comprehensible and comprehensive, the turns of phrase in many cases were either slightly too familiar or rather stilted. However, the publication dates of the Korean and English manuals are very close together and the quality of translation is more than acceptable for such an apparently speedy job.

The manual is illustrated throughout with screenshots which, when combined with the text, leave very little to chance as far as information is concerned. One thing that would have been appreciated, however, is some form of context-sensitive help within the program itself since, although complete, the manual is sufficiently large to make locating the relevant information a slow process. For the same reason an index in the printed manual would be helpful.

Web Presence

NTW's website is available in English, Korean and Japanese. While the Korean version can be found at http://www.viruschaser.com/, the English version is at the rather less intuitive http://www.virusdesk.com/. Appearances suggest that the English website is a close translation of the Korean site.

With the main page comes an additional pop-up screen for current news which, thoughtfully, may be configured so as to appear only once per day. During the review process the pop-up consisted of a large pronouncement of the product's recent VB 100% achievement.

The slight mistranslations found in the manual are more prevalent within the website, probably as a result of the much larger and faster turnover of information.

Virus descriptions, news and product descriptions account for the bulk of the site. The marketing information does veer somewhat towards hyperbole in some cases. The slogan 'Meet the world's top product' is, perhaps, a reflection of the fierceness of the competition *NTW* faces.

Installation and Update

Virus Chaser is packaged in an InstallShield wrapper for distribution purposes, providing both look and feel that are no different from thousands of other applications on the market. For the purposes of review, installation was carried out on *Windows XP*, *Windows 98* and *Windows 98 SE* machines. Unless otherwise specified, the results mentioned refer to the version installed on *XP*.

As is standard with InstallShield installations, the initial stage involves identification of user and company name and destination. Once these formalities are over, the all-important installation options are reached. By default, updates will be triggered as soon as installation is complete, whereas local disks, memory and boot sectors will not be scanned.

Upon selecting the 'next' button, a few progress bars flash, tray icons appear, and the product is installed without further interaction. Matters were even more automated when installing the software from the CD, since the installation process is set up to autorun.

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. Tel +44 1235 555139. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

Updating was triggered at the end of installation and this, again, was a fairly transparent activity. The servers were accessed quickly on most occasions when updates were triggered, though there was one occasion on which the servers could not be contacted for five minutes or so. This small interruption in service does not give great cause for concern – although the lack of a warning within the program is a little more worrying.

One feature that remained untested was *Virus Chaser*'s ability to detect the presence of other products on a target machine and remove these as part of its installation procedure.

The lack of interactive requirements is surprising given the lengthy processes by which many products are applied to a target machine. The end results of the all-but-silent installation are to place a link to the on-demand scanner on the desktop, an icon in the tray, and an entry in the program menu. Right-click scanning is also supported.

Features

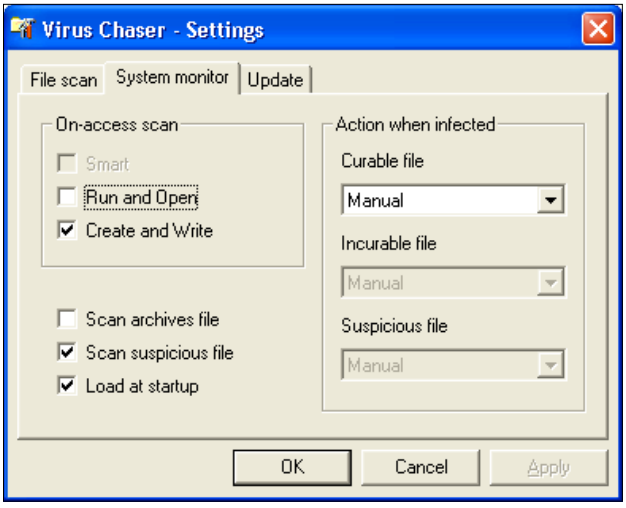
Covering the program menu first, the entries in this area are variable between the very simple and the fairly complex. However, of the programs that have been reviewed recently, *Virus Chaser* is one of the least complicated to operate.

Entries in the program menu are Configration (*sic*), Real time monitor, Scan memory and boot sector, Uninstall, Update, and Virus Chaser – the last of these being a link to the main application.

Configuration

This dialog consists of three tabs, labelled 'File scan', 'System monitor' and 'Update'. Not surprisingly, these settings reflect the product's *Dr.Web* ancestry.

The default settings under the File scan tab are with scanning activated for 'Suspicious file' and 'Archives' and deactivated for 'Executable file only'. If 'Executable file only' is selected for scanning, the entry for 'Archives' becomes unselectable, though the 'Suspicious file' setting is independent of the other two.



The action taken upon detection of an infected file is also selected in this area. 'Manual' and 'Automatic cure' are the two options available.

Also under the File scan tab 'Scan priority', which starts at a medium level, can be adjusted on a slide. It is also possible to activate a feature which sends suspicious files to an unmentioned target.

Memory/boot sector scans can be activated here, though it was not apparent at what stage these scans would occur. Since there is no context-sensitive help, the manual was consulted: the scanning is activated when the program is executed.

Moving onto the second tab, System monitor (representing the on-access scanner), there are more options as to what action should be taken when infected files are detected. The options are the same as previously – that is, manual or automatic curing – although the scanned areas are broken down into 'Curable', 'Incurable' and 'Suspicious' subsets. The default setting for this action is manual curing on curable files – which leads, automatically, to manual curing for the other two categories. Only when curable files are set for automatic disinfection does the automatic disinfection option become available for other file designations.

Other options within the System monitor configuration tab include the selection of which file actions will trigger scanning. Originally this is set to 'Create and Write', with 'Run and Open' unchecked and 'Smart' unavailable. 'Smart' becomes selectable only when the other two options are unselected – though any combination of one or both of 'Create and Write' and 'Run and Open' are considered valid.

As for which targets are to be scanned and when, the default settings are 'Scan suspicious file' and 'Load at startup'. The option to scan archives is disabled. Each of these settings may be toggled on or off, independently of the others.

The last of the three tabs is Updates. This is the area in which the Product Key may be inserted and the update URL edited. The preset update URL, free.viruschaser.net, was used for the updates described above. Update status seems to be hard-wired to Automatic update. The remaining part of this tab relates to the pattern number, the number of viruses detected, last update and date of expiry of the product licence.

Virus Chaser

The main program is reached either through the program menu or the link on the desktop. It follows the usual screen division of scanners: left-hand pane for view selection, right-hand pane for the view itself and above these a selection of drop-down menus and icons. As ever, there is

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. http://www.virusbtn.com/. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

some degree of duplication among the various menus, icons and standalone programs.

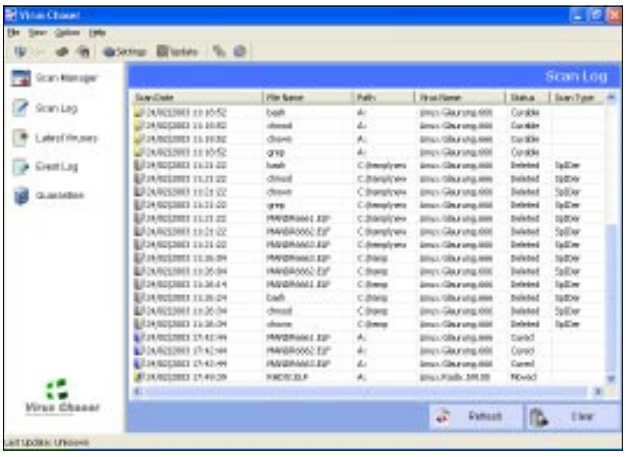
The menus are divided into File, View, Option and Help. File is a somewhat bizarre title for this menu since, from here, scans of the selected areas on disks, memory or boot sectors may be initiated. The View menu duplicates the actions of the view selection icons in the left-hand pane and will be covered later. Option either launches the settings dialog or initiates the update with parameters selected elsewhere. Both of these actions have been covered earlier in the review. Finally, Help provides program information and a link to the online version of the English manual.

The icons on the top bar all represent commands duplicated elsewhere. Several of these are unlabelled, but using the cunning investigative method of pressing each in turn it was discovered that the effects were, from left to right, scan, scan memory, scan boot sectors, initiate the settings dialog, update, program information and a link to the online manual.

This leaves the five icons in the left pane and the corresponding five views in the right-hand pane. These are, from top to bottom, Scan Manager, Scan Log, Latest Viruses, Event Log and Quarantine. There is an additional logo icon which links to the *Virus Chaser* website – although, even in the English version of the program, this links to the Korean homepage.

The Scan manager view is a familiar tree representation of the system, where objects can be selected for scanning. Objects may be selected for scanning on a drive or folder basis, but individual files cannot be selected.

The view here is clearly derived from the way in which *Windows* regards the computer to be configured, rather than a physical representation of the machine and its folder structure. This is useful in that it allows easy scanning of networked areas and the Recycle Bin. However, it does seem excessive to allow all machines on a network to be flagged for scanning, through selection of My Network Places. Similarly, the ability to scan the Control Panel menu would seem to be of little practical use.



The Scan Log window fills the right-hand pane with information about infected files. The details provided are scan date, file name, path, virus name, status and scan type. The data is sorted by scan date. The logging here is for both the on-demand and on-access components of the scanner – scan type showing up as blank for the former, SpIDer for the latter. On demand, file status refers to the state of disinfection of the file – for example whether it is disinfectable, non-disinfectable, or already disinfected. With the on-access scanner the status may also be 'Locked'.

The Latest Viruses view is one which will probably be used very little in day-to-day use. This displays the names of the viruses whose definitions are the latest to have been added to the definitions file. In almost all cases this is limited to the date of update and the name of the virus. A 'Remark' column is also supplied, though this was not used for any of the entries produced during test updates.

The Event Log view will be of more value. Events seem to be limited to updates – thus this is a means by which to determine when the last update was performed, broken down into individual files. However, this area logs only successful updates, and information about failed updates remains unrecorded.

The last of the views available is Quarantine. This view is odd in that, under the default settings at least, there seemed to be no possible way to invoke quarantining of files. The most likely option, that of 'Move File' when the on-access scanner is triggered, did not move the file in question but deleted it. This was found to be the case with all combinations of settings for the on-access scanner. Quarantining was eventually achieved through the on-demand scanner and the process is discussed further later in the review.

The Tray Icon

The tray icon defaults to the spider which will be familiar to anyone who has run *Dr.Web* with its on-access component SpiderGuard. Right-clicking on the icon opens up a small menu. The topmost contents of this menu are Open, Settings, Update and Memory/Boot Sector Scan. Each of these is a direct link to areas already discussed.

The next two entries in this area are slightly more intriguing. 'Change Icon' acts as might be expected, allowing the user to swap between the SpIDer icon and one that matches the *Virus Chaser* icon. This works perfectly, though begs the question as to why this bizarre functionality should have been included.

Equally perplexing is the 'Send Log File' setting, which appeared at first to serve no purpose whatsoever. Triggering it on an isolated machine generated no warnings or errors, which would be expected if this were attempting to email log files to some outside destination. The manual was consulted with respect to this mystery and it transpired that this is a method of sending information to *NTW*, though not one which has an error message upon failure.

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. Tel +44 1235 555139. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

The final entry in this menu is the function to stop and start the on-access scanning. However, while it was possible to activate the on-access scanner from here, the option to stop the scanning did not seem to function.

Others

The remaining entries in the program menu are small enough that they cannot really justify a full entry of their very own. Real time monitor simply initiates the on-access scanner if this is not already loaded. Scan memory and boot sector and Uninstall are self-explanatory, while Update has already been discussed.

Scanning Tests

A preliminary scanning test was performed upon a collection of files, simply to gain an idea of the interface. In this case the result is a rather empty-looking dialog, stating 'Virus(es) not found'.

There are two icons on the right of the dialog, one of which displays the settings for the scan, while the other gives a basic report of how many scanned, infected, suspicious, disinfected, deleted and quarantined files are present. In the case of a scan without any detections involved, no notes are added to any log within the program. No logs are created in the program folder or the log folder in such a circumstance. When infections are noted a log file is produced in the log folder, though not mentioned in, or configurable through, any of the programs supplied.

Given that only two options are available as far as scanning action is concerned – manual or automatic disinfection – a scan using manual treatment was performed upon an infected sample set. In the default mode this caused some problems from the start.

For the scanning test a floppy disk containing infected files was selected. However, the action of inserting this disk triggered the on-access scanner. The option of 'lock' was chosen for each file detected and an on-demand scan was then performed. However, the file lock imposed by the on-access scanner was sufficient to deny access to the infected files and the on-demand scanner declared the floppy to be free from infection.

For the remainder of the on-demand testing the on-access scanner was disabled. Rescanning the same files resulted in detection, as expected, and disinfection of infected files was attempted.

Disinfection was the only option available for files which were disinfectable – deletion, quarantine or renaming were not available. For those viruses which were flagged as being non-disinfectable there was a choice between deletion or quarantine. When either the delete or quarantine option is chosen, it is possible to set this as the default choice for all further files (though there appeared to be no obvious way to reset this selection once made).

On-access scanning also has manual and automatic settings; the manual method was chosen. As mentioned previously this produces a 'move' option – which, in reality, deletes files – and options to delete, disinfect (if relevant), ignore and lock. However, selecting the delete option simply re-engaged the scan so as to create a never-ending sequence of dialog boxes, so the move option was at least faster despite its faults.

Conclusion

Virus Chaser is a different creature from its *Dr.Web* originator in a number of ways, both positive and negative. On the positive side, the look and feel of the product is significantly more modern and the manuals supplied are more useful than those supplied the last time *Dr.Web* was reviewed as a standalone product (see *VB*, March 2001, p.21).

On a somewhat less pleasing note, there are a number of features missing in terms of options upon detection. It is possible that these issues are solved by the management system for corporate environments, but this was not available for review. However, detection rates are something in which both products can take pride.

As for an overall comment, the small usability issues are those which provide the most cause for concern. Given some tweaking to improve the friendliness of these features, the product could advance from awkward, yet effective, to truly impressive.

Technical Details

Test environment: For in-lab tests 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 running *Windows XP Professional* and *Windows 98*. For tests performed outside the secure labs, an Athlon XP 1600+ machine with 512 MB RAM, 20 GB dual hard disks, DVD/CD-ROM drive and USB ADSL internet connection running *Windows 98 SE*.

Prices: (per user) one user $32, 10 users $25, 100 users $23. Renewals are at a 50% discount. Prices include updates and upgrades for a one-year period.

Developer: *New Technology Wave. Inc.*, 5th Fl., Bowon Bld., 183 Bangi-Dong, Songpa-gu, Seoul, Republic of Korea; Tel: +82 2 414 0983; Fax: +82 2 418 0054; email: sales@virusdesk.com; website http://www.virusdesk.com/.

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. http://www.virusbtn.com/. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

ADVISORY BOARD:

Pavel Baudis, Alwil Software, Czech Republic
Ray Glath, Tavisco Ltd, USA
Sarah Gordon, WildList Organization International, USA
Shimon Gruper, Aladdin Knowledge Systems Ltd, Israel
Dmitry Gryaznov, Network Associates, USA
Joe Hartmann, Trend Micro, USA
Dr Jan Hruska, Sophos Plc, UK
Eugene Kaspersky, Kaspersky Lab, Russia
Jimmy Kuo, Network Associates, USA
Costin Raiu, Kaspersky Lab, Russia
Charles Renert, Symantec Corporation, USA
Péter Ször, Symantec Corporation, USA
Roger Thompson, ICSA, USA
Joseph Wells, Fortinet, USA
Dr Steve White, IBM Research, USA

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/

No responsibility is assumed by the Publisher for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions or ideas contained in the material herein.

This publication has been registered with the Copyright Clearance Centre Ltd. Consent is given for copying of articles for personal or internal use, or for personal use of specific clients. The consent is given on the condition that the copier pays through the Centre the per-copy fee stated on each page.

END NOTES AND NEWS

The 12th Annual SysAdmin, Audit, Networking and Security Conference (SANS) takes place 7–12 March 2003 in San Diego, USA. The conference will feature 12 tracks, night activities, a vendor exhibition, and additional special events. See http://www.sans.org/.

InfoSec World Conference and Expo 2003 takes place 10–12 March 2003 in Orlando, Florida. See http://www.misti.com/.

CeBIT, one of the world's largest information technology trade fairs, runs for one week in Hannover, Germany from 12–19 March 2003. All aspects of IT are catered for, with well over 7,000 exhibitors. For full details see http://www.cebit.de/.

SACIS Expo (Security, Audit & Control of Information Systems) takes place 25–26 March 2003 in Istanbul, Turkey. Hear about the latest information security and audit developments from IT security professionals, and meet with product developers and academics. Early registrations qualify for a discount of up to 20%. For details see http://www.smartvalley.net/sacis/.

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/ +44 709 211 1950 or check http://conference.eicar.org/ for details.

Black Hat Europe 2003 takes place 12–15 May 2003 at the Grand Krasnapolsky, Amsterdam, the Netherlands. For more details 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 see http://www.gocsi.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. Those interested in sponsorship or exhibiting at the event should contact Bernadette Disborough on +44 1235 555139 or email vb2003@virusbtn.com (for details of how to submit a paper for the conference see p.7). More information can be found at http://www.virusbtn.com/conference/.

Sophos* has released *Remote Update, an application that updates remote computers with the latest upgrades of *Sophos AntiVirus* software. The application is designed to integrate with the company's *Enterprise Manager* suite. See http://www.sophos.com/.

***AhnLab, Inc.* has received the first Korea Corporate Ethics Annual Award** in the Transparent Corporate Management category. The Award was established this year in order to promote global competitiveness by Korean companies through the discovery and promotion of good corporate management examples among domestic businesses. See http://www.ahnlab.com/.

The beta version of *BitDefender*'s home user product *BitDefender Antivirus - Standard Edition* has been released. Users are invited to trial the software and to take part in 'a testing contest endowed with valuable prizes'. See http://www.bitdefender.com/.

VIRUS BULLETIN **©2003 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. Tel +44 1235 555139. /2003/$0.00+2.50**
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.